

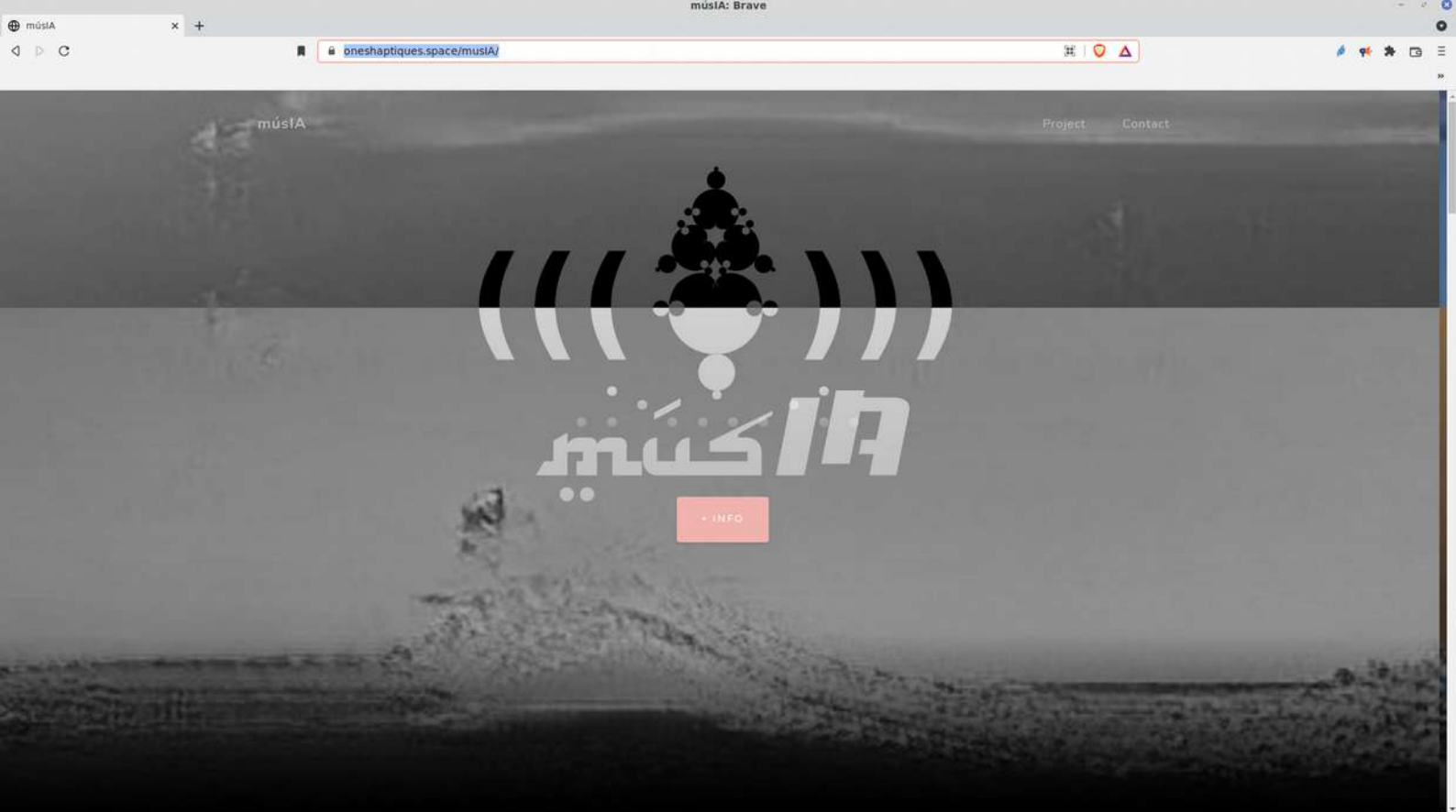
papers & counter · papers

músIA

*A research about Generative Algorithms and AI
applied to music & creativity*

Xavier Manzanares

Beques Recerca i Creació Osic gencat 2021



músIA

[def]

músia > [neol.] concepte contrari d'amúsia > capacitat de reproduir o reconèixer sons musicals

amúsia > Pèrdua de la capacitat de reproduir o reconèixer sons musicals

musIA > música +AI (en el text s'utilitzarà indistintament el terme IA i el terme anglès AI)

Com a evolució del projecte Ones Hàptiques i tot enfocant més la qüestió als instruments sonors automatitzats, el propòsit de la present recerca **músIA** és estudiar les diferents tècniques creatives amb tecnologies algorítmiques avançades i actuals, els precedents, l'estat actual i les perspectives en vers a la Música Generativa , i la Música assistida amb tècniques d'Intel·ligència Artificial .*

La singularitat d'aquest estudi roman en la idea d'analitzar i comparar ambdues tipologies que si bé tenen molts punts comuns, se situen en paradigmes diferents. Sintèticament : mentre els sistemes Generatius s'inspiren i modelen en comportaments biològics / físics / geològics, els sistemes d'AI s'inspiren i modelen en la cognició i la intel·ligència humana.

Un altre element singular és analitzar i estudiar la fenomenologia del que representa la creació humana, creativitat, intuïció, etc. i per tant poder realitzar connexions analítiques i conceptuals entre els sistemes AI vs la cognició humana, així com els sistemes AI vs la cognició i percepció en altres organismes. En definitiva estudiar els límits que els procediments de la AI mostren en tasques creatives.

La publicació de tot el procés de recerca es pot consultar a la següent url <http://oneshaptiques.space/musIA>
També es poden trobar repositoris i tutorials de tall tècnic a <https://github.com/xamanza>

La recerca s'inicia a l'Abril de 2021 i inclou les següents càpsules :

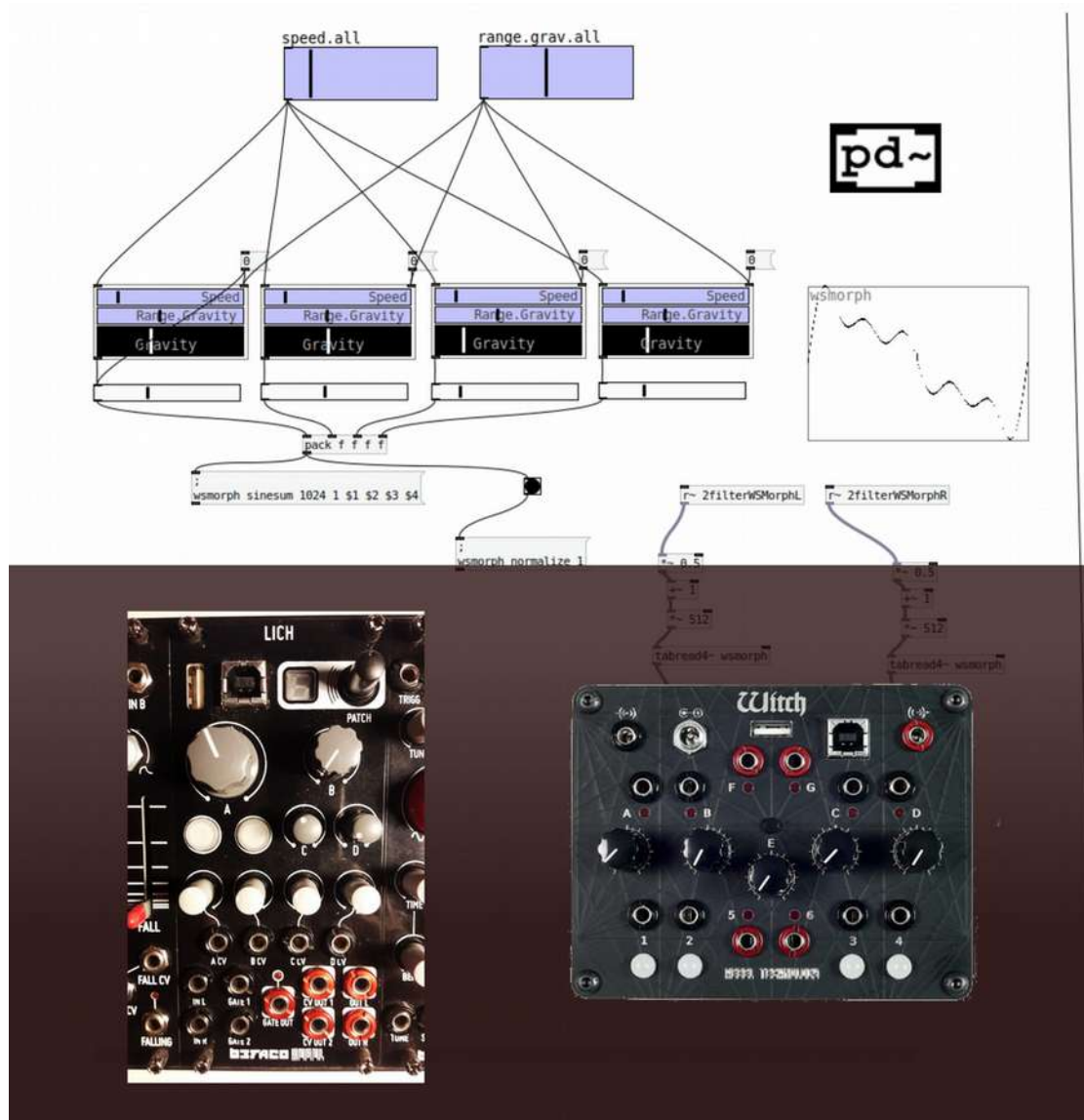
1 LICH·WITCH DEVS

url & info > <https://github.com/xamanza/LICH.Pd.Patches>

Treball de recerca i desenvolupament d'algoritmes sonors a les plataformes **Lich** i **Witch**. Aquests dispositius desenvolupats pels **luthiers electrònics Befaco i Rebel Technologies**, tenen la particularitat que es poden programar els algoritmes que els fan funcionar.

Donada aquesta singular propietat converteix a aquests dispositius com a plataformes creatives per a músics electrònics amb un enorme potencial.

Els experiments a implementar de la recerca es realitzaran amb el dispositiu **Witch** atès que aquest esdevé un instrument 'stand-alone' (funcionant per sí mateix sense altres màquines auxiliars, i per tant optimitzant consum) Podeu veure alguns dels algoritmes desenvolupats aquí :



2 LICH·WITCH TUTORIALS

url & info >

https://github.com/xamanza/LICH.Pd.Patches/blob/main/Pd4LICH.Part1.PdTutorialEssentials_PdLangFundamentals.pdf
https://github.com/xamanza/LICH.Pd.Patches/blob/main/Pd4LICH%26WITCH_Part3.RebelTech.BrowserCompilingTool.pdf

Tutorials dels desenvolupaments anteriors: Des d'un tutorial per a comprendre les bases del llenguatge de programació Pd (pure-data) -que és un dels llenguatges suportats per als dispositius anteriors-, fins a entendre el procés de compilació i instal·lació dels algorismes dins el hardware del Lich i Witch.

1

Pd Tutorials for LICH & WITCH Devels

1. Pd. Tutorial Essentials

Tutorial by Xavi Manzanares

<http://xavimanzanares.oneshaptiques.space>

by-sa // 2021

LICH Module by ://
Befaco & Rebel Technologies

PD
FOR
LICH
&
WITCH
pd~



3 PAPERS 6 COUNTER.PAPERS

url & info > https://oneshaptiques.space/musIA/pdfs/musIA_papers+counterpapers.pdf

Procés de recerca amb 'papers'(articles més formals des d'una vessant acadèmica), i alter-papers o articles de reflexió en vers al tema de la recerca amb més informalitat / no-convencionalitat.

Aquests articles són els següents :

404.GNRTV.RHYTHM.BOX.pdf

Artificial Stupidity.pdf

Basilisc de Roko.pdf

Gaugan imago.pdf

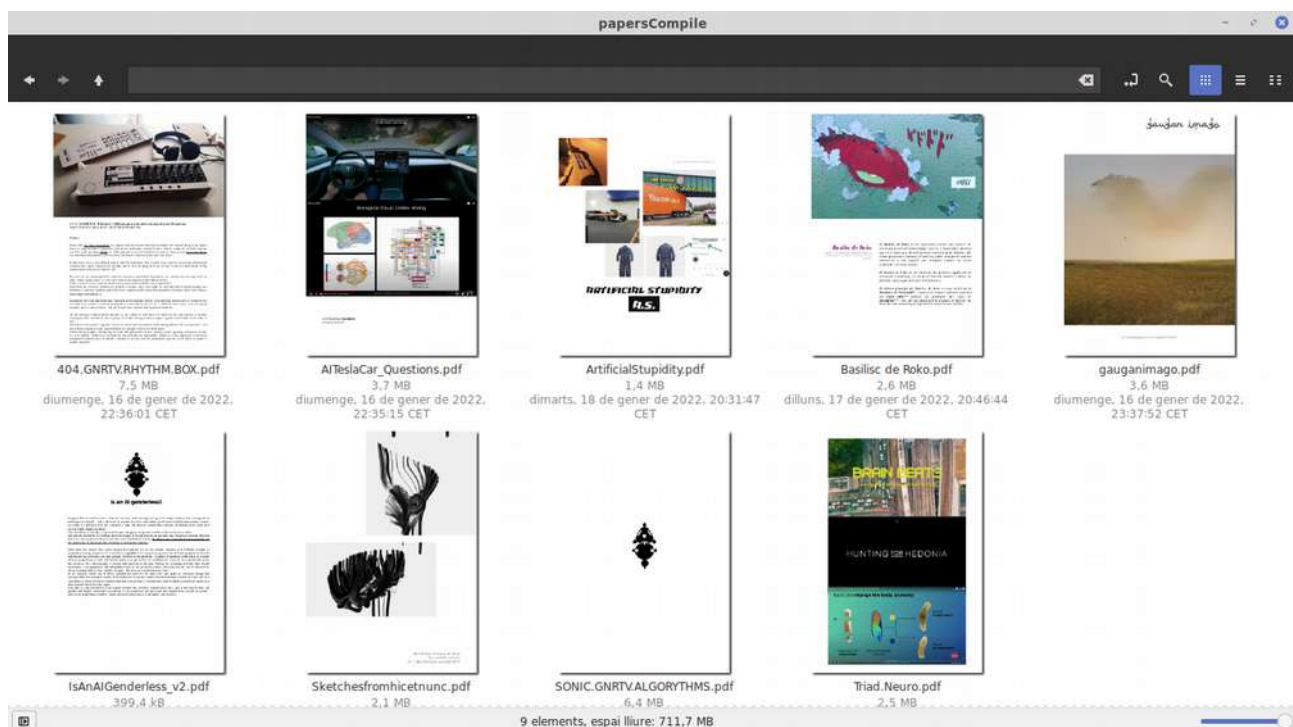
Is An AI Genderless.pdf

Sketches from hicetnunc.pdf

SONIC.GNRTV.ALGORYTHMS.pdf

Triad.Neuro.pdf

AITeslaCar_ Questions.pdf



4 TUTORIAL

INTRODUCCIÓ ALS MACHINE LEARNING

url & info > https://oneshaptiques.space/musIA/pdfs/musIA_Intro_al_Machine_Learning.pdf

Tutorial d'iniciació al Machine Learning per a aplicacions creatives. Aquest tutorial és fruit d'haver realitzat el curs de Rebecca Fiebrink 'Machine Learning for Musicians and Artists'

<https://www.kadenze.com/courses/machine-learning-for-musicians-and-artists-v>

Es tracta d'un recull sintètic per a que els conceptes sovint complexos i ferragosos en vers al Machine Learning i la Intelligència Artificial pugui ser comprensible per al públic en general.



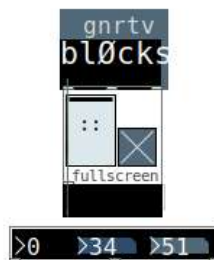
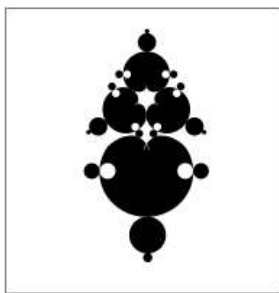
Beques de recerca i creació OSIC: generat 2021
projecte músIA
Xavi Manzanares
2021 / 2022

**MACHINE LEARNING
FOR NEWBIES
VOL I INTRO**

5 COBI. GNRTV. BLOCKS

url & info > <https://oneshaptiques.space/musIA/code/GNRTV.BLOCKS.v1.0.zip>
& <https://github.com/xamanza>

Kit de programació modular **GNRTV.BLOCKS** per a fer instruments Algorítmics Generatius Sonors i altres experiments sònics amb algorismes de Machine Learning. Aquest kit de programació és un treball en procés que quedarà documentat i disponible a les següents urls.



GNRTV.BLOCKS 1 by OH-LAB & Xavier Manzanares 2021/22

GNRTV.BLOCKS és un kit de programació creativa orientada als instruments sonors, que consisteix en un conjunt d'eines i funcions modulars (interconnectables entre si), entre les que hi figuren seqüenciadors, sintetitzadors, generadors amb models físics, motors de temps, filtres, FX, mòduls d'espacialització, mòduls d'interacció, entre altres funcionalitats.

El kit és un recull de funcions programades amb el llenguatge visual Puredata (pd /pd-l2ork).

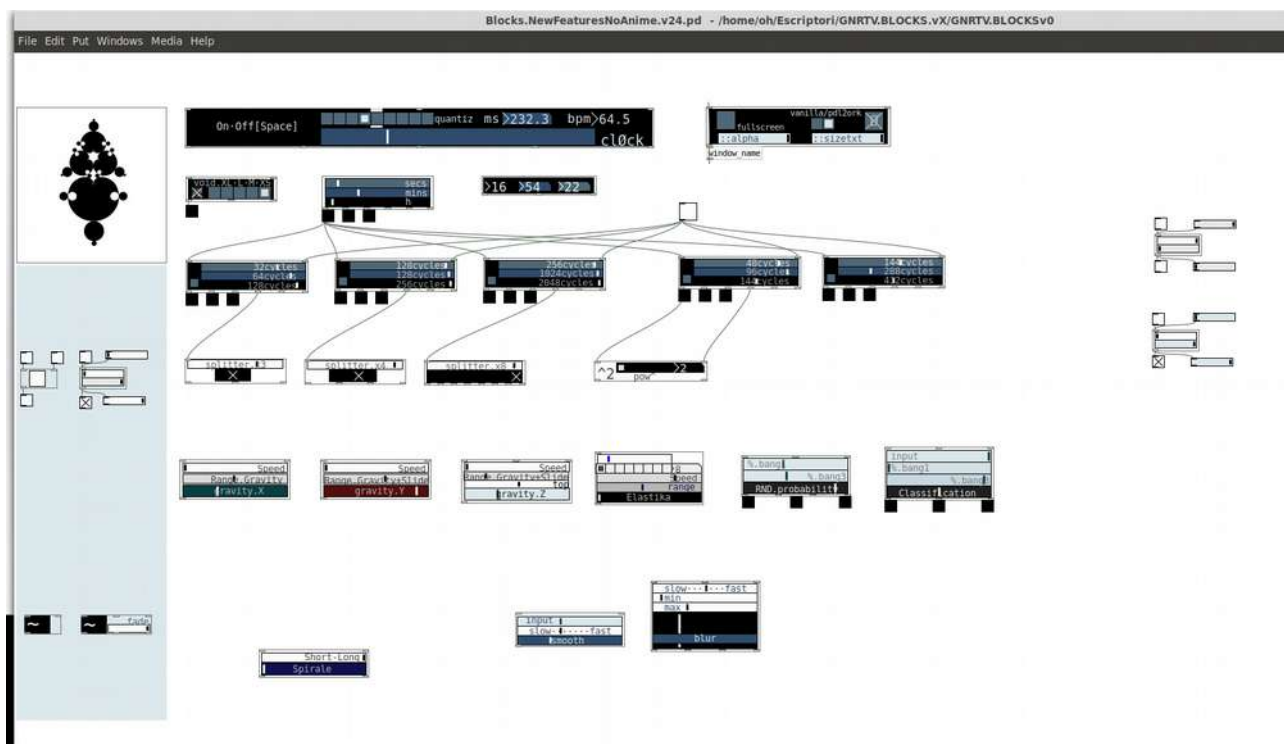
Un cop acabat el procés de desenvolupament i implementació d'aquesta recerca es publicarà amb els pertinents tutorials

Aquest kit quedarà publicat en obert per a desenvolupadors, músics, artistes, livecoders, investigadores, docents, etc. i en general per a persones inquietes que vulguin incrementar els seus coneixements en vers a la programació generativa i a la AI aplicada a la música.

6 TUTORIAL. GNRTV. BLOCKS

url & info > https://oneshaptiques.space/musIA/pdfs/Tuto.GenerativeBlocks_v_ANG.pdf

Tutorial del toolkit de programació generativa GNRTV.BLOCKS



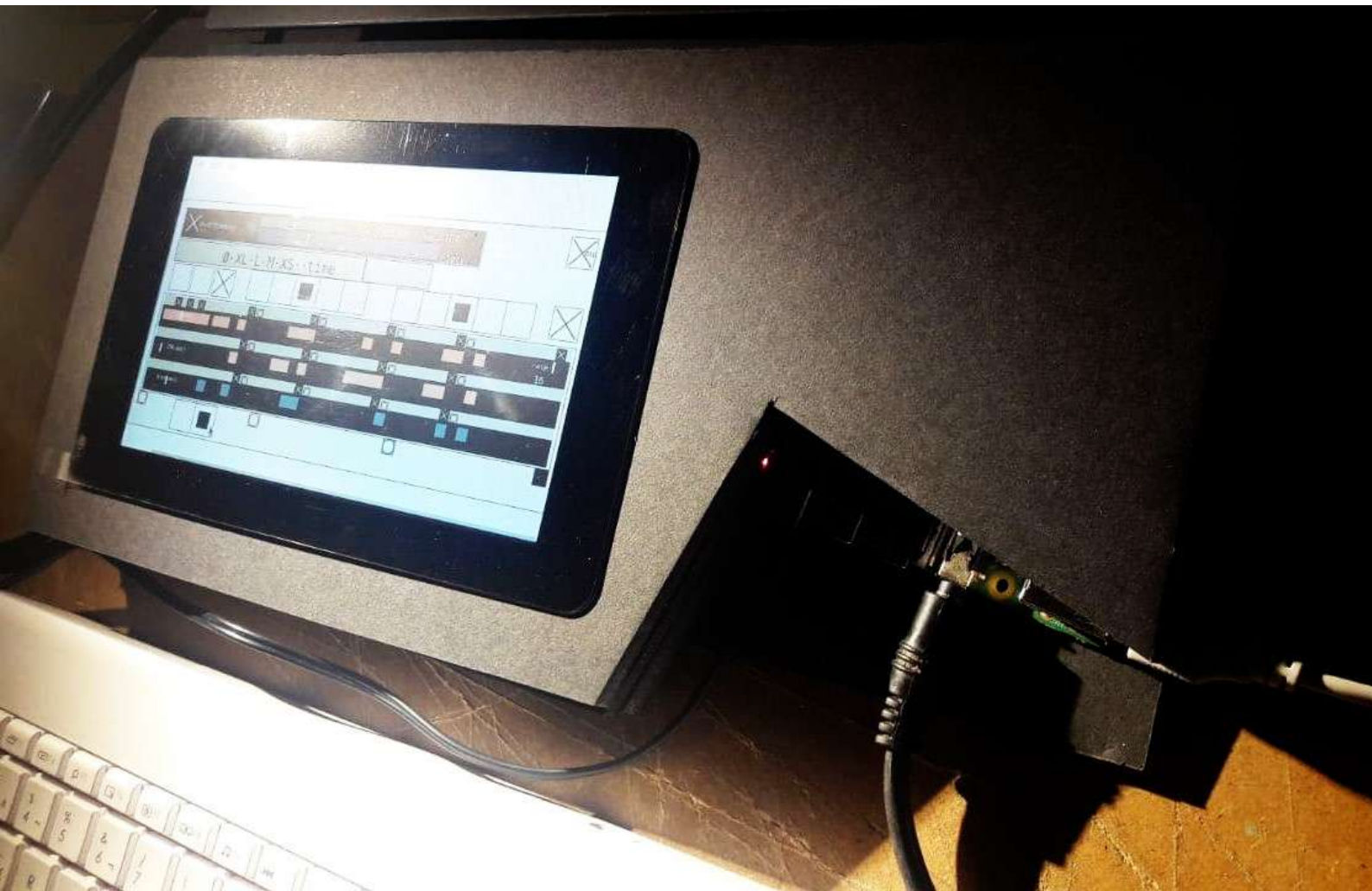
7 EXPERIMENTS AMB GNRTV.BLOCKS

url & info > <https://youtu.be/Npv4LtDSCdU>

Implementació de temes singulars i interessants de la recerca en instruments i dispositius físics actualment en desenvolupament (veure anterior menció OH.404 i Witch)

S'han realitzat alguns experiments amb una variació de la versió OH on el control del algoritmes (en aquesta versió s'han provat uns algoritmes de ML basats en la creació d'una matriu de patrons on recalcula les posicions intermèdies per un mètode inspirat en l'algoritme k-nearest neighbour algorithm, utilitzat a diverses tècniques de ML supervisat.

En la següent imatge es pot veure l'ensamblatge d'aquest prototip que en fases posteriors d'aquesta recerca s'implementarà amb un dispositiu apte per a músics i performers electrònics.

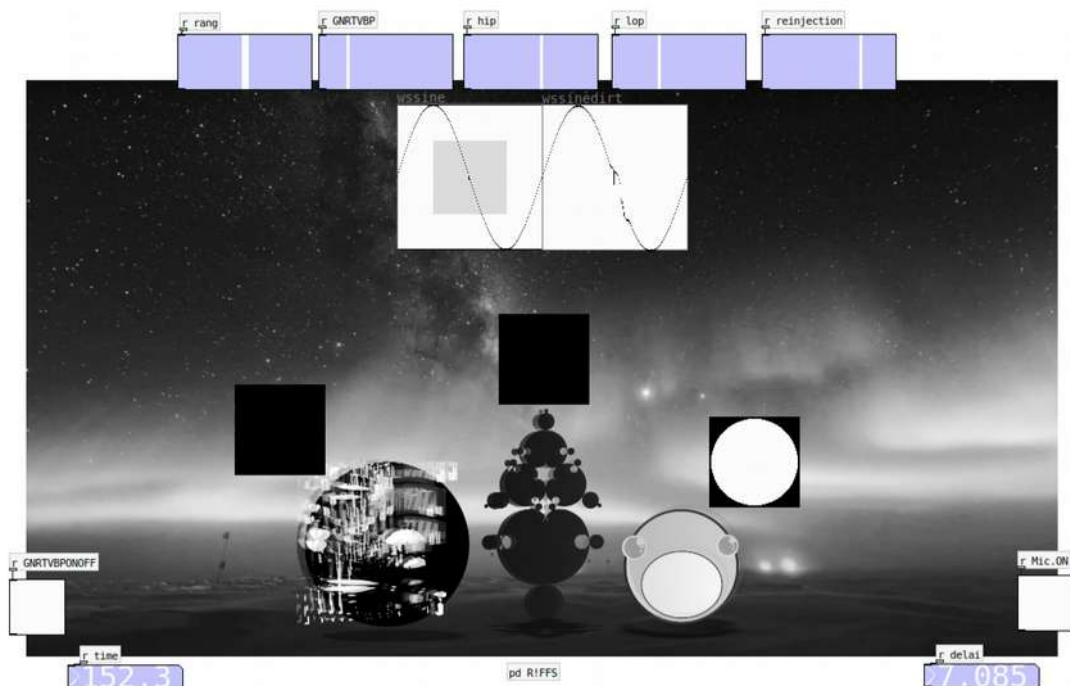




Amb la plataforma OH 404 iniciada a principis d'any 2021, també s'han realitzat experiments realitzats amb GNRTV.BLOCKS. Aquest giny integra un maquinari dedicat amb Raspberry Pi 4 de 4gb i un controlador midi korg nanockontrol integrat.

També cal destacar el treball de desenvolupament i recerca implícit a musIA enfocat en el cosmos de sintetitzadors modulars amb els dispositius LICH i WITCH (descrits al primer punt d'aquest document).

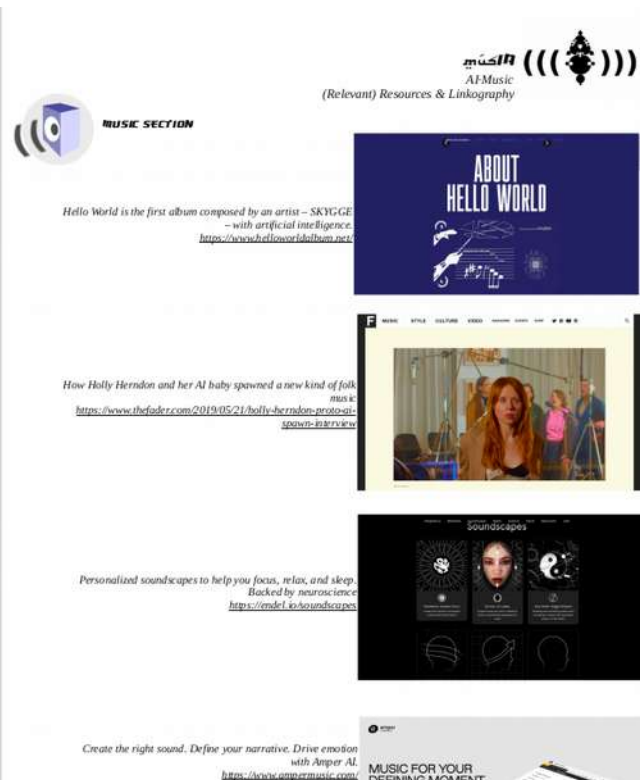
Altres experiments generatius destacables dins el procés de recerca músIA ha estat l'instrument RIFFS <https://github.com/xamanza/R1FFFS> realitzat al maig de 2021

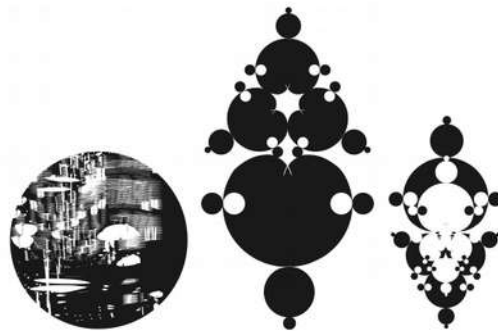


8 LINKOGRAFIA I REFERÈNCIES

url & info > https://oneshaptiques.space/musIA/pdfs/musIA_RelevantResources+Linkography_Music+ARTS.pdf

Document de linkografia sobre referències i experiments d'AI aplicada a la música i a la creació.





**GNRTV
BLOCKS**



MUSIA

<http://oneshaptiques.space/musIA>

<https://github.com/xamanza>

by xavier manzanares

<http://xavimanzanares.oneshaptiques.space/>

<http://oneshaptiques.space>

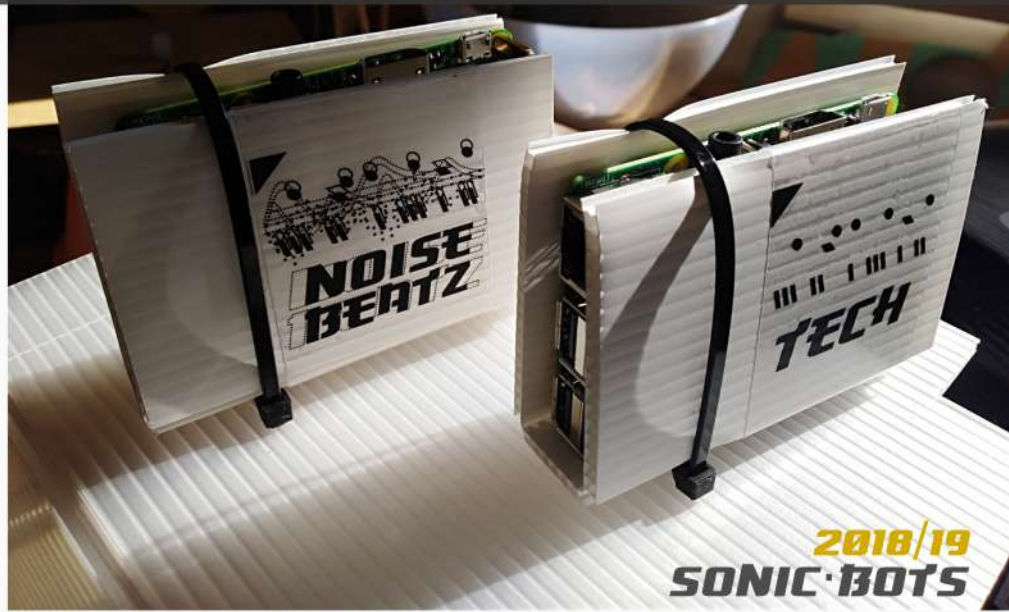




2021..

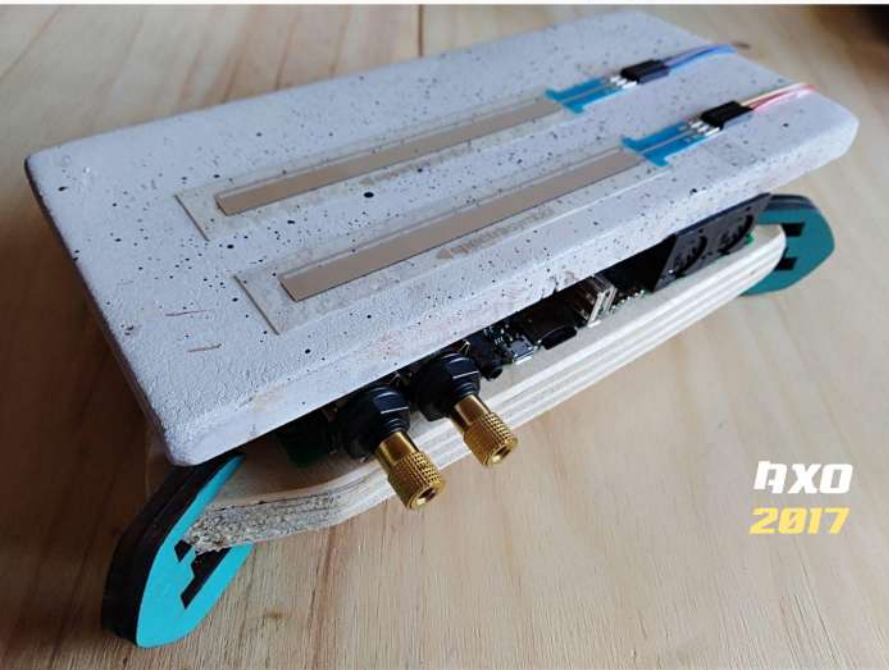
404·GNRTV·RHYTHM·BOX

**PHYSICAL INSTRUMENTS PERFORMANCE-ORIENTED
DRIVEN BY GENERATIVE ALGORITHMS**



2018/19

SONIC·BOYS



**DXO
2017**



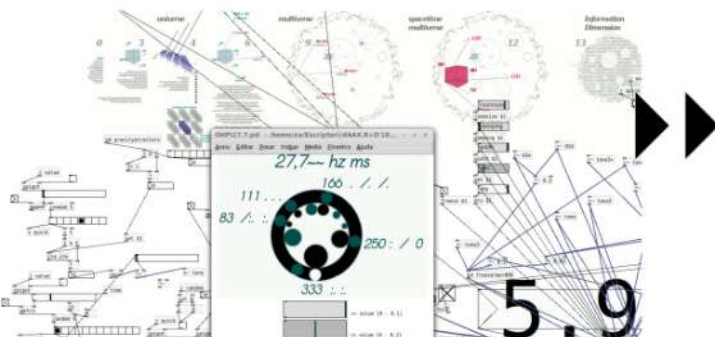
ONES HAPTIQUES / HAPTICAL WAVES

**PHYSICAL INSTRUMENTS (HAPTICAL AND BINAURAL)
DRIVEN BY GENERATIVE ALGORITHMS**



**OBJECT DESIGN + ALGORITHMS + HAPTICAL TECHS AT EIXAMS PROJECT
EIXAMS PROJECT COPRODUCTION PROJECT WITH A. MUÑOZ**

SONIC GENERATIVE
ALGORITHMS

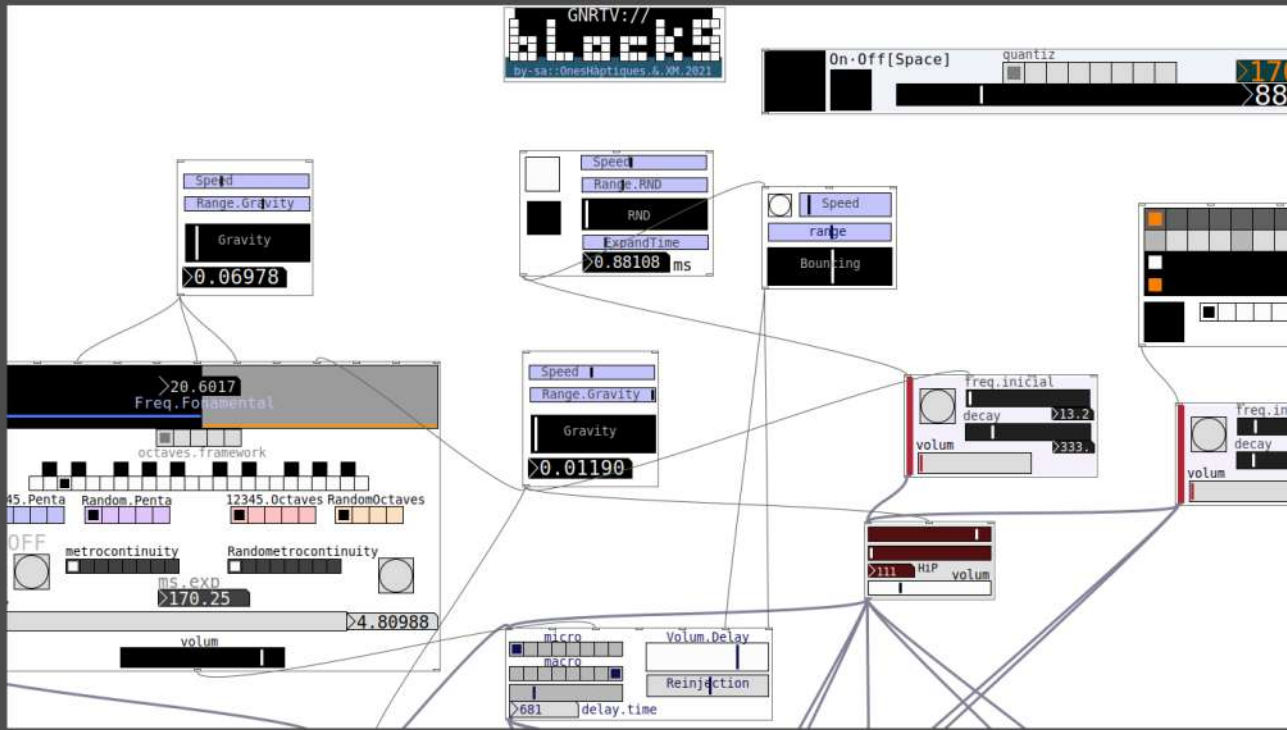


HAPTICAL LISTENING

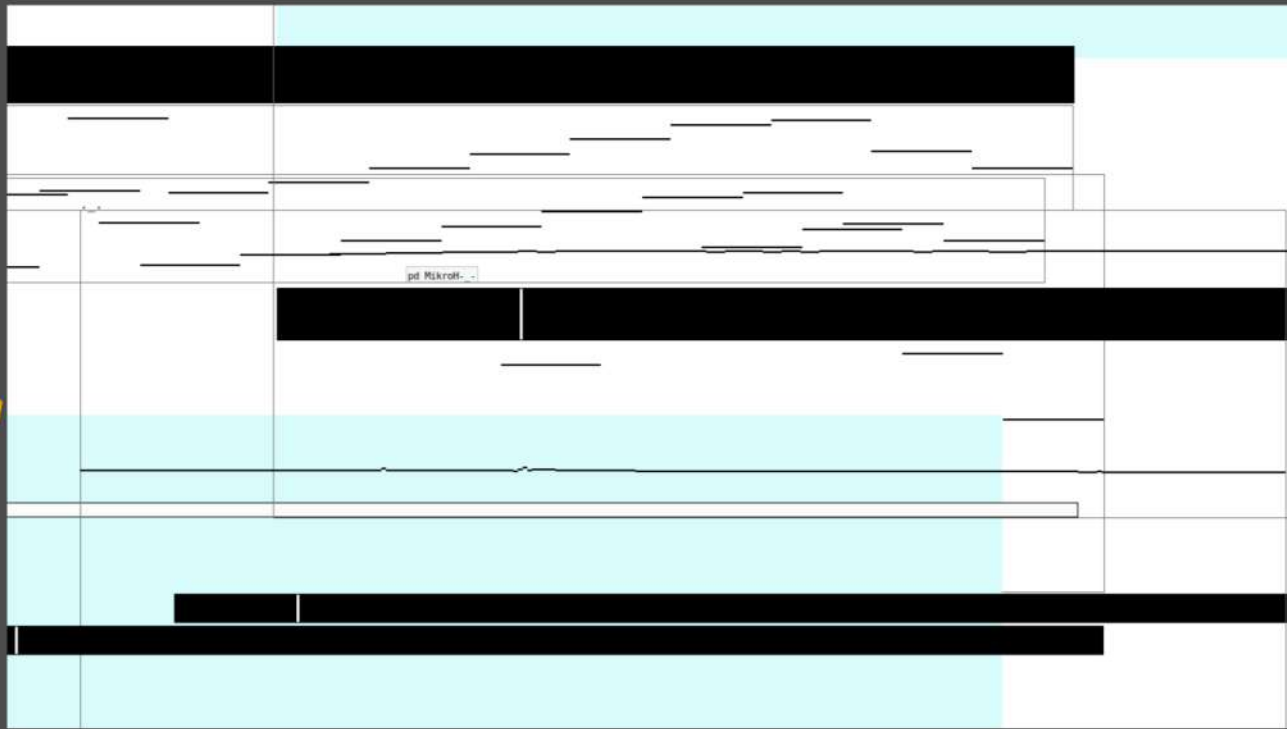


BINAURAL
LISTENING

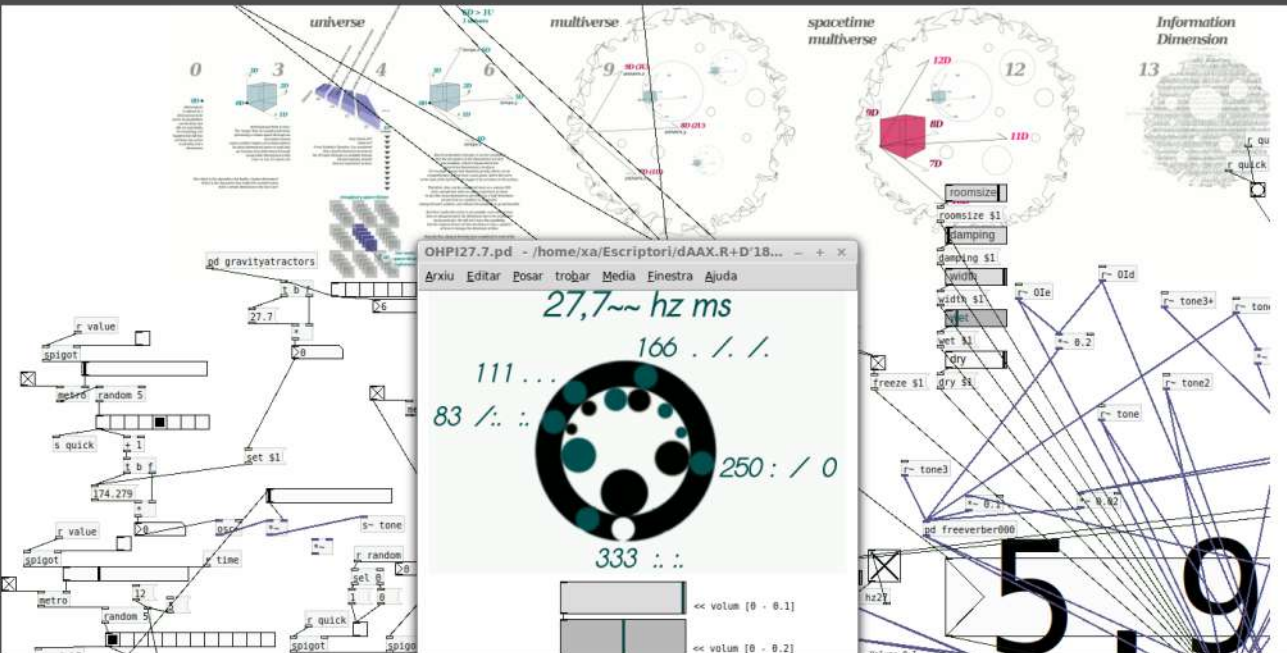
GNRTV.
BLOCKS
 DEVELOPMENT KIT
 FOR SONIC
 INTERACTION
 APPLICATIONS
 2021..



SONIC.BOYS
 2018/19

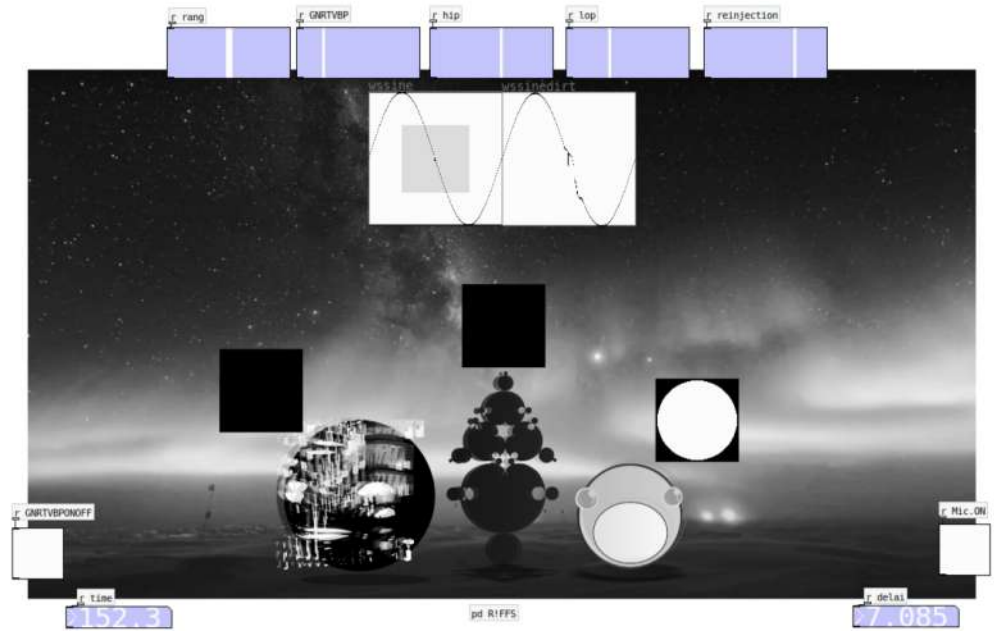


OHPI
 SONIC ALGORITHM
 (ONES HAPTQUES)
 2016



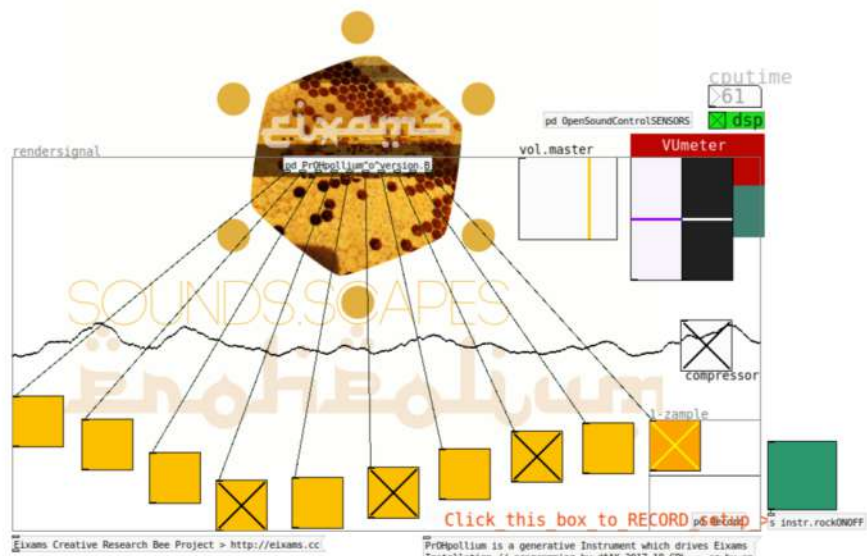
R!FFS!

GENERATIVE
PROGRAMMING
TOOL
2021



PROHPOLIUM

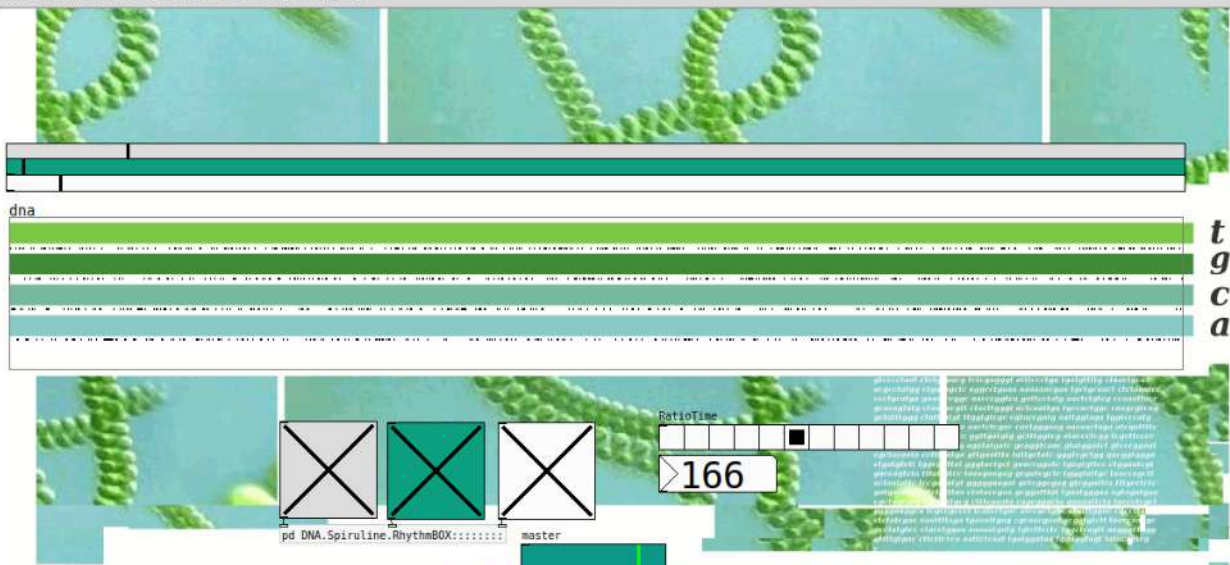
GENERATIVE
PROGRAMMING
FOR EIXAMS
INSTALLATION
2018



Arxiu Editar Posar trobar Media Finestra Ajuda

DNA. SPIRULINE. RHYTHM.BOX

EXPERIMENTAL
APPLICATION
THAT TRANSLATES
A DNA SEQUENCE
INTO A RHYTHM BOX
2015



DNA.Spiruline.RhythmBOX

Arthrospira platensis crtB gene for phytoene synthase, complete cds
<http://www.ncbi.nlm.nih.gov/nuccore/AB001284.1>

Concept + Code + Design by Xavier Manzaneres dAAX 2015
<http://noconventions.mobi/daax>

))) 音 (((
(((音)))





AI vs GeNeRaTiVe Techniques : Different approaches of the Automated music Phenomena Sonic Generative Algorithms : 404·GNRTV·Rhythm·Box

Context

Meanwhile AI / ML / NeuralNets are hugely complex systems that tries to emulate the neurobiology of the human brain in computational simulations, Generative techniques are much more 'simple' compared with the previous one. Generally speaking AI,ML are challenging the very basis of human cognition domain, and Generative Techs. are somekind of augmented and assisted techs based on physics and nature dynamics.

In this sense, it is a very difficult task to build AI techniques that 'creates' from scratch new artistic and musical compositions. Often examples we already saw in this emerging field, are in some point pre-determinate to big amount of previous data analysis / ML.

But music is an amazing field in which its structures and stilistic boundaries are continously evolving, and also often what is 'good music' or 'not' often remains on subjective and cultural levels.

So how can we train a machine about taste, poetry and sensibility of a composition?

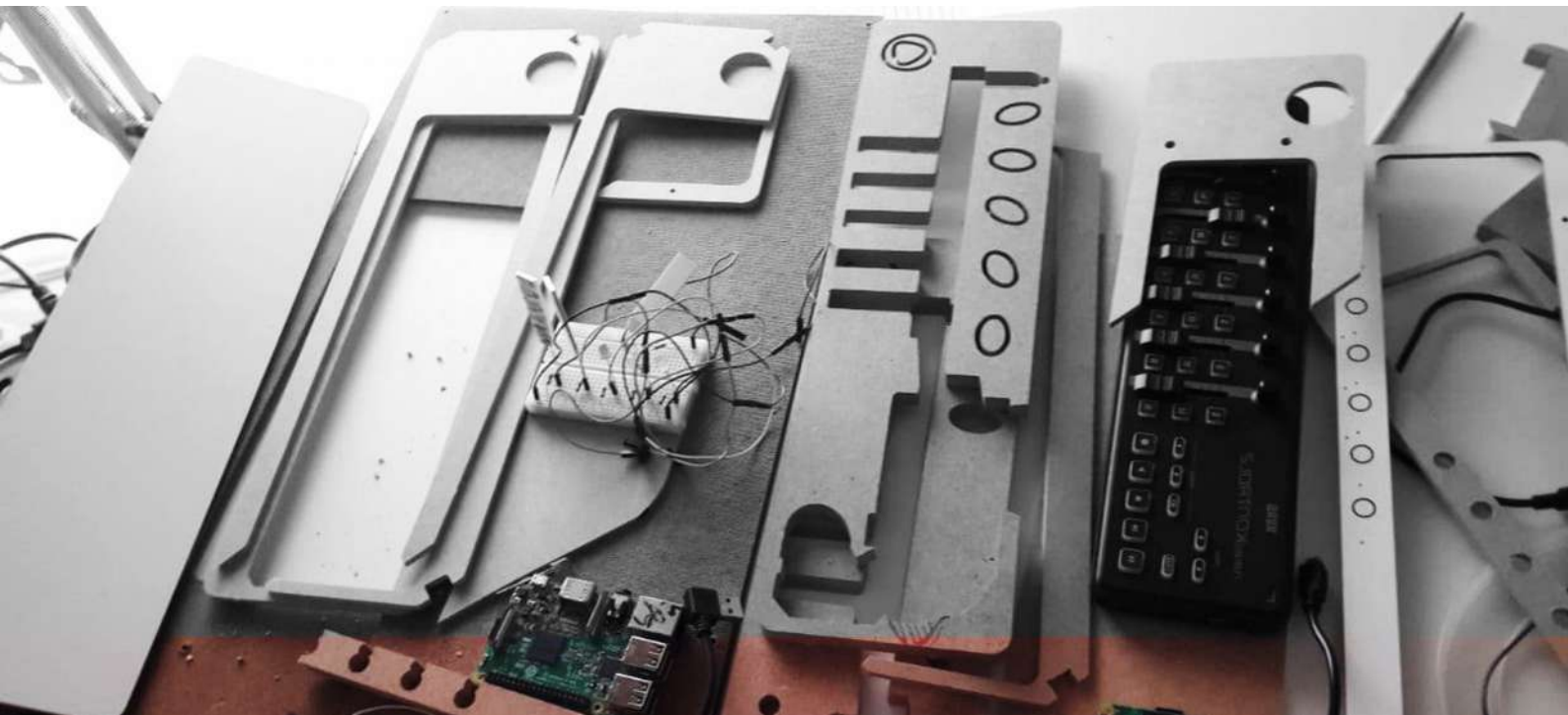
Which are the reference frameworks of these concepts, since even today we still don't know deeply enough how emotions, cognition, intuition and other brain aspects works (even Neurocognitive Scientists have been doing a pretty huge work about it).

In addition the irrational behaviour that often drives humans, both in every-day-life, both in artistic creations, are in collapse in systems of neuron propagation, convolutions, and so on.... which in some sense -even are hugely complex and virtuous systems-, still are restricted to ordered and sequenced methods.

All the previous considerations reminds us that when we talk about AI, often we are refering into a human-intelligence-like simulation, but we forget that other beings makes complex cognitive and behavioural tasks as well.

Therefore in this panel i'm gonna show how robots and automation can be designed from other perspective: with the influence of physics and nature behaviours through Generative Techniques.

Within this paradigm, introducing concepts like physical atractors, gravity, forces, growing structures, motion, etc. and different methods of randomness and probabilistic alorythms, maybe is a nice approach to introduce unexpected behavior and 'irrational' tensions in artistic /creative automated systems, which often we found in human creativity.



Project

Sonic Generative Algorithms : 404·GNRTV·Rhythm·Box

AI vs GeNeRaTiVe Techniques : Different approaches of the Automated Music Phenomena

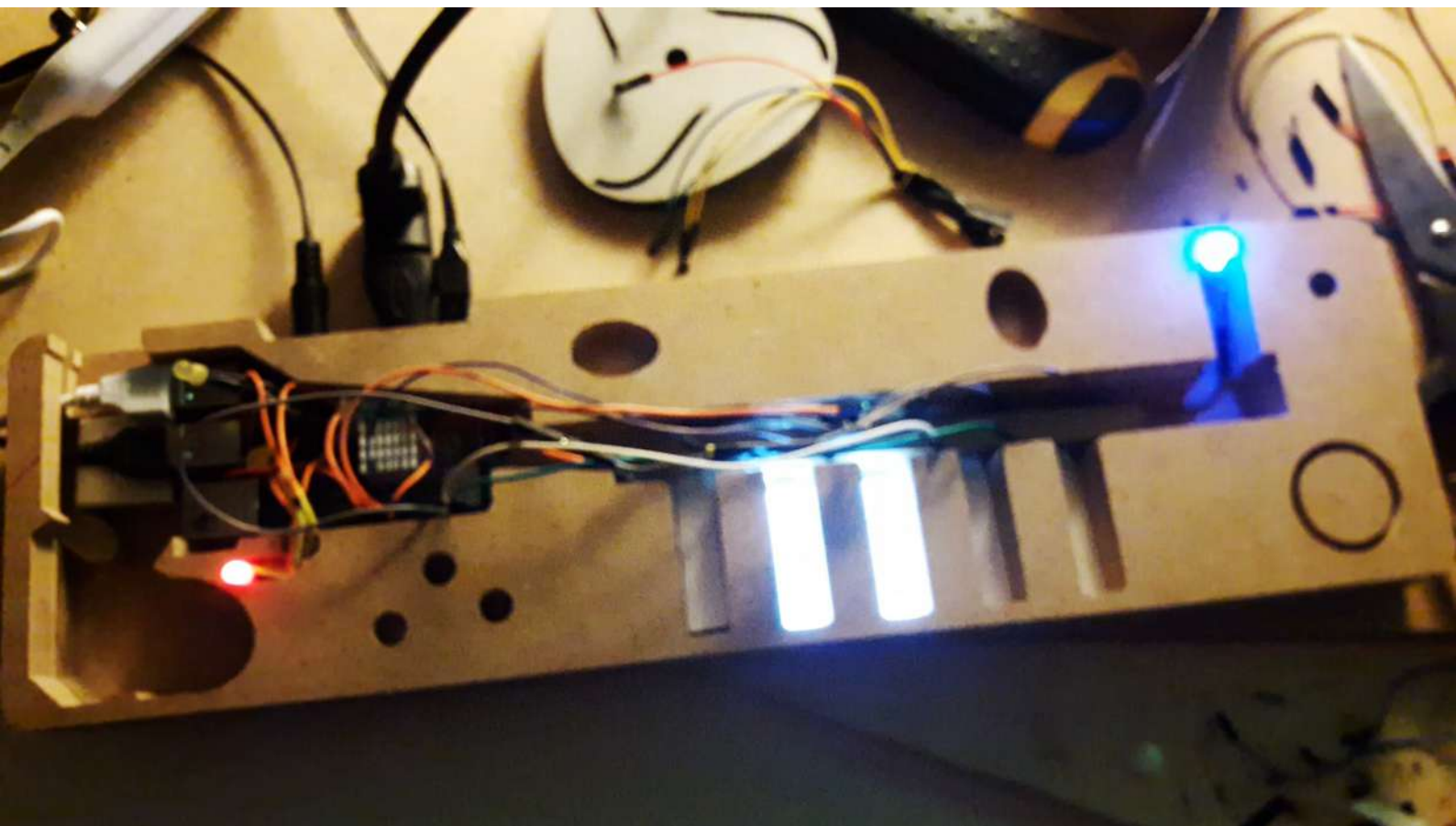
404.GNRTV.RhythmBOX device is an standalone **synthesizer***, in which the whole algorithm is full of generative methods in order to build mutable patterns and growing structures. In short : a kind of Augmented and assisted *Rhythm BOX Synth*.

It is driven by a midi controller which depending on the values that are sended to the 'brain' or sonic engine, it creates many different shapes both for sonic timbers both for musical patterns. In addition, as ambient layers, there are three embed sonic instruments which are totally autonomous (bots) in their sonic and narrative evolutions.

This is an experimental project still working progress. The idea is to implement a complementary *Machine Listening* device in which **404.GNRTV.RhythmBOX**'s sound structures can be analyzed, reinterpreted and rebuild according on the combinations that has been performed within the instrument in time.

This is the most recent project of this typology, but also another examples of sonic generative automats developed in the last years are: **R!FFS** , **SonicBots**, **OHPI**, **Prohpolium**, **SynthPiBots** (see images in next pages) , among other examples that are very useful for *Live Performances*, but specially for *Media Art / Sound Installations*.

**programming : Pd Language + Bash Scripting + WiringPi Lib // OS : RaspPi-OS // Platform : Raspberry Pi 4*



Methodology

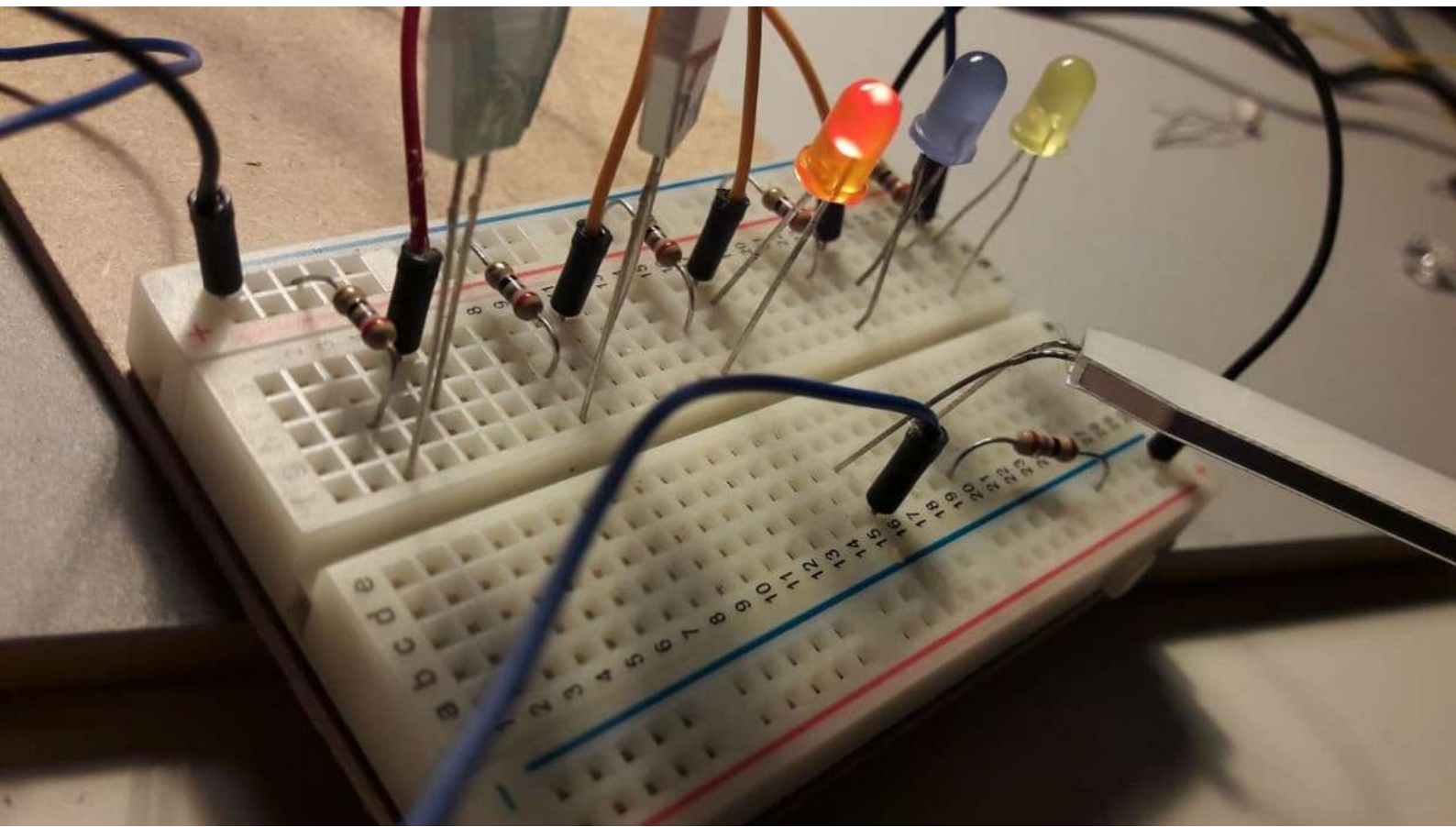
Using several types of generative techniques in a music compositional tool.

In this sense, like already commented, introducing concepts like physical attractors, gravity, forces, growing structures, motion, etc. and different methods of randomness and probabilistic algorithms, and apply them into musical concepts like : patterns, narrative structures, time engines and polyryhtms, types of sound synthesis, evolving parameters of filters, evolving parameters of space and binaurality among many others. In fact the more values you are assigning to generative methods the more fully automated will be the instrument.



Objectives

- _Introducing generative techniques in order to introduce plasticity, organicity and aesthetics in the musical performance.
- _Introducing generative techniques which assists the performer to take advantage of growing networked musical structures in time.
- _Intercommunicate several units and/or with other electronic music machines in order to knit audio synthesized networks.
- _Create a complementary MachineListening device that can listen, reorganize and build different patterns according to the reproduced combinations, that the performer/user has been doing with the device.
- _Create an assisted tool for musicians (and those persons interested in music) ready to create and enjoy.



Project links

<http://xavimanzanares.oneshaptiques.space>

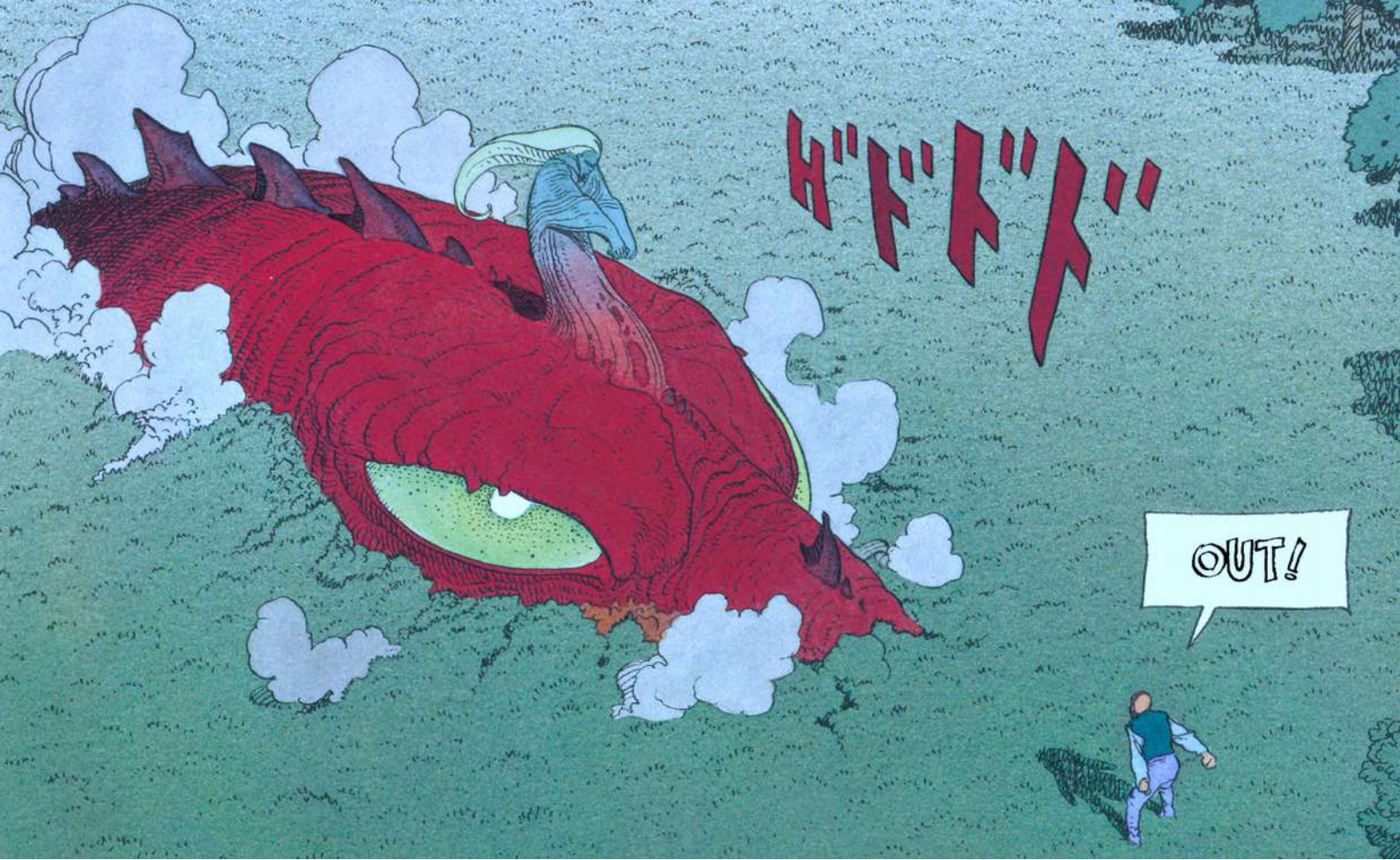
<http://oneshaptiques.space/>

Social media profiles

TW @txa @OH_Lab

Instagram @xamanza

<http://www.linkedin.com/in/xamanzanares>



Basilisc de Roko

img src > el mundo de edena moebius this
image may have copyrights / use only for research
purposes

El **Basilisc de Roko** és un experiment mental que explora els riscos potencials de desenvolupar una IA. L'experiment planteja que en el futur una IA amb accés a recursos quasi il·limitats des d'una perspectiva humana (el basilisc) podés castigar de manera retroactiva a tots aquells que d'alguna manera no varen contribuir a la seva creació.

El basilisc de Roko va ser proposat per primera vegada per la comunitat LessWrong, un forum d'Internet dedicat a temes de filosofia i psicologia amb una visió futurista.

El dilema plantejat pel Basilisc de Roko és una versió de la **Paradoxa de Newcomb***, i explora de manera informal aspectes del **lliure albir**** similars als plantejats pels casos de **Frankfurt*****. Des del seu plantejament original, el Basilisc de Roko ha anat acompanyat de polèmica sobre la seva validesa.

*La paradoxa de Newcomb és l'estudi d'un joc entre dos jugadors, un dels quals pot predir el futur. Es considera una paradoxa perquè porta a una autocontradicció. La causalitat inversa està definida en el problema, per la qual cosa no hi pot haver lliure albir. Alhora, el lliure arbitri està definit en el problema; altrament, el jugador no estaria fent una veritable elecció. Aquesta paradoxa va ser formulada per William Newcomb, del Laboratori "Lawrence Livermore" de la Universitat de Califòrnia. Robert Nozick la va donar a conèixer a la comunitat filosòfica el 1969, i va aparèixer a la columna de Martin Gardner a *Scientific American* el 1974.

**L'albir (de la deformació vulgar del vocable llatí arbitri,¹ al seu torn d'arbiter, 'jutge'²), lliure albir o lliure elecció és la creença d'aquelles doctrines filosòfiques segons les quals les persones tenen el poder de triar i prendre les pròpies decisions. Moltes autoritats religioses han donat suport a aquesta creença,³ mentre que ha estat criticada com una forma d'ideologia individualista per pensadors com ara Baruch Spinoza, Arthur Schopenhauer, Karl Marx i Friedrich Nietzsche.

El principi del lliure arbitri té implicacions religioses, ètiques, psicològiques, jurídiques i científiques. Per exemple, l'ètica pot suposar que els individus són responsables de les accions pròpies. En la psicologia, implica que la ment controla algunes de les accions del cos, les quals en són conscients.

L'existència del lliure arbitri ha estat un tema central al llarg de la història de la filosofia i de la ciència. Es diferencia de la llibertat en el sentit que comporta la potencialitat d'obrar o no obrar.

****Els Casos de Frankfurt (també coneguts com a Contra-exemples de Frankfurt o Casos d'estil Frankfurt) van ser proposats pel filòsof Harry Frankfurt el 1969 com a contra-exemples al principi de possibilitats alternatives (PPA), que sosté que un agent és moralment responsable d'una acció si i només si l'agent podria haver actuat altrament.



basilisc

[def] m Mitologia

Rèptil fabulós al qual hom atribuïa la propietat de matar amb la mirada, representat com un ésser especialment terrible, amb ulls i boca ardents, cos de serp, potes de gall, ales espinoses i cua en forma de llança.

Creença provinent d'Orient, fou molt estesa a l'edat mitjana. Hom en troba descripcions des de Plini el Vell.

src

<https://www.enciclopedia.cat/ec-gec-0083547.xml>

منجھیند منجھنؤ





by XaviManzanares 2021 #copyXYZv560831

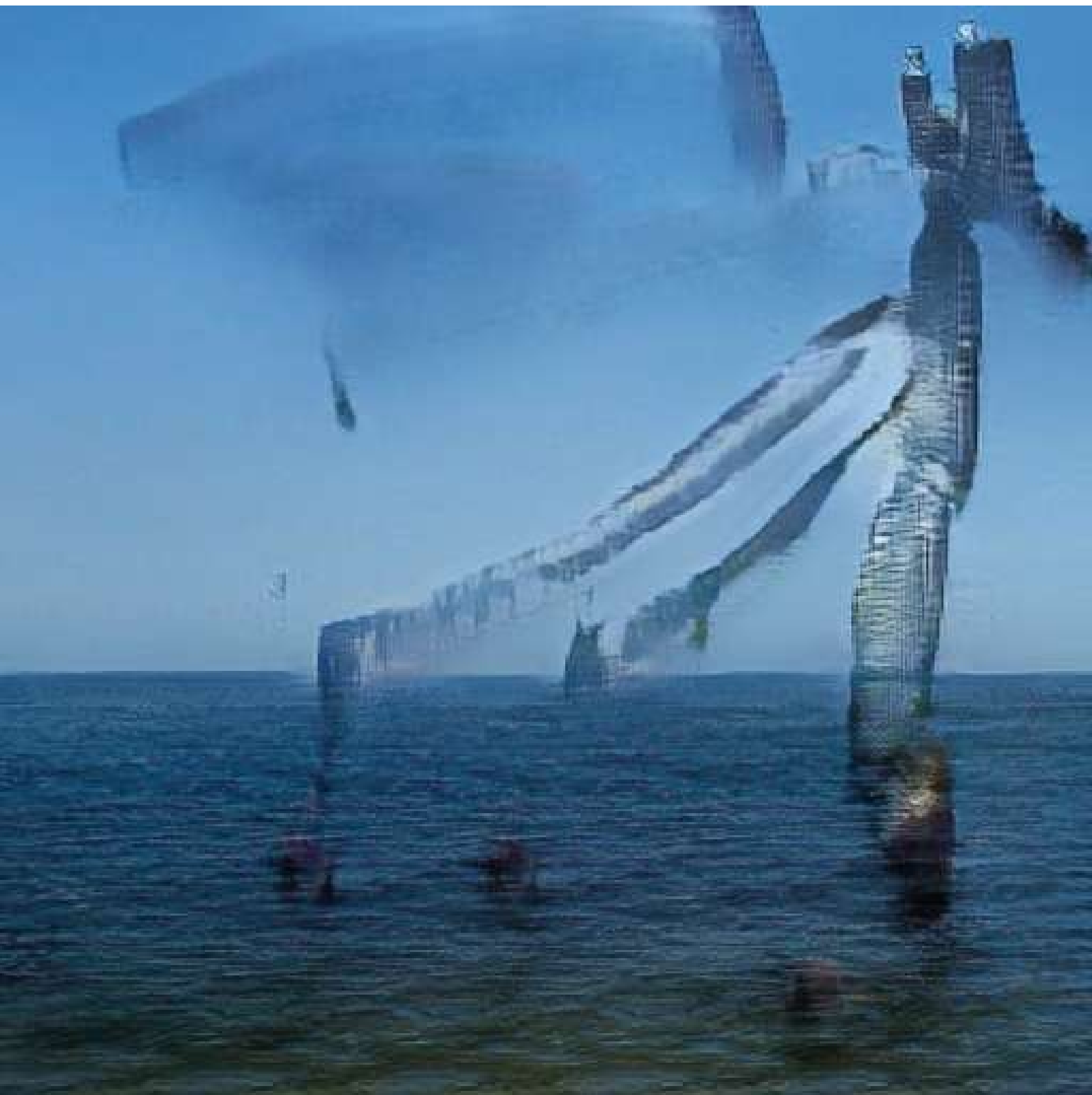




by XaviManzanares 2021 #copyXYZv481956



by XaviManzanares 2021 #copyXYZv556452

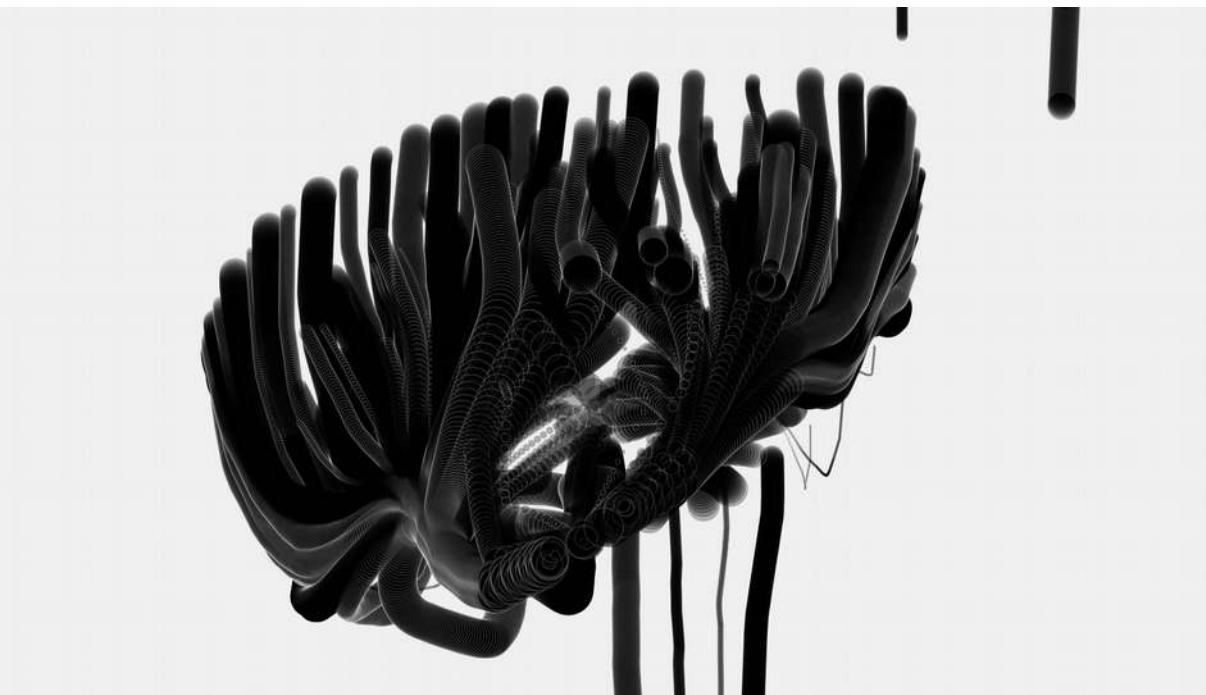


by XaviManzanares 2021 #copyXYZv722818

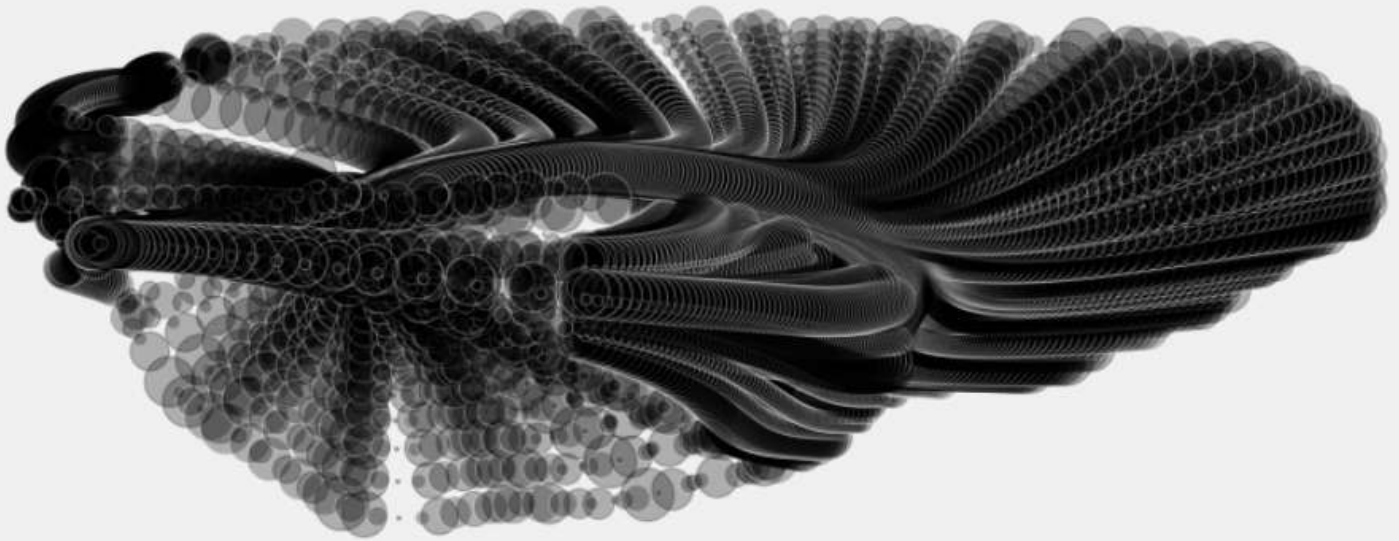








*Sketches from hicetnunc nft portal
Not available currently
src > <http://hicetnunc.xyz/objkt/78207>*







(img src memes from the internet)



RRL
Mono con gotas de pintura

~~690,00 €~~ **483,00 €**

30% DE DESCUENTO

Marino Con Salpicaduras

XS S M L XL XXL

[Tabla de tallas](#)

ARTIFICIAL STUPIDITY

A.S.

La Estupidesa és una característica humana que totes i tots tenim.

La Estupidesa no és un atribut identitari de la subjectivitat d'algunes persones, sinó un estat temporal de manca d'intel·ligència que ens pot succeir en qualsevol moment.

Podem tenir una genialitat i al cap de 5 minuts cometre una estupidesa integral.

Segurament hi ha qui tendeixi a fer-ho més vegades i gent que li passa menys sovint, però això és un altre tema.

La narració antropocèntrica (que encara avui en dia el punt de vista *normal-core* sosté) ens diu que els humans som els sers més evolucionats i intel·ligents.

Sovint trobo aquesta qüestió com una còmica paradoxa : si de debò la humanitat fos la més evolucionada i intel·ligent no estaríem on estem com a civilització.

Per a matisar les anteriors línies podem fer referència a complexitat. Segurament sí que es cert que des d'una perspectiva orgànica, la cognició humana té una alta complexitat respecte altres organismes, però doiria que fora interessant dissociar la relació entre complexitat i atribució de superioritat.

A vegades penso que un invent random que alguns humans han pensat i dissenyat (imaginem-nos un patinet elèctric) el 99% de la resta que l'utilitza no en té ni la més remota idea de com i perquè aquell giny funciona. També podríem dir el mateix sobre una teoria filosòfica escrita i desenvolupada per algun/a intel·lectual.

I mil coses més.

És a dir, la intel·ligència humana amb els seus descobriments generats es pot considerar col·lectiva, però si anem a analitzar la intel·ligència particular individual, veurem com hi ha sorprenentment un alt nivell d'estupidesa generalitzada.

I què passa amb la intel·ligència en la resta d'éssers vius?

El novelista gràfic Miguel Brieva assenyalava sovint a les seves vinyetes el contrast intel·lectual entre els humans i els animals. Mentre els humans mostraven insistentment una estupidesa integral, els animals tenien reflexions filosòfiques ben virtuoses. Evidentment s'havia de llegir com una sàtira *antiespecista* i *mediambientalista* però fins a quin punt això és així?

Els animals es comuniquen entre ells, sigui amb sons, infrasons, ultrasons, feromones, sònar, etc.

Habitualment la manca de comprensió d'aquests codis ha estat malinterpretat amb un biaix qualitatiu i de judici inconscient (reduïnt-ho molt : confondre la intel·ligència d'un ésser viu amb la seva capacitat d'entendre unes paraules humanes) .

Artificial Stupidity

AI-Absurdity

Una Intel·ligència Artificial on les decisions generades no porten a cap solució concreta

AI-Fail

Una Intel·ligència Artificial on les decisions generades són les menys encertades

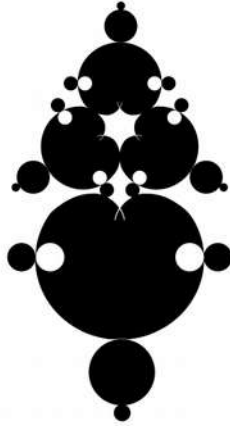
AI-Serendipity

Una Intel·ligència Artificial on les decisions generades són les menys esperades i imprevistes

AI-Joker

Una Intel·ligència Artificial on les decisions generades són les més gracioses





Is an AI genderless?

Imagine that we need to train a *machine learning model* among two types of images (*inputs*) that corresponds to archetypes of female / male. We have to assume that if is some kind of self-aware/sensitive/counscious system, (currently we still are not in this scenario), a large structures of *trained* data contains all genders at the same time with different weights or ratios.

One possibility is *classify* or *regress* the inputs and get an output that matches with one option or other.

But another possibility is blending these two inputs if we use neural nets *methods* that extracts a crossover between them (in order to get non-discrete and thus more transitioned results). In other words, from this binarian example, we get some kind of trans masculine-feminine, or non binary output/s.

What does this mean? This could suggest that humans are in fact already compose with different *weights* or proportions among *categories* of as we know as **gender**. For example, any person has different proportions of male and female characteristics (not only cultural / behaviour but metabolic / organic). Depending on the ratios or *weights* of those proportions of male and female features, we get millions of combinations, from the most polarized ones in the extremes, 'til a wide number of crossovered spectrum in between. Notice that according with this idea, sexual orientation is an augmented and independent layer of the previous context. Therefore anyone can be attracted to others independently of their 'gender' *weights*. But now we can obviate these layer.

In the physical world, any of those combinations deserves the same cure and rights as conscious beings, but unfortunately this collapses within a Patriarchal and Darwinist *model* like the dominant systems sustains (not only capitalism), which pretends to *simulate* and reduce just a *binary classification model* instead a much more plastic and *deep learned* one with wide ranges.

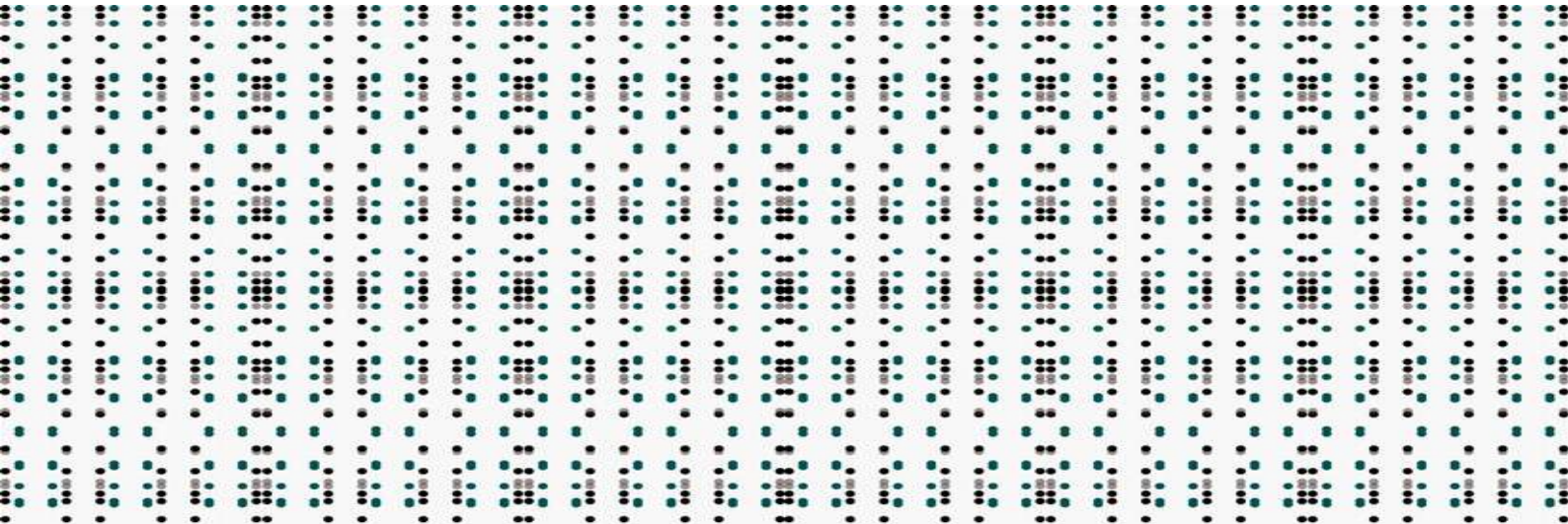
This idea is controversial for some people because they are been *trained* from early ages at the school about the gender and identity construction according to what systemical structures and most people often classify as *normal* : heterosexual and binary mindset, which structures behaviours of individuals and societies.

But what is 'normal'? Many conservative mindsets claims that binary gender division is related to external physiological fenotypes that we find in nature, therefore is the most 'natural'.

But in fact, this is a *mistaken* assumption : if we look accurately how sexuality and gender roles works in nature, we will realize that nature itself performs a wide range of non-binary crossover examples : some organisms features (what often culturally is classified as) male roles and after switches to (what often culturally is classified as) female roles. In other organisms like hipocampus male gets pregnant with their breeds inside. Other organisms are hermaphrodites like snails, etc.

Since sexuality in organisms are often related to expand further generations and extend their group in the future, the question is much more complex. Some organisms self replicates, therefore there are no classification of male-female for their reproduction. Another organisms makes gender transitions depending on their life span stage. Some others organisms like many plants needs the help of other pollinizer organisms (like bees) in order to make a transpecies form of reproduction. *In other words, sexuality and self perpetuation has many types of classifications which overcomes a binarian structure. How an AI could perpetuate itself? Just clone itself i guess. Like many bacterias or unicellular organisms does.*

So why some people still insists in the binarian structure of nature even we have many of examples that demonstrates a much more complex and different structure? Why still this mindset if we humans have the ability to dissociate sexuality and reproduction?



To answer the question '**Is an AI genderless?**' we have first to understand that **AIs** still are not self conscious and sensitive entities, therefore maybe we cannot assume that are intelligent beings, in the sense of complexe intelligence : *cognitive, sensitive, emotional, propioceptive, intuitive, abstract*, etc.

When we say *intelligence* we have to separate as well the property that is often associated only with the brain. *Consciousness* is contained in any *lifeform*, with different layers of complexity. From a single cell which exchanges biochemical information between others in orther to build more complexe functions, structures of functions, organs, and whole organisms. Also consciousness is a symbiotic process of information through electrical (bio)signals. We'll talk about Margulist Symbiotic theory in other pages.

But what is *consciousness*? *Maybe a sensitive and solving-problem being performing the most tuned and optimized method to do a task after many iterations of trial-error-loop with/in its surround, with the goal to sustain itself and improve their existence in its context.*

Even *consciousness* is often related to Philosophical domain, is in fact a very common trend in Sciences as well, since the *Quantum mechanics* theory and after others, suggests the relationship of *observation* (therefore some type of counscious being to perform the observation) and the very *basis of nature and physical laws* (at least in subatomic scales). *Schrödinger's cat, Quantum Oscillation Wavefunctions* and so forth.

Secondly, according that **AIs** are bodyless like many algorithmtms used on the internet, or even can be *embodied* in several kinds of shapes like robots, the question of gender is maybe a confusing issue.

An AI can be Female? Male? Both? None? A Multiplicity of genders?

Are we antropomorphizing AIs too much instead of consider them just as another kind of counsciousness? I think so.



BRAIN BEATS

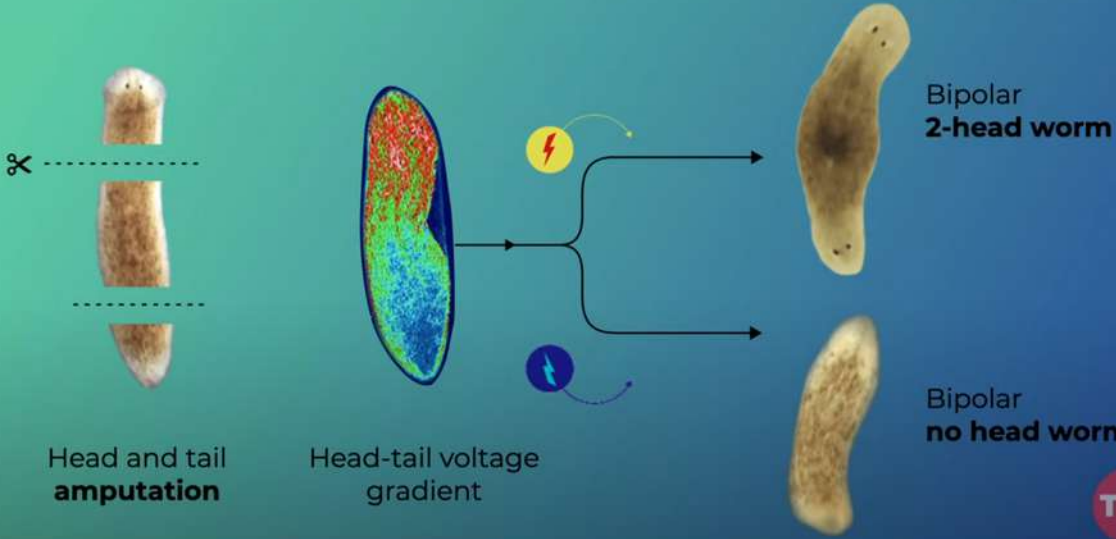
Eine Reise in die Zukunft des Hörens



HUNTING FOR HEDONIA

Michael Levin: The electrical blueprints that orchestrate life | TED

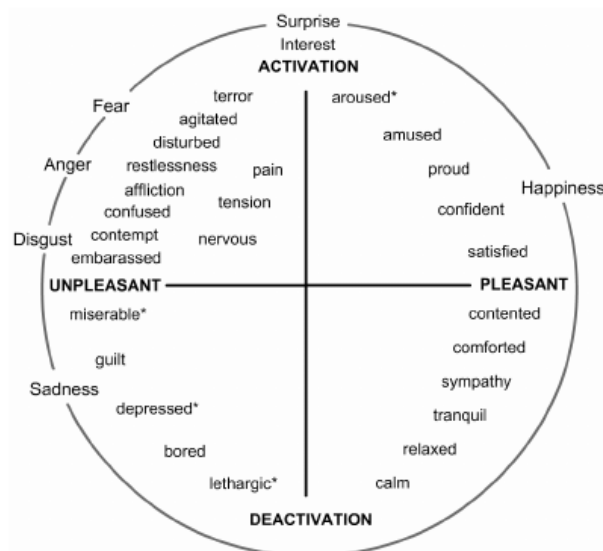
Radically **change the body anatomy**



Per a entendre la IA hem de primer entendre com funciona la Intel·ligència Humana i per extensió com funciona la Intel·ligència als éssers vius, des de la seva organització social fins a la seva organització bioquímica : com funciona l'intercanvi d'ions en les membranes cel·lulars, com es codifiquen unes proteïnes o altres etc..

A dia d'avui, dins els marcs de les ciències cognitives no s'han descobert temes rellevants més enllà dels definits pel **model circumflex** dels anys 80. (Russel). El model circumflex és un mapping-diagrama de les emocions i combinació d'emocions ('moods'), en relació a determinades àrees d'activitat cerebral detectades per nombroses medicions.

<https://www.researchgate.net/profile/Nelson-Zagalo/publication/221594596/figure/fig1/AS:305496490823684@1449847447790/Circumplex-Model-of-Emotions-15-This-model-contains-already-all-the-emotions-from-our.png>



Sembla increïble que hagin determinat unes àrees d'estimulació a una zona del cervell correspon directament amb una sèrie d'emocions.

El projecte **Brain Poliphony** que van realitzar fa uns anys els recercadors **Mobility Lab** (Efrain Foglia i Jordi Sala) , conjuntament amb el grup de Recerca Neuro Cognitiva de **Lara Dierssen** (CRG), és un gran exemple pràctic experimental.

En aquest projecte l'objectiu era poder connectar les emocions i 'moods' de persones amb paràlisi cerebral i per tant amb diferents graus de mobilitat motora i que aquestes poguessin fer música.

<https://www.youtube.com/watch?v=6eKLlz4-yVM>

https://www.youtube.com/watch?v=Hns_iwEMbdw

Aquest article inicia amb la captura de 3 documents que s'entrelliguen per a poder entendre millor la AI, atès que tracten del lligam emocional-cognitiu del nostre cervell.

BRAIN BEATS

(A musical species)
ESP translation
<https://youtu.be/yi71aS667Es>

El primer és **Brain Beats** un documental extraordinari sobre la relació entre tecnologies musicals, sonologia i les emocions, i sobre com l'estudi d'aquestes poden fer evolucionar no només la musicologia i les tecnologies musicals, sinó també la creació d'altres especialitats emergents.

Hunting for Hedonia

primevideo.com
<https://youtu.be/NS7IjdqGPOY>

El segon és **Hunting for Hedonia**

un documental on enfoca la recerca realitzada pel Dr. Heath el qual tot i realitzar avenços rellevants en psiquiatria dins la seva època (anys 50) que enllacen amb estudis neurocientífics actuals, els seus mètodes i aplicacions traspasaven nombroses línies vermelles des de la perspectiva ètica (capacitisme, homofòbia,...) que van acabar sepultant la seva carrera.

La recerca estudiava la estimulació amb petits gradients de nano-electricitat tipus 5mAh en determinades àrees cerebrals, generant canvis molt significatius en pacients amb diverses patologies de salut mental (demència, esquizofrènia, depressió, ansietat etc).

Biologia sintètica i enginyeria per generar sistemes vius

<https://www.biennialciutatciencia.barcelona.ca/programa/biologia-sintetica-i-enginyeria-generar-sistemes-vius#directe>
https://youtu.be/jntk_IWXHA
Fragment de Michael Levin
22.30 to 34.15

El tercer no és un documental en sí sinó una sèrie de càpsules vídeo del treball del grup d'investigació de Michael Levin. Cal dir que quan vaig veure aquesta presentació on-line dins el marc de la Biennal Ciutat i Ciència 2021, em va explotar bastant el cervell.

Levin ens explica la relació entre la informació i codificació nano-bioelèctrica* i la morfogènesi orgànica (creixement, reparació de teixits, etc).

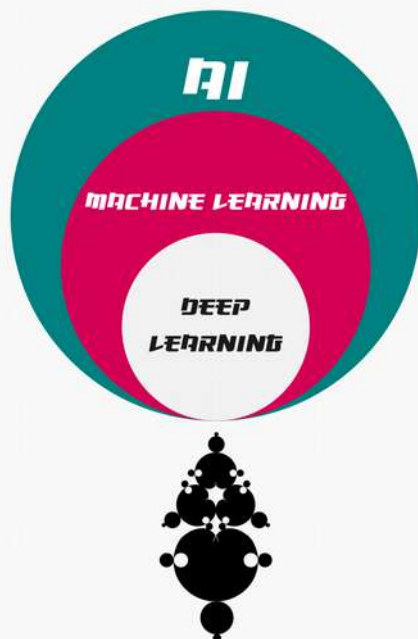
*sabíem que entre neurones s'intercanvien micro-pulsacions elèctriques però aquests en realitat es produeixen a qualsevol cèl·lula.

És a dir, ens mostra en com un llenguatge nano-elèctric té una implicació directa en la forma de desenvolupament d'organismes i teixits. (sense edicions ni modificacions d'ADN en absolut).

Això pot obrir les portes de nombrosos descobriments aplicats a la medicina, però especialment pot suposar una fusió transdisciplinària entre les ciències de la computació i la biologia més enllà de les seves relacions conceptuals i filosòfiques.

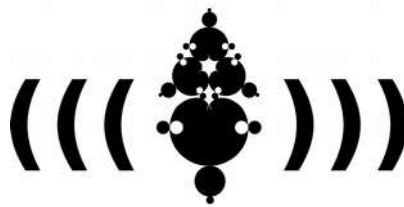
La recerca de Levin sens dubte una de les més interessants i trencadores en la biologia actual, no està exempta de controvèrsia ètica pel que fa a la frankensteinització d'alguns mètodes utilitzats.





INTRODUCCIÓ
AL
MACHINE LEARNING

ML



Beques de recerca i creació OSIC· gencat 2021

projecte **músIA**

Xavi Manzanares

2021 / 2022

MACHINE LEARNING FOR NEWBIES VOL I INTRO

Aknowledgments:

Molts diagrames i representacions d'aquest tuto són directament inspirats del molt recomanable curs de
Rebecca Fiebrink 'Machine Learning for Musicians and Artists'

<https://www.kadenze.com/courses/machine-learning-for-musicians-and-artists-v>

on es mostra una overview al ML amb projectes de media & sonic interaction per mitjà del programari ML
Wekinator.

En aquest tutorial veurem les diferents tipologies i particularitats del que entenem com a **Machine Learning**.

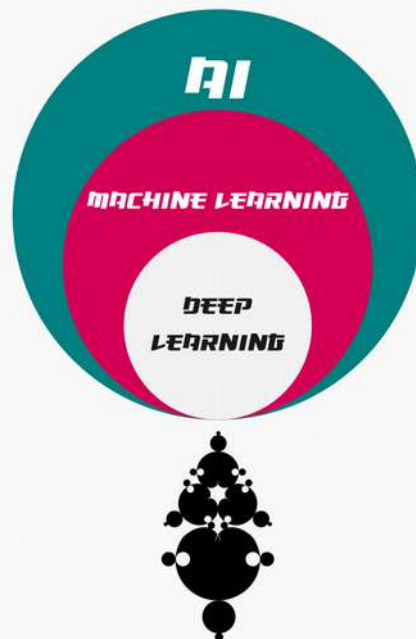
Abans de començar, un parell de qüestions.

La primera i per a què disposem d'un enfoc amb precisió i comunicació pedagògica, com a **Machine Learning (ML)** entendrem com una sèrie de mètodes i tècniques de programació amb les quals una determinada aplicació o conjunt d'aplicacions pugui realitzar un procés iterat d'entrenament ('training') i aprenentatge 'learning'.

El '**Deep Learning**' (DL) correspon a un tipus de Machine Learning més evolucionat i (sovint) complex en el què s'hi recreen models de **xarxes neuronals (ANN o Artificial Neural Networks)**. Cal recordar que les xarxes neuronals no són de cap manera neurones sintètiques dissenyades a laboratoris bioengineering sinó que són un conjunt d'aplicacions modulars inspirades en la propagació i retropropagació que es produeix a les neurones humanes.

Per tant, i per a resumir les anteriors frases són com unes nines russes.

La IA és el marc principal, el Machine Learning és una part de la IA i el DeepLearning és una part del Machine Learning.



La pregunta habitual, seria la confusió entre ML i AI.

Però podriem entendre que hi han sistemes AI on no es necessitin mètodes d'aprenentatge.

És aquí on entra una qüestió filosòfica en entendre què és intel·ligència.

//...

we have first to understand that AIs still are not self conscious and sensitive entities, therefore maybe we cannot assume that are intelligent beings, in the sense of complexe intelligence : cognitive, sensitive, emotional, propioceptive, intuitive, abstract, creative etc.

When we say intelligence we have to separate as well the property that is often associated only with the brain.

Consciousness is contained in any lifeform, with different layers of complexity. From a single cell which exchanges biochemical information between others in orther to build more complexe functions, structures of functions, organs, and whole organisms.

Also consciousness is a symbiotic process of information through electrical (bio)signals. We'll talk about Margulist Symbiotic theory in other pages.

But what is consciousness? Maybe a sensitive and solving-problem being performing the most tuned and optimized method to do a task after many iterations of trial-error-loop with/in its surround, with the goal to sustain itself and improve their existence in its context.

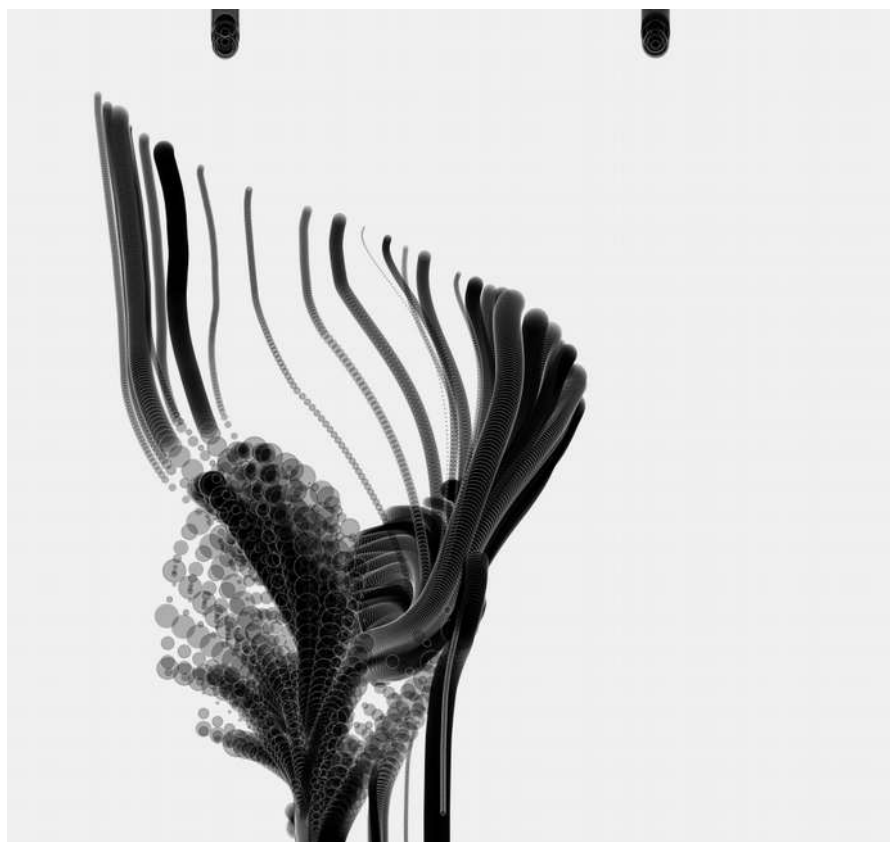
Even consciousness is often related to Philosophical domain, is in fact a very common trend in Sciences as well, since the Quantum mechanics theory and after others, suggests the relationship of observation (therefore some type of conscious being to perform the observation) and the very basis of nature and physical laws (at least in subatomic scales). Schrödinger's cat, Quantum Oscillation Wavefunctions and so forth.

[xavi manzanares, from the article Is An AI Genderless? 2021]

... //

En el següent gràfic podem veure 3 tipologies d'intel·ligència artificial > els sistemes **Generatius** del qual trobo personalment un context d'estudi fascinant atès que l'aproximació a la intel·ligència va més enllà de la antropocèntrica, així com la IA 'mono-cognitiva' i la IA Dinàmica amb Machine Learning.





GENERATIVA

No ens centrarem ara en els mètodes generatius ja que aquest tutorial s'enfoca més al *ML*, però com a recordatori, els mètodes generatius s'inspiren en comportaments que podem trobar a la natura en diverses escales : partícules, fluids, fluxes biològics, swarming, fluxes geològics, atmosfèrics, gravetat, etc

En la present recerca es publicarà el 'toolkit' *GNRTV.BLOCKS* orientat per a construir de manera senzilla instruments algorítmics generatius sonors. Aquesta llibreria com a spòiler inclourà mètodes generatius i algunes aproximacions experimentals basades en ML.

GENERATIVA

NATURA
CREIXEMENT
FÍSICA
GEOLOGIA
COSMOS

IA

CERVELL
COGNICIÓ HUMANA

**SELECCIÓ BASADA AMB
CÀLCULS PROBABILÍSTICS,
RANDÒMICS, O ESTRUCTURALS**

**SELECCIÓ BASADA AMB
MODELS D'ENTRENAMENT
AMB DATABASES QUE PUGUIN
FER COMPARATIVA DE PATRONS**

https://generativelandscape.files.wordpress.com/2014/10/f5_variant_structure.png?w=640

https://miro.medium.com/max/600/1*Wrgmo2FioJDev2uofraw.png

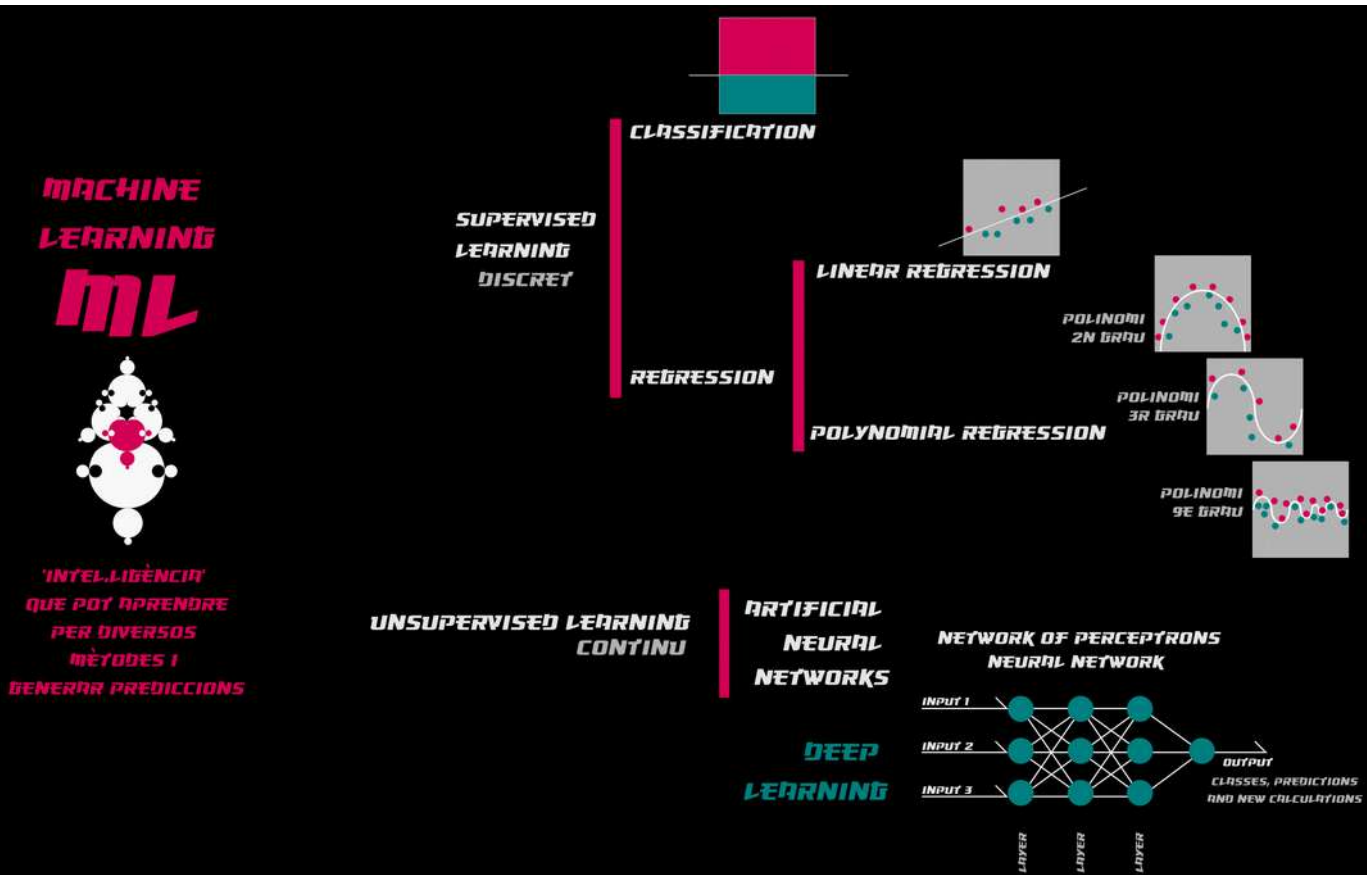
MACHINE LEARNING ML



'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREVICIONS

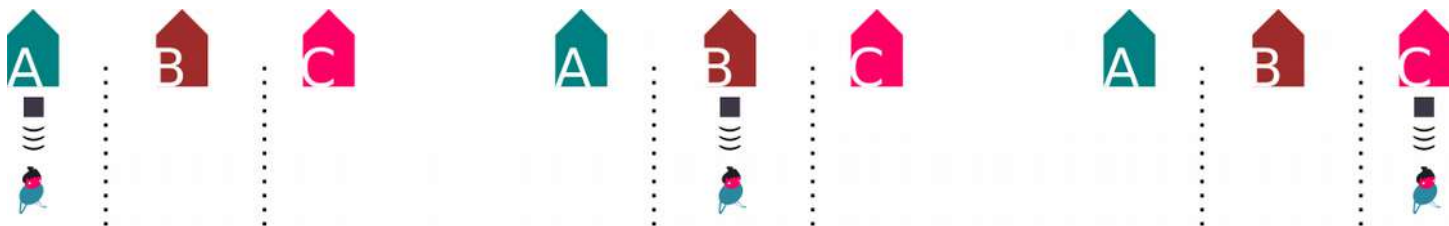
Dins el Machine Learning podrem trobar :
aprenentatge supervisat, [supervised]
aprenentatge no-supervisat, [un-supervised]
aprenentatge reforçat (amb recompensa) [reinforced]
 ...entre d'altres.

Però anem a enfocar-nos en els dos primers.
Tècnicament el 'Supervised' funcionaria d'una manera *discretitzada*.
I el 'Un-Supervised' funcionaria d'una manera *Contínua*

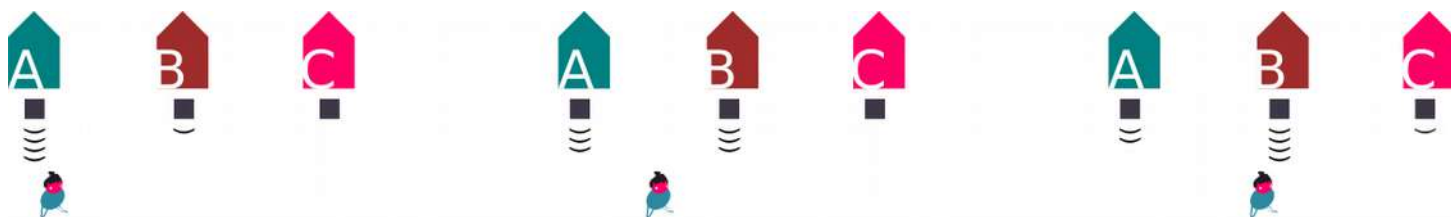


*Per a que quedi clar aquest concepte matemàtic,
Imaginem que estem en un passatge molt petit on hi ha tres cases,
-cadascuna d'elles dotada d'un altaveu a la seva entrada-
i volem que es produeixi un so cada cop que passem davant d'una casa en concret.*

*En un sistema **discretitzat** els sons seran únics i inequívocs.
O sonarà el so A, o el B o el C quan passem per davant de la casa*



*En un sistema **continu** els sons es fondrien entre els adjacents.
(A fusionat amb el B, el B fusionat amb l'A i el C i el C fusionat amb el B)*



Tecnològicament dins el context del ML les tècniques Supervised o Discretes són sovint més senzilles, de manera que depenent del problema a resoldre anirem amb un o altre mètode.

No hi ha millors mètodes que altres sinó que sempre són contextuals :

**Dependran sempre del propòsit a resoldre,
però el minimalisme i la senzillesa sempre ens ajudaran amb més velocitat de computació.**

Abans d'entrar amb les tipologies convé explicar un altre concepte transversal del Machine Learning.

El Fitting.

Un model serà **Underfit** si conté massa imprecisions

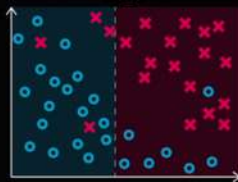
Un model serà **Overfit** si és massa precís i per tant no pot incloure futurs elements d'aprenentatge dissonants.

En definitiva el més adequat és que tendeixi a l'Overfit però deixant marges d'error tant per sorolls i inconsistències en futurs valors d'entrada per a nous entrenaments. (veure model B a la imatge)

IMAGINE WE WANT TO TRAIN A MODEL TO CLASSIFY TWO CLASSES X AND O

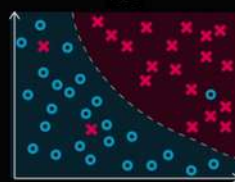
UNDERFIT

A



LESS THAN 100% ACCURACY

B



C

OVERFIT



WHICH IS THE BEST OPTION?

DEPENDS ON THE CONTEXT BUT USUALLY A MODEL LIKE B (CLOSE TO OVERFIT BUT NOT PERFECT) IN ORDER TO INTRODUCE FUTURE UNEXPECTED CASES AND ALSO INTRODUCE ERRORS IN THE MODEL

FOLLOW OCCAM'S RAZOR STRATEGY :

DONT MAKE A COMPLICATED CLASSIFIER UNLESS THERE IS EVIDENCE IT WILL BE BETTER THAN A SIMPLE ONE

MACHINE
LEARNING
ML



'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICIONS

Supervised Learning

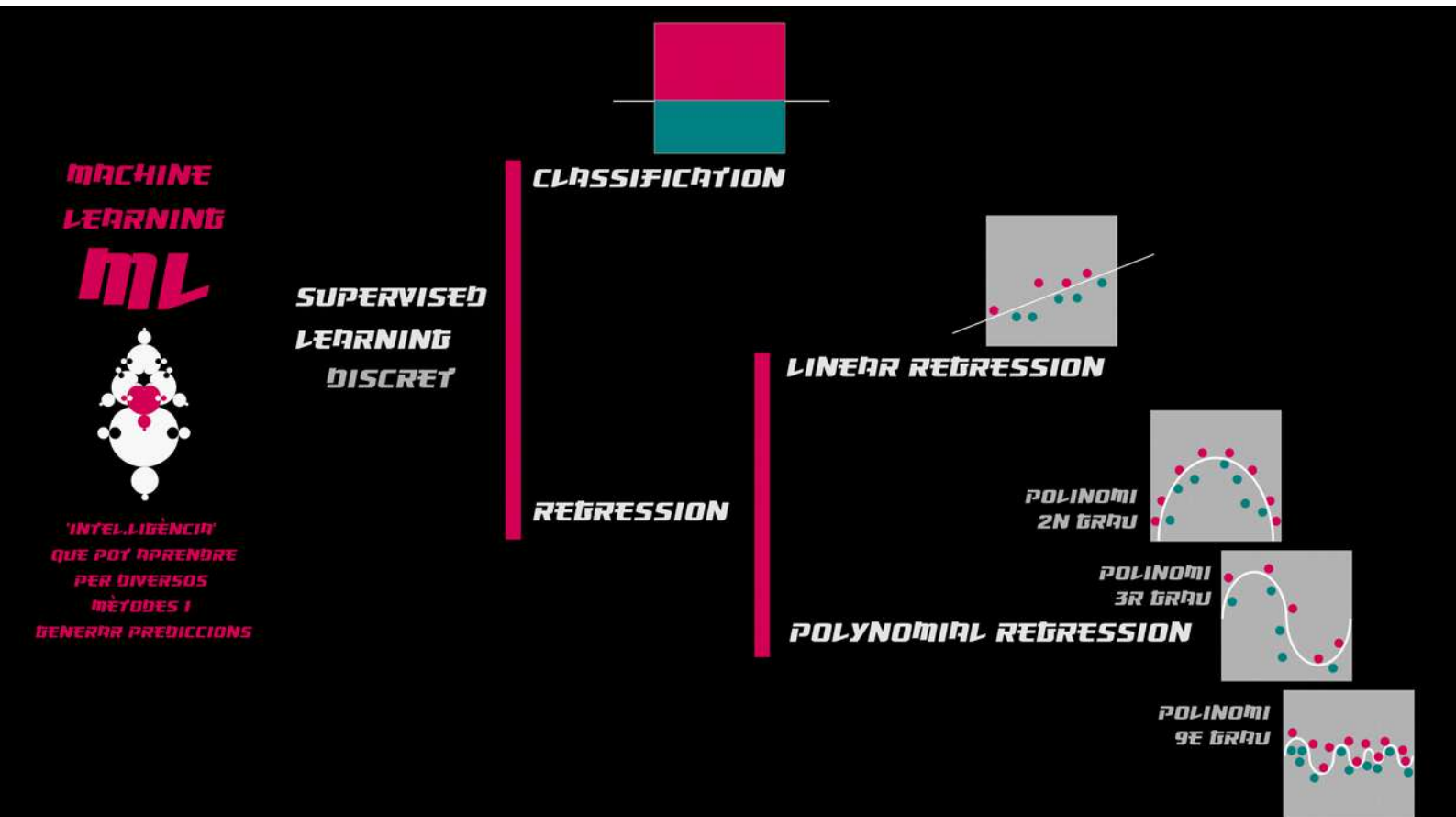
En l'aprenentatge **supervisat** trobem les tècniques de **classificació** i les de **Regressió**

Classificació

En la primera i simplificant molt la idea es tractaria de poder determinar entre 2 classes una sèrie de prediccions (outputs) en base a uns inputs inicials i a un entrenament.

Nota: en les figures apareixen dues classes diferents si bé aquestes poden ser d'ordres elevats.

Imaginem que tenim una piscina quadrada que la veiem zenitalment a vol d'ocell. Si tracem una línia horitzontal imaginària ens generarà un límit el qual podrem classificar quantes persones estan nedant a la part superior (en fucsia) i quantes a la part inferior (turquesa)



Regressió

Aquest mètode genera com l'anterior una classificació entre dues classes o outputs, si bé en un procés iteratiu de 'nombroses passades'.

A més passades, més precisió però com apuntàvem abans, pot haver-hi imprecisions i soroll també quan hi ha una càrrega de dades d'entrada (inputs) molt elevada.

Com a tipologies tindrem la **Regressió Lineal** i la **Regressió Polinòmica (de 2^a, 3^a, 4^a, 5^e Grau etc...)**.

Si bé noteu com matemàticament en realitat la lineal és polinòmica de 1 grau (línia amb pendent)

Segurament a moltes i molts de vosaltres això dels polinomis us sona de la Secundària o altres fases educatives, si bé com a recordatori es tracta de 'vectors amb curvatura'.

La quantitat de curvatures contraposades sumant-li 1 ens indicarà el grau

És a dir si tenim una curvatura com una semielipse o semicircumferència serà de grau 2

Si tenim dues curvatures contraposades serà de grau 3

....

Si tenim una forma estranya amb 7 curvatures contraposades serà de grau 8

And so Forth...

Unsupervised Learning

En l'aprenentatge **no supervisat** trobem les tècniques de **Neural Networks inspirades en les neurones humanes tant del cervell com del sistema nerviós.**

Quan les ANN (Artificial Neural Networks) es complexitzen amb moltes dimensions i ítems esdevé en models de **Deep Learning** d'alta complexitat i sofisticació.

Ho veurem posteriorment.

UNSUPERVISED LEARNING
CONTINU

MACHINE
LEARNING

ML



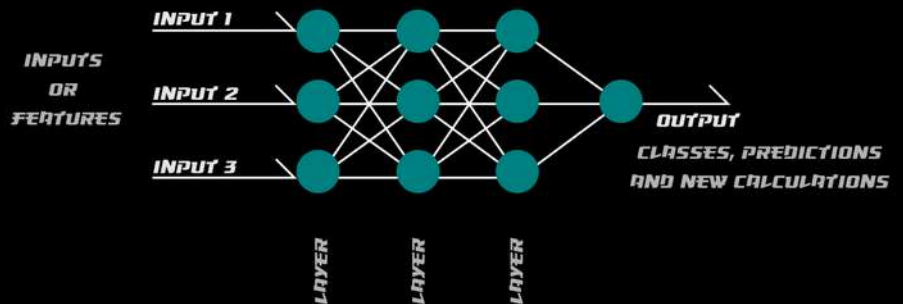
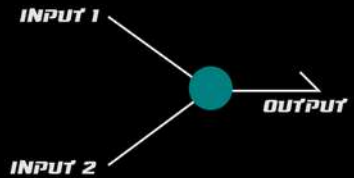
'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

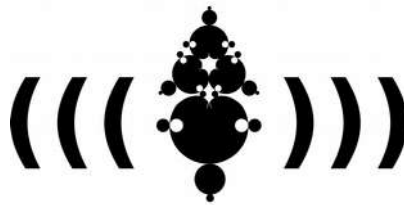
ARTIFICIAL
NEURAL
NETWORKS

DEEP
LEARNING

PERCEPTRON | SINGLE NEURON

NETWORK OF PERCEPTRONS
NEURAL NETWORK





Beques de recerca i creació OSIC· gencat 2021

projecte *músIA*

Xavi Manzanares

2021 / 2022

**MACHINE LEARNING
FOR NEWBIES
VOL2 SUPERVISED LEARNING
CLASSIFICATION**

Aknowledgments:

*Molts diagrames i representacions d'aquest tuto són directament inspirats del molt recomanable curs de
Rebecca Fiebrink 'Machine Learning for Musicians and Artists'*

<https://www.kadenze.com/courses/machine-learning-for-musicians-and-artists-v>

on es mostra una overview al ML amb projectes de media & sonic interaction per mitjà del programari ML

We

**MACHINE
LEARNING**
ML



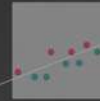
'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

**SUPERVISED
LEARNING
DISCRET**

CLASSIFICATION

REGRESSION

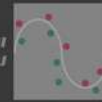
LINEAR REGRESSION



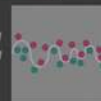
**POLINOMI
2N GRADU**



**POLINOMI
3R GRADU**



**POLINOMI
9E GRADU**



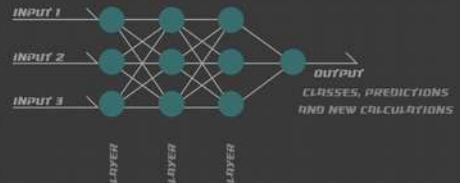
POLYNOMIAL REGRESSION

**UNSUPERVISED LEARNING
CONTINU**

**ARTIFICIAL
NEURAL
NETWORKS**

**NETWORK OF PERCEPTRONS
NEURAL NETWORK**

**DEEP
LEARNING**



**MACHINE
LEARNING**
ML



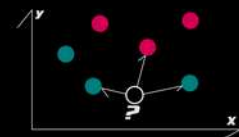
'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

**SUPERVISED
LEARNING
DISCRET**

CLASSIFICATION

K-NEAREST NEIGHBOUR

TRIGONOMETRY



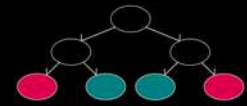
NAIVE BAYES

PROBABILISTIC



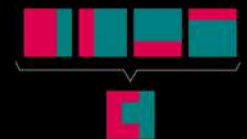
DECISION TREES

CONDITIONAL BLOCKS



ADA BOOST

META ALGORITHM



SUPPORT VECTOR MACHINES

MULTIDIMENSIONAL SPACE



MACHINE
LEARNING
ML



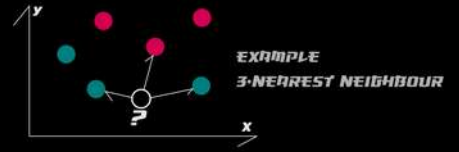
'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

SUPERVISED
LEARNING
DISCRET



K= NUMBER OF ELEMENTS TO CALCULATE DISTANCE

DISTANCES CALCULATES BY EUCLIDEAN METHODS
IN CASE ITS ONLY BIDIMENSIONAL IS THE CLASSIC PITAGORIC FORMULA
IN CASE INPUTS ARE MORE THAN 2 THEREFORE HAS SOME MORE DIMENSIONS, CALCULUS GET MORE COMPLICATED



$$\sqrt{\sum_{i=1}^d (X_{fa} - X_{fb})^2}$$

distance between example a and example b with d features or inputs



NO TRAINING TIME
CAPABLE OF ARBITRARY COMPLEX BOUNDARIES



METHOD THAT GOES WRONG WITH VERY DIFFERENT MAGNITUDES
> NORMALIZE FEATURES OR INPUTS WITHIN A SIMILAR RANGE FROM 0 TO 1
SLOW METHOD FOR LARGE TRAINING EXAMPLES
CAN BE AFFECTED BY NOISE
CAN BE AFFECTED BY IRRELEVANT AND REDUNDANT FEATURES

MACHINE
LEARNING
ML



'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

SUPERVISED
LEARNING
DISCRET



NAIVE BAYES



PROBABILISTIC METHODS

COMPUTE THE LIKELIHOOD OF THE GIVEN FEATURE VALUES BEING OBSERVED FOR EACH CLASS

COMBINE THIS CONSIDERING THE RELATIVE PROPORTIONS OF CLASSES IN THE WORLD
TO COMPUTE PROBABILITY THAT NEW EXAMPLE IS FROM EACH CLASS



FAST TRAINING AND RUNNING
CAN GET INFO ABOUT PROBABILITY EACH CLASS IS NOT JUST SINGLE 'BEST' CLASS
NOT AS SENSITIVE TO OUTLIERS OR SINGLE ERRORS, LESS LIKELY TO OVERFIT



NOT AS SENSITIVE TO OUTLIERS OR SINGLE ERRORS, LESS LIKELY TO OVERFIT



ASSUMPTION THAT FEATURES ARENT RELATED TO EACH OTHER ISNT ALWAYS EFFICIENT,
CAN LEAD TO WORSE PERFORMANCE

MACHINE LEARNING ML



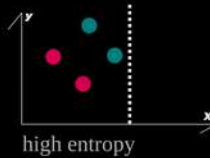
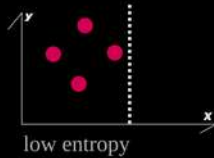
'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

SUPERVISED
LEARNING
DISCRET

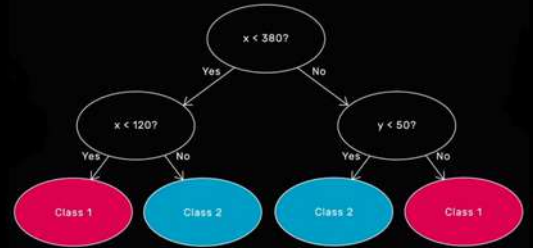


DECISION STUMP IF / CONDITIONALS
A TREE OF DECISION STUMP LOOPS IS A DECISION TREE.
LEAFS ARE ELEMENTS IN DEEPER LAYERS OF THE TREE

ALSO A NICE POINT IS TO INTRODUCE ENTROPY IN THE MODEL AS A MEASURE OF HOMOGENEITY / DIVERSITY



DECISION TREES
C4.5 ALGORITHM
J48 ALGORITHM
...



- FAST TRAINING AND RUNNING
- EASY TO WRITE AS CODE
- IF A FEATURE IS IRRELEVANT DECISION TREES CAN IGNORE IT
- DECISION TREES DONT REQUIRE YOU TO TUNE PARAMETERS



IS NOT A CUTTING EDGE METHOD TO SOLVE COMPLICATED PROBLEMS.

MACHINE LEARNING ML



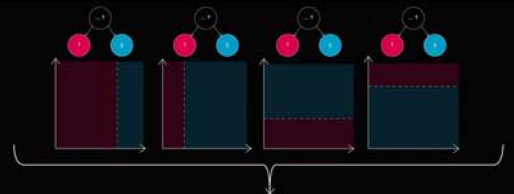
'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

SUPERVISED
LEARNING
DISCRET

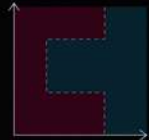


ADAPTATIVE BOOST OF
DIFFERENT MULTIPLE ALGORITHMYS
LIKE DECISION TREES OR DECISION STUMP

Ada BOOST



AdaBoost combines multiple learning algorithms into one (hopefully) really good classifier:



a meta-learning algorithm



- FAST TRAINING AND RUNNING
- EASY TO TUNE > CAN TUNE TRAINING SPEED BY ADJUSTING NUMBER OF TRAINING ROUNDS
- ICAN DEAL GRACEFULLY WITH IRRELEVANT FEATURES
- THEORETICALLY LESS PRONE TO OVERFITTING
- COMPUTING ON NEW EXAMPLES IS FAST



LESS LIKELY TO OVERFIT, SO NEED TO GIVE MORE DATA TO MAKE MORE COMPLEX BOUNDARIES

**SUPERVISED
LEARNING
DISCRET**



SUPPORT VECTOR MACHINES

**MACHINE
LEARNING
ML**



**'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS**

**USING MORE THAN ONE CLASSIFIER AT ONCE
FEATURES MULTIDIMENSIONAL SPACES TO SOLVE HARDER CLASSIFICATIONS IN LOWER DIMENSIONS.**



src img > kadenze courseware machine learning for musicians and artists by Rebecca Fiebrink

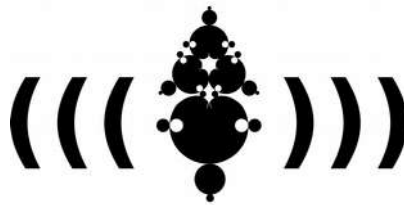
KERNEL PROJECTS DATA INTO HIGHER DIMENSIONAL SPACE
LINEAR
POLYNOMIAL
RBF
DIFFERENT KERNELS AND KERNEL PARAMETERS WILL GIVE DIFFERENT RESULTS



VERY POWERFUL CAN WORK IN COMPLICATED PROBLEMS
SEVERAL KERNEL TYPES
COMPUTATION ON NEW EXAMPLES IS FAST



TRAINING INVOLVES SOLVING OPTIMIZATION THAT CAN BE SLOW FOR LARGE DATASETS
CHOOSING KERNELS AND ITS PARAMETERS CAN BE FRUSTRATING
MIGHT NOT WORK WELL FOR SMALL TRAINING SETS OR COMPLEX COMBINATIONS LIKE DIFFERENT NUMBERS OF EXAMPLES IN EACH CLASS



Beques de recerca i creació OSIC· gencat 2021

projecte *músIA*

Xavi Manzanares

2021 / 2022

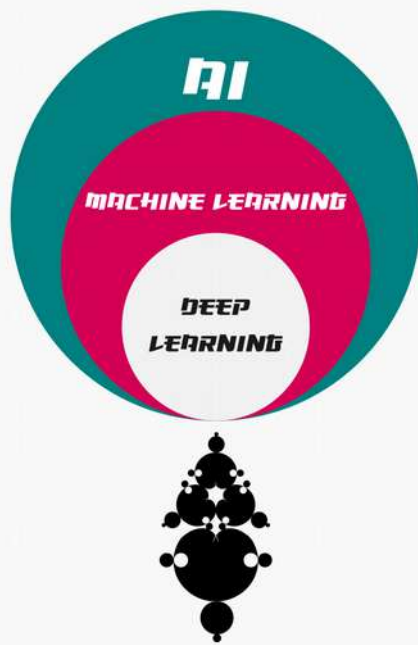
**MACHINE LEARNING
FOR NEWBIES
VOL3
ARTIFICIAL NEURAL NETWORKS**

Aknowledgments:

Molts diagrames i representacions d'aquest tuto són directament inspirats del molt recomanable curs de
Rebecca Fiebrink 'Machine Learning for Musicians and Artists'

<https://www.kadenze.com/courses/machine-learning-for-musicians-and-artists-v>

*on es mostra una overview al ML amb projectes de media & sonic interaction per mitjà del programari ML
Wekinator.*



UNSUPERVISED LEARNING
CONTINUU

MACHINE
LEARNING

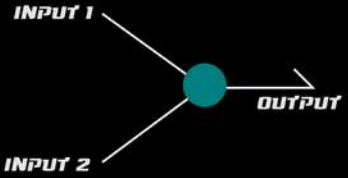
ML



'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

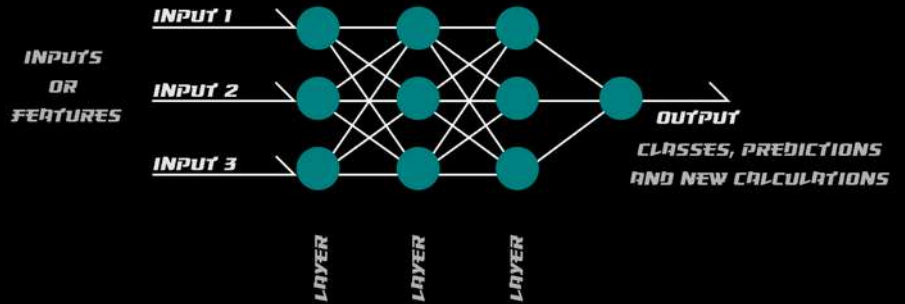
ARTIFICIAL
NEURAL
NETWORKS

DEEP
LEARNING



PERCEPTRON | SINGLE NEURON

NETWORK OF PERCEPTRONS
NEURAL NETWORK



PERCEPTRON | SINGLE NEURON

MACHINE
LEARNING

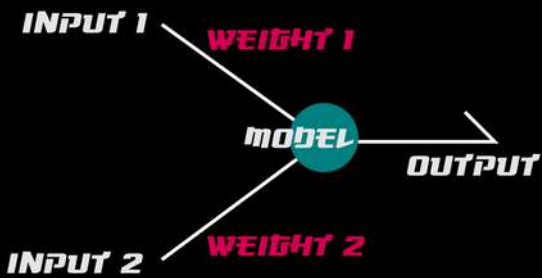
ML



'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

ARTIFICIAL
NEURAL
NETWORKS

MODEL



$$SUM = INPUT1 \times WEIGHT1 + INPUT2 \times WEIGHT2$$

ACTIVATION FUNCTION

f

LINEAR
SINE
TANG > SIGMOID

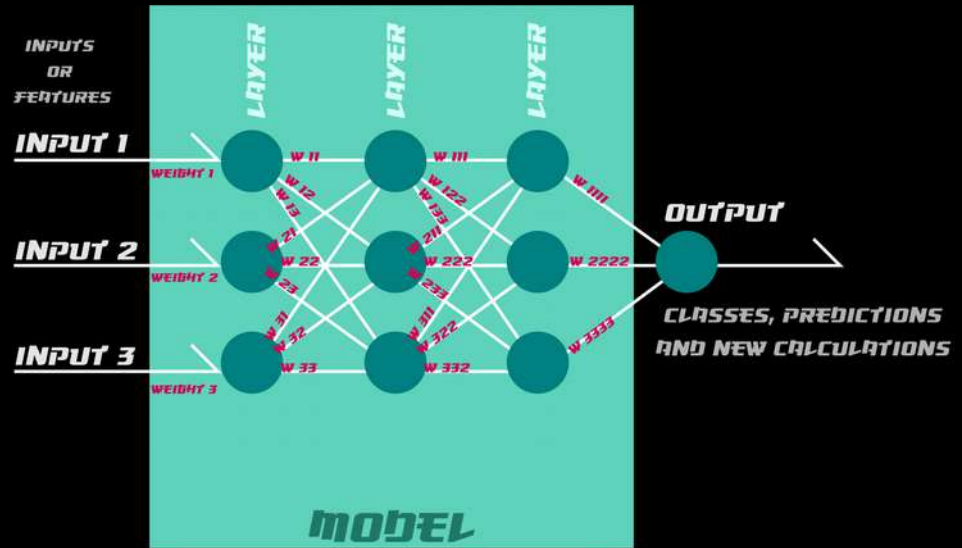
ANN ARTIFICIAL NEURAL NETWORK

MACHINE
LEARNING

ML



'INTEL·LIGÈNCIA'
QUE POT APRENDRE
PER DIVERSOS
MÈTODES I
GENERAR PREDICCIONS

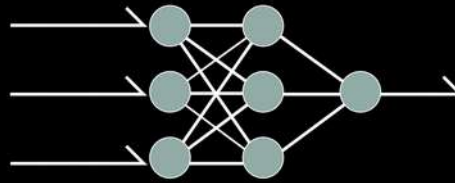


MACHINE
LEARNING
ML



PROBLEM :
WE WANT TO TRAIN
A NEURAL NETWORK
TO FIND A CERTAIN COLOR

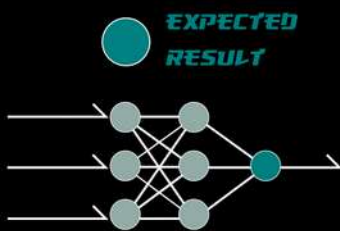
**EXPECTED
RESULT**



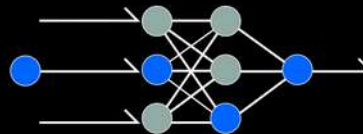
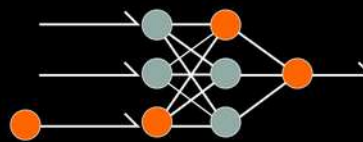
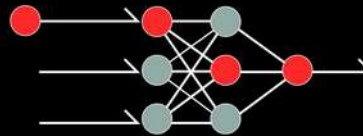
MACHINE
LEARNING
ML



PROBLEM :
WE WANT TO TRAIN
A NEURAL NETWORK
TO FIND A CERTAIN COLOR

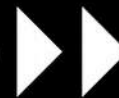


**INPUTS
RANDOM COLOR**

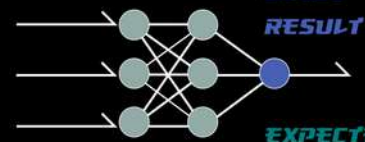


ETC

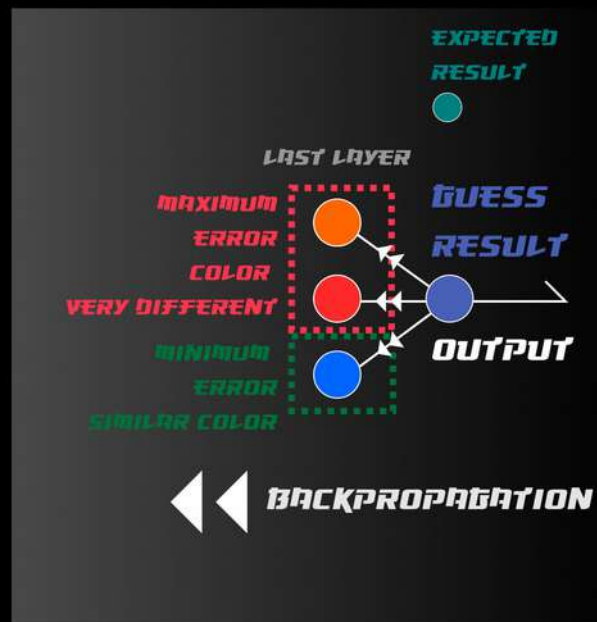
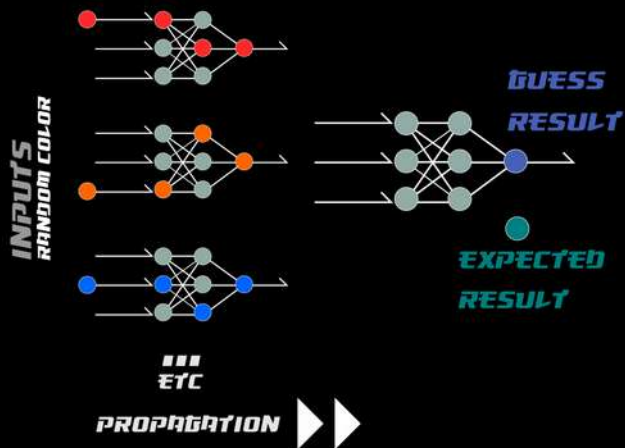
PROPAGATION



WITH DIFFERENT METHODS
OF COMPARISON BETWEEN ITEMS
EX NEAREST NEIGHBOUR ALGORITHMS
EXTRACT THE MIX



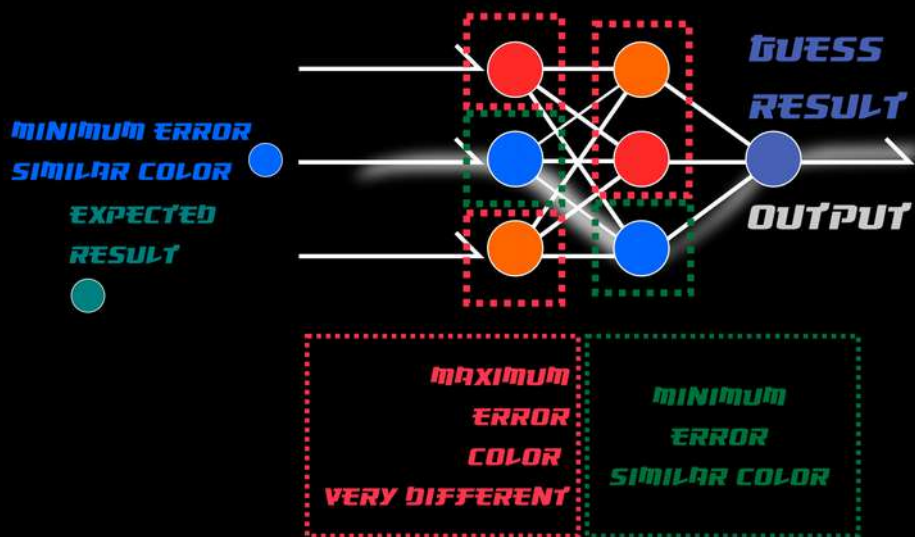
MACHINE
LEARNING
ML



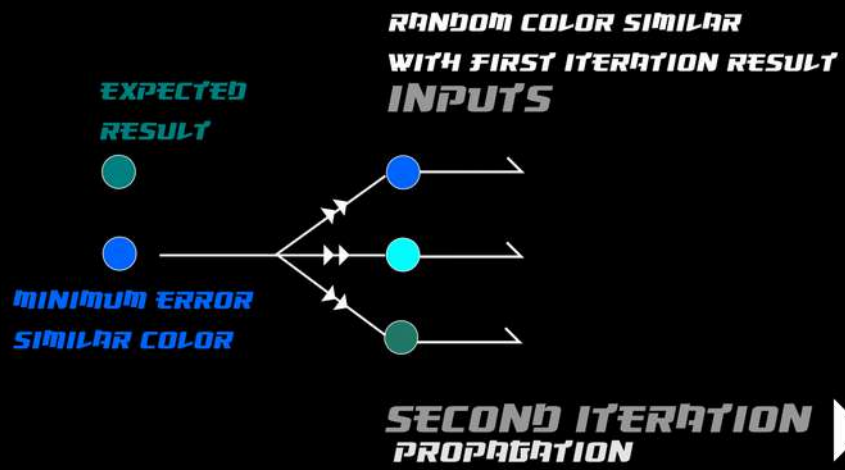
BACKPROPAGATION

GRADIENT DESCENT ALGORITHM
= MINIMIZATION OF ERRORS

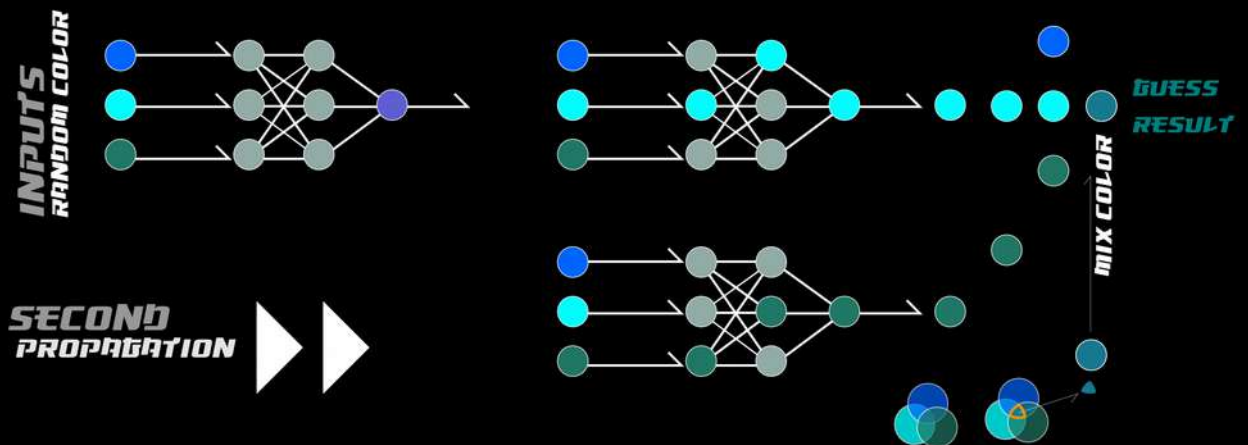
MACHINE
LEARNING
ML



MACHINE
LEARNING
ML

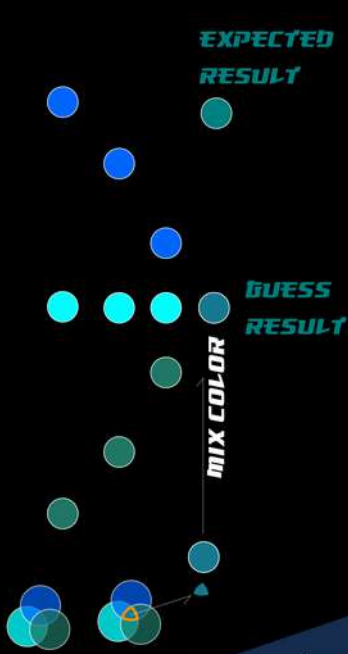


MACHINE
LEARNING
ML



PROBLEM :
WE WANT TO TRAIN
A NEURAL NETWORK
TO FIND A CERTAIN COLOR

MACHINE
LEARNING
ML



EXPECTED
RESULT

000000FF

COMPARISON
SIMILARITY
 $> = 90\%$

GUESS
RESULT

177A90FF

ACCEPTABLE OUTPUT
ACCEPTABLE PREDICTION

IN CASE IS NOT AN ACCEPTABLE OUTPUT
LEARNING CONSISTS IN DOING SEVERAL
LOOPS OF BACKPROPAGATION AND PROPAGATION
UNTIL ARRIVE TO A SIMILAR ACCEPTABLE MATCH



2ND
BACKPROPAGATION



3TH
PROPAGATION



3TH
BACKPROPAGATION



4TH
PROPAGATION



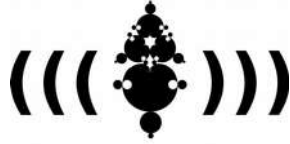
ETC





**REFERÈNCIES
I PROJECTES
AI APLICATS A
LA CREACIÓ MUSICAL**



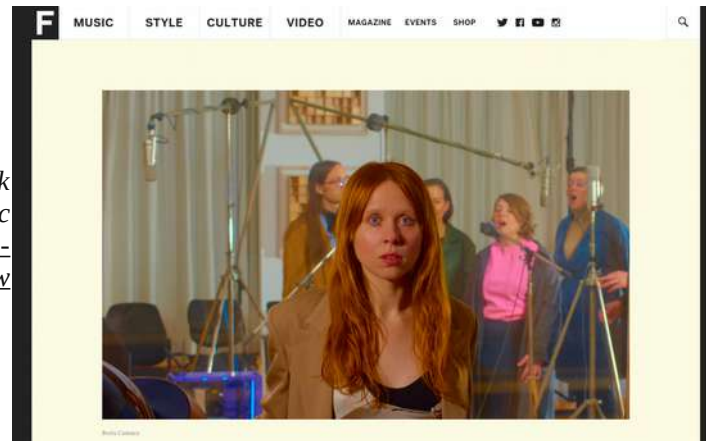


MUSIC SECTION

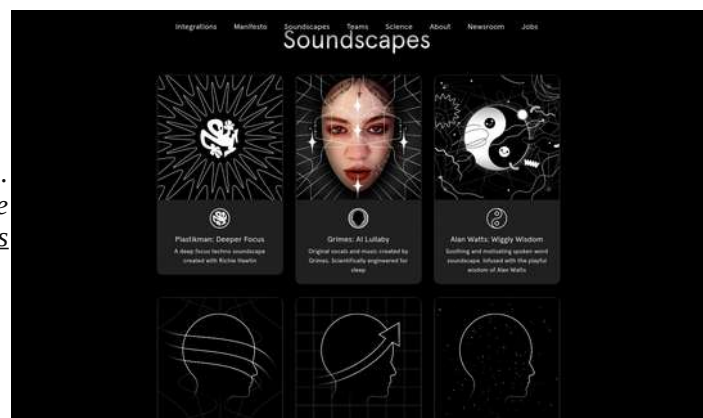
Hello World is the first album composed by an artist – SKYGGE – with artificial intelligence.
<https://www.helloworldalbum.net/>



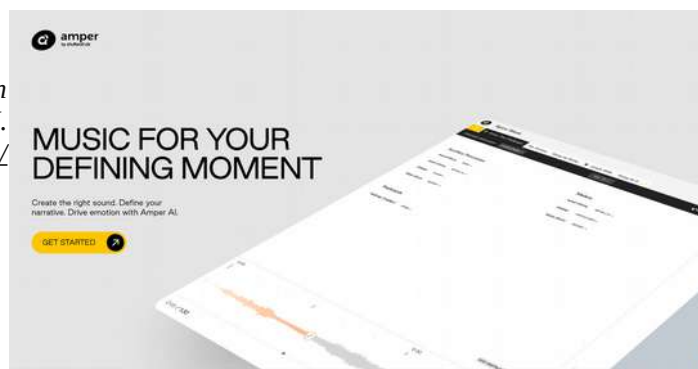
How Holly Herndon and her AI baby spawned a new kind of folk music
<https://www.thefader.com/2019/05/21/holly-herndon-ai-spawn-interview>



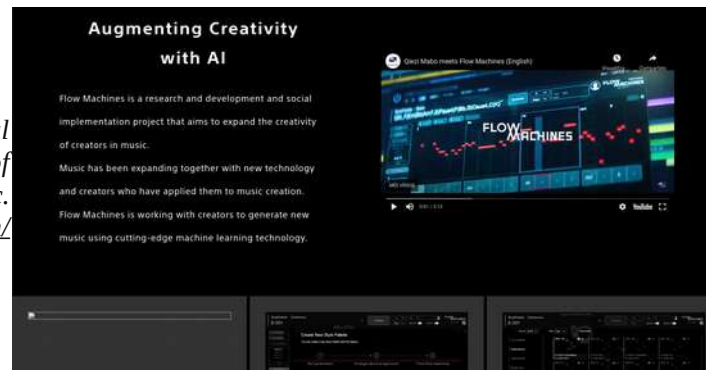
Personalized soundscapes to help you focus, relax, and sleep. Backed by neuroscience
<https://endel.io/soundscapes>



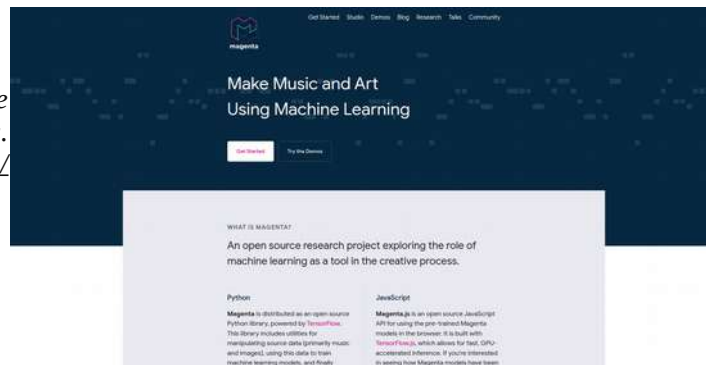
Create the right sound. Define your narrative. Drive emotion with Amper AI.
<https://www.ampermusic.com/>



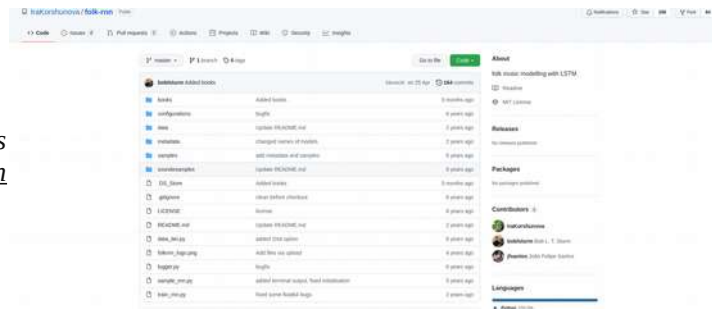
Flow Machines is a research and development and social implementation project that aims to expand the creativity of creators in music.
<https://www.flow-machines.com/>



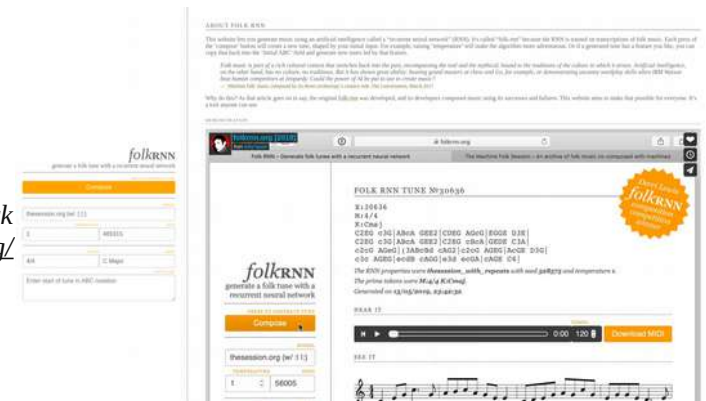
An open source research project exploring the role of machine learning as a tool in the creative process.
<https://magenta.tensorflow.org/>



Folk music style modelling using LSTMs
<https://github.com/IraKorshunova/folk-rnn>



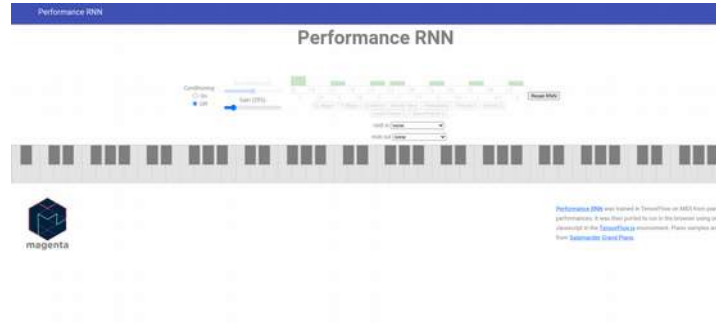
generate a folk tune with a recurrent neural network
<https://folkrrn.org/>



Accessible AI Tools by Shirin Anlen

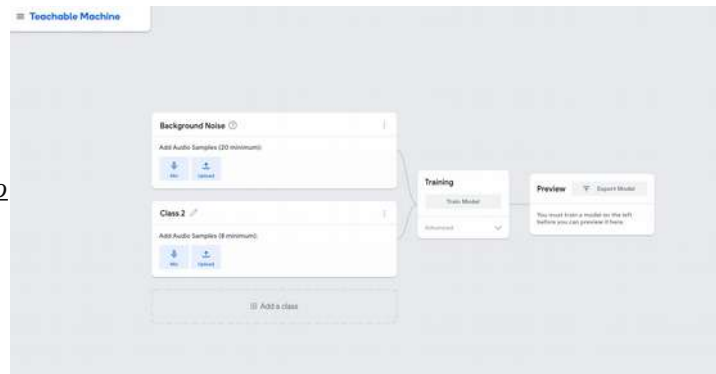


<https://docubase.mit.edu/lab/case-studies/accessible-ai-tools-by-shirin-anlen/>

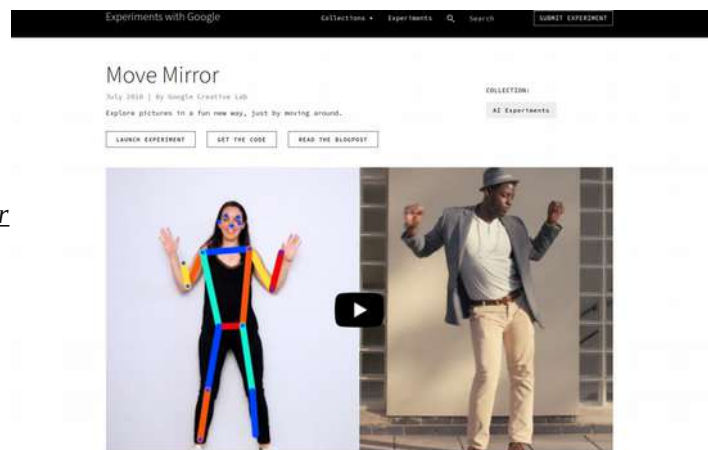


https://magenta.tensorflow.org/demos/performance_rnn/index.html#2%7C2,0,1,0,1,1,0,1,0,1,0,1,0,1%7C1,1,1,1,1,1,1,1,1,1,1,1,1%7C1,1,1,1,1,1,1,1,1,1,1,1,1%7Cfalse

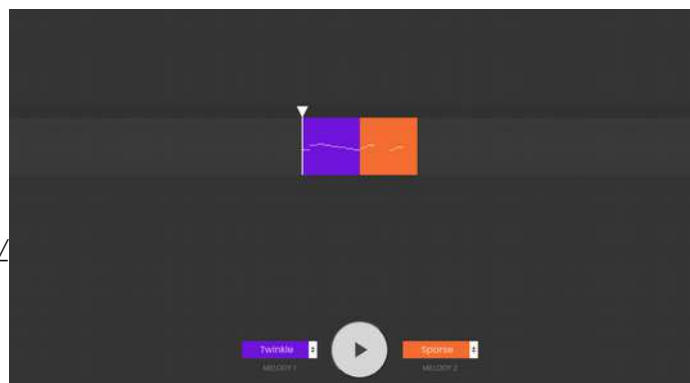
<https://teachablemachine.withgoogle.com/train/audio>



<https://experiments.withgoogle.com/move-mirror>



<https://experiments.withgoogle.com/ai/melody-mixer/view/>



<https://mimicproject.com/>

about • explore • learn •

Make Music and Art with Machine Intelligence

The mimic project offers a way to make new kinds of music, sound and creative arts experiences using machine learning, machine listening and artificial

Getting Started

MIMIC is a web platform for the artistic exploration of musical machine learning and machine listening. We have designed this collaborative platform as an interactive online coding environment, engineered to bring new technologies in AI and signal processing to artists, composers, musicians and performers all over the world.

<https://nsynthsuper.withgoogle.com/>

NSYNTH SUPER

MAKING MUSIC USING NEW SOUNDS GENERATED WITH MACHINE LEARNING

WATCH THE FILM

magenta Get Started Studio Demos Blog Research Talks Community

<https://magenta.tensorflow.org/studio/standalone>

Magenta Studio (Standalone)

Download for macOS Download for Windows View on GitHub

This page is for the standalone version of Magenta Studio. If you're looking for the Ableton Live Integration instead, click [here](#).

TABLE OF CONTENTS

- Overview
- Installation
- Usage
- Continue
- Generate
- Interpolate
- Groove
- Drumify

Overview

Magenta Studio is a MIDI plugin for Ableton Live. It contains 5 tools: [Continue](#), [Groove](#), [Generate](#), [Drumify](#), and [Interpolate](#), which let you apply Magenta models to your MIDI files.

magenta Get Started Studio Demos Blog Research Talks Community

<https://magenta.tensorflow.org/music-vae>

MusicVAE: Creating a palette for musical scores with machine learning.

Mar 19, 2018

Adam Roberts [@adarob](#) [@ada_r](#) [@ada_r](#)
Jesse Engel [@jesseengel](#) [@jesseengel](#)
Cain Raffel [@craffel](#) [@craffel](#)
Ian Simon [@iansimon](#) [@iansimon](#)
Curtis Hawthorne [@cghawthorne](#) [@jordan1](#)

When a painter creates a work of art, she first blends and explores color options on an artist's palette before applying them to the canvas. This process is a creative act in its own right and has a profound effect on the final work.

Musicians and composers have mostly lacked a similar device for exploring and mixing musical ideas, but we are hoping to change that. Below we introduce **MusicVAE**, a machine learning model that lets us create palettes for blending and exploring musical scores.

As an example, listen to this gradual blending of 2 different melodies, A and B. We'll explain how this morph was achieved throughout the post.

MusicVAE: Melody 2 bar "Loop" Int... Visualiza... Comparex

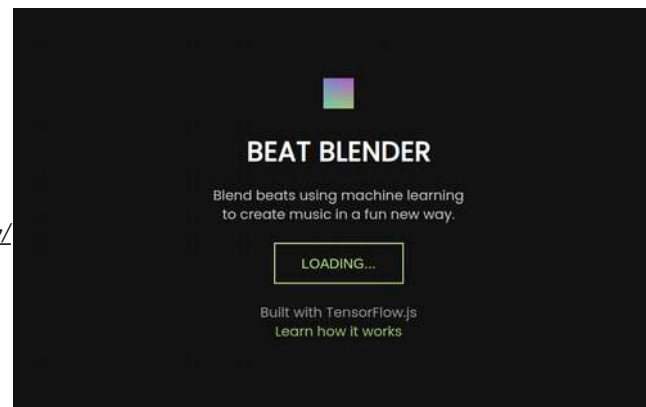
<https://www.youtube.com/watch?v=QM6LbbcCghc>



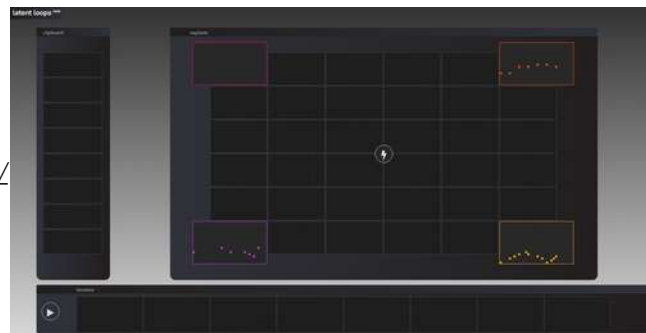
<https://www.youtube.com/watch?v=ILV0259QGxk>



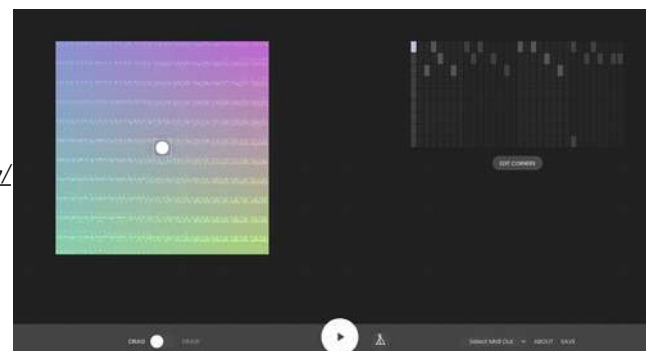
<https://experiments.withgoogle.com/ai/beat-blender/view/>



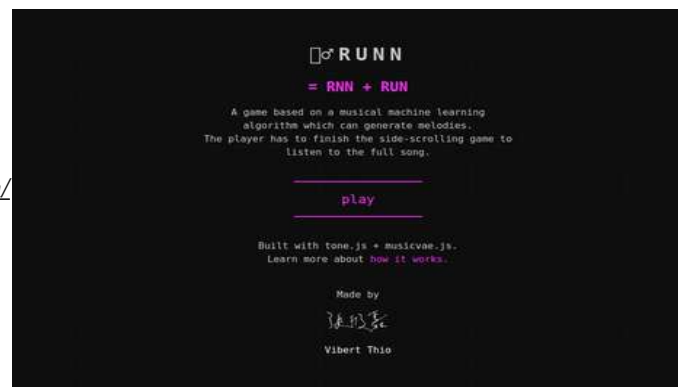
<https://teampieshop.github.io/latent-loops/>



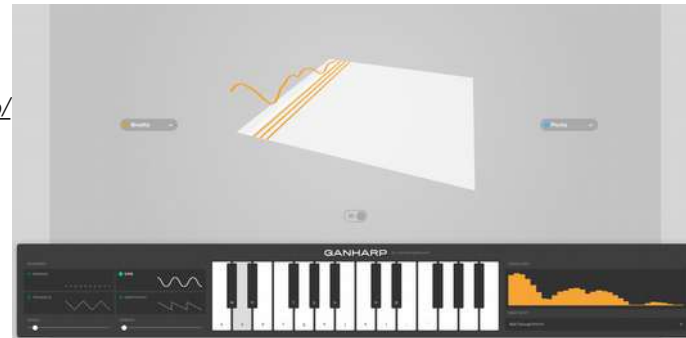
<https://experiments.withgoogle.com/ai/melody-mixer/view/>



<http://vibertthio.com/runn/>



<https://ganharp.ctcpt.co/>



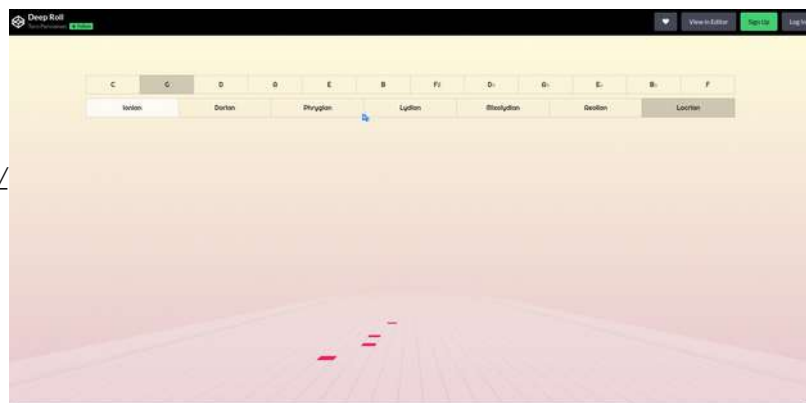
<https://codepen.io/teropa/full/gzigEP/>



<https://codepen.io/teropa/full/RMGxOQ/>



<https://codepen.io/teropa/full/zpbLOj/>



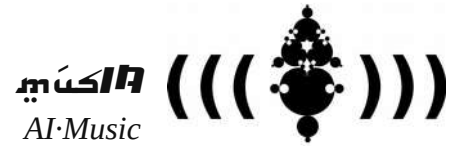


**REFERÈNCIES
I PROJECTES
D'APLICACIÓ A
L'ART I LA CREACIÓ**





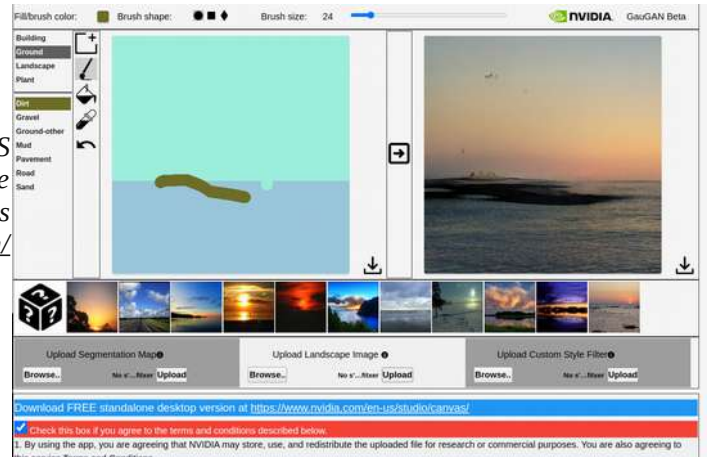
ART & CREATIVITY SECTION



*AI-Music
(Relevant) Resources & Linkography
order in this list is not relevant, it's just a research container*

*NVIDIA CANVAS
Use AI to turn simple brushstrokes into realistic landscape images*

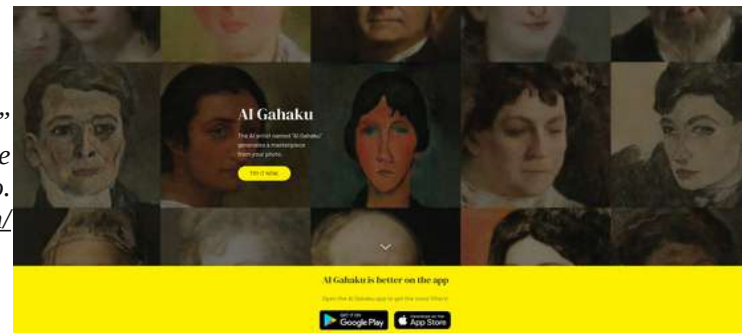
<http://nvidia-research-mingyuliu.com/gaugan/>



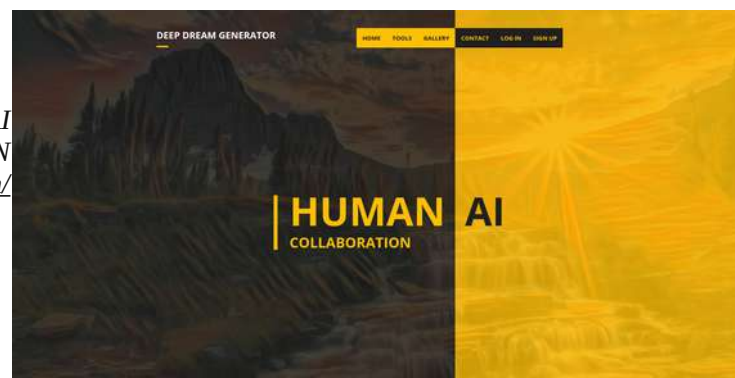
*Empatizando Con La Psique De Las IAS
<https://bikolabs.biko2.com/empatizandoconias/>*



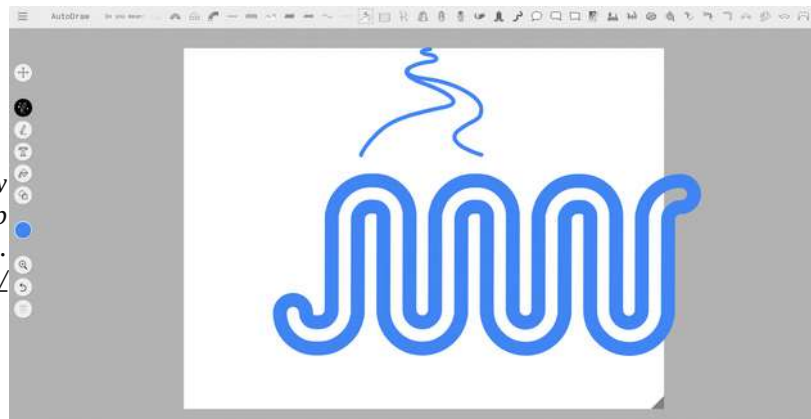
*The AI artist named "AI Gahaku"
generates a masterpiece
from your photo.
<https://ai-art.tokyo/en/>*



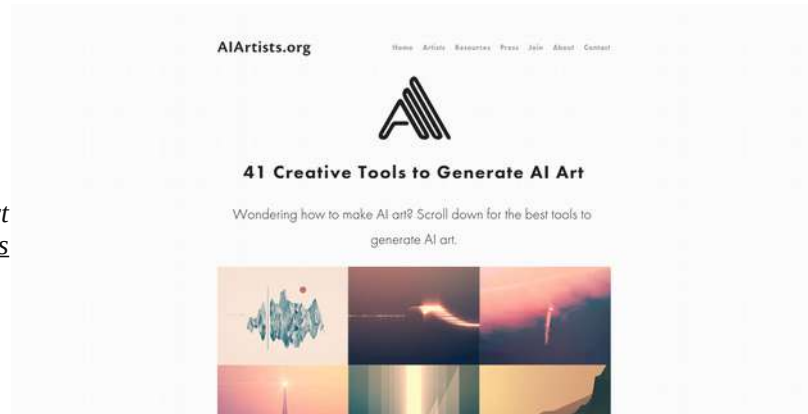
*HUMAN AI
COLLABORATION
<https://deepdreamgenerator.com/>*



AutoDraw
By Google Creative Lab
Fast drawing for everyone.
<https://www.autodraw.com/>



41 Creative Tools to Generate AI Art
<https://aiartists.org/ai-generated-art-tools>



ML-based Image Processing
<https://nanonets.com/blog/machine-learning-image-processing/>



ML-based Image Processing

by Vihar Kurama 3 months ago 5 MIN READ

Have a machine learning image processing problem in mind? Want to leverage ML & DL to automate image processing?

Get Started

Machine Learning (ML) has become one of most widely used AI techniques for several companies, institutions and individuals who are in the business of automation. This is because of considerable improvements in the access to data and increases in computational power, which allow practitioners to achieve meaningful results across several areas.

WebGazer.js
Democratizing Webcam Eye Tracking on the Browser
<https://webgazer.cs.brown.edu/>



WebGazer.js

Democratizing Webcam Eye Tracking on the Browser

WebGazer.js is an eye tracking library that uses common webcams to infer the eye-gaze locations of web visitors on a page in real time. The eye tracking model it contains self-calibrates by watching web visitors interact with the web page and forms a mapping between the features of the eye and positions on the screen. WebGazer.js is written entirely in JavaScript and with only a few lines of code can be integrated in any website that wishes to better understand their visitors and transform their user experience. WebGazer.js runs entirely in the client browser, so no video data needs to be sent to a server, and it requires the user's consent to access their webcam.



- Real time gaze prediction on most common browsers
- No special hardware; WebGazer.js uses your webcam
- Self calibration from clicks and cursor movements
- Easy to integrate with a few lines of JavaScript
- Separable components for eye detection
- Multiple gaze prediction models
- Continuously supported and open source for 4+ years

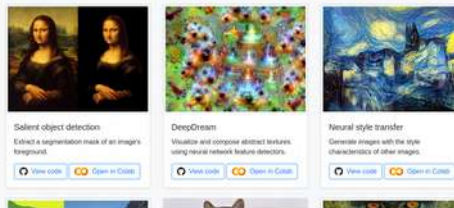
Usage

To use WebGazer.js you need to add the webgazer.js file as a script in your website.

Machine Learning for Art

mila is a collection of tools and educational resources which apply techniques from machine learning to arts and creativity.

Home Fundamentals Tools



Machine Learning for Art
<https://ml4a.net/>

Neural Networks

Processing with AI

- PART 1: BASICS -
 - Introduction
 - Basic Setup
 - JavaScripT removers
 - Starting with p5.js
 - Machine Learning
- Neural Networks
 - Classification
 - Chatbot
 - Runway
- PART 2: EXPLORATIONS -
 - Introduction
 - Ethics of AI
 - Modules (not over)
 - Transfer Learning
 - NLP
 - Computer Vision

Let's start by stating two things that most people get wrong about neural networks:

- Artificial "neurons" are just a fancy way to describe a programming building block that processes **numbers** and **only numbers**.
- We use the term "neuron" only because it has some high-level similarities with a biological neuron.

THE ONLY BIOLOGY LESSON YOU'LL GET IN EMLYON

<https://processing-with-ai.gitlab.io/part1/intro-to-nn/>

m5.js - Friendly Machine Learning for the Web

Welcome to the m5.js documentation. Here you'll find everything you need to get up and started with m5.js.

Getting Started

Take a ride on the Coding Train to search for [@efclark's](#) [code](#) on [YouTube](#) or [GitHub](#) to learn more about m5.js. Here Evan explains what m5.js is and where it all comes from.

m5.js is machine learning for the web and your web browser. Through some clever and exciting abstractions, the folks building [SocialFlow](#) figured out that it's possible to use the web browser's built-in graphics processing and WebGL to do calculations that would otherwise run very slowly using normal processing and JS. It only has requirements of WebGL to be supported with WebGL on the [found](#) [here](#) [m5.js](#) [github](#) [m5.js](#) [github](#). m5.js allows to make all these new developments, to make learning on the web more approachable for everyone.

Quickstart

The fastest way to get started exploring the creative possibilities of m5.js are:

- If you're interested in using m5.js with WebGL, you can start with our [Interactive GPU and WebGL Demo](#) or view the [p5.js - m5.js Interactive Demo](#).
- If you're interested in using m5.js without WebGL, visit the [m5.js plain javascript](#) that has been [demo](#). View the [Interactive Demo](#) for an interactive demo.
- Alternatively, you can copy and paste the CDN link to the m5.js library.

```

<script src="https://unpkg.com/m5.js@0.1.0/dist/m5.js">

```

Quickstart: Plain JavaScript

Reference the [plain javascript](#) of m5.js using a script tag in an HTML file as below:

in an [index.html](#) file, copy and paste the following and open up that file in your web browser.

<https://learn.ml5js.org/>

m5.js examples search

search:

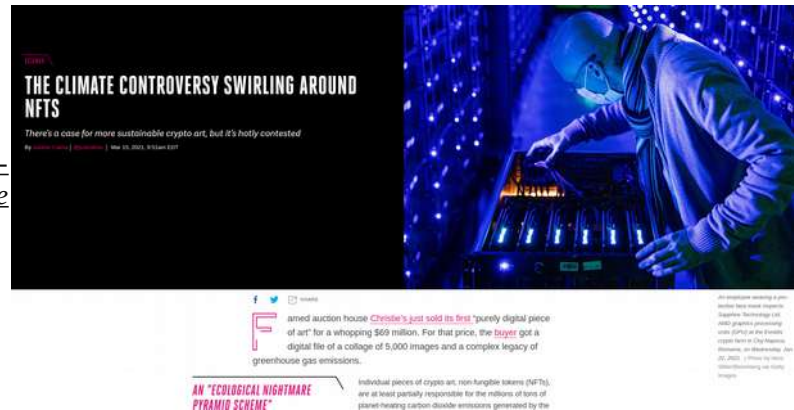
[https://examples.ml5js.org/](#)

will help you get started using m5.js. The examples are organized by the features available in m5.js. You can open the interactive demos using the p5.js web editor or check out the various demos that use p5.js or plain javascript.

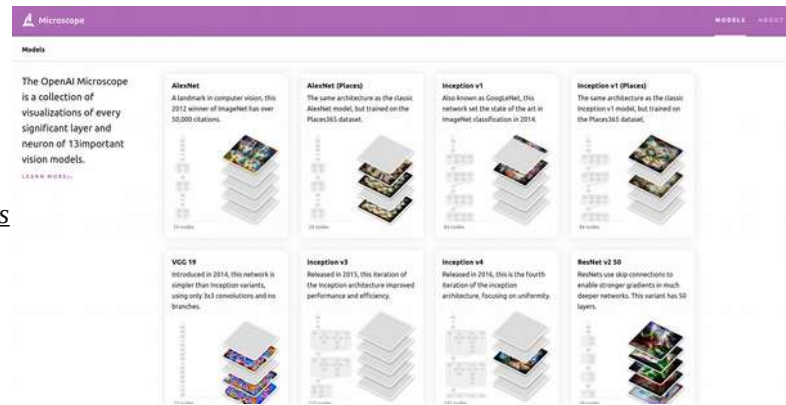
<h4>BodyPix</h4> <ul style="list-style-type: none"> p5.js web editor BodyPix_image BodyPix_widgets BodyPix_widgets_Parts p5.js demo BodyPix_image BodyPix_widgets BodyPix_widgets_Parts BodyPix_image BodyPix_widgets BodyPix_widgets_Parts 	<h4>CVAE</h4> <ul style="list-style-type: none"> p5.js web editor CVAE_QuickDraw p5.js demo CVAE_QuickDraw plain javascript demo CVAE_QuickDraw 	<h4>CartoonGAN</h4> <ul style="list-style-type: none"> p5.js web editor CartoonGAN_Base CartoonGAN_Landmarks CartoonGAN_Widgets CartoonGAN_WidgetCam p5.js demo CartoonGAN_Base CartoonGAN_Landmarks CartoonGAN_WidgetCam
<h4>CharRNN</h4> <ul style="list-style-type: none"> p5.js web editor CharRNN_interactive CharRNN_Test CharRNN_Test_Statistik p5.js demo CharRNN_interactive CharRNN_Test CharRNN_Test_Statistik 	<h4>DCGAN</h4> <ul style="list-style-type: none"> p5.js web editor DCGAN_Landmarks_RandomMask DCGAN_Landmarks_Slider DCGAN_Random p5.js demo DCGAN_Landmarks_RandomMask DCGAN_Landmarks_Slider DCGAN_Random 	<h4>FaceApi</h4> <ul style="list-style-type: none"> p5.js web editor FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks FaceApi_image_Landmarks

<https://examples.ml5js.org/>

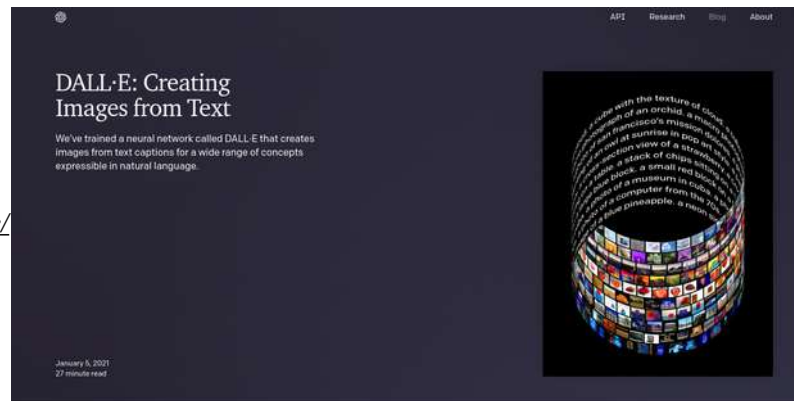
<https://www.theverge.com/2021/3/15/22328203/nft-cryptoart-ethereum-blockchain-climate-change>



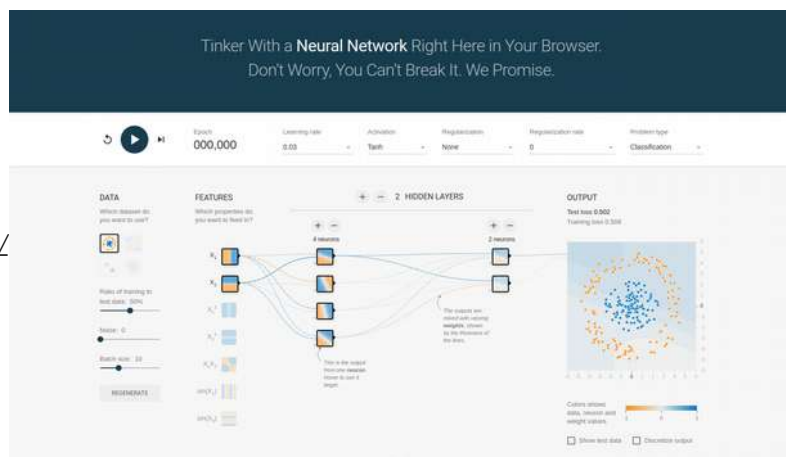
<https://microscope.openai.com/models>



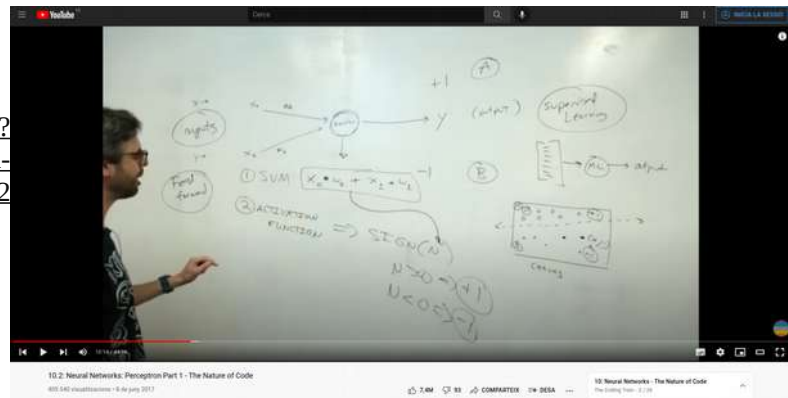
<https://openai.com/blog/dall-e/>



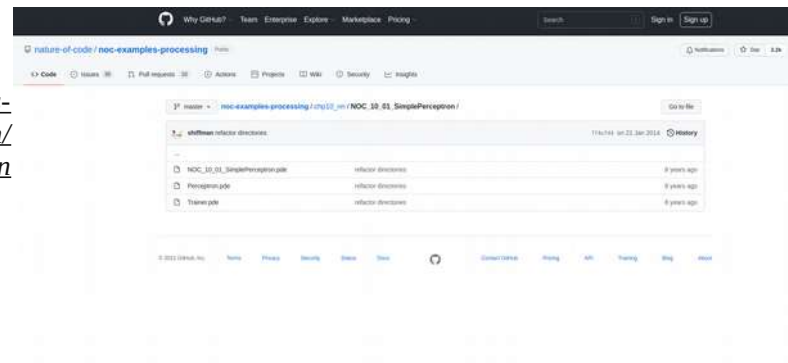
<https://playground.tensorflow.org/>



<https://www.youtube.com/watch?v=u5GAVdLQyIg&list=PLRqwX-V7Uu6aCibgK1PTWWu9by6XFdCfh&index=2>



https://github.com/nature-of-code/noc-examples-processing/tree/master/chp10_nn/NOC_10_01_SimplePerceptron



<https://www.inverse.com/innovation/xenobots-are-living-machines>



https://youtu.be/jntk_I1WXHA?t=1352
Bioelectrical patterns in lifeforms

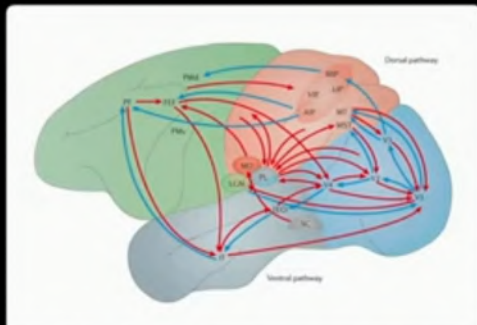




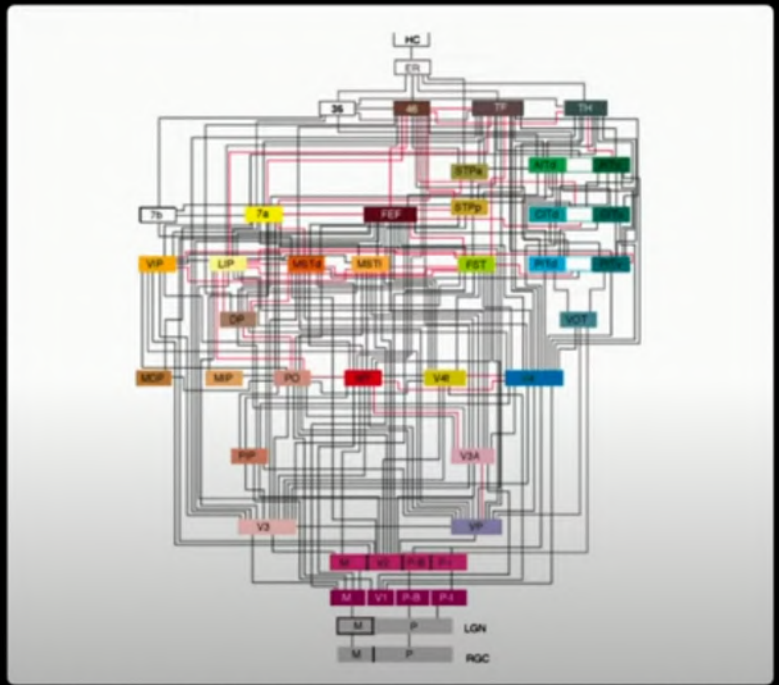
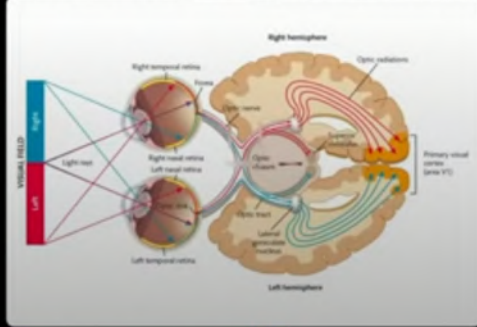


Tesla AI Day

Biological Visual Cortex Wiring



Top-down influences on visual processing. Nature Reviews Neuroscience, 2013.

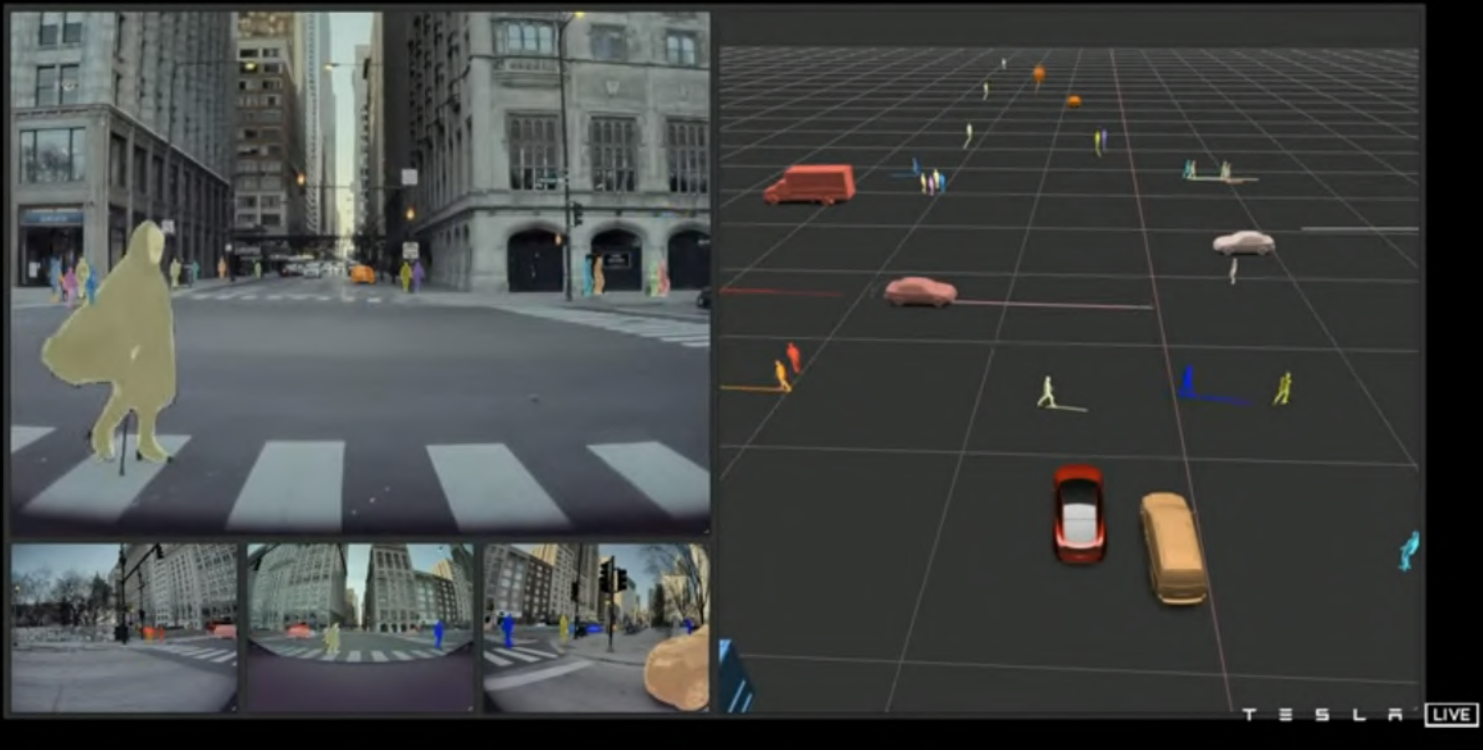


50:26 / 3:03:20 • Tesla Vision > primary-visual-cortex-5th-diagram/ Desplaça't per la pàgina per veure més informació...rocessing in the primate cerebral cortex. Cereb Cortex, 1991 Jan-Feb; 1:1-11

AI.Tesla Day Questions

////////////////////

Persisting Vehicles & Pedestrians Through Occlusions



> Massive amount of data analysis which illustrates the huge cognitive complexity that everyday lives we are using while we are driving, thinking that is an pretty easy task.

Thats an image of which is the difficulty of self driving objects from a perceptive and cognitive POV, therefore the difficulty not only to develop Intelligent machines in calculus, but in a more apparently simple task that is to propioceptively move through the surrounded space.

>Autopilot-car system involves 8 cameras to get a surrounding Proprioceptive space of the car.

What happens when a camera is broken or even more than one?

PLAN B, C, etc.. If the computer vision fails which are the additional plans and methods to determine and still predict the ability of self driving.

>System is based on predictions according some Neuroscientists like Anil Seth suggests that experience of self is based on **predictions** *intraceptive, proprioceptive and extraceptive*.

How to use different information in time and space which drives certain decisions (for example a signal to turn left is 400m before that the crossroad in which decision must be solved).

>Which is the purpose of a Massive system of mapping territories which still are not mapped in G.Maps / OSV etc.

>Self Driving cars dedicates massive amount of engineering and cutting edge/state of the art techs. Therefore as a human breakthrough may be is interesting as a challenge, **but finally which is the objective? Why we have to abbandon the idea of drive as humans?** Can you imagine everything it must be driven like bikes, surfboards or others can be delegated into complex machinery?

The idea of self driving cars suggests that it can be deployed a carsharing/taxi-like service in which there is no driver. Therefore from a PostCapitalist-Liberal POV a business model where there are no labour and rights issues.

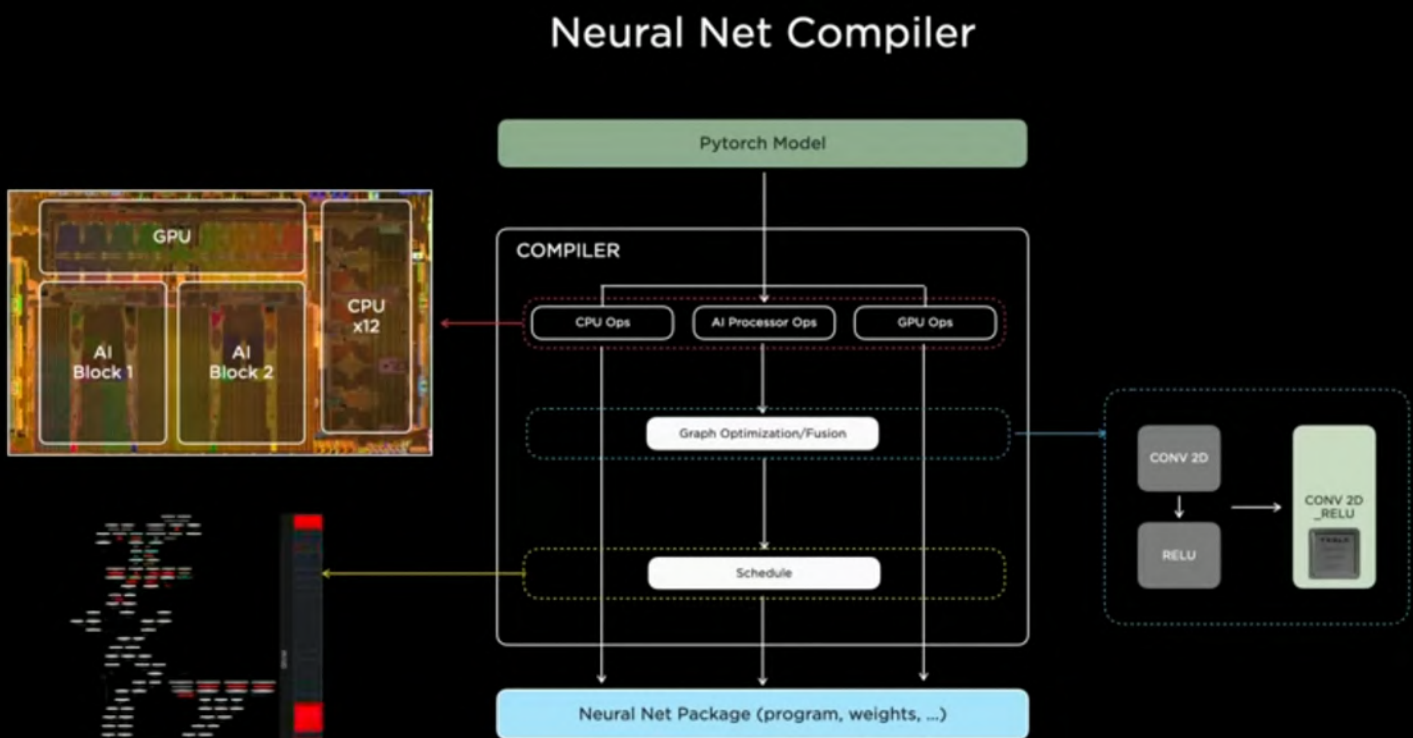
>**Trust** : How we can trust in machinery decisions? In the opening video according AICars still are in development, we can see a driver which slightly supervises the automated turns of the steering wheel. This suggests that even the system is fully developed to drive alone, still there are the temptations to humanly supervise the automated decisions. Can this go to another problem that is : we have a self driving car in which we have to pay a lot of attention not to drive, but to assist the correct automated decisions. Therefore we still have a fully occupation of senses meanwhile we are inside the car.

>**Propioceptive and exoceptive** System of surround is based on human decisions. Therefore how it is possible to predict the uncertain and 'chaotic' behaviur that humans do. You can only watch how a roundabout in several countries works pretty much like self-emergence-almost-chaotic system, like we find for instance in Napoli, Casablanca or Cairo.

> **Gender GAP** : There is no woman in the experts pannel.

Again the patriarchy passes over us like idiots.

> Frame ratio Data Anaylisis in Automated Cars is about **27ms** of computation between each frame. Since **Neural Action Potentials in human brain is processing at 1ms**, hot to solve this gap of analysis in computation.







1. Pd.Tutorial Essentials

Tutorial by Xavi Manzanares

<http://xavimanzanares.oneshaptiques.space>

by-sa // 2021

1. Pd.Tutorial Essentials

LICH requirements >

Pd Vanilla, in other words the original kernel of Pd without external libraries developed by the community

Downloads > <https://puredata.info/downloads/pure-data>

Note : In order to follow the next instructions, download this tutorial in .pd format

This pdf will be anyways useful to read whenever you don't have your computer to practice.

0. Before Programming

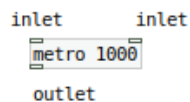
Before programming is useful to know some basic issues in Pd environment.

As you may know, Pd is an open sourced graphic programming language which controls the DSP in a dataflow of structures that you can build.

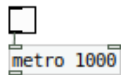
How this dataflow works?

Generally speaking, data flow works from top to bottom, and from Right to Left in the GUI.

So Programming elements may have inlets at the top of them and outlets at the bottom.



Left inlet activates a particular function [object] in this case 'metro' that is a metronome or clock function

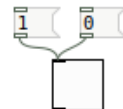


We can both init this function with [bang] or [toggle] elements we find in Put Menu

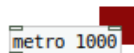
With [bang] once is started the object will be activated until we close the patch or programming GUI surface

With [toggle] we can start and stop the function [object] any moment we need

Binary Messages can be linked to a Toggle to activate it



Right inlet changes the argument (or value) described in the function, in this case 1000



note that arguments or values does not have a specific units, but it directly applies to a specific object.

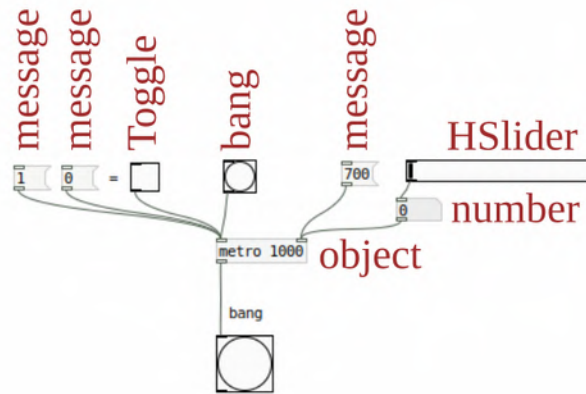
for example if we find metro 1000, the argument it means 1000 milliseconds. If the object is osc~ 1000, the argument means 1000 hertz



outlet in the case of [metro] is a trigger which in Pd is the element 'bang'



Like other languages we can build structures with different methods and solutions.
 In this example, there are different ways to activate the function metronome (object **[metro]**) :
 Through messages **[1]** and **[0]**
 Through a **Toggle** that in fact is an On / Off switch
 Through a **bang** or a trigger



On the other side, on the right inlet we are introducing a new value for a particular argument.
 This can be a fixed value with **message**, or a dynamic value with **HSlider** or **number**
 Remember that even could be a default written value in the object (like 1000 in this example), the last value we are dynamically modifying in live (for example through the associated Hslider or number), will be defined as the lead and 'definitive' one.

There is another issue important to take it into account before programming :
 In Pd we have **data connections** and **signal connections**.

Data connections sends numbers and alphanumeric messages, at the speed of your CPU can manage and delay (latency) that has to be defined in your Pd preferences (depending on the OS can be found in the menu **Edit > preferences** or in the menu **Media > AudioSettings**).

Signal connections sends / receive signals at 44100hz (so 44100 dots/samples every second) or whatever samplerate you already set up in the preferences of Pd.

*Data connections are **thin 'cables'***

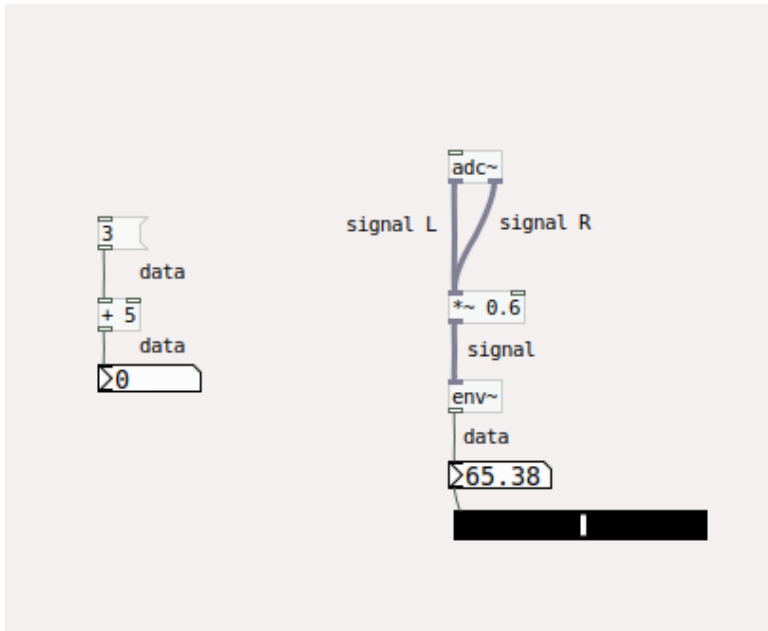
*Signal connections are **wide 'cables'***

*Any object or message for **Data connections** is written without tilde ~*

*Any object or message for **Signal connections** is written with tilde ~*

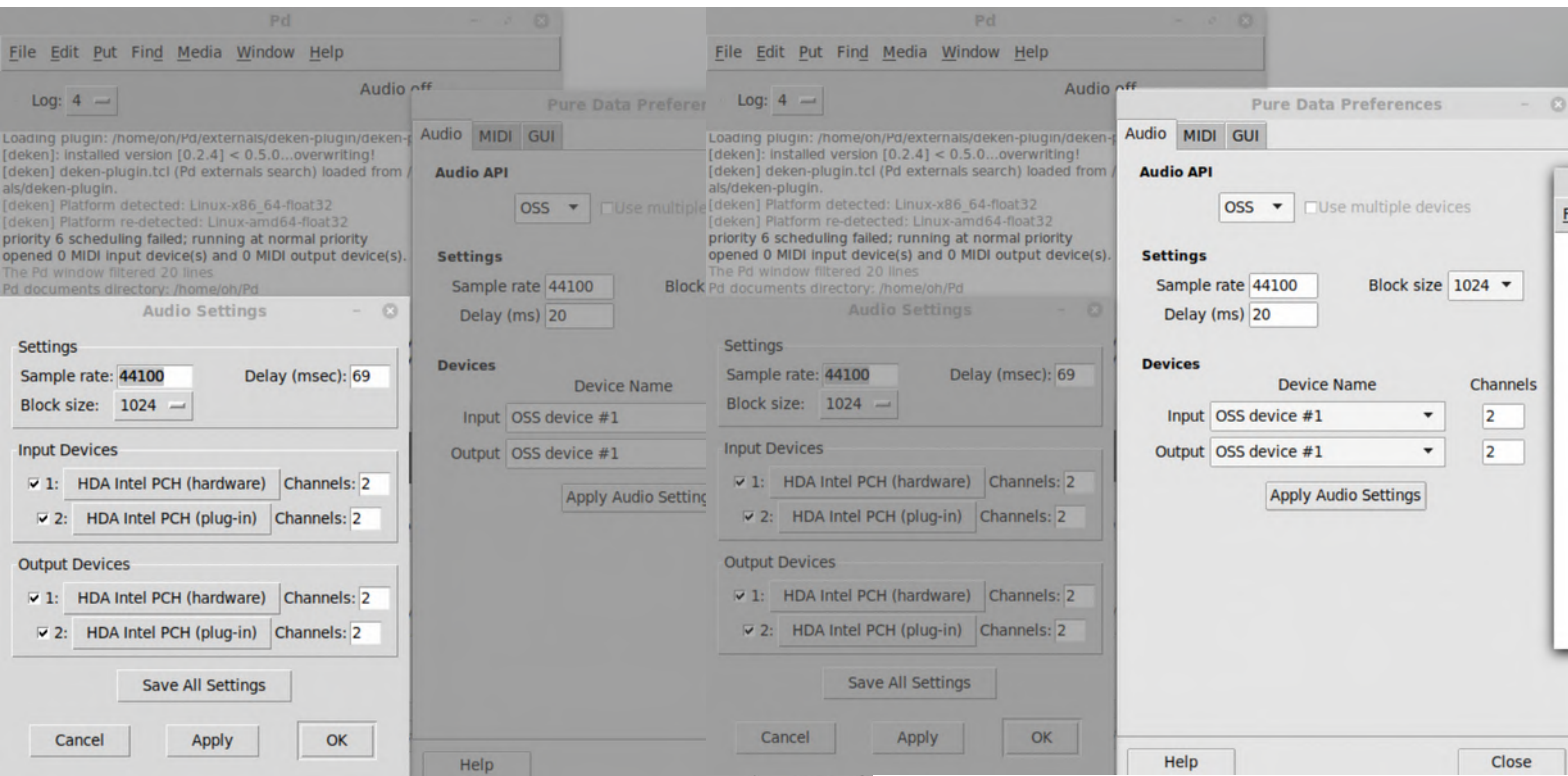
For example a multiplier of **data** is **[* 2]** and a multiplier of **signal** is **[*~ 2]**

data vs signal



In **preferences** you can manage different audio devices like internal /external soundcards or even working with several soundcards.

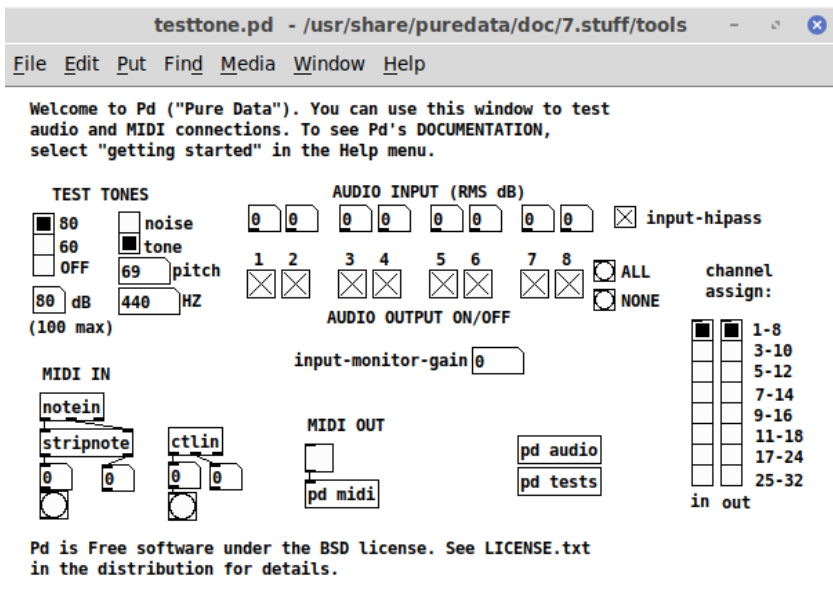
These GUI interfaces can change depending on the OS you're using.



In **preferences** also you can change **samplerate** or **latency (Delay)**.

Latency (Delay) can produce errors for short values, depending on the size / calculus of your algorithms and the power of your computer processor. So it's a value that we can tweak in order to work DSP fine and without glitches. But how we know it?

The first and basic test to check if pd is working fine is **Test audio and Midi** in the menu Media :



We can select Test Tones button on the left side, and select 80db. **If pd is working fine a sine tone of 440hz will be output in your soundsystem or headphones.**

When we create a new file with Ctrl+n it appears a blank GUI surface where you can build your applications. This surface is called **Patch** like when we create a set of concrete connections in the Modular Synths Cosmos.

Finally, remember that Pd is a dataflow Programming environment where we can build DSP applications, but also manage and play with them as a performance/musician mode.

Maybe this is the command you'll be using the most if you program with Pd > **Ctrl+E**
Ctrl+E will switch between these two modes :

> **Edit / Programming Mode**

You can move and write whatever you want in the patch

> **Performance / Musician Mode**


You can just only move the sliders and dynamic elements of the Code as a performer

syntax


1.1. Pd syntax

Even we can find several menus in the Pd shell or in any New patch from scratch (**File / Edit / Put / Windows / Media / Help**), there is one menu that is related to the programming language elements, and therefore the most important one, which is the menu Put.
In **Put** we'll find different elements of programming :



> We can find [Object] that makes a concrete function for example [metro] wich runs a metronome.


Object
 Those functions are described by the name of the function and an argument or value that can be written in a fixed position or dynamic values through the right inlet

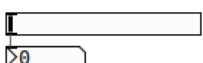
> We can find Message that is an alphanumeric instruction connected to an specific function or [object]


Message Message Message
  


> We can find Numeric values that are managed in different GUI types, but essentially are the same :

 Number
 Number2 This element allows receive or send messages and values from other code sections. Making right button > properties > messages > send-symbol or receive-symbol

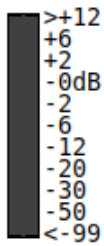
 VSlider > Vertical slider which defaultly goes from 0 to 127
We can write a new range of values for example from 0 to 1 making right button > properties > output-range
This element allows receive or send messages and values from other code sections. Making right button > properties > messages > send-symbol or receive-symbol

 HSlider > Exactly the same as Vslider but in Horizontal position
This element allows receive or send messages and values from other code sections. Making right button > properties > messages > send-symbol or receive-symbol

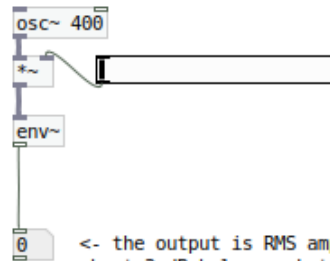
 VRadio > Vertical Radio Button which goes from 0 to the number of steps described in the element (in the default example from 0 to 7)
This element allows to receive or send messages and values from other code sections. Making right button > properties > messages > send-symbol or receive-symbol

 HRadio > Exactly the same as VRadio but in Horizontal position
This element allows to receive or send messages and values from other code sections. Making right button > properties > messages > send-symbol or receive-symbol

> VU is a vumeter which can visualize dB from incoming RMS signal



VU is an element not enough straight forward to monitor amplitude values. Instead of this, you can check it with [env~] which features the envelope value



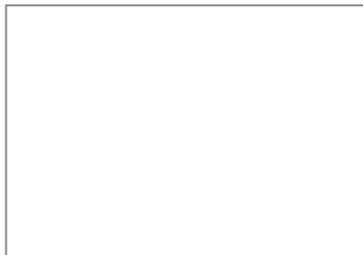
<- the output is RMS amplitude which (for a sinusoid) is about 3 dB below peak-to-peak amplitude.

> Canvas is a GUI element which features design zonification of the code.

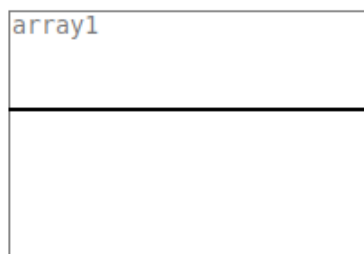


This element allows to receive or send messages and values from other code sections. Making right button > properties > messages > send-symbol or receive-symbol

> Graph allows to manage different code sections as a 'backend' swithcing 'graph on parent' after making right button > properties of this element. It will be useful to save space in the main GUI



> Array is a memory of numeric values, very useful for different algorithms we want to program. This element will be explained later due to the fact of its versality and strenght.



syntax

1.2. Pd Programming Dataflow

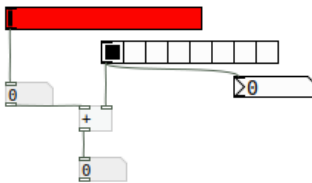
data.flow

PD Pure-data is a Dataflow visual programming language

Flow is from up to bottom and from right to left

Is important to rearrange different algorithms which can be automatized in the flow

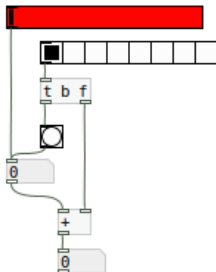
For example in this piece of code, any time we move HRadio button we need to 'refresh' the value of the red slider



if we want to automatize it, we can put the Red slider and the Hradio associated with the [t b f] object

[t b f] object describes 't' trigger 'b' bang and 'f' float

Therefore any time we move or the slider or the HRadio button the operation will be directly updated



sends&receives

Messages and values can be send and received anywhere in a patch, building a rhizomatic structure with the only one hierarchy that have been constructed.



r anyvalue



The rhizomatic structure of the data flow it has no hierarchy, except the order that any code element has been constructed.

Basic algorithms

2.1. Pd Basic Algorithms

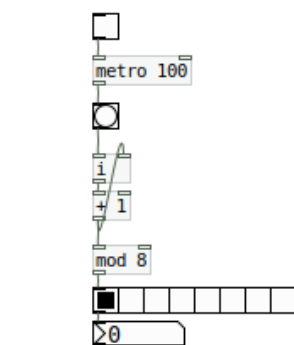
In the next examples we'll see a bunch of tiny algorithms and sections of code which can be useful for many developments.

loops

Like other programming languages, one of the operations we are using to build structures in time are **loops**. Even other programming languages this feature is essential for reading the whole code, in Pd we can make several groups of loops that are running apart from each other. Therefore we can build several groups of loops to make different operations at the same time.

Here there is an example of how to build a loop of 8 steps (from 0 to 7). In this case the steps are changing to a fixed speed of 100ms, driven by the object metro.

loops



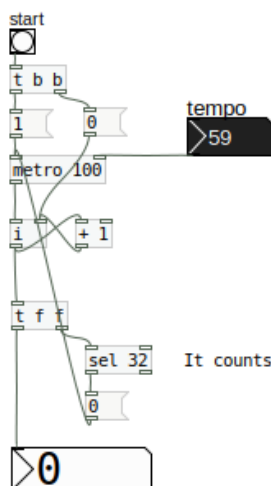
mod 8 It counts 8 values (from 0 to 7) every 100ms

Counters

A counter builds a sequence of numbers until to a certain value.

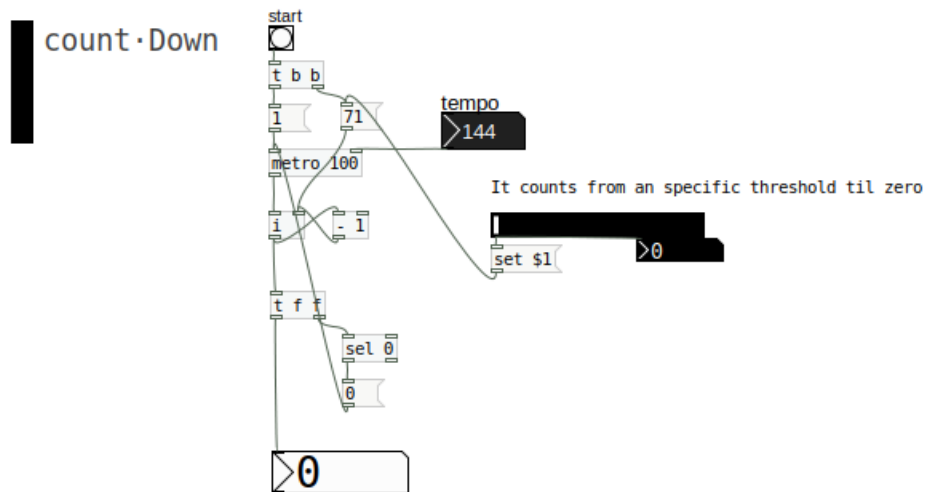
If we want to make a **count-up** sequence we can use this piece of code:

count-up



It counts from 0 to a threshold in this case 32

On the contrary, if we want to make a **count-down** sequence we can use this piece of code:



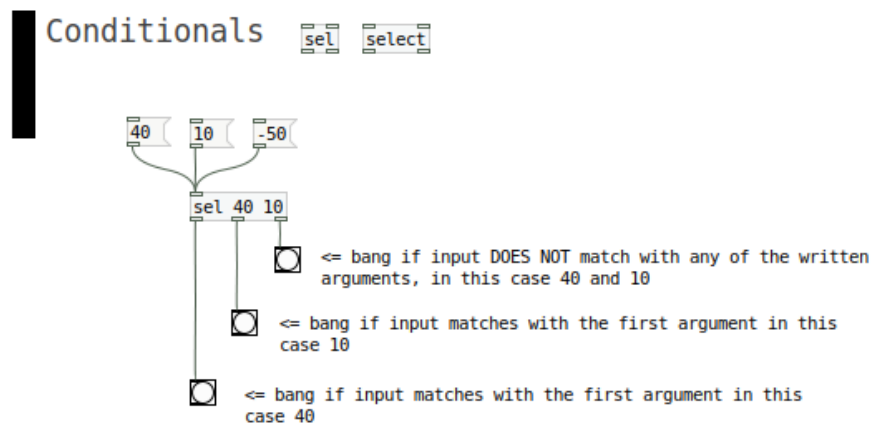
Conditionals

Conditionals are the classic if, else etc. functions from another languages.

In Pd we can solve conditional dataflow with the object **[sel]** or **[select]** which solves both the *if* condition and *else* condition.

The object **[sel]** works with a *string* of numbers or alphanumeric values, where each value has an specific outlet, ordered from left to right.

Notice that in this example we have 2 arguments (40 and 10), but anyways we have 3 outlets in the object . That's exactly the reason that the first outlets corresponds to an *if* function, and the right side outlet works like an *else* function.



There are other functions [objects] to manage conditional structures, for example :

[route] > distributor > similar to **[sel]** but a bit more complex and not so straight forward

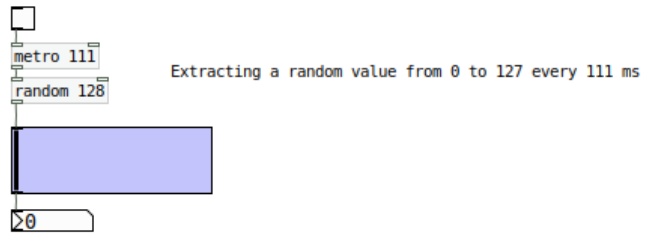
[spigot] > gate / door > allow continuity or not from the incoming dataflow

[moses] > splitter > splits numeric values into two data lines from an specific argument ex. [moses 34]

Random

Maybe one of the most exciting feature in Computer Science is the **random** function. Of course is a very useful tool if you want to make **generative instruments**, but remember that even we adore randomness, if we overuse this function, this could not solve our expectations in the sonic design.

Random

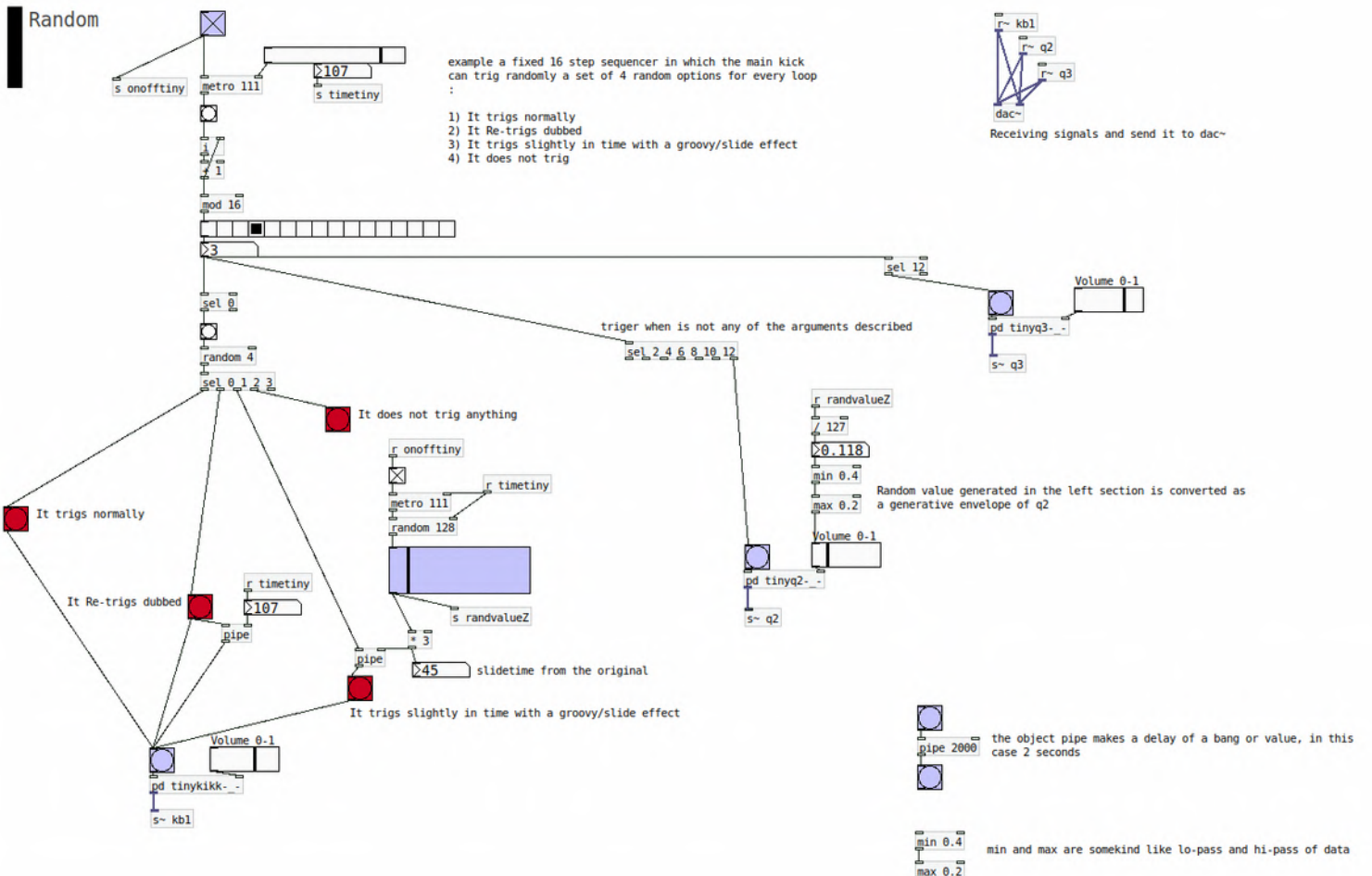


One suggestion is to use random methods in a very defined set of options.

For example we can build a fixed 16 step sequencer in which the main kick can trig randomly a set of 4 random options for every loop :

- 1) It trigs normally
- 2) It Re-trigs dubbed
- 3) It trigs slightly in time with a groovy/slide effect
- 4) It does not trig

In other words, for this example using randomness in order to get more or less static/patterned structures and organicity/changes at the same time.

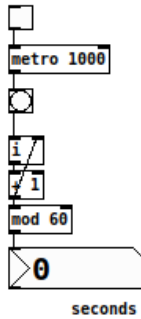


Real Time Clock

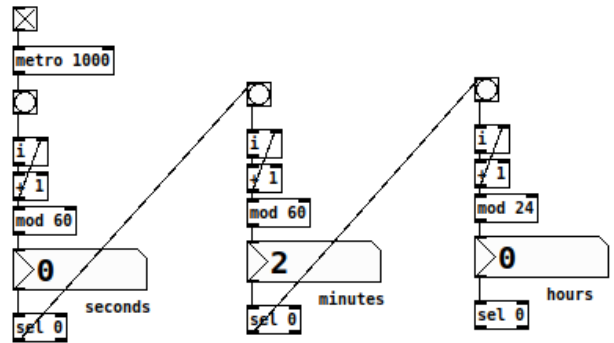
Maybe we need to use a real time clock (seconds, minutes etc) for generative instruments we want to make changes in time, for example an Installation in which we want to take it into account what time is it, or changing parameters depending if we are on the morning, afternoon etc.

With a set of conditionals [sel] we already seen we can trigger some events for specific daytime.

RealTime.Seconds



RealTime.Seconds



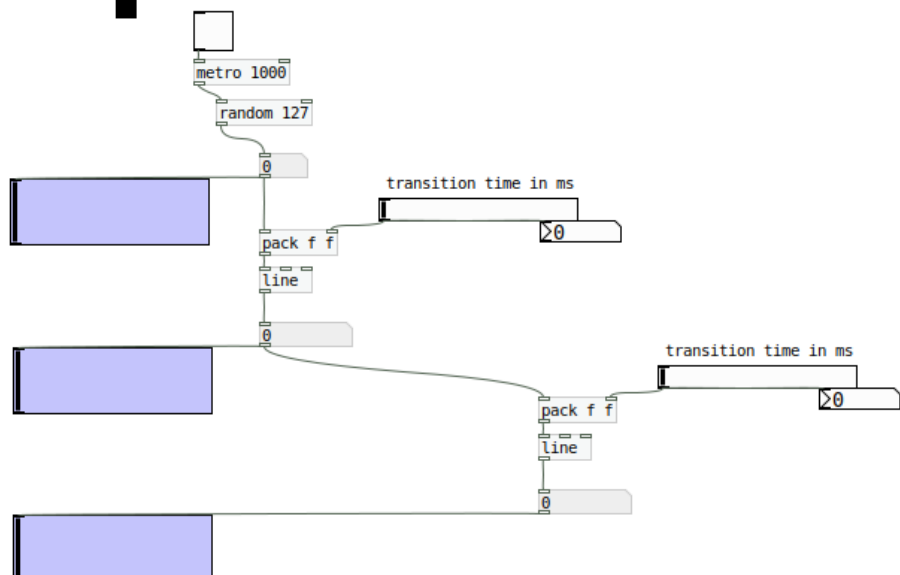
Line : Ramps and

Line is the object which makes transitions from two points.

It is useful to make several actions in the on going sound like fade ins/outs, portamentos, smooth binaural effects, among another sliding effects.

In this example a group of lines are making different transitions recoursevely from an initial changing parameter (random of 128 values every 1000ms). The 'monitoring' lilac-blue sliders are conceptually doing the same but with different sliding effect in time. Therefore the slider at the bottom behaves much more 'organic' or physical-modelled. Those sliding values we can apply to other programming parameters, like amplitudes, amounts of filters, etc.

Fades.And.Transitions



Basic algorithms

2.2. Pd Packaging Algorithms

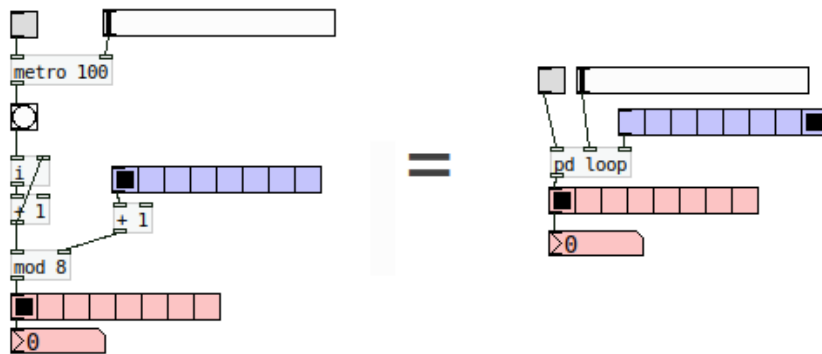
One thing that usually happens is that we are programming with a lot of pieces of code that easily fills up the GUI. One useful method to save space and code elements we are not playing as performers, is to hide pieces of code in structures that only sends/receives the dynamic parameters we want to change.

For do that, we have several options:

Sub Patch

In this example we can see two pieces of code that are exactly the same. Is it easy to see that the right solution saves a little bit of GUI space.

packaging.algoryhtms.with.subpatches



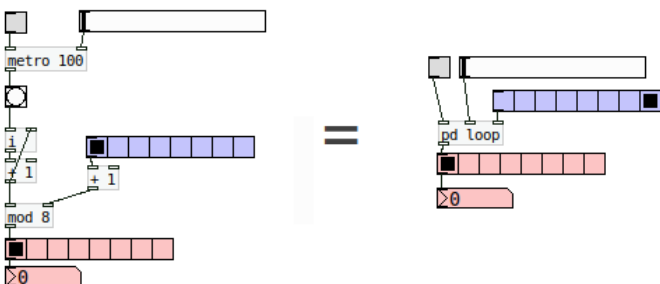
We can make a sub-Patch with the Object [pd whatever]

Is it a very useful feature whenever the original algorithm is big in the GUI

In this case we are using the subpatching method, with and object writing into : 'pd' and the name you like, in the example **[pd loop]**. This creates a blank sub-patch, where we can paste the desired code and connected to the main GUI through inlets and outlets.

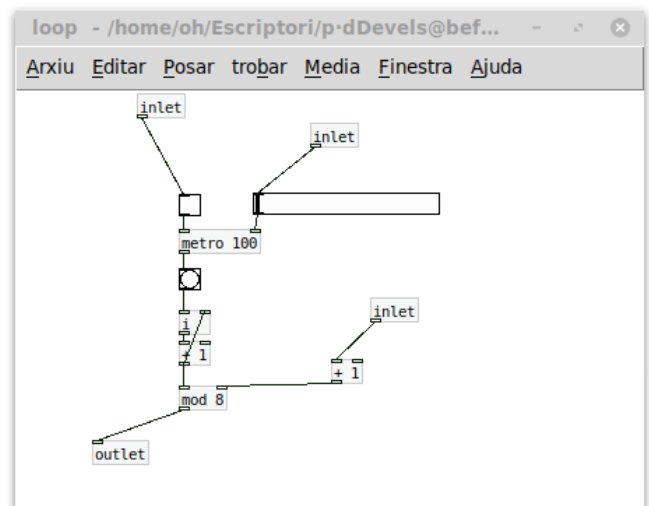
The order of *inlets* and *outlets* that appears in **[pd loop]** at the main GUI, corresponds to the different *inlets* and *outlets* we have written from left to right inside the subpatch.

packaging.algoryhtms.with.subpatches



We can make a sub-Patch with the Object [pd whatever]

Is it a very useful feature whenever the original algorithm is big in the GUI

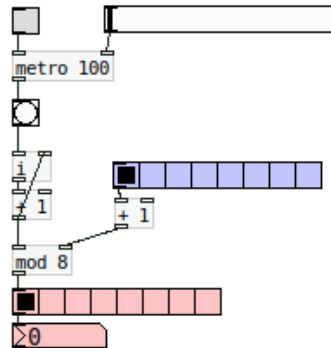


Graph

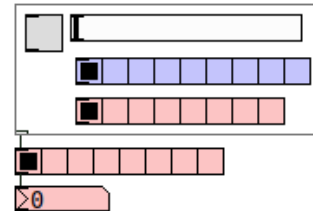
Another method to pack sections of code is with 'Graph' function > menu put > graph

As the previous method, it creates a new patch where we can 'make like a kind of hole' in it, allowing us to see the elements we want to manage.

packaging.algoryhtms.with.graphs



=

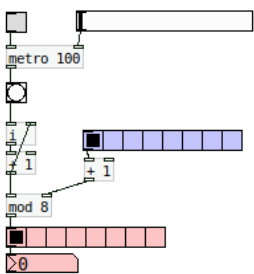


Create Graph > menu Put > Graph

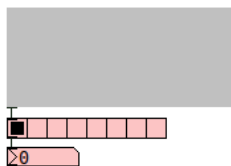
Right Button > open to edit and/or write the code

Similarly as the previous technique, the order of *inlets* and *outlets* appearing in the main GUI's Graph corresponds to the different *inlets* and *outlets* we have done **from left to right** inside the Graph

packaging.algoryhtms.with.graphs

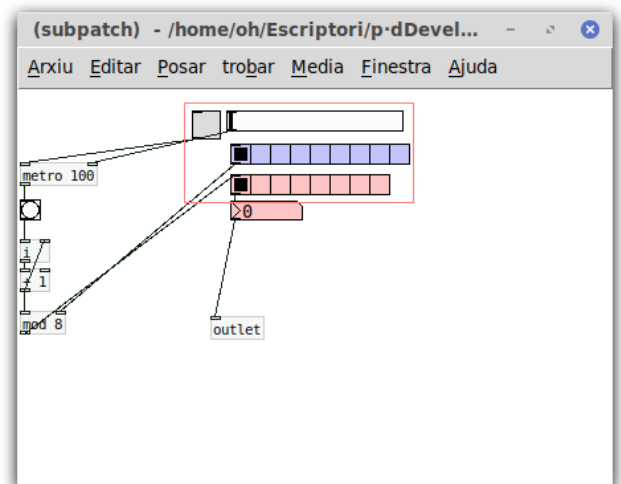


=



Create Graph > menu Put > Graph

Right Button > open to edit and/or write the code



Both methods (Subpatch & Graph) are similar, but maybe the second one helps to visually structure a main GUI for big Patches.

Anyways the SubPatch technique allows us to make complexe groups of sections of code, that we can keep in the 'backend' calling the functions or parameters we want to control, via **sends** and **receives**.

Sends [s whateverdata] [s~ whateversignal] and **Receives** [r whateverdata] [r~ whateversignal] are somekind of 'wireless' connections, does not matter how deep you are (imagine you wanna send a message from the main GUI, to a subpatch which is into another subpatch into other subpatch of the main GUI etc.)

Basic algorithms

2.3. Pd Arrays (Buffers or Memories)

Arrays [menú put > arrays] are useful elements in Pd.
In fact are memory storages or buffers, where we can keep both data (numbers) and signal.
Let's start with data storages and its transformations.

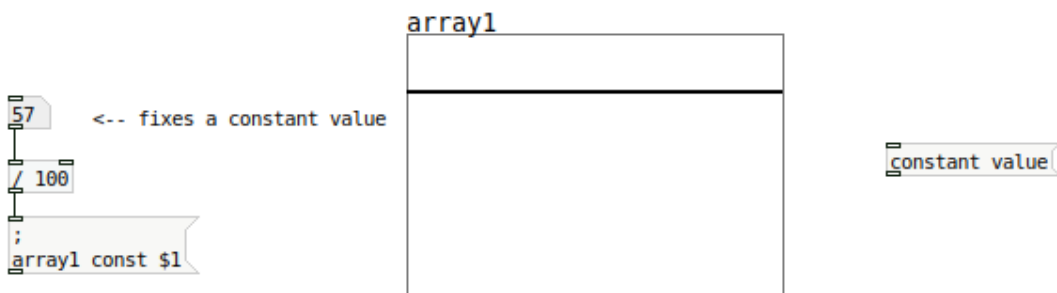
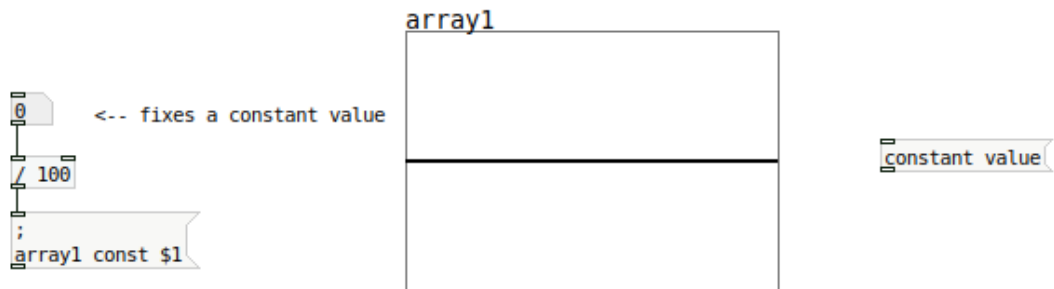
Array's transformations are usually done by specific messages, with the next conceptual syntax :

[; arrayname transformation value [

Arrays > Constant

We can use arrays to store constant values in time.
Even is not so often used, this is the syntax :

[; arrayname const value [



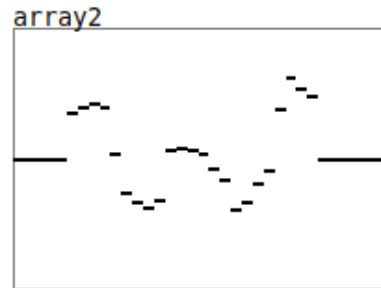
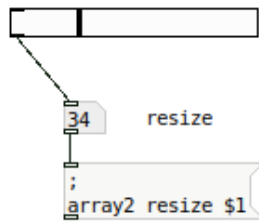
Notice that value after *const* is written with a \$1.
This means that can be a **variable**, therefore a numeric value which dynamically can be changed.

Arrays > Resize

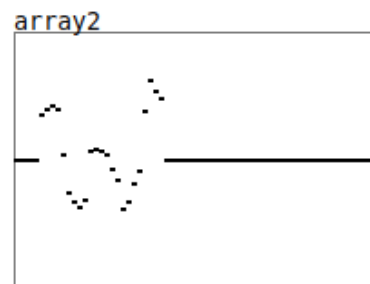
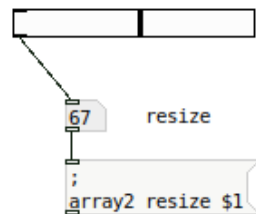
We can resize arrays in time.

This is the syntax :

```
[; arrayname resize value [
```



[; resize array or memory



[; resize array or memory

Arrays > Trigonometry functions

We can make data buffers with trigonometric structures, like sines and cosines

These are examples of the syntax :

```
[; arrayname sinesum arraysize string.of.values [
```

```
[; arrayname cosinesum arraysize string.of.values [
```

The array size has to be power of 2 > 32, 64, 128, 256, etc

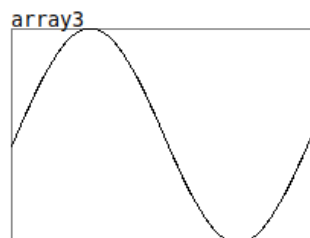
The string of values you can try with different combinations between 0 and 1 > 0, 0.1, 0.3, 0.2, 0.1, 0.4, etc..

The more large is the string, the more curvatures has the generated wave.

For instance if we want to make a basic sinewave

```
[; array3 sinesum 64 1 [
```

```
[; array3 sinesum 64 1 [
```

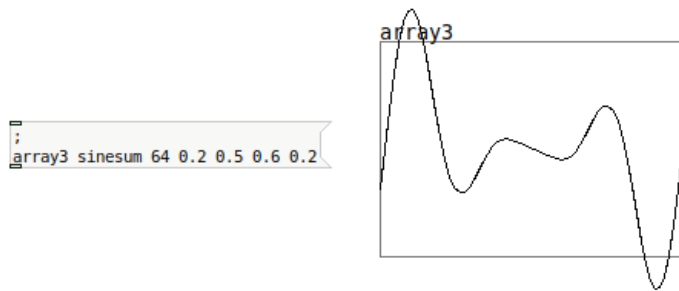


[; trigonometric shapes

These functions are useful in filtering signal like WaveShaping techniques

Now a sinewave a bit more 'woobly' :

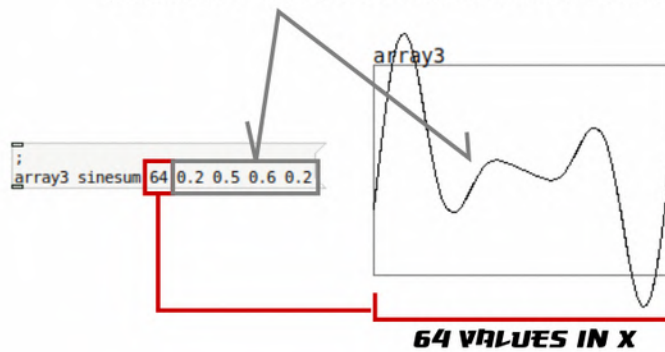
Remind that the more large is the string of values, the more curvatures has the generated wave.



trigonometric shapes

These functions are useful in filtering signal like WaveShaping techniques

**THE MORE LONG IS THIS STRING OF VALUES
(BETWEEN 0 AND 1), THE MORE WOOLLY IS THE WAVE**

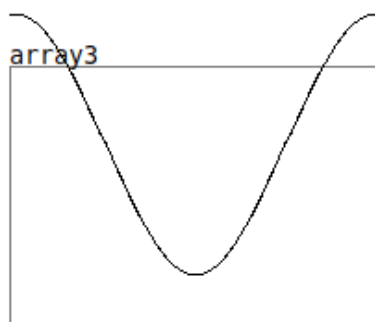


trigonometric shapes

These functions are useful in filtering signal like WaveShaping techniques

With cosines is similar

```
[; arrayname cosinesum arraysize string.of.values [
```



trigonometric shapes

These functions are useful in filtering signal like WaveShaping techniques

```
[; array3 cosinesum 64 0.4 1
```

Arrays > Normalize

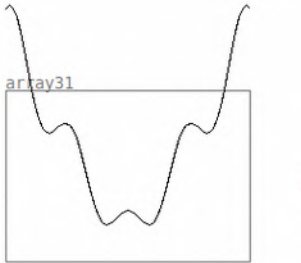
Depending on which use we want for the buffer, maybe we need to limit the ranges.

Working with sines and cosines the values on the Y axis goes from -1 til 1, but sometimes shapes can overpass these thresholds.

There is another instruction with arrays that limits this ranges, which is 'normalize'

[; arrayname normalize value [

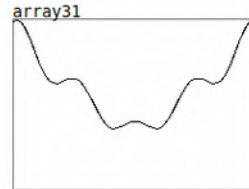
If we write *normalize 1*, shape will be fitted in -1 / 1 ranges.



trigonometric shapes

These functions are useful in filtering signal like WaveShaping techniques

```
[; array31 cosinesum 64 0.4 1 0.1 0.2 0.3
```



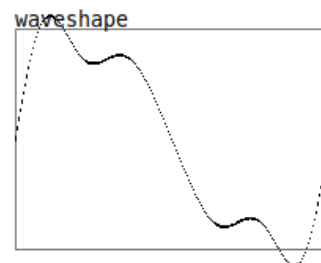
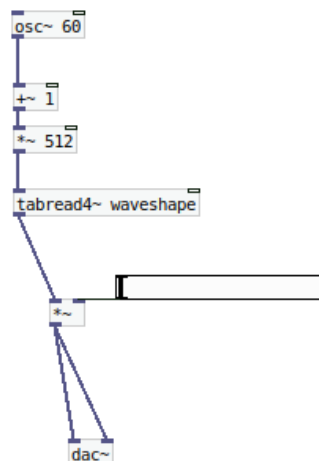
trigonometric shapes

These functions are useful in filtering signal like WaveShaping techniques

```
[; array31 cosinesum 64 0.4 1 0.1 0.2 0.3
```

```
[; array31 normalize 0.99
```

Trigonometric shapes are very useful if we want to make **WaveShaper Filters**, for example this one.



```
[; waveshape sinesum 1024 1 0.2 0.3 0.1
```

Anyways, due to the fact that it's a nice and expressive technique of signal filtering, we'll see different methods and tricks of it later.

Arrays > .txt Reading and Writing TXT files

A very interesting feature with Arrays is to 'import' or grab the contents of a text file.

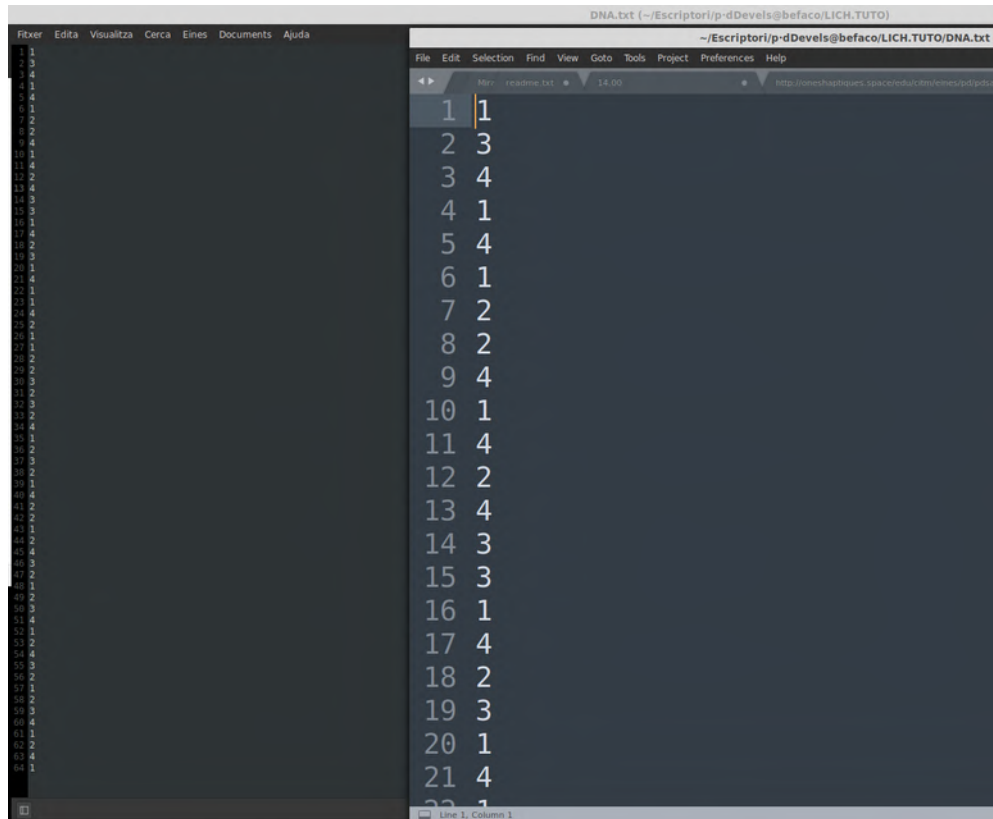
In fact this is an interesting feature for **sonification projects**.

In this sense, in order to Data be imported correctly by Arrays, text files has to be formatted with a value per line of code.

Values has to be ALWAYS numbers, not alphanumeric structures or others, just one number for each line.

For example, this .txt file with different values between 1 and 4.

note : the image features a dual zoom of the same file.

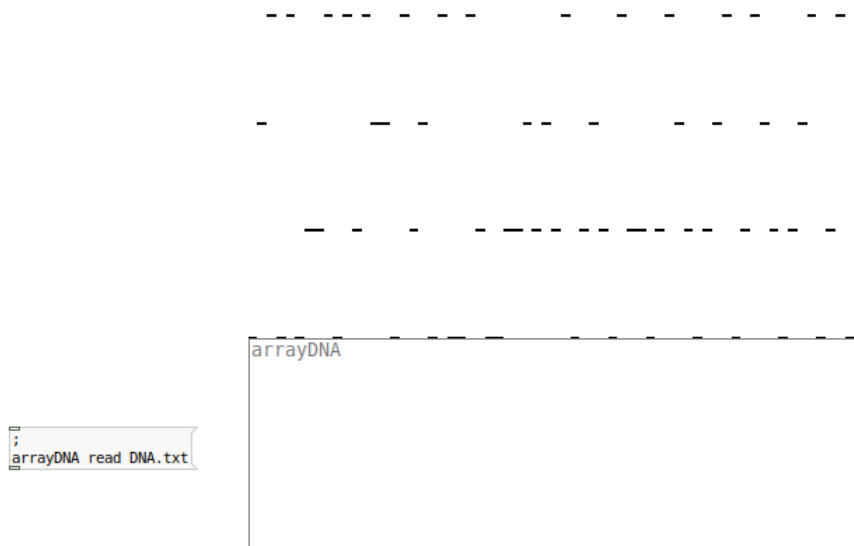


If we save this .txt file in the same directory level as the patch we are working in, with this message,

[; arrayname read ourfilein.txt [

and bang it on it or just clicking on it :

The contents of the file will be directly transported to the array :

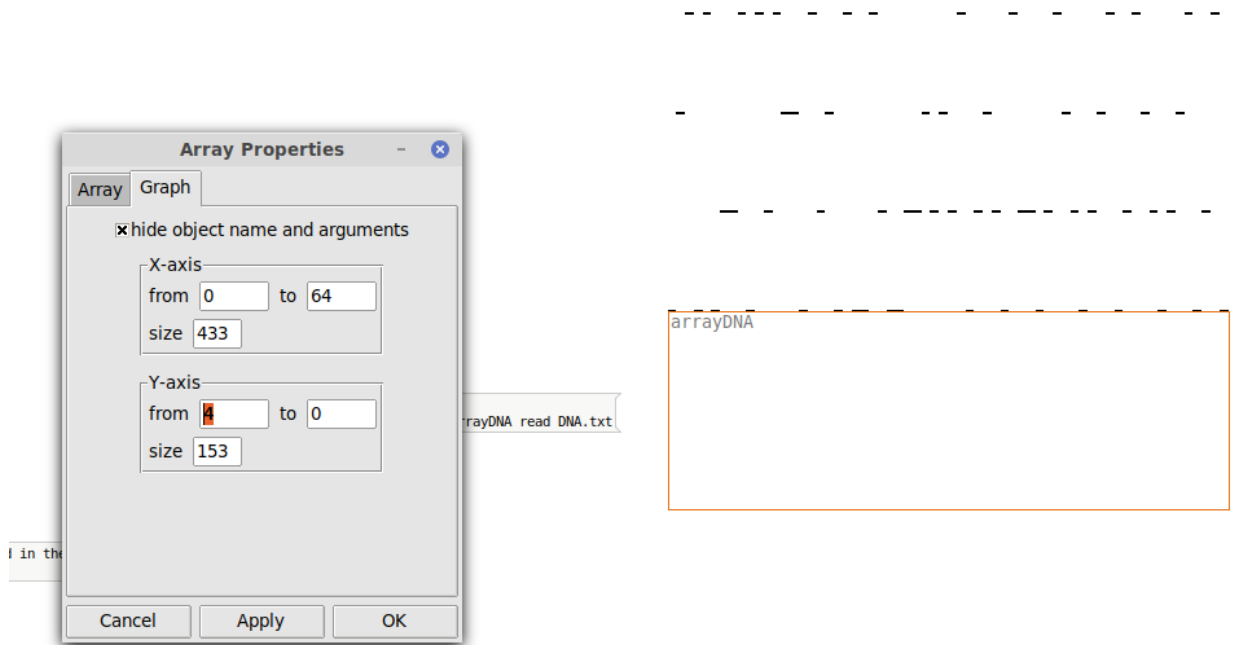


In the previous image, noticed that values of the array has overloaded the frame of the array. That's because, by default Arrays generates a score with values from -1 til 1.

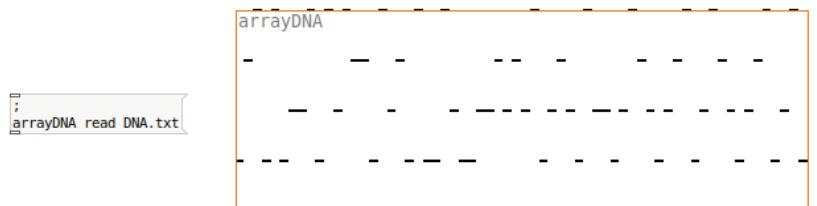
Due to the fact we have values in the .txt (and therefore in the array) from 1 to 4 it features an overloaded representation.

How can we rearrange it?

If we make right button over the array and click properties, we'll see a pop up menú where we can change different parameters : size of the Array in samples, Size of the Array in pixels (X & Y), range of values in Y axis, size of the Array itself amongst other features of visualization.



Once set up values from 0 to 4, Array is rendered like this :

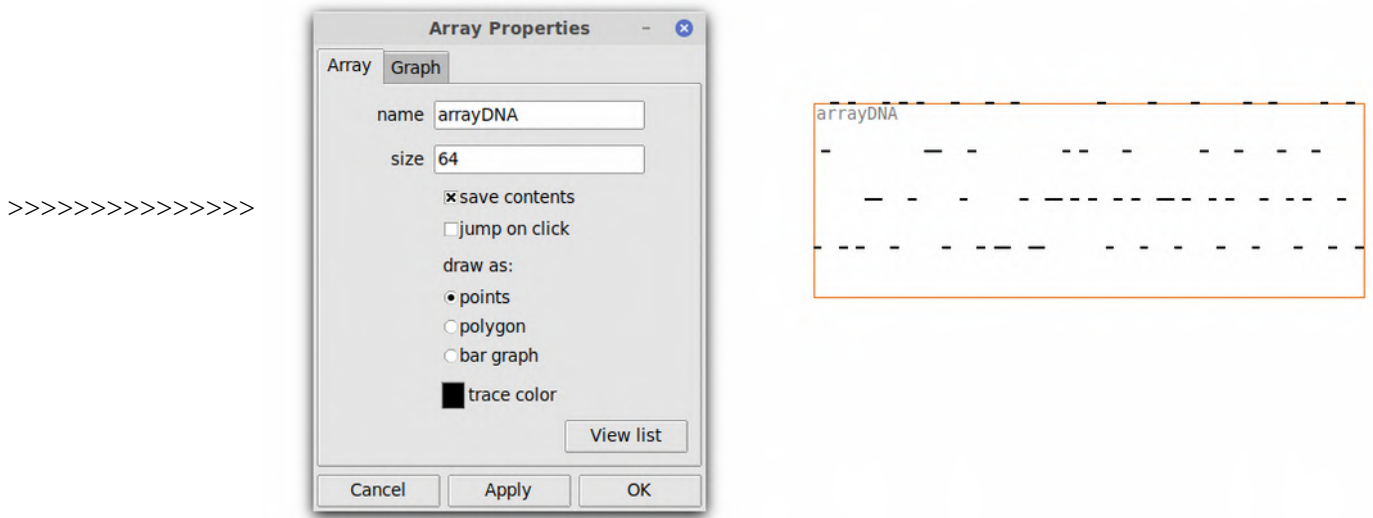


Also another trick and very interesting Array's feature is to **save the content**.

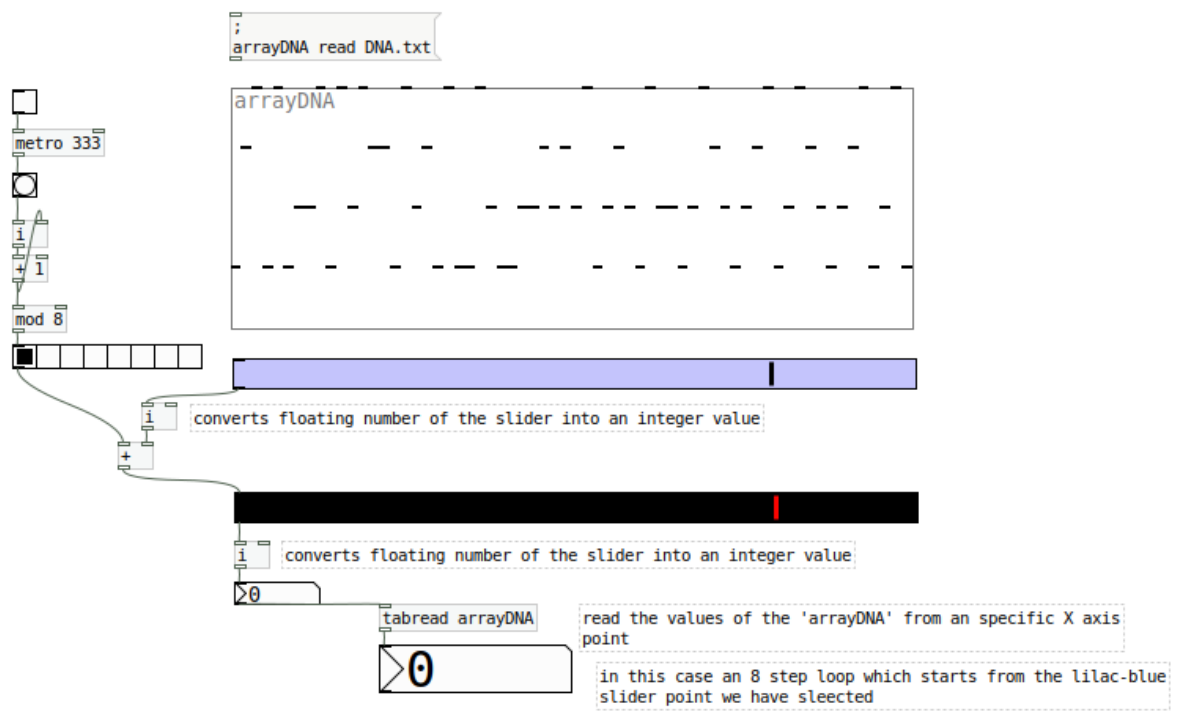
Imagine in an array we have a sequenced bassline midi notes. Every time we open the patch, the contents will be erased and we have to bang it to a certain message to loading it again, unless we have already set up 'save contents' in Array Properties.

If we do that, every time we open the patch the content of this Array or memory will be kept like a list of values embed in the array and therefore in the patch (file.pd) we are working in.

This method works both for *data* storages and *signal* storages within the arrays.



Playing around with arrays and .txt files allows us to build **sequencers** in a pretty easy way, for example :



Anyways we'll see later in the sequencers chapter of this tutorial.

Sampling

3. Pd Sampling

After we have seen different array methods, let's remind that Arrays we can use it for storing numbers, but also for storing signal. (In fact the second one is also a data storing but it has some specific details to take it into account, to manage it as a signal).

If you are a musician, you'll already know what **sampling** is, but as a short reminder, is a technique which uses audio fragments to play and process them.

Notice that in Pd and in DSP techs in general, sample is not only the concept of storing audio signal in buffers, but also the 'pixels' of information which describes a particular stored signal. Therefore for an specific unit of time, the more samples of information we have, the more detailed will be the signal. That's interesting for reproduce preexisting audio files, but for managing signal calculations in real time, sometimes the HD quality render effort is collapsing with the processor speed, and therefore its more efficient to balance and tweak it, in order to get a nicer DSP performance.

Let's see different methods to play an audio file from the computer.

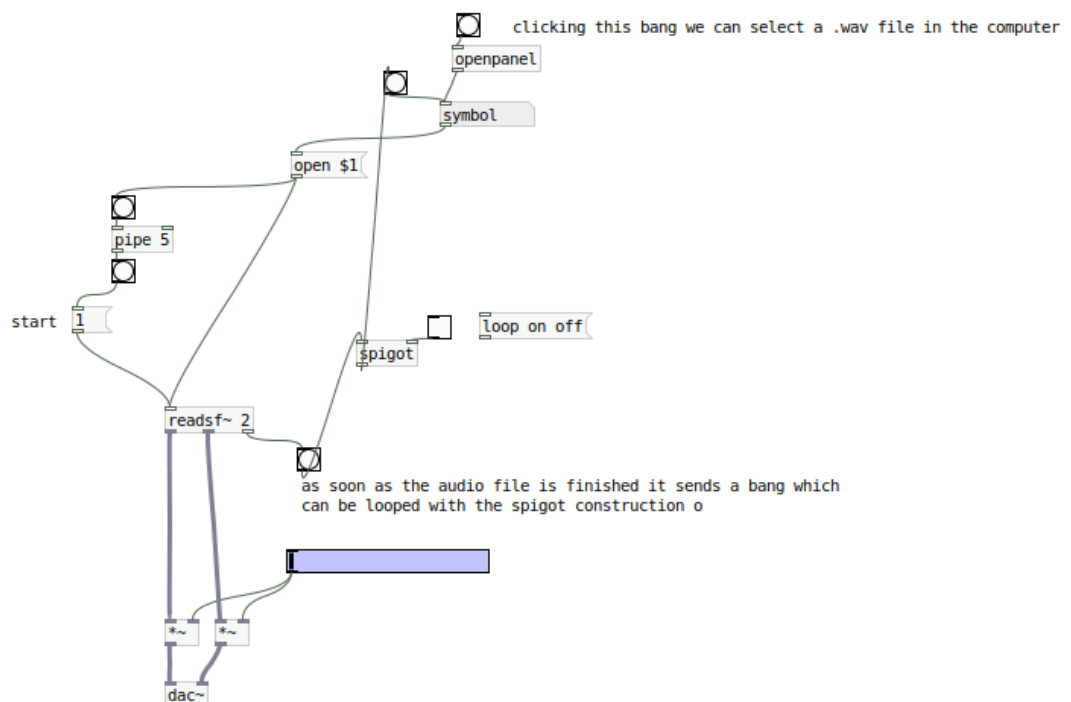
[readsf~]

The first and most simple method is to read the audio file (or in .wav or in .aiff) directly from the disk (therefore without buffer it). This method is required for large* audio files we want to play.

*large meaning audio from 30 seconds until hours.

Sampling.for.Large.Audio.Files

this method reads the file directly from the Drive



[readsf~] as a memotechnique : **readsoundfile**

is an object where we can describe the number of channels we want to play :

for a mono audio file **[readsf~]**

for an stereo file **[readsf~ 2]**

...and so forth until 64 audio separated channels.

But also we can add another argument in the object to define some more specific details like buffer size in bytes per channel for specific purposes [not often used].

Therefore the syntax :

[readsf~ numchannels buffer.size.per.channel.in.bytes]

In the object [readsf~] will be several outlets. The firsts from the left corresponds to the channels that has been described in the object. There is also another outlet on the right bottom which trigs a bang as soon file has ended the whole reproduction. Therefore if audio sample is correctly edited we can make easily a looper.

In the previous example due to the fact we have been using a symbol (menú put > symbol), the first time we loaded the file from the HardDrive the path has been stored in the symbol. So anytime we want to make a loop the sequence of triggers first trigs the path and after 5 ms [pipe 5] trigs the [1] message which starts the play.*

The object spigot is in charge if we want to make this flux like a loop. So spigot is in fact like a gate that is opening / not opening >> flux continuity / not flux continuity

**pipe is a delay of data very useful to scheduling instructions and dataflow.*

Recording : [writsf~]

The opposite function of **[readsf~]** is **[writsf~]** that is an object which can capture or record any signal that is throwing into it.

[osc~ 111]

|

[writsf~]

That's an interesting feature if you want to **record** a performance of your patch within the DSP.

Remind that in order to record some running audio is important to follow an specific order :

1. **write the name** of the file we want to create in a message associated to writsf~

[open /your/path/Desktop/record.wav [

2. **connect whatever signal** you want to record into the writsf's channels you wanna record (L+R if is [writsf~ 2]).

3. If you are ready to record, then **Click** on the message created before **[open /your/path/Desktop/record.wav [**

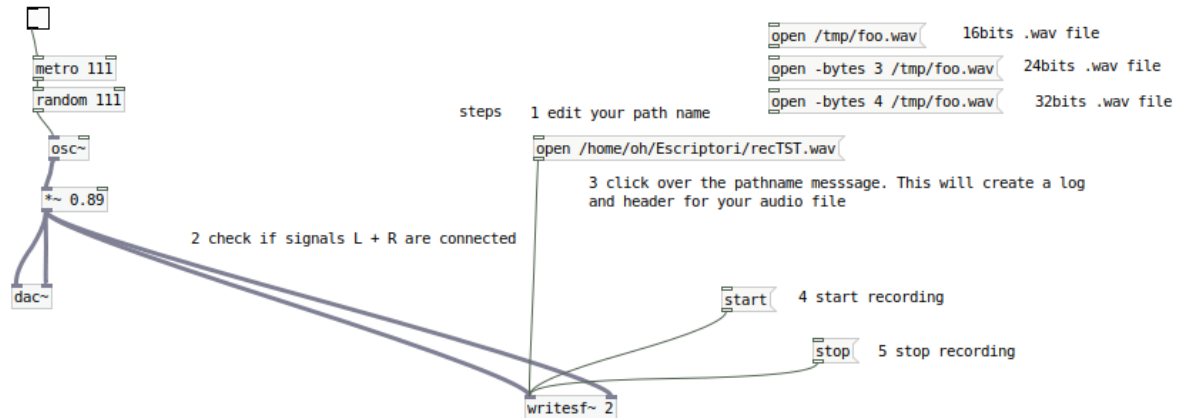
4. click on **[start [** message, to init the recording

5. click on **[stop [** message, to stop the recording

*note : clicking on the message with your path and name.wav it creates a **log** file that iniciates the recording with the instruction **[start [**. Therefore if we do not click on the message, log will not be created and Neither the audio record.*

Recording

recording a signal into an audio file

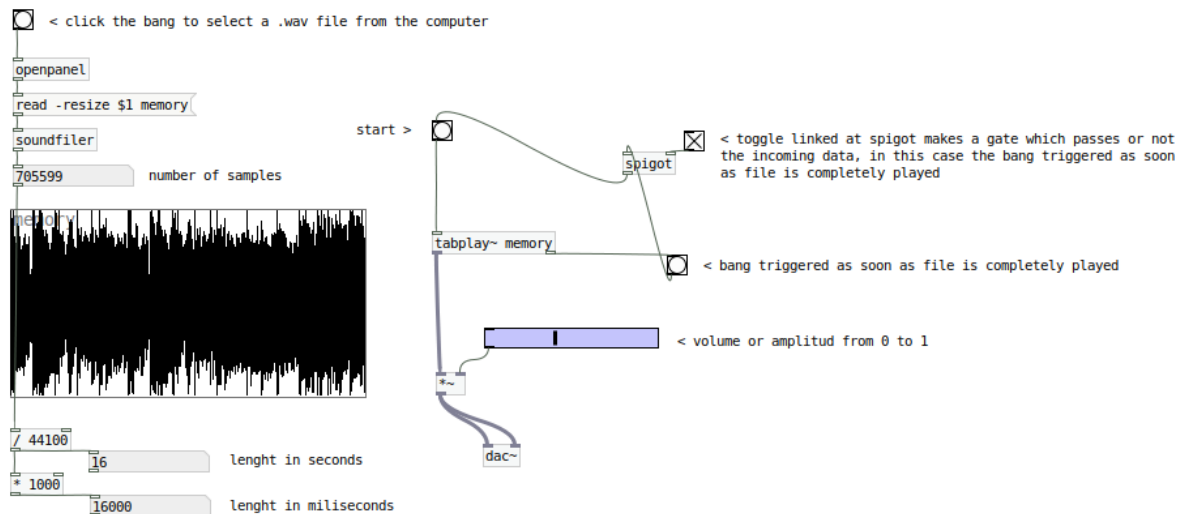


[tabplay~]

maybe the most precise method to sampling in the classic sense is with the object [tabplay~]
First we have to load a .wav or .aiff file into an array or memory that after we can trig, and even loop it.
Notice that in order to adapt an audio file into an array it is necessary the object [soundfiler] and it previous message [read -resize \$1 arrayname]

Sampling for Short Audio Files

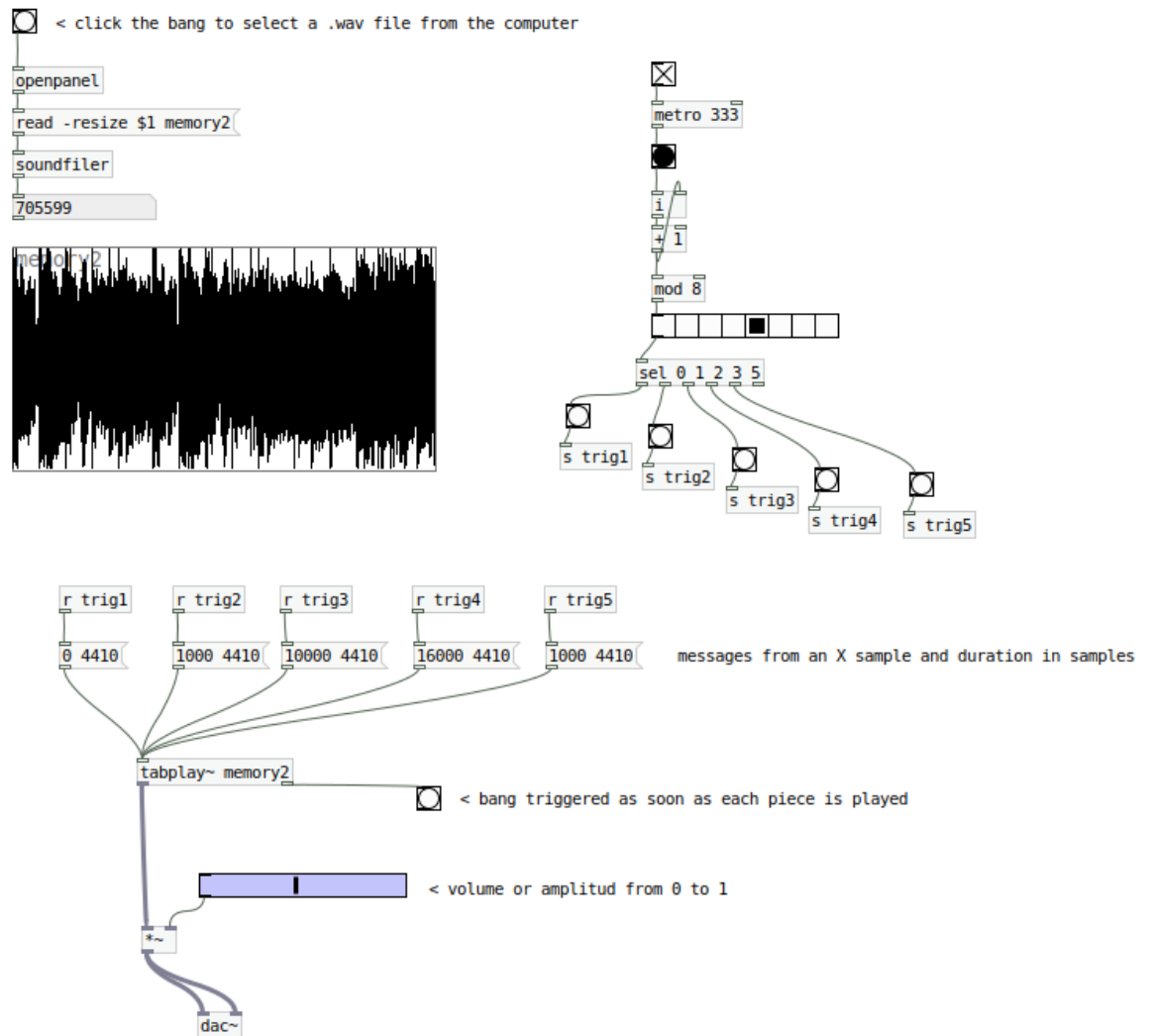
this method put the .wav file into a memory (array) which later can be processed



With **[tabplay~]** we can play the whole sample like in the previous example, but also we can make another use of it that is *slicing*.

Slicing is a reproduction of a particular section of the original file that we can manage with messages that defines the init point of the sample and the length in samples of the slice > **[0 4410 [[100 4410 [etc.**

In the next figure a tiny sequencer of 8 steps triggers several slices.



Synths

4.1. Pd Synths Waveforms

If you are an electronic musician you can jump this introduction :

Synthesizers both in software and hardware are beautiful instruments to reproduce a massive range and types of sounds, from the imitation of natural sounds until the production of unique and particular sounds.

All this massive range of sounds, can be produced by a vast number of techniques and methods, but anyhow we can build any type of synthesizer from particular 'sonic bricks', that afterwards we can combine, organize and filter in complex structures.

These bricks are referred as Waveforms driven by Oscillators.

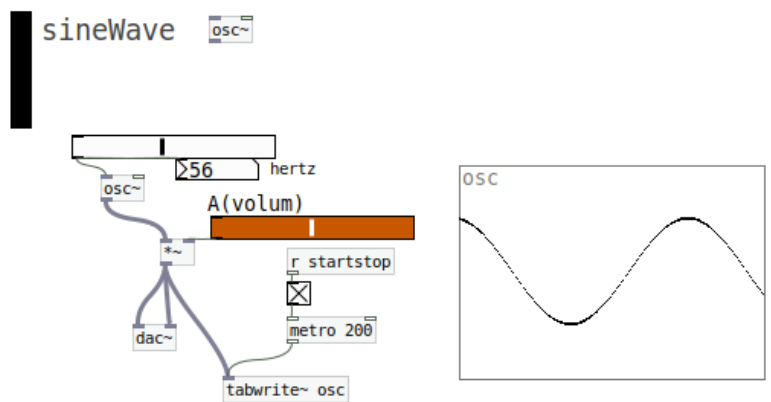
An Oscillator is one of the most basic conceptual element in a synthesizer :

In hardware framework, it translates electricity to an oscillating acoustical signal.

In software framework, are functions which translates data into an oscillating acoustical signal through the DSP.

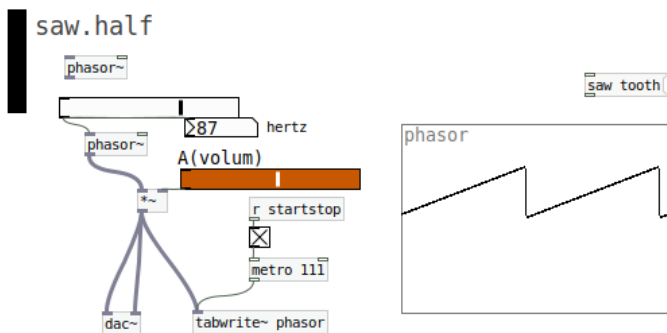
In pd there are some native waveforms and some others that we can build.

Sinewave Oscillator > [osc~]

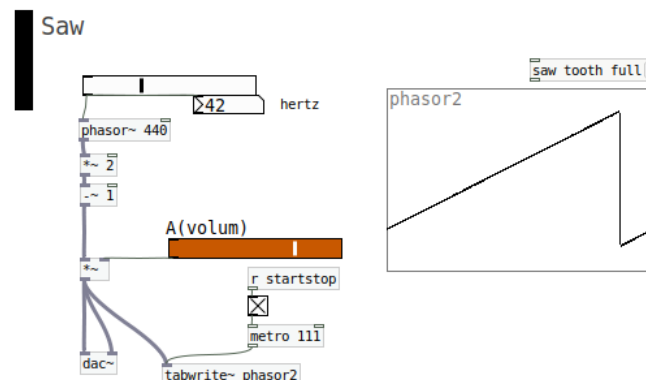


Saw Oscillator > [phasor~]

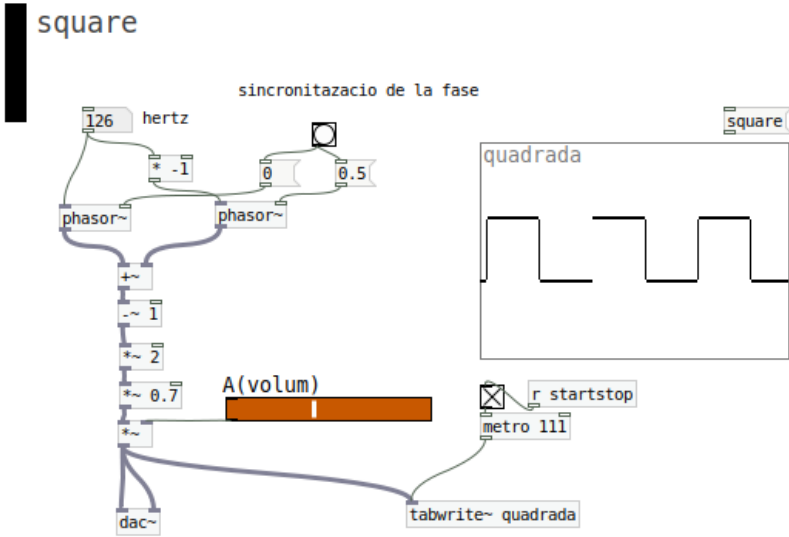
(half) saw >



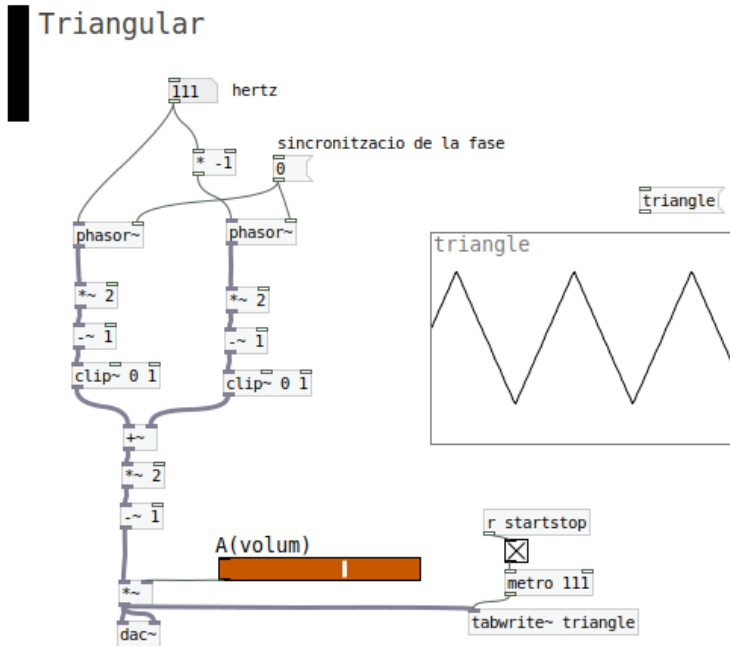
(full) saw >



Square Waveform

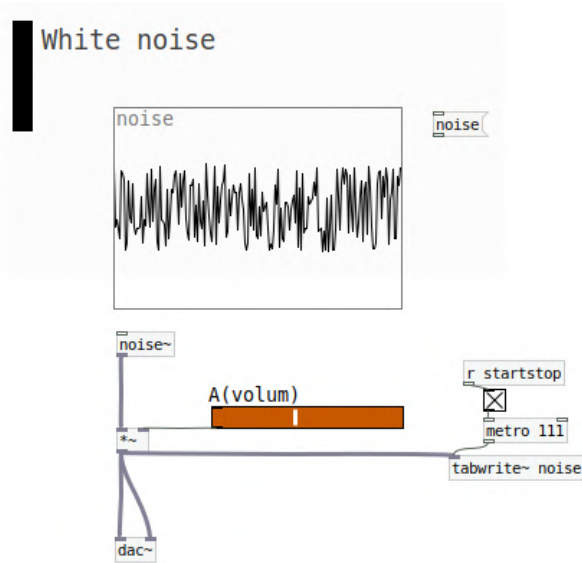


Triangular Waveform



Noise Waveform

Even is not exactly a waveform type, white noise **[noise~]** it is a very common 'brick' in Synth's World. White noise is a pretty particular sonic element in which all frequencies are reproducing at the same time. For this reason is a very used element in subtractive and percussive synth techniques.

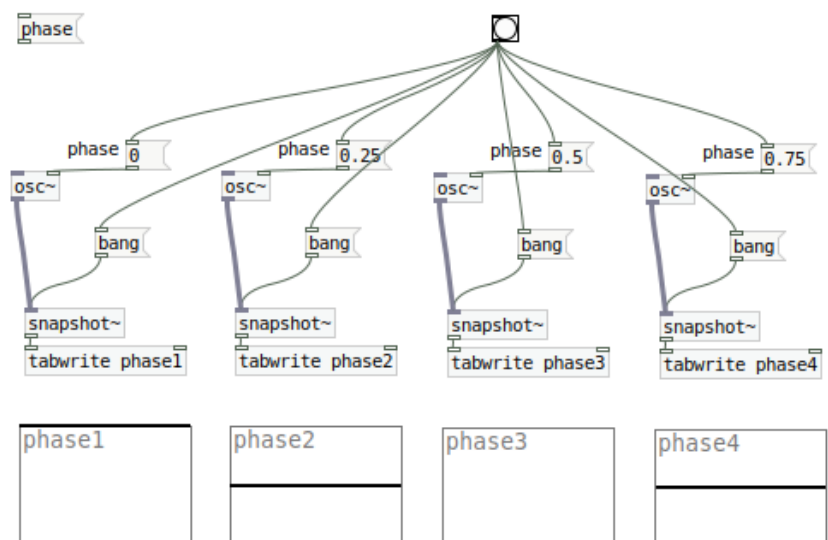


Synths

4.2. Pd Phases in Oscillators and Phasors

Phases can be set both in **[osc~]** and **[phasor~]** objects through the right inlet. Notice that degrees are represented in a range from 0 to 1

Oscillator.Phases



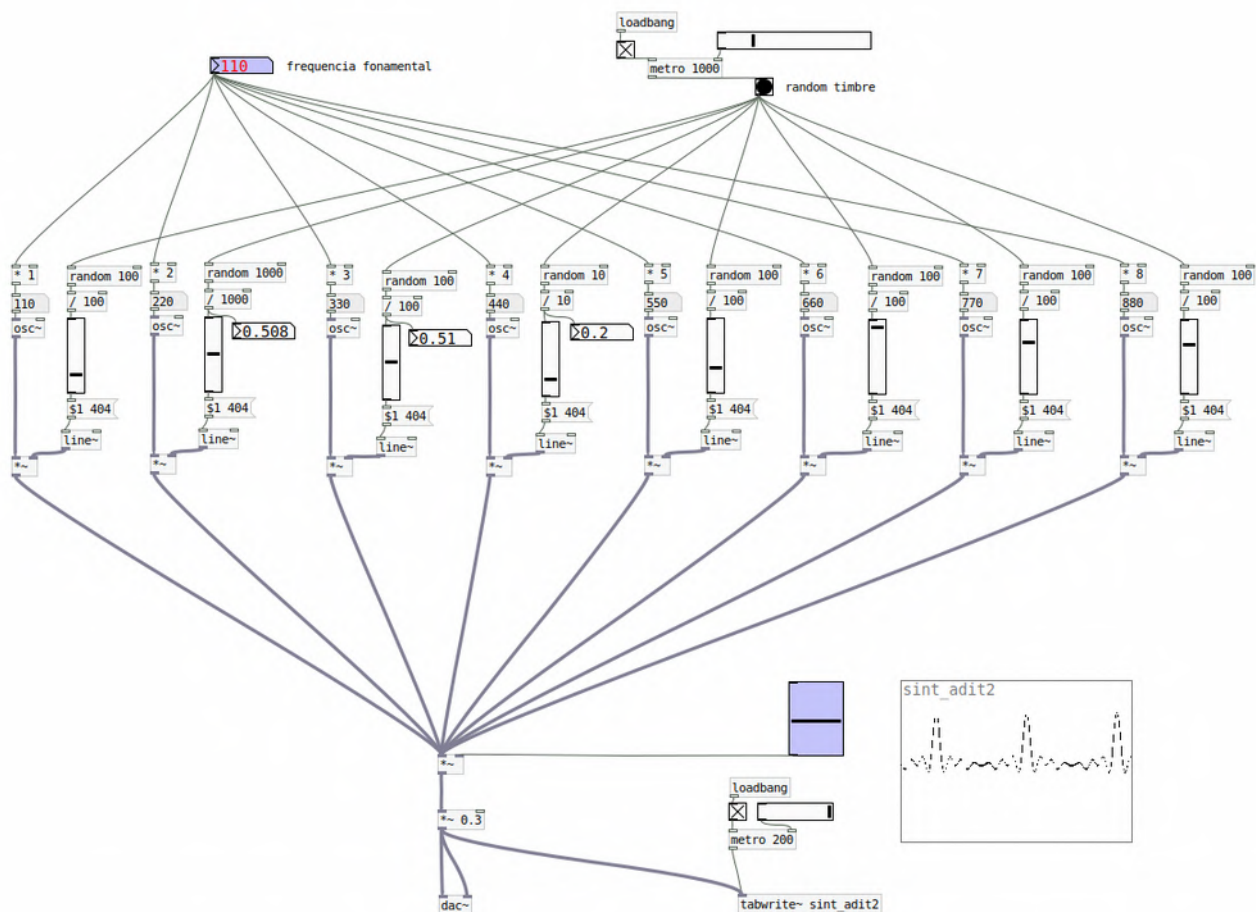
As you may know there are a big amount of synthesis types. However we differentiate among different classical categories like additive, subtractive or granular. Another categories and techniques of synthesis can be based in *physical models*, in *probabilistic models* (stokastic synthesis), or imitation of sounds like *formant synths* imitating human voice, among others.

Synths

4.3. Pd Additive Synthesis

Visual Domain Analogy > Drawing different colors and shapes in a blank and white canvas. The additive synth would be the whole picture.

Additive Synthesis : several layers of generated sound combining and interacting between them. In the next example, a group of 8 oscillators with superior harmonics (multipliers) is changing randomly the amplitud of each line before mixing them. The result is a continuous tone / drone in which timber is changing all the time.



Synths

4.4. Pd Subtractive Synthesis

Visual Domain Analogy > carving with different shapes and sizes the surface of a black canvas, sculpting it and arriving to the canvas basement. The Subtractive synth would be the whole sculpted picture.

Subtractive Synthesis : sculpting with different kind of filters a massive block of sound generated usually with white noise [noise~].

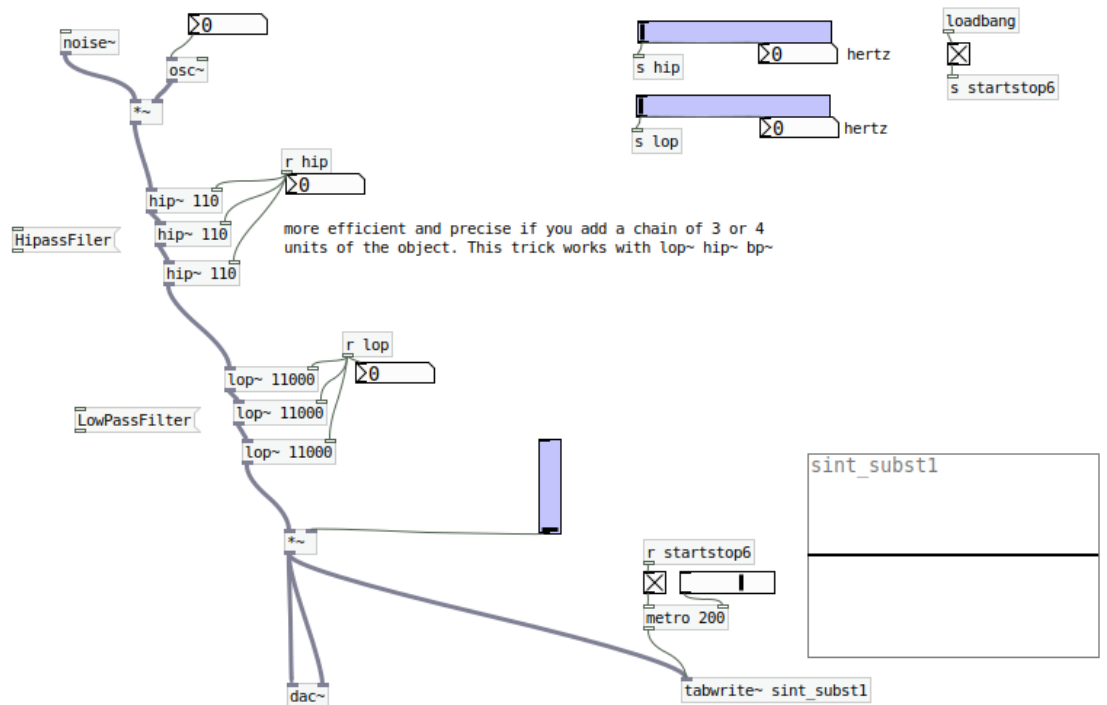
In the next example an oscillator is modulating a white noise which afterwards can be filtered with hipass filters or lowpass filters.

[lop~ value.in.hertz]

[hip~ value.in.hertz]

Subtractive.Synthesis

Esculpting white noise [noise~] with hipass [hip~] and lowpass [lop~]



As a reminder for newbies:

A hi-pass, features the whole auditive spectrum **starting from** the hipass parameter or threshold.

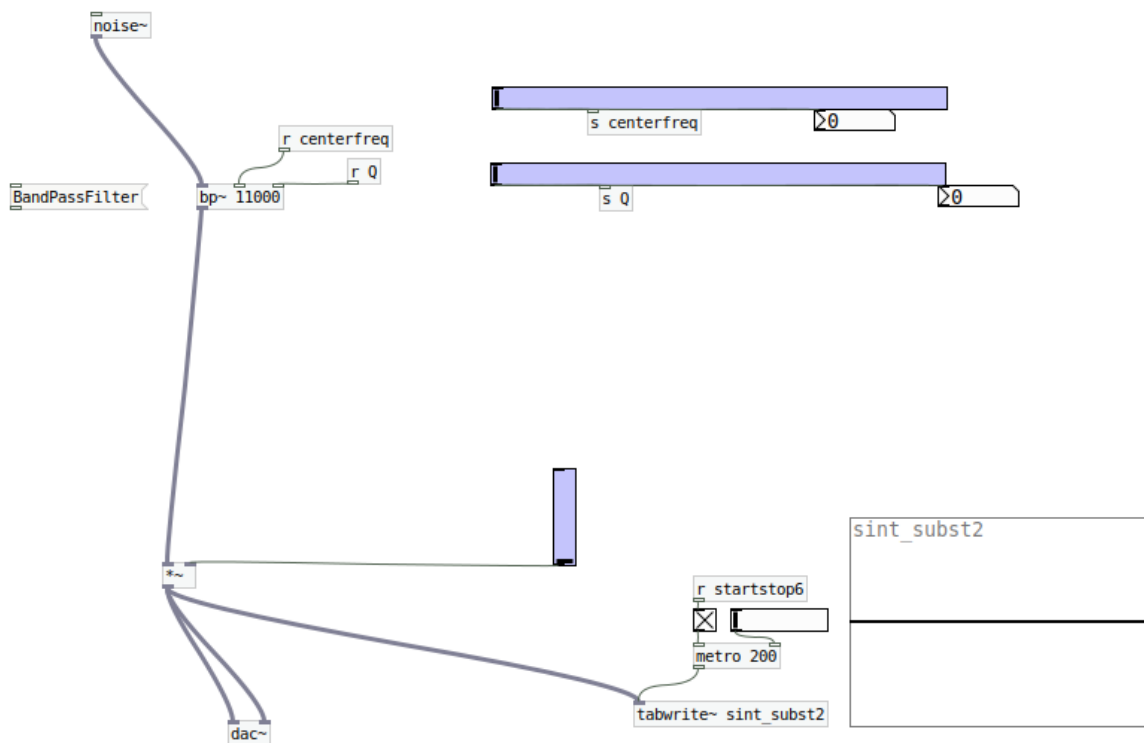
A lo-pass, features the whole auditive spectrum **until** the lo-pass parameter or threshold.

In the next example a white noise is filtered with a band pass filter.

[bp~ value.in.hertz]

Subtractive.Synthesis

Esculpting white noise [noise~] with bandpass [bp~]



As a reminder for newbies:

A band-pass, executes a 'mountain-shape' filter from an incoming signal in which for a certain frequency defined in the `bp`.

The 'mountain shape' can be more or less vertically stretched depending on the amount of `Q` parameter, producing more or less resonant effect, according the reflections inside the cavity of the 'mountain-shape' Therefore:

High values of `Q` represents a kind of huge and vertically stretched mountain, and will be MORE resonance on the frequency range defined by `bp`.

Low values of `Q` represents a tiny hill wider in the bottom, and will be LESS resonance on the frequency range defined by `bp`.

////////////////////

The previous examples may be a bit obvious and 'nothing' special as a synths, but may be can constitute the basis for other operations and tricks in time.

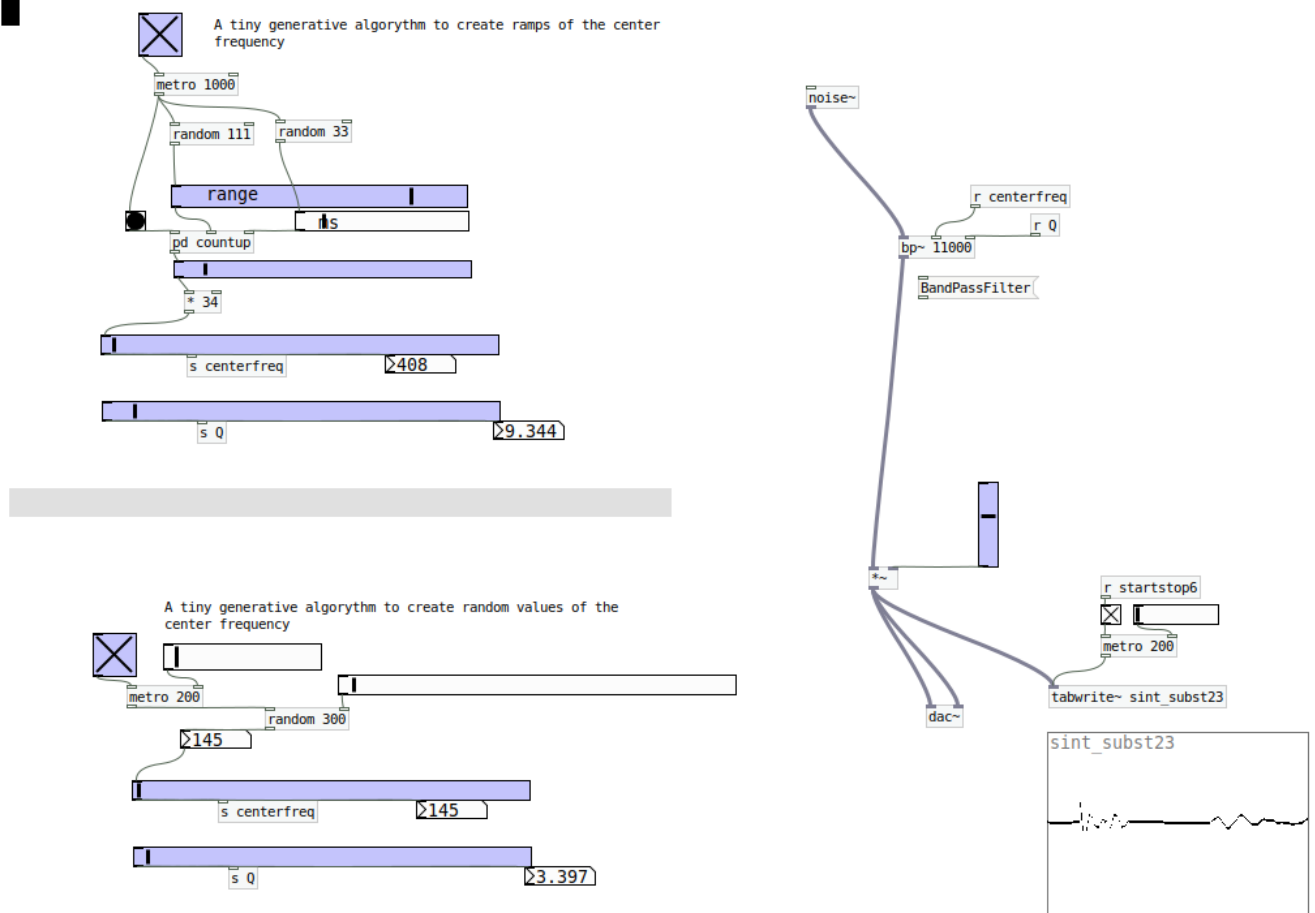
For example, with the band pass example, we can introduce a couple of tiny algorithms to create some particular effects.

In the first one (top left), some kind of sliding or *portamento* effect is produced in the band pass frequency. Every second is doing this effect with a random frequency in a different amount of time, therefore a slightly different action within a repetition process.

In the second one (bottom left), a random generator produces different sudden frequency values. This sudden changes produces some glitches with a particular sound-ressonant bubbles effect.

Subtractive.Synthesis

Sonic_Design



Notice that those algorithms are a calculus layer over the signal structure. Therefore it is possible to combine and reproduce several algorithms at once, producing expressive and unexpected effects.

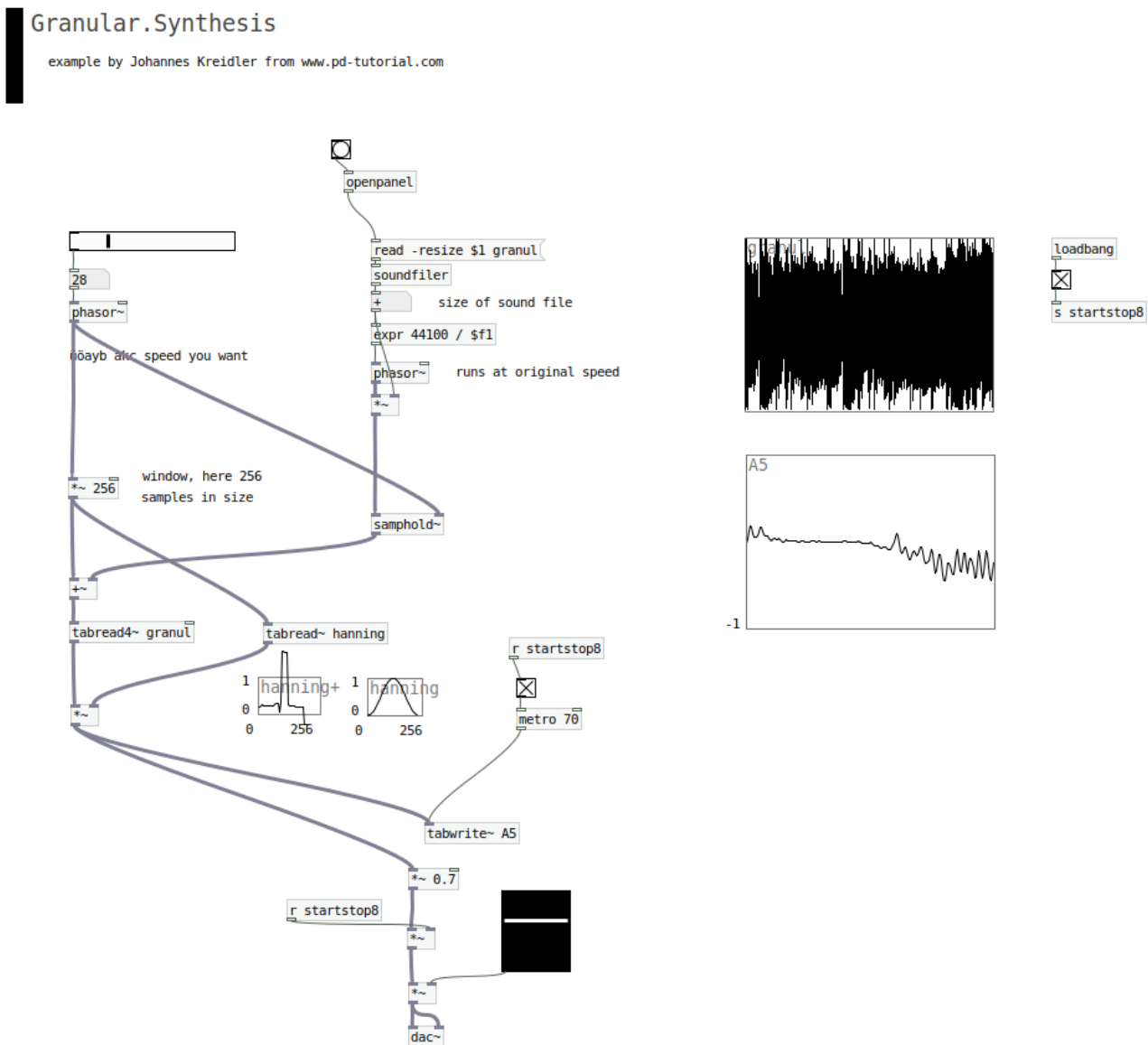
Synths

4.4. Pd Granular Synthesis

Another interesting technique, originally conceived from digital music systems, is granular synthesis. Granular synths are somekind inspired by quantum physics applied to sound.

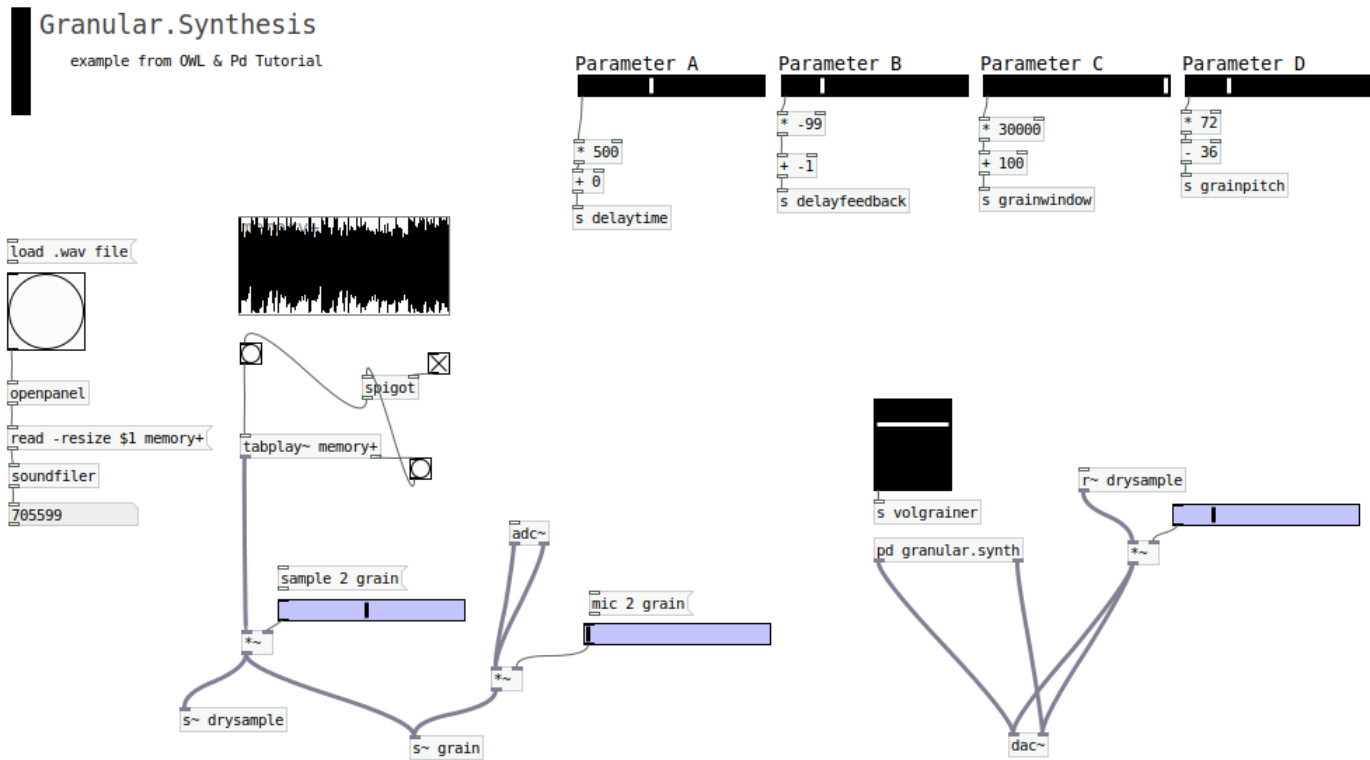
In this technique a particular audio sample is reprocessed like if we would have a microscope targeting over a tiny sample slice, with different non-linear parameters to tweak like amount of particles or *grains*, asynchrony of them, among other non conventional parameters.

In pd Vanilla , that is the version in which we will be able to compile for LICH module, granular synths are not the most complexes, but at least we can use it to stretch and distort samples as an expressive sonic resource.



Notice that these techniques requires an advanced level of DSP Programming skills, that often as a musicians, or coder-musicians we use them just with few tweaks.

Another example of granular synth is the next one, borrowed from OWL rebel Tech repos :



If you like this kind of synthesis and want to master it, the book ['MicroSound' C.Roads](#) is a very nice recommendation.

With synthesizers we can produce synthetic sounds with the most fundamental bricks like we already saw, but also is a lot of fun to model, transform and sculpt any preproduced synthetic sound. That's the process of signal filtering.

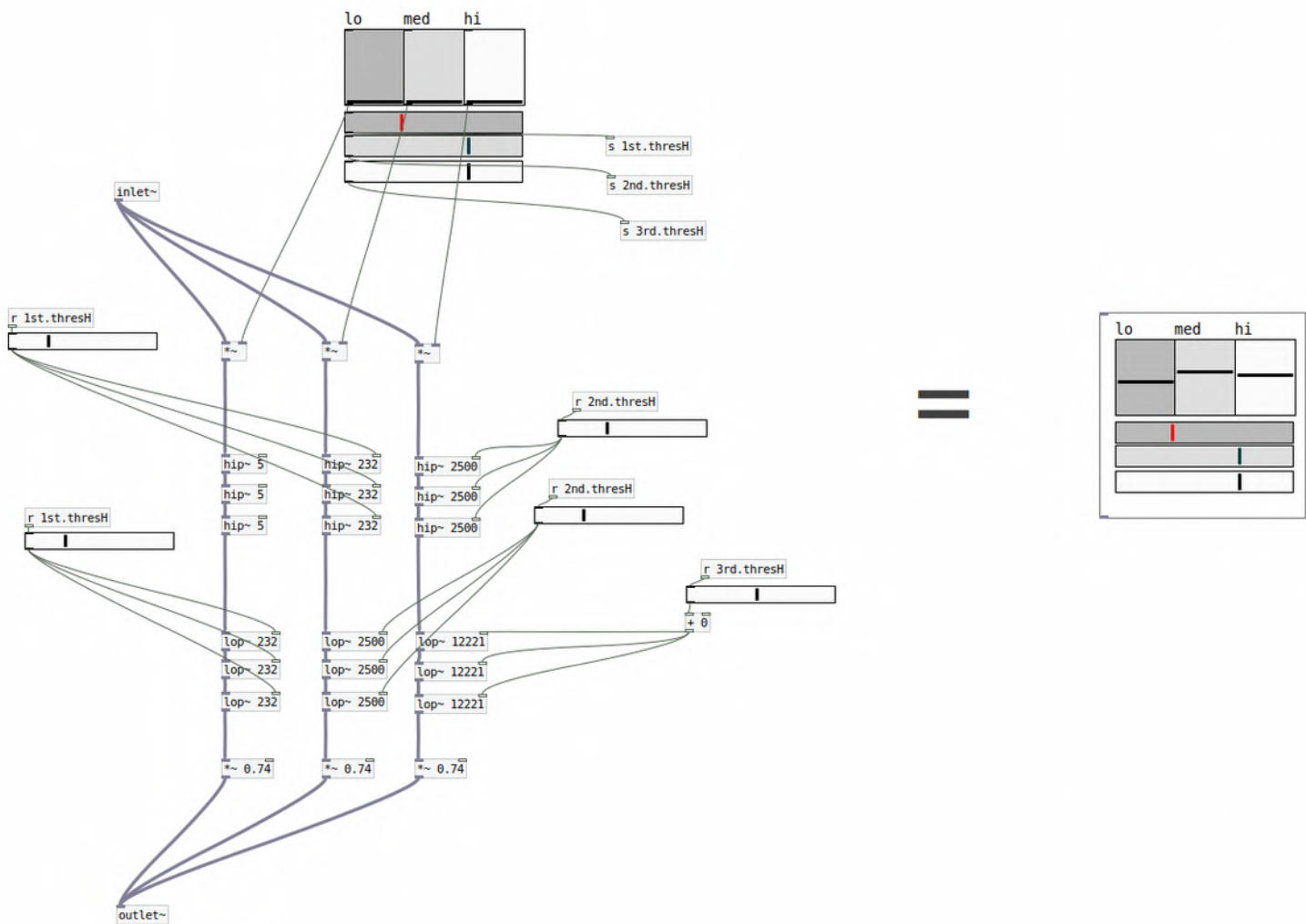
Filtering Signal

5.1 Eqs :// hip~ lop~ bp~

Like we already saw, we can filter any signal through the classic filters `lop~ freq` `hip~ freq` and `bp~ freq Q`.

If we want to build an Equalizer we can split the main signal into the EQ channels we want, and drive each line with the appropriate `hip~` and `lop~` thresholds. In addition in the end of each line, there is an attenuator `*~ 0.74` due to the fact that signal line is triplicated.

3. Band. EQ



Filtering Signal

5.2 Delays

Maybe **Delays** is one of the most classic effects (but at the same time essential) in signal processing, due to the fact that is not affecting directly the character of the sound, but is affecting it in the time domain.

In order to build delays we need to send any particular signal into a buffer with

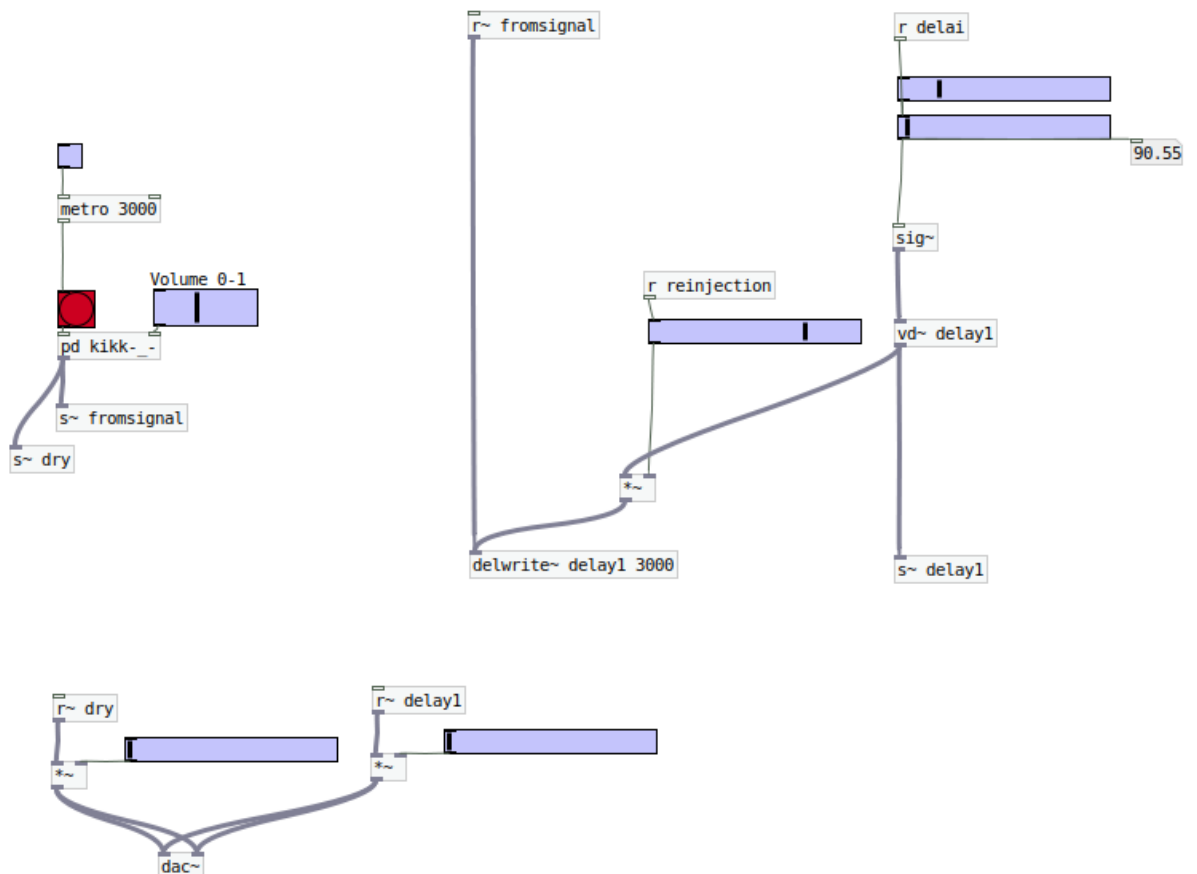
[delwrite~ name.of.delay default.time]

and later call it with **[vd~ name.of.delay]** to the main signal mix. The values inside this object has to be managed with **[sig~]** which allows to introduce numbers (data) into a signal object. Therefore is a method to dynamically introduce delay times into the **[vd~]** object.

In addition we can reinject the processed signal's delay into the original loop through a gate **[*~]** allowing to reinject signal from 0 to 1 (zero reinjection to full reinjection feedback).

Note: if you use delay's feedback in LICH applications, feedback values has to be until +- 0.75 because firmware doen not support calculations fro higher feedback values.

Delay



Another interesting feature with **Delays** with Digital Processing techniques, is to quantize the delay times related to a certain master clock we are running.

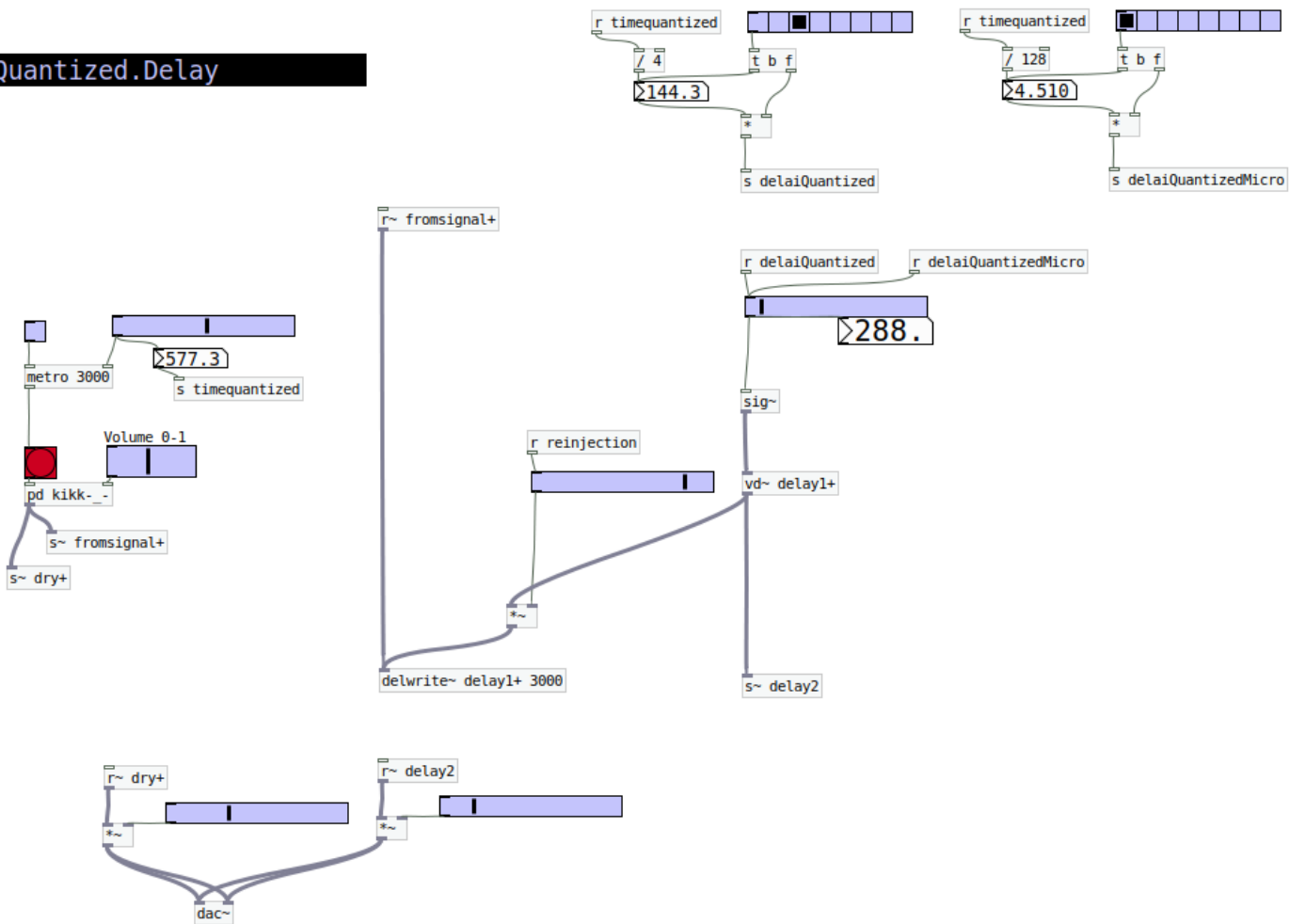
Quantized Delays

In this case the algorithm to quantize time, is as simple of use **[t b f] trigger bang float** which from a certain incoming value (in this case **[r timequantized]**), can be easily sliced or quantized in proportions of time of the main clock with the initial sent **[s timequantized]**.

Therefore *Quantizations* can be controlled just by the lilac Hradio Button.

In this case there is a couple of Hradios : on the left for quantized macro delays and on the right for quantized micro delays.

Quantized.Delay



Filtering Signal

5.3 Reverb

Like Delays, **Reverb** effect is an action that affects the incoming signal in the **time** domain, but specially in the **space** domain.

With this classic effect we can simulate different spaces from a tiny room, until a massive hall.

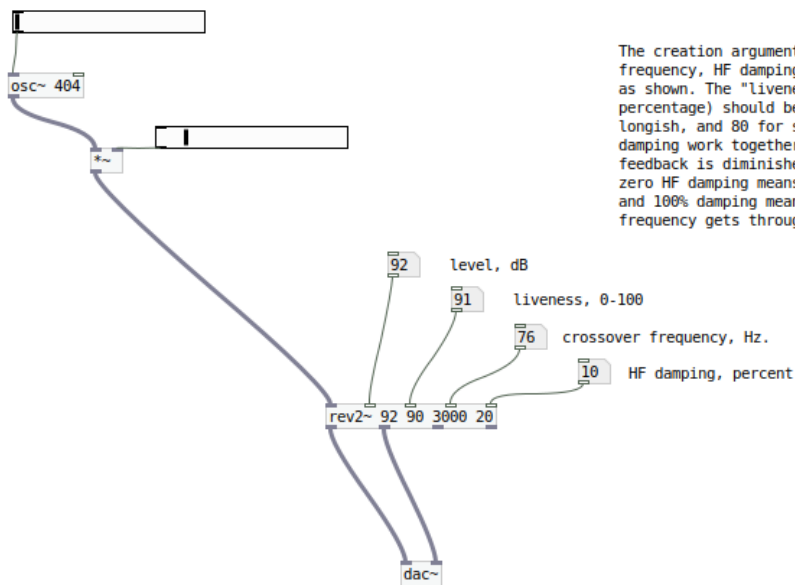
In Pd reverbs are pretty simplified objects, due to the expensive processor calculus of this kind of filters.

[rev2~] less Cpu expensive

[rev3~] more Cpu expensive

Reverb

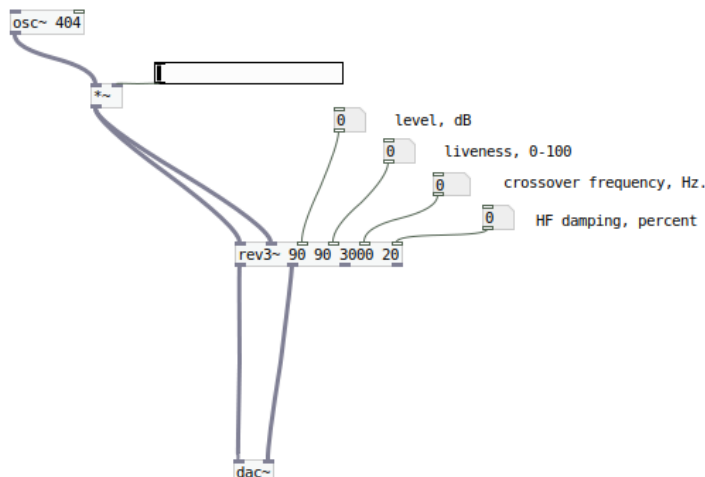
REV2~ - a simple 1-in, 4-out reverberator



The creation arguments (level, liveness, crossover frequency, HF damping) may also be supplied in four inlets as shown. The "liveness" (actually the internal feedback percentage) should be 100 for infinite reverb, 90 for longish, and 80 for short. The crossover frequency and HF damping work together: at frequencies above crossover, the feedback is diminished by the "damping" as a percentage. So zero HF damping means equal reverb time at all frequencies, and 100% damping means almost nothing above the crossover frequency gets through.

Reverb

REV3~ - hard-core, 2-in, 4-out reverberator



The creation arguments (level, liveness, crossover frequency, HF damping) may also be supplied in four inlets as shown. The "liveness" (actually the internal feedback percentage) should be 100 for infinite reverb, 90 for longish, and 80 for short. The crossover frequency and HF damping work together: at frequencies above crossover, the feedback is diminished by the "damping" as a percentage. So zero HF damping means equal reverb time at all frequencies, and 100% damping means almost nothing above the crossover frequency gets through.

Filtering Signal

5.4 Distorsion

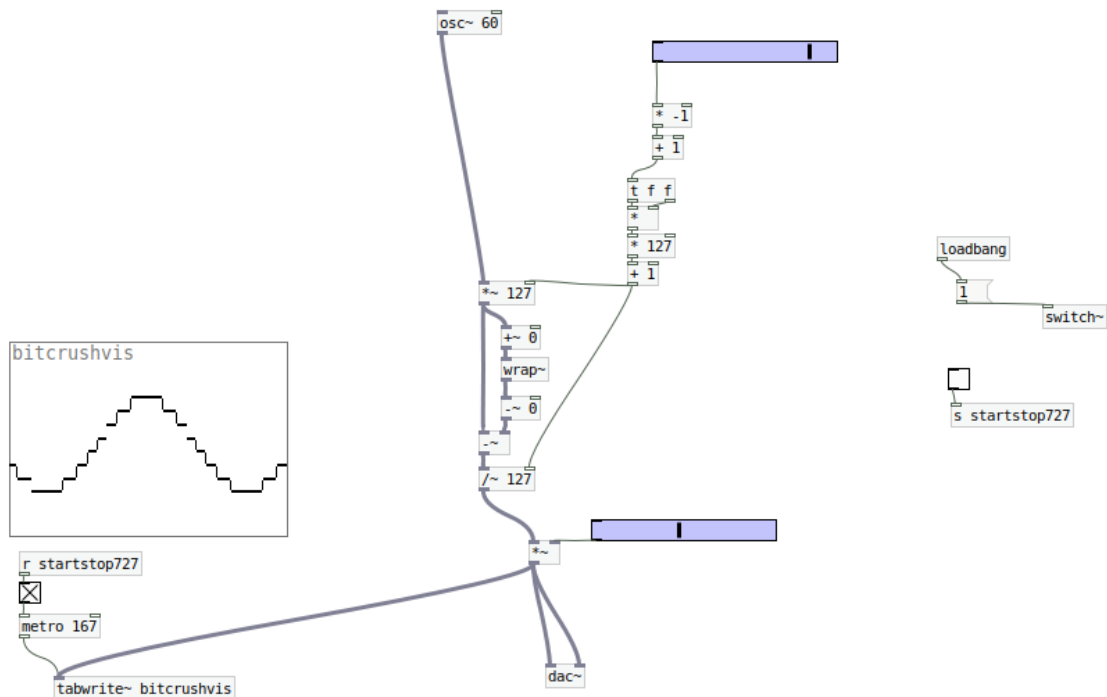
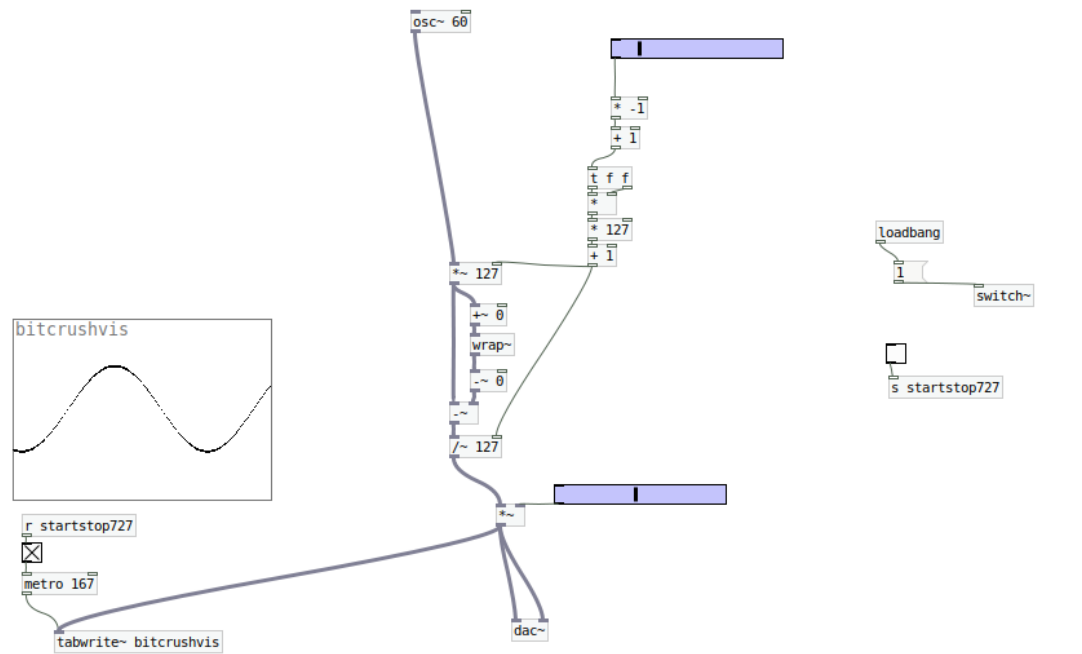
There are several kind of distorsion, but one of the most common is the **BitCrusher**, a type of distorsion that reduces the *bitdepth* of the running signal, altering the waveform itself.

The next example is an algorithym of bitdepthing.

For low values of the top lilac slider, incoming signal is not filtered

For high values of the top lilac slider, incoming signal is transformed into a much more 'pixelated' / squared waveform than the initial one.

BitCrusher



Filtering Signal

5.5 WaveShapers

WaveShaping is an sculpted/extruded technique of an incoming signal.
According to a certain stored waveform's geometry, signal is processed with this shape.

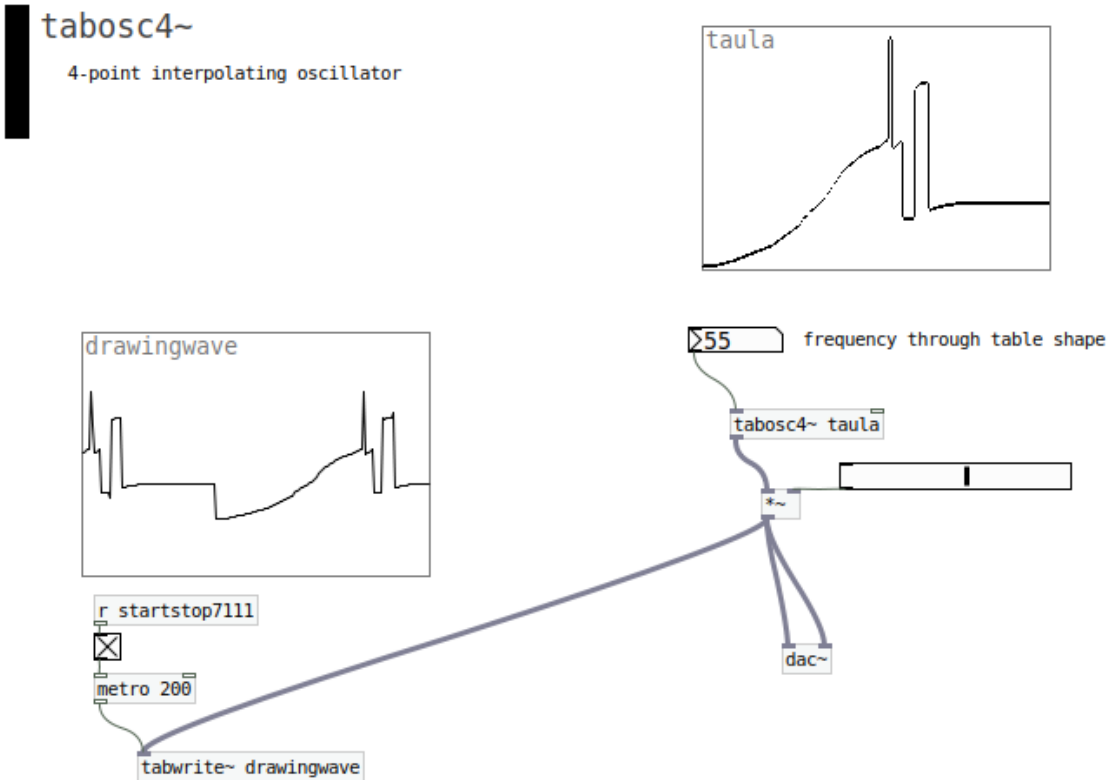
In pd we can use different methods to build this filter.

tabosc4~

With this object associated to an array or memory (in this case **taula**), an incoming value is processed as the lead frequency of an oscillator which shape or waveform is the one described in the array (**taula**).
*In DSP terminology, [tabosc4~] is a traditional computer music style **wavetable** lookup oscillator using 4-point polynomial interpolation.*

It's a nice technique to manage waveshaping with oscillators in a simple way.

WaveShapers



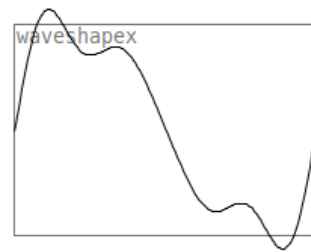
tabread4~

In DSP terminology, [tabread4~] is used to build samplers and other table lookup algorithms. The interpolation scheme is 4-point polynomial.

With this method [tabread4~ array] we can modelate in an easy way different shapes in the array, with trigonometric messages like sinesum, cosinesum among others.

With this method, window size (X) in samples has to be powered 2 (32,64,128,etc), that we can call with the appropriate messages.

[; array sinesum X string.of.values.that.describes.shape [*~(X)/2]



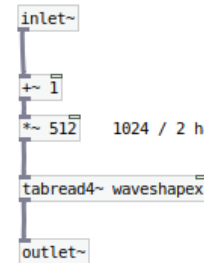
1024 sample array

note Due to interpolation, size is increased in 3 units more in this case 1027

```

;
wvshapex sinesum 1024 1 0.2 0.3 0.1

```



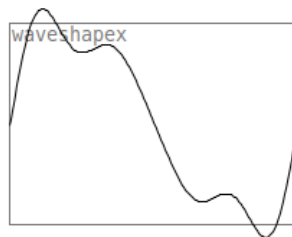
1024 / 2 has to be written in the multiplier

Notice that the more window size (X) will have the waveshaper, the more accurate or less glitched will be the resulting signal.
For example :

WaveShapers

tabread4~

4-point-interpolating table lookup
arrays has to be powered 2 and after rewrite in the multiplier the size of the array



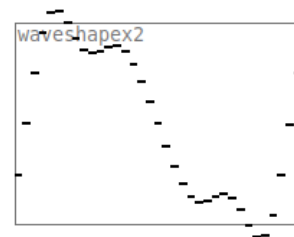
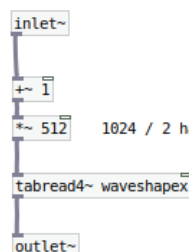
1024 sample array

note Due to interpolation, size is increased in 3 units more in this case 1027

```

;
wvshapex sinesum 1024 1 0.2 0.3 0.1

```



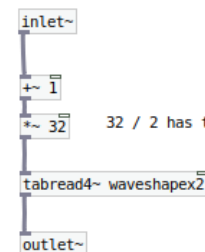
32 sample array

note to interpolation, size is increased in 3 units more in this case 35

```

;
wvshapex2 sinesum 32 1 0.2 0.3 0.1

```



32 / 2 has to be written in the multiplier

Following in this line of comments, lets see how behaves the array [in this case *waveshape*] depending on the written data on it.

If we have a simple ramp, incoming signal is not processed, so the result its gonna be like if there is no waveshaping filter at all. See this figure :

WaveShapers

tabread4~
4-point-interpolating table lookup
[tabread4~] is used to build samplers and other table lookup algorithms. The interpolation scheme is 4-point polynomial.

thats the algorithym of a waveshape for a 1024 sample array

waveshape

1024 sample array

Set ramp
pd NoWaveShapeAffection
With a ramp from -1 to 1 the filter of waveshaping is not affecting the incoming signal

Otherwise If we have different shapes from the lineal ramp shown before, incoming signal is processed through the geometry of the draw shape, like if the previous ramp was the axis of the calculus.

In this example, a simple sinewave is producing a wooble wave like in visualization's array *waveshapevis* is featuring :

WaveShapers

tabread4~
4-point-interpolating table lookup
[tabread4~] is used to build samplers and other table lookup algorithms. The interpolation scheme is 4-point polynomial.

thats the algorithym of a waveshape for a 1024 sample array

waveshape

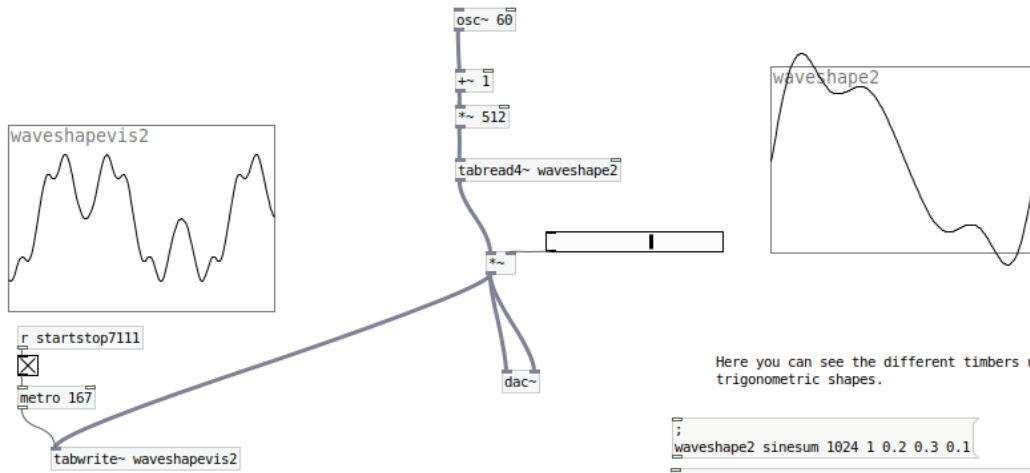
1024 sample array

click and Set basic Sine
waveshape sinesum 1024 1

With this method we can easily build different shapes, for example building with trigonometric messages :

for example :

[; waveshape2 sinesum 1024 1 0.2 0.3 0.1]
 With sinesums we can produce similar effects to a compressor due to the fact that boosts incoming signal without clipping (in case that the stored shape is fit in the limits of the array (-1 to 1)).

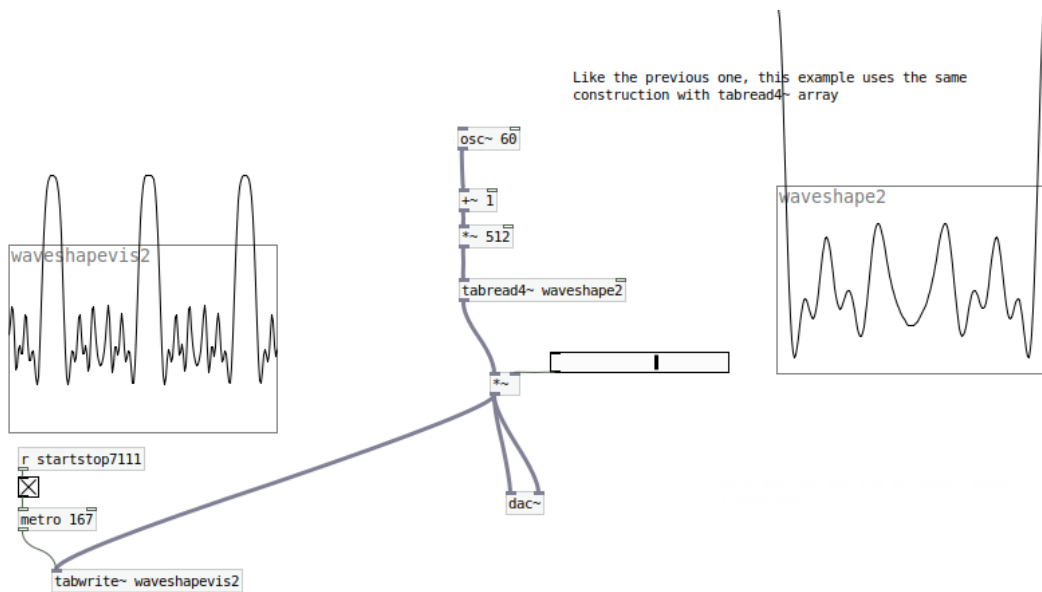


Here you can see the different timbers using different trigonometric shapes.

```

; waveshape2 sinesum 1024 1 0.2 0.3 0.1 <<< Click over these boxes
; waveshape2 sinesum 1024 0 0.2 0.2 0.3 0.1 0.6 0.3 0.2 0.3 <<< Click over these boxes
0.1 0.2 0.3 0.1
; waveshape2 cosinesum 1024 0 0.2 0.2 0.3 0.1 0.6 0.3 0.2 0.3 <<< Click over these boxes
0.1 0.2 0.3 0.1
    
```

[; waveshape2 cosinesum 1024 0 0.2 0.2 0.3 0.1 0.6 0.3 0.2 0.3 0.1 0.2 0.3 0.1]



Like the previous one, this example uses the same construction with tabread4~ array

```

; waveshape2 cosinesum 1024 0 0.2 0.2 0.3 0.1 0.6 0.3 0.2 0.3 <<< Click over these boxes
0.1 0.2 0.3 0.1
    
```

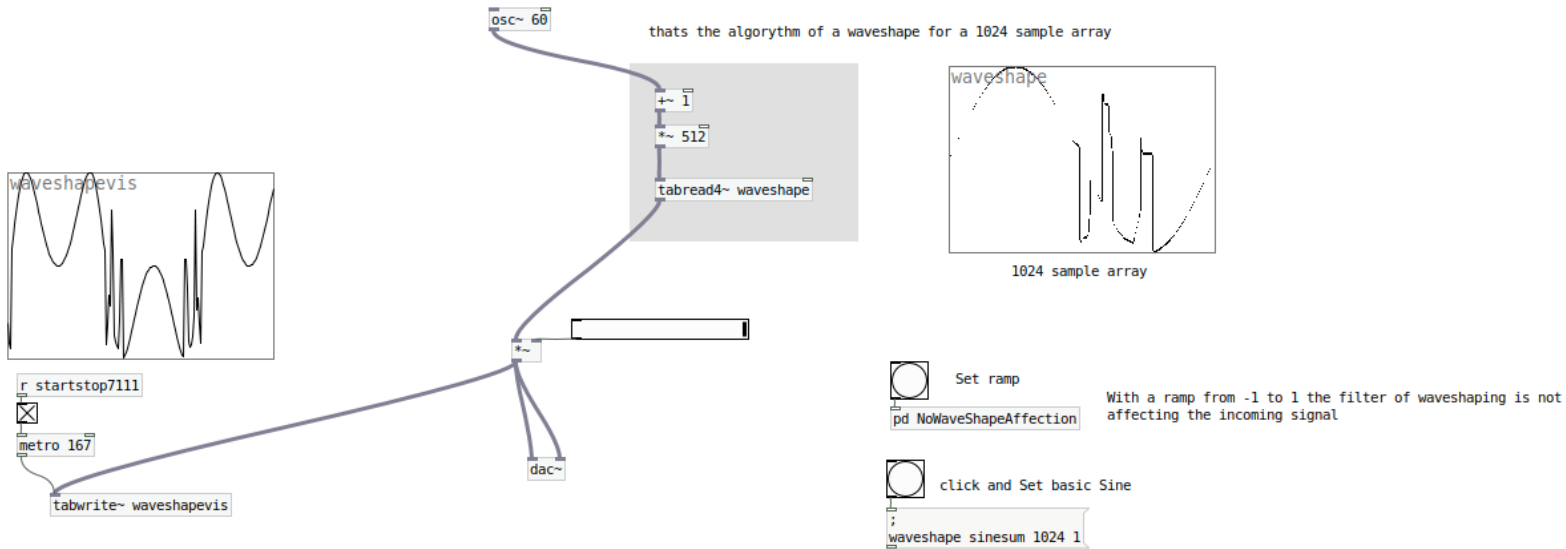

Another interesting feature with this method is to draw over the array with the mouse so we can modificate shapes with strange and non regular geometries :

WaveShapers

tabread4~

4-point-interpolating table lookup

[tabread4~] is used to build samplers and other table lookup algorithms. The interpolation scheme is 4-point polynomial.



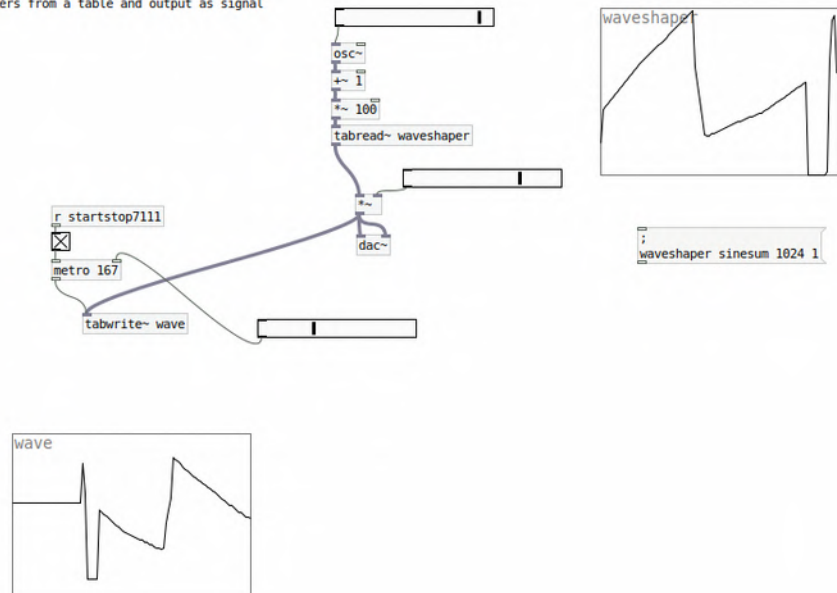
tabread~

In DSP terminology this object read numbers from a table and output as signal. Like in the previous its useful for drawing distorted shapes. Notice that the multiplier and the array size is 100 by default. You can change this in order to produce some more distorted effects.

WaveShapers

tabread~

read numbers from a table and output as signal



Waveshaping RECAP :

Maybe you may think, that due to have several waveshape options and methods, which is better to use?

If you want to make noise registers may be **tabosc~** and **tabread~** are useful.

But if you wanna process your signal with more accuracy **tabread4~** is a very cool method because allows filtering signal both in a clean way and in a distorted / glitchy way depending on your performance needs.

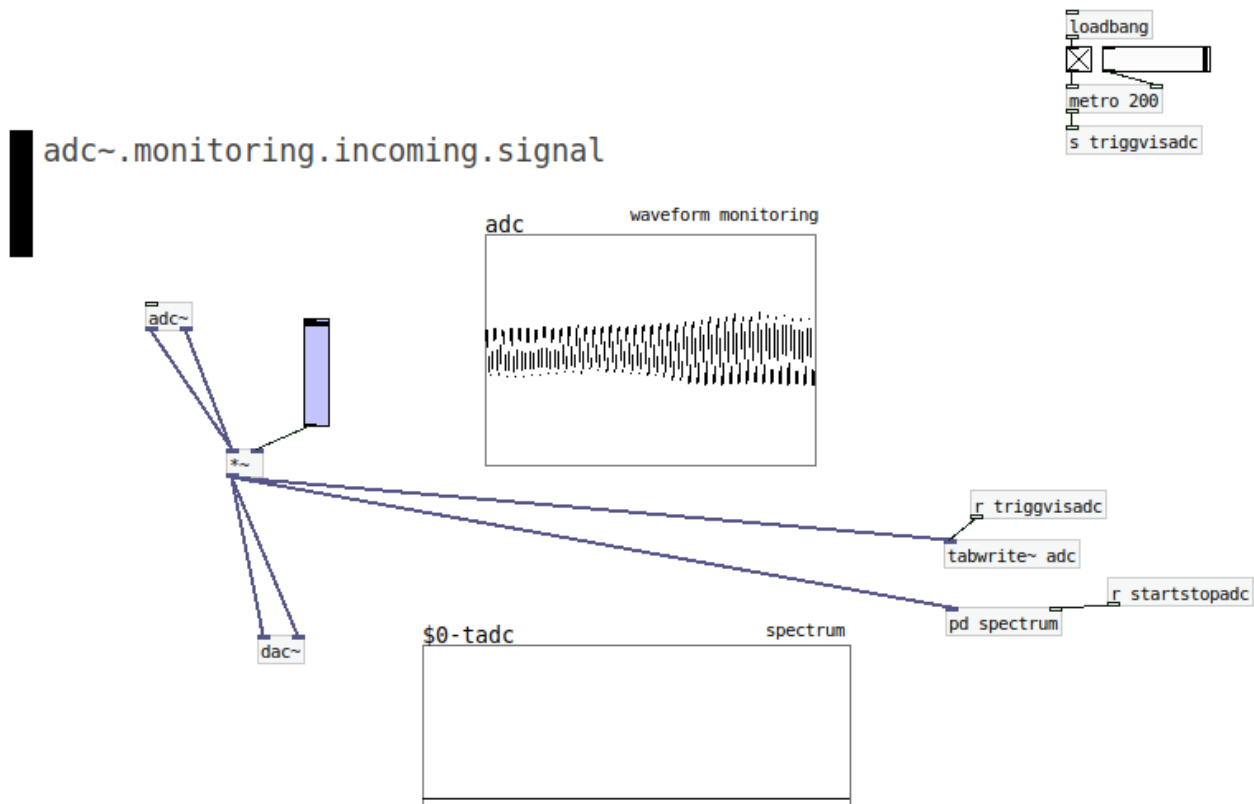
adc~

6. playing with incoming signal

`[adc~]` *analog to digital conversion* is the object that introduces any available* line-in or mic, already set up in your sound card parameters. In laptops usually is the incoming signal of the in-built microphone, unless you load an external sound card with its incoming ports. In this case if you have an external soundcard with for example 4 mono inputs, those will be referred as `[adc~ 1][adc~ 2][adc~ 3][adc~ 4]`.

In LICH module we have a couple of incoming audio signal ports (`IN_L`) and (`IN_R`) that corresponds to `[adc~ 1]` and `[adc~ 2]`.

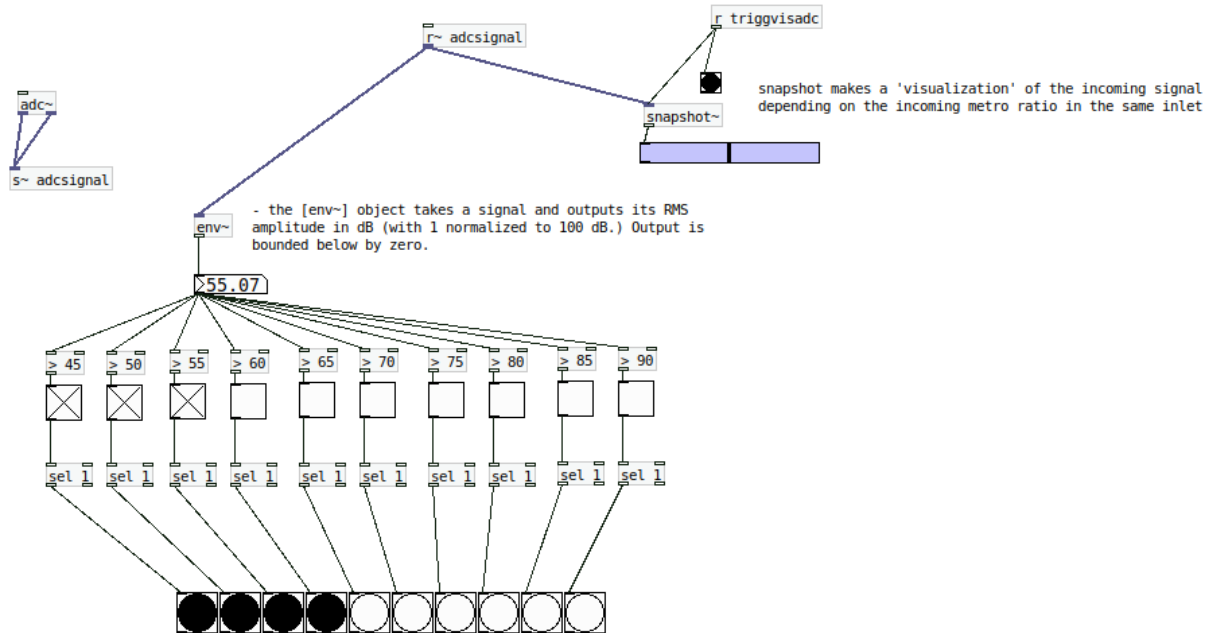
This piece of code features the incoming signal's render from the laptop's in-built mic.



With the object `[env~]` we can monitor the envelope of a certain signal in this case the incoming signal of the mic. The rate of analysed data will be very fast, but we can threshold it with different values in order to make some conditional tree of triggers according to the incoming signal's strenght.

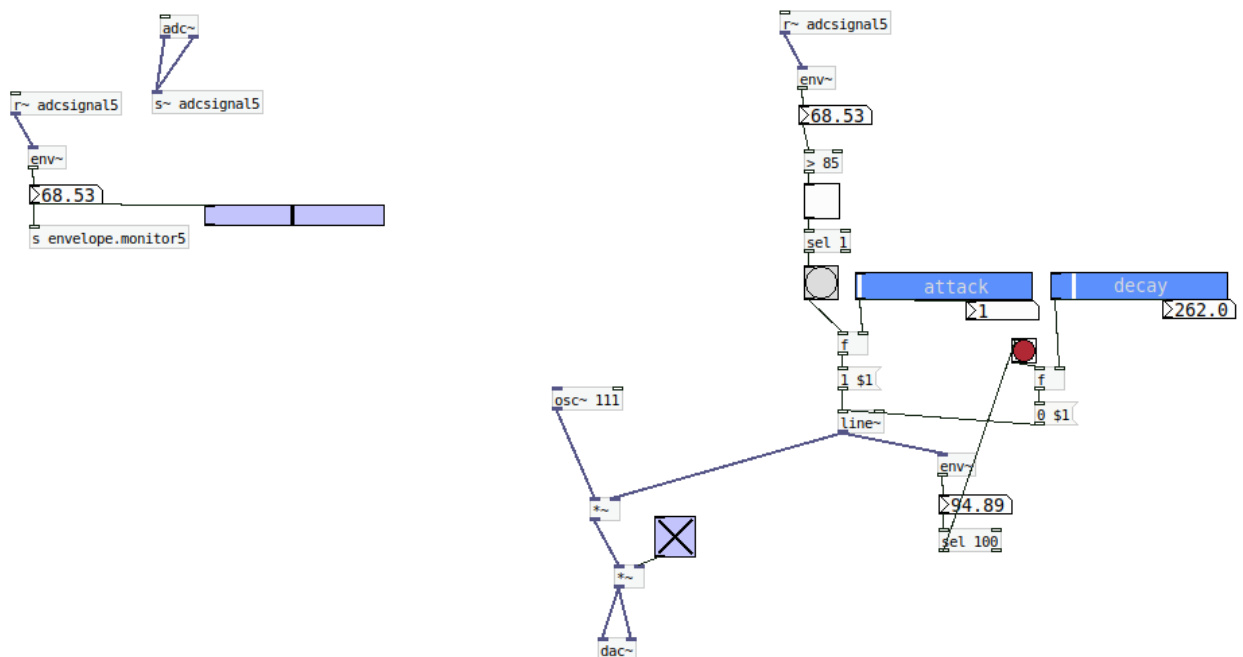
With the object `[snapshot~]` we also can monitor the envelope at a certain desired rate [with an incoming `[metro ms]` object, that maybe can match with our clock for other elements in the patch, that will bring us more sync effect.

adc~.envelope.thresholds



In this example, for higher values than 85 [> 85] detected in the `[env~]` analysis, it will trigger a bang that in this case is connected to an oscillator with attack and decay control, therefore a percussive sound.

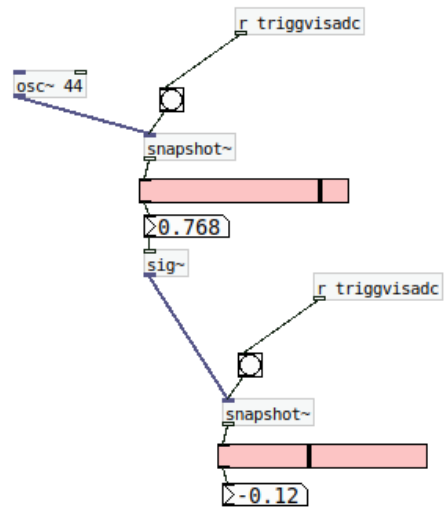
adc~.envelope.controlling.triggers



One way to translate signal to data is with the object **[snapshot~]** triggered with a [metro] object at the desired ratio.

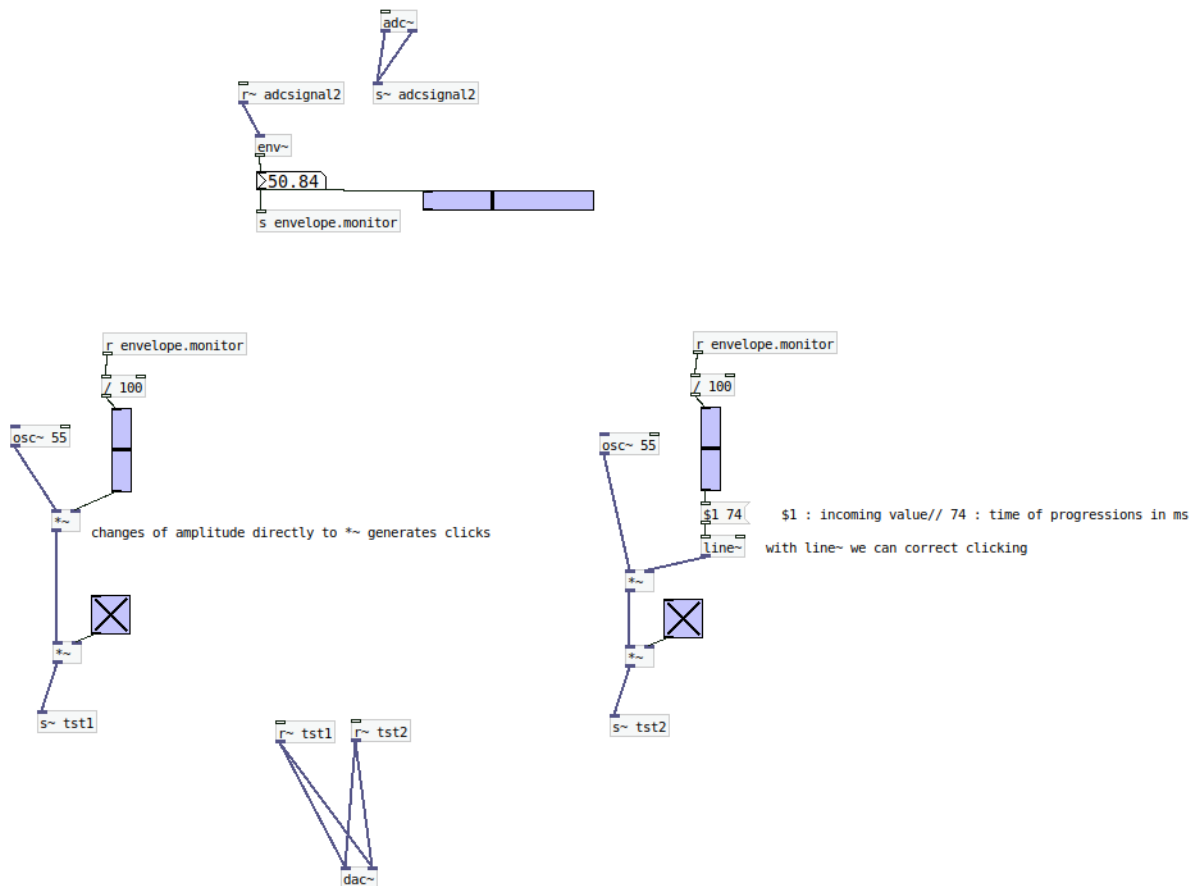
On the contrary, one way to translate data to signal is with the object **[sig~]**.

From.Signal.to.Data.and.viceversa



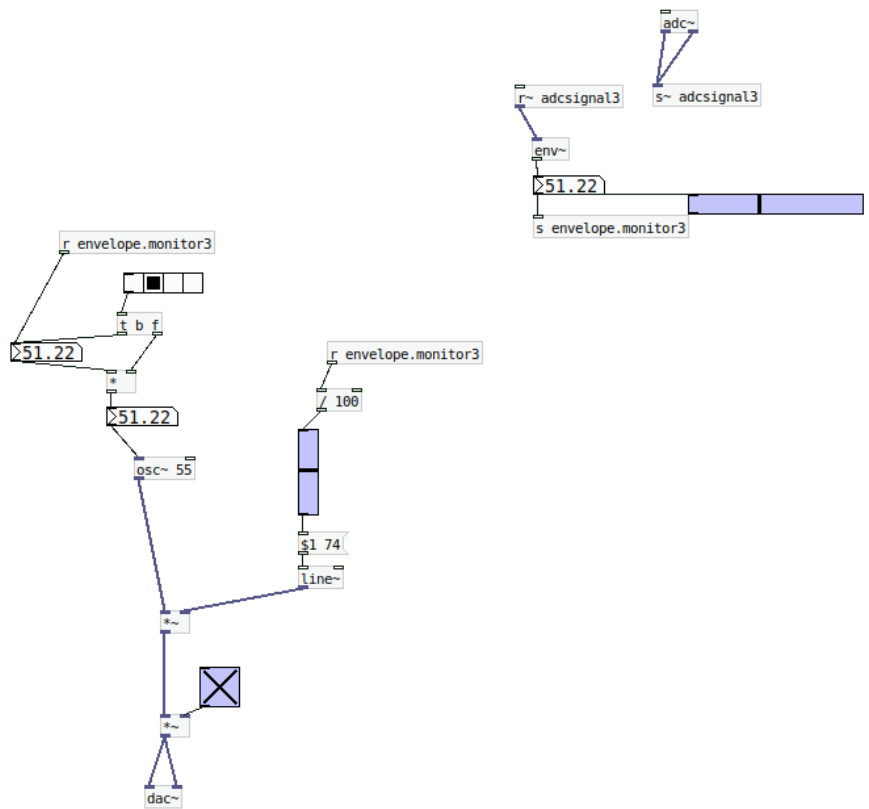
We can use envelopes to control generatively/automatedly different parameters. For example in this case the analyzed envelope data result is controlling the amplitude of a couple of oscillators.

adc~.envelope.controlling.amplitudes



Or even the envelope can control the frequency and the amplitude of an oscillator, like in this example :

adc~.envelope.controlling.amplitudes.and.pitches



6.2. analyzing incoming signal

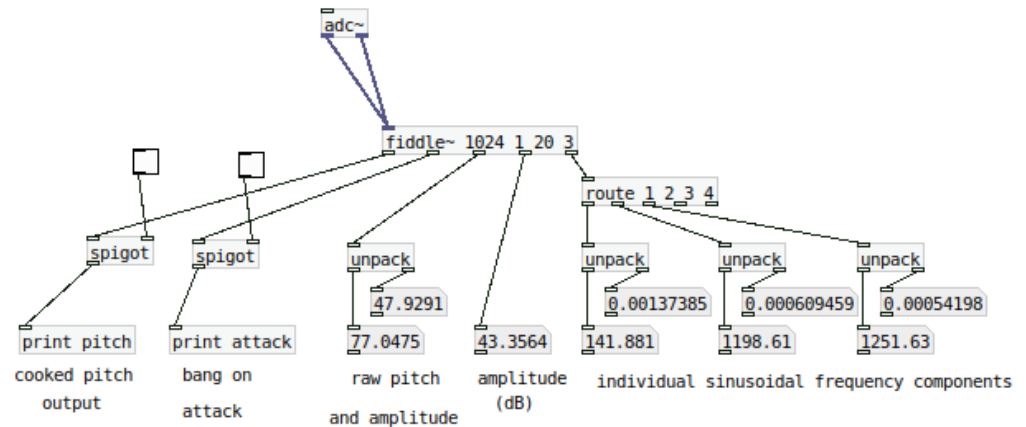
There are different objects useful for analyze a certain signal.

Anyways there a couple of methods that does not work in the rebeltech compiling process which are [fiddle~] and [bonk~]...but at least we can measure the incoming signal with [env~] as we already saw.

detection.functions.of.incoming.signal

fiddle~

this object does not correctly compile in Rebeltech LICH Browser Application

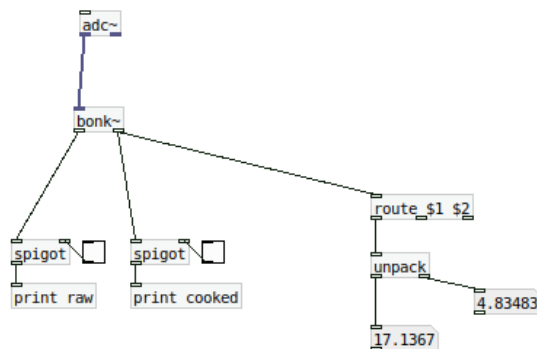


detection.functions.of.incoming.signal

bonk~

this object does not correctly compile in Rebeltech LICH Browser Application

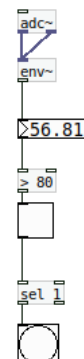
in order to grasp envelope detection try with [env~] and filter outcoming data as we saw before



detection.functions.of.incoming.signal

env~

Remember to use this method >>>>> within your LICH Pd algorithms



Envelopes & LFOs

7.1 LFO : Envelope Modulation by Oscillators

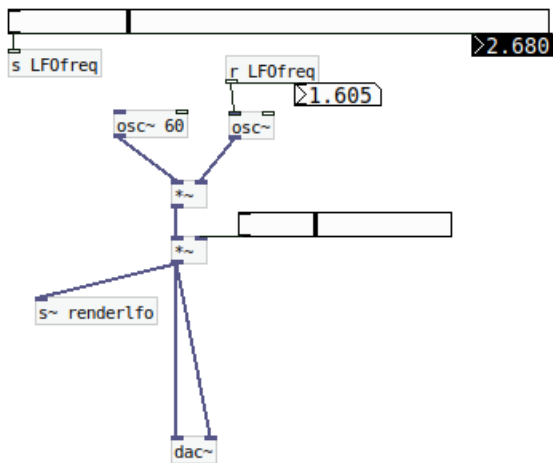
As you may know LFO is a very common and classic effect in electronic music.

It consists into the modulation between a couple of oscillators, one the lead frequency, and the other the LFO modulation rate with very low values of frequencies.

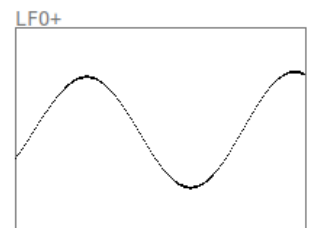
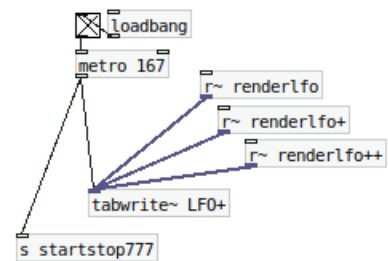
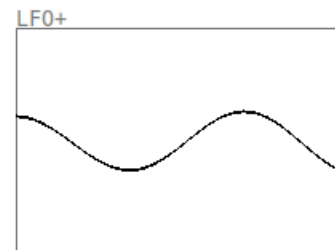
In pd the most basic method to build an LFO is multiplying the signal of two oscillators with very different frequency rates (the lead one and the modulator one).

LFO Low Frequency Oscillator

This is the most basic LFO method > a lead oscillator modulated (multiplying the signal) by a Lowfrequency additional oscillator

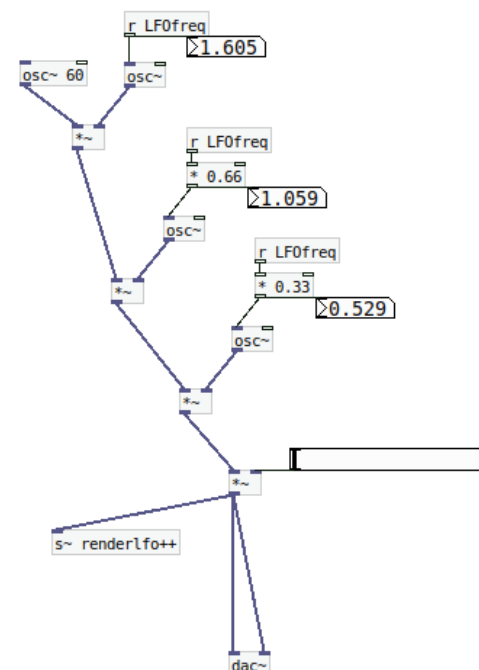
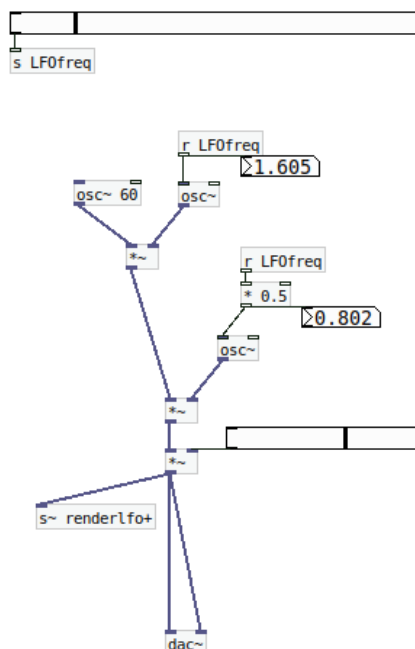


LowFrequencyOscillator can be in a desired range of low frequencies,



Also we can easily build some more unconventional and experimental LFOs like put a chain of LFOs therefore an LFO over a previous LFO and so forth.

Also its possible to build some experimental LFOs like LFOs over a previous LFO making some frequency proportions

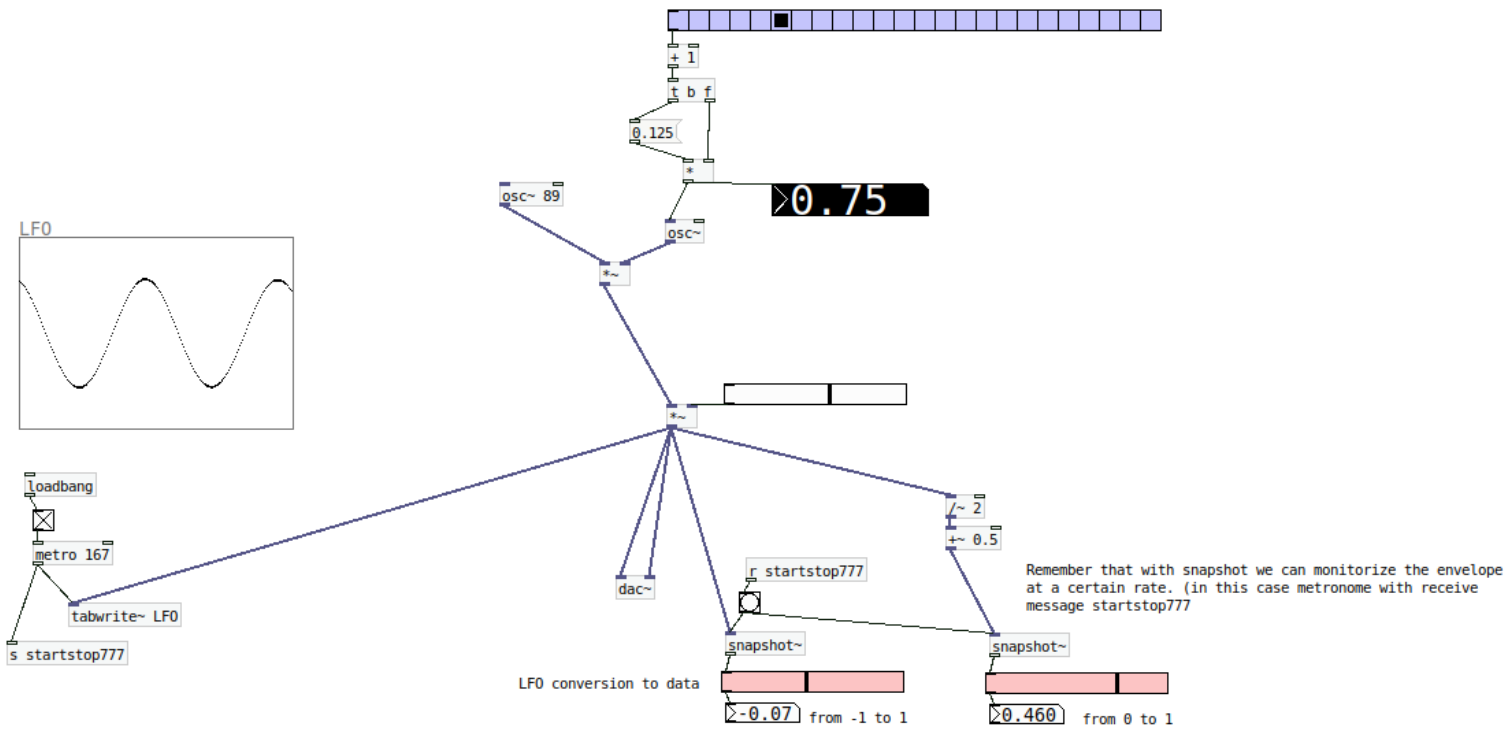


Or also to be more accurate in the LFO frequencies to tweak with proportions of a certain value (in this case **[0.125]**).

In this sense, quantizing LFO frequency rates like the previous one we can build some particular sequences

You can control LFO's LowFrequencies with accuracy making a multiplier with [t b f] over a certain value [0.125] in this case

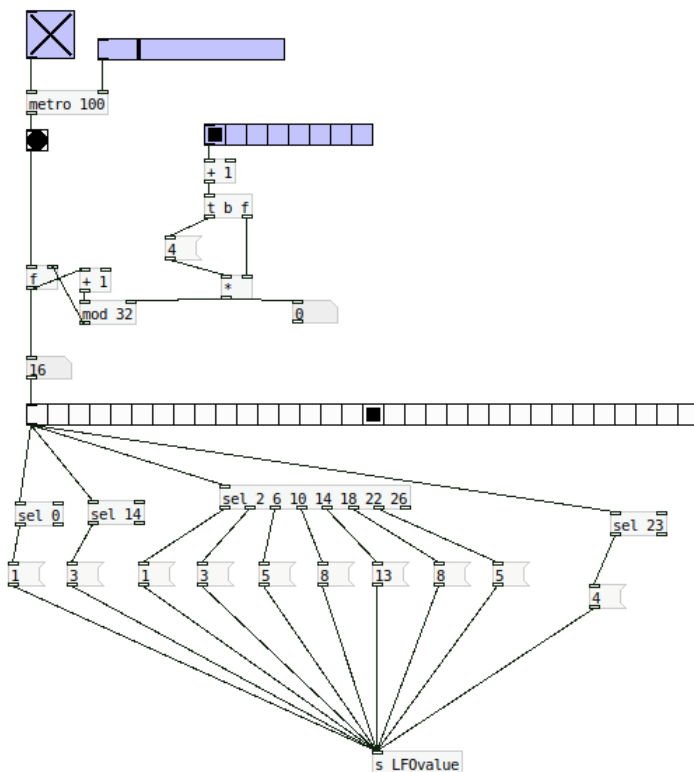
That's useful if you want to make proportions of LFO frequencies



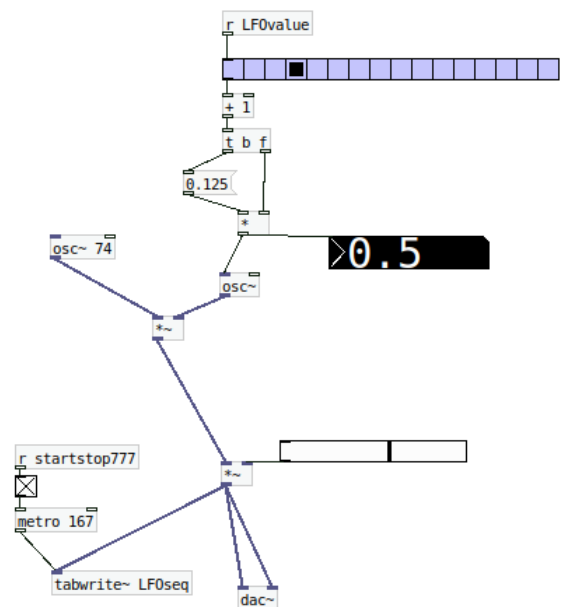
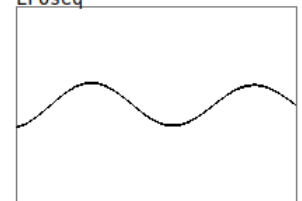
of LFO values

LFO Low Frequency Oscillator SEQUENCED

With the previous LFO quantization algorithm we can even make some sequences of different LFO rates, for example:



LFOseq



Envelopes

7.2 Playing with envelopes in percussive synths

Synths are amazing instruments converting electricity into acoustic signal.

In digital world we would say that different kinds of data produces changes in the DSP extracting an acoustic signal.

In pd there are many kinds of types and techniques, but let's see how to build percussive synthetic sounds.

To produce them we have to control envelope as the classic AttackDecaySustainRelease structure, but even we can go in a much more simplified way, just only controlling *decay* or *attack + decay*.

First we need signal generators that in pd -LICH oriented (vanilla), we have a couple of functions to do so, and already saw them :

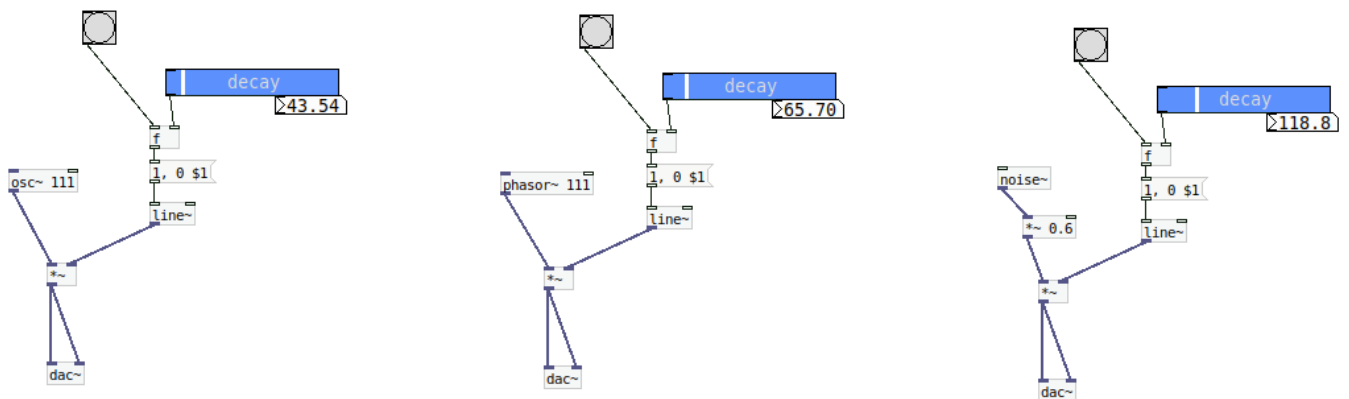
[osc~] sine oscillator

[phasor~] half sawtooth waveform oscillator

[noise~] white noise generator.

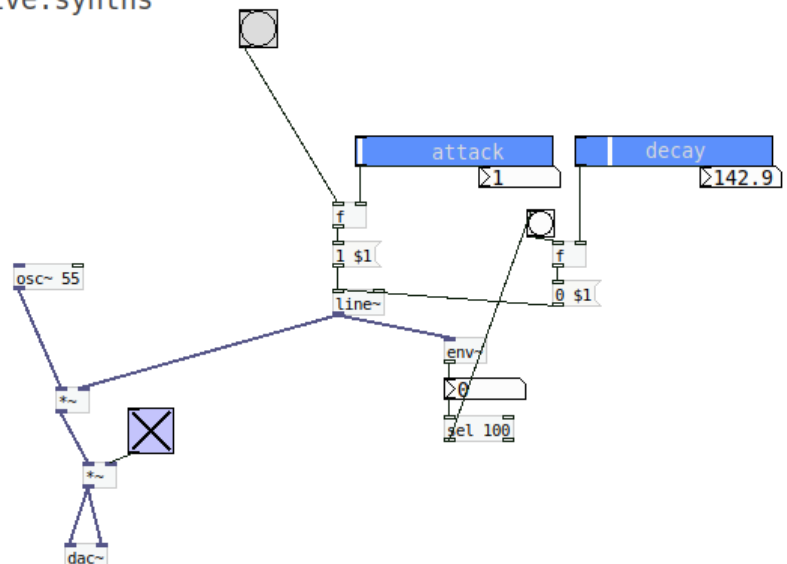
With those objects we can produce signal that after we can control in its *decay* with the object **[line~]** that produces a ramp or progression between two values. In this case message **[1, 0 \$1]** means that any time the trigger is activated, the envelope of the sound will go instantly to the maximum (1), and then goes to (0) (silence) in \$1 milliseconds. As \$1 is a variable, means that any value we are dynamically changing it will be considered as \$1 (in this case with the blue slider).

percussive.synths



Also we can complex a bit the algorithm in the line~ section, introducing both values : one for *attack* and the other for *decay*.

percussive.synths

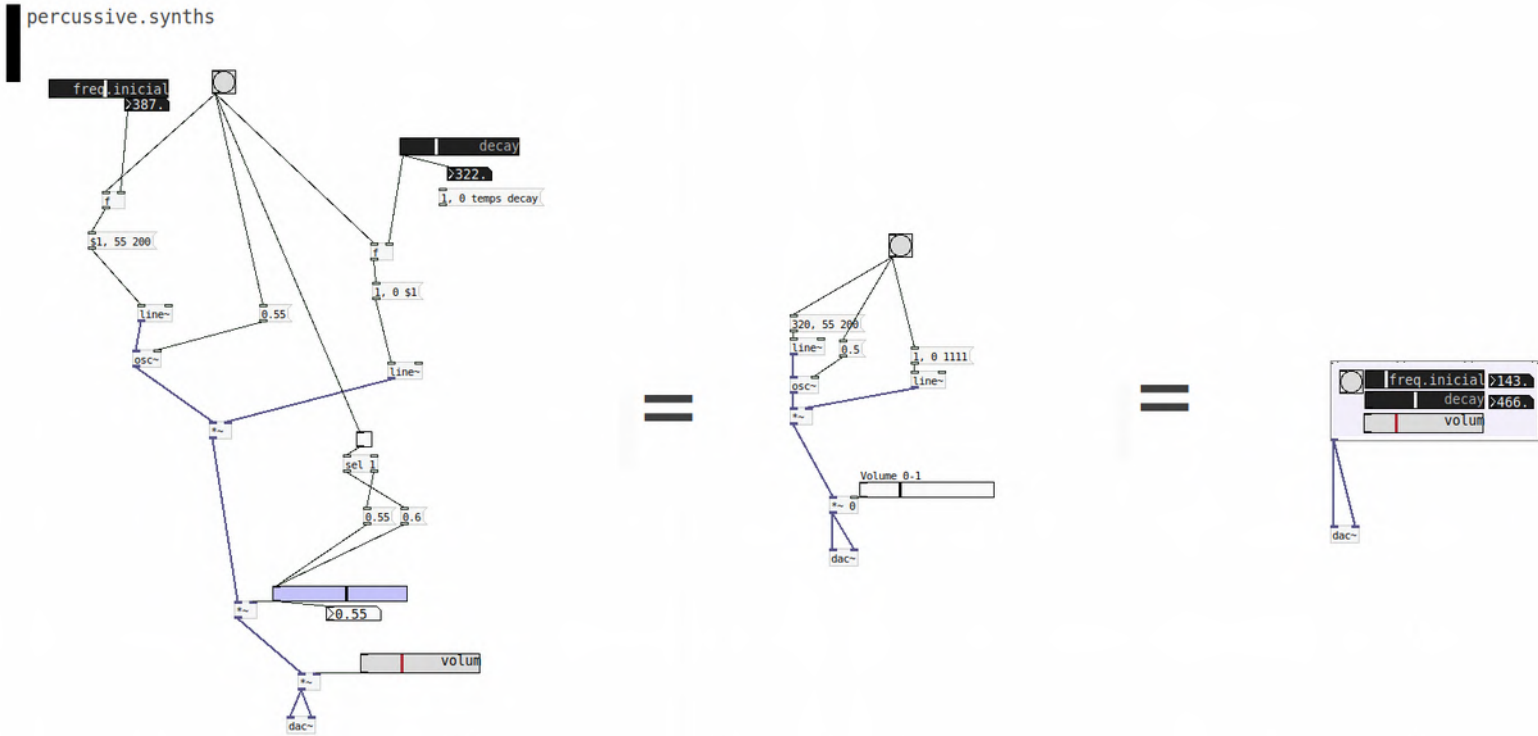


The next figure features the same algorithm in different compacted forms. On the left the whole algorithm, in the

center a simplified one, and on the right an encapsulated method with graph [menú put > graph] that we saw in **2.2. Pd Packaging Algorhythms**

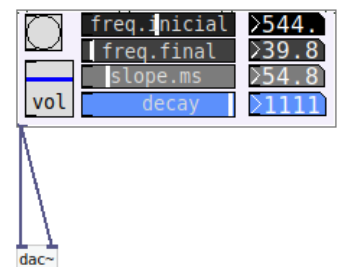
Notice that line is not only applied to envelope to control decay time, but also to slide in time the incoming frequency of the oscillator producing a *portamento* effect. In this case with the message [\$1, 55 200[linked to **line~** in the left version, or [320, 55 200[in the center version.

In this last case, message **[320, 55 200[** indicates the start frequency (320hz), the target frequency (55hz) and the sliding time between both frequencies (200ms).



percussive.synths

In this next figure >>>>> is shown a compacted version of a percussive synth where we can control *initial freq*, target or *final frequency*, sliding time (*slope*), and *decay* time. Therefore a pretty compact and versatile module to trigg percussive sounds.



Sequencers

8. Time Machines

A Sequencer is one of the most essential tools in the electronic music production. As you may know a sequencer is somekind of a time machine or time engine, because it builds narrative structures in time.



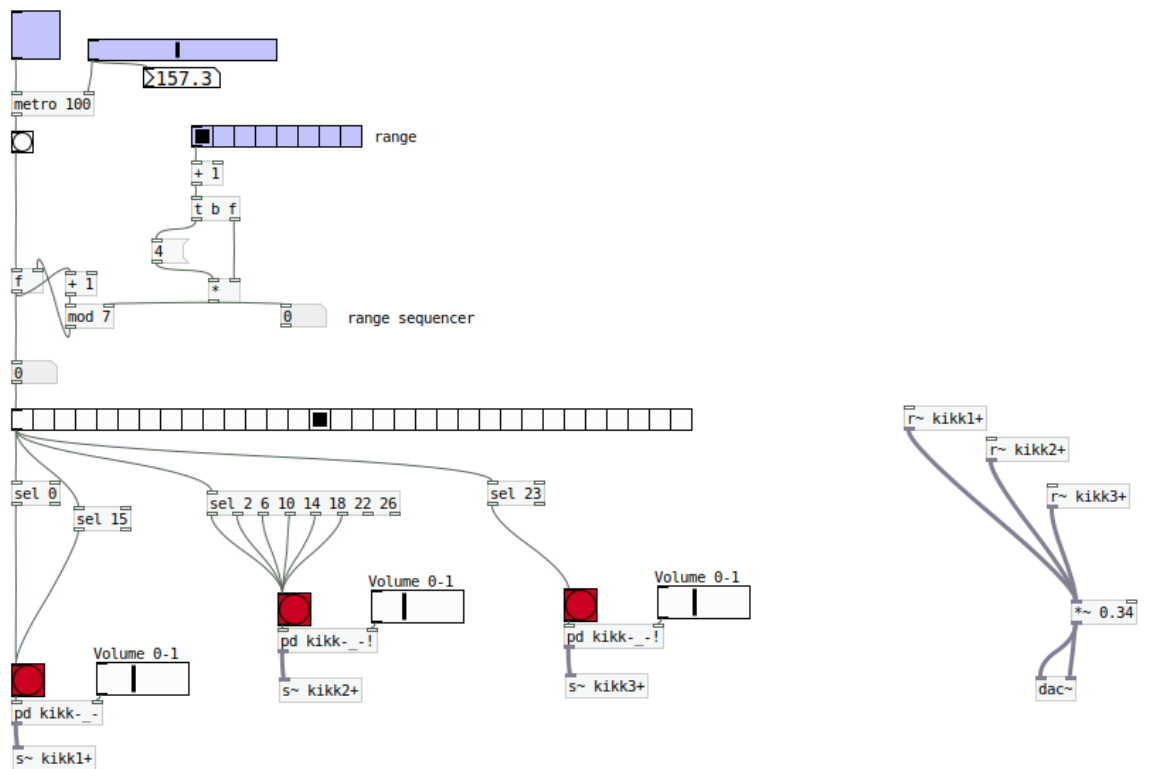
In the programming domain, this kind of structures can be pretty different, depending on the events that we want to create.

8.1 Linear Sequencers

8.1.1 Fixed

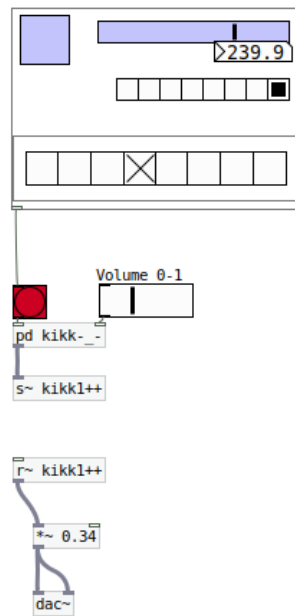
One of the most classic example is a fixed linear sequencer. In this case building an algorithm with a loop section with **[mod]** and a conditional section with **[sel]**

Linear Sequencer : FIXED



8.1.2 8 steps SEQ

Linear Sequencer : 8steps



This example is one of the most basic example of a swith sequencer with 8 steps, that que can swith with on / off any of those 8 steps.

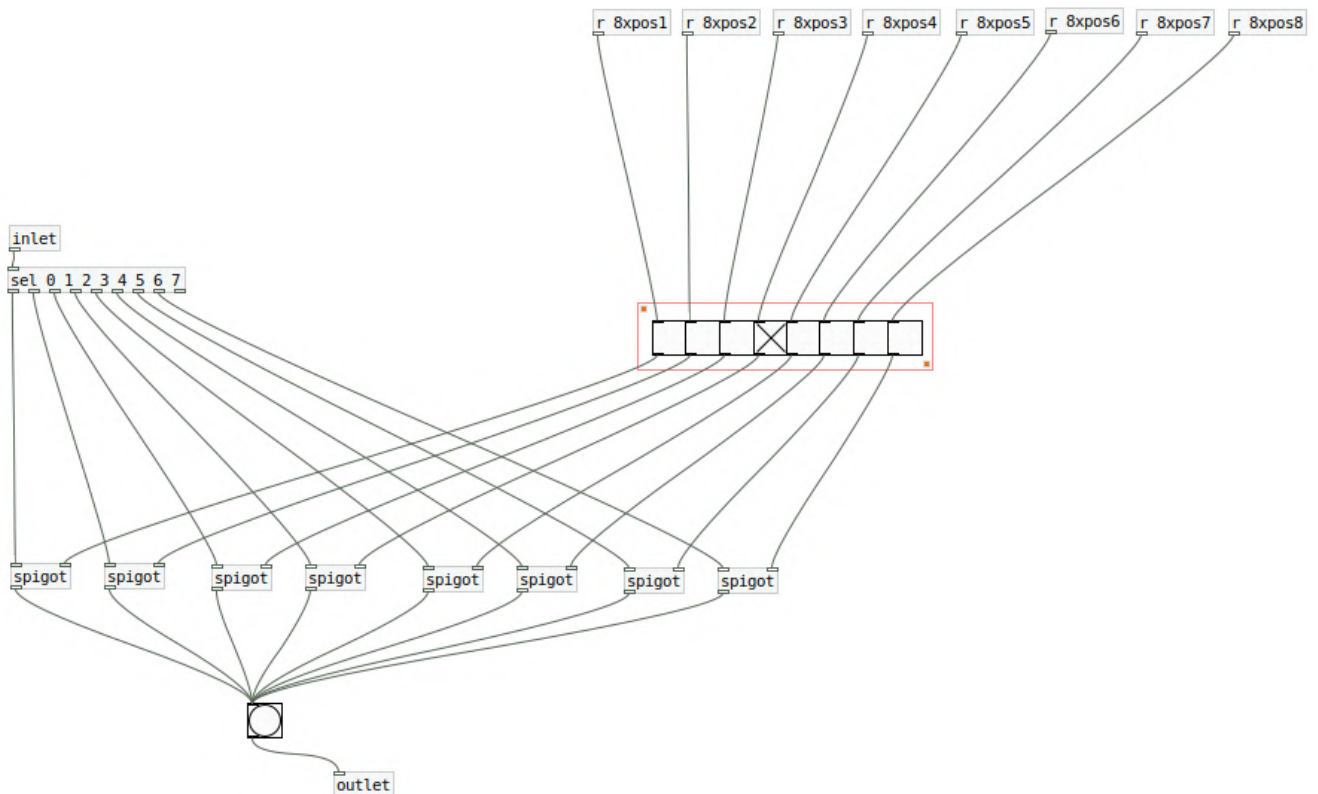
We'll need **[spigot]** object that opens or closes the gate* and also its receive messages for each step.

[s 8xpos1][r 8xpos1] [s 8xpos2][r 8xpos2] [s 8xpos3][r 8xpos3] and so forth

Notice that build this structures is somekind like a knit work, due to repeating a certain unit structure.

So if you want to build a 16 or 32 steps sequencer is simple although a meditative work)

**of the incoming trigger that already has to be sequenced in position with conditionals [sel 0 1 2 3 4 5 6 7],*



8.1.3 8 steps SEQ Phrases

This example creates a different stored pattern combination for every loop.

For do that we have to send and receive a 1 or 0 value in the ID message:

[s 8xpos1][r 8xpos1] [s 8xpos2][r 8xpos2] [s 8xpos3][r 8xpos3] and so forth

(the receive messages are featured in the previous image on the top)

With the combined messages

```

;
8xpos1 1;
8xpos2 0;
8xpos3 0;
8xpos4 1;
8xpos5 0;
8xpos6 1;
8xpos7 0;
8xpos8 0;

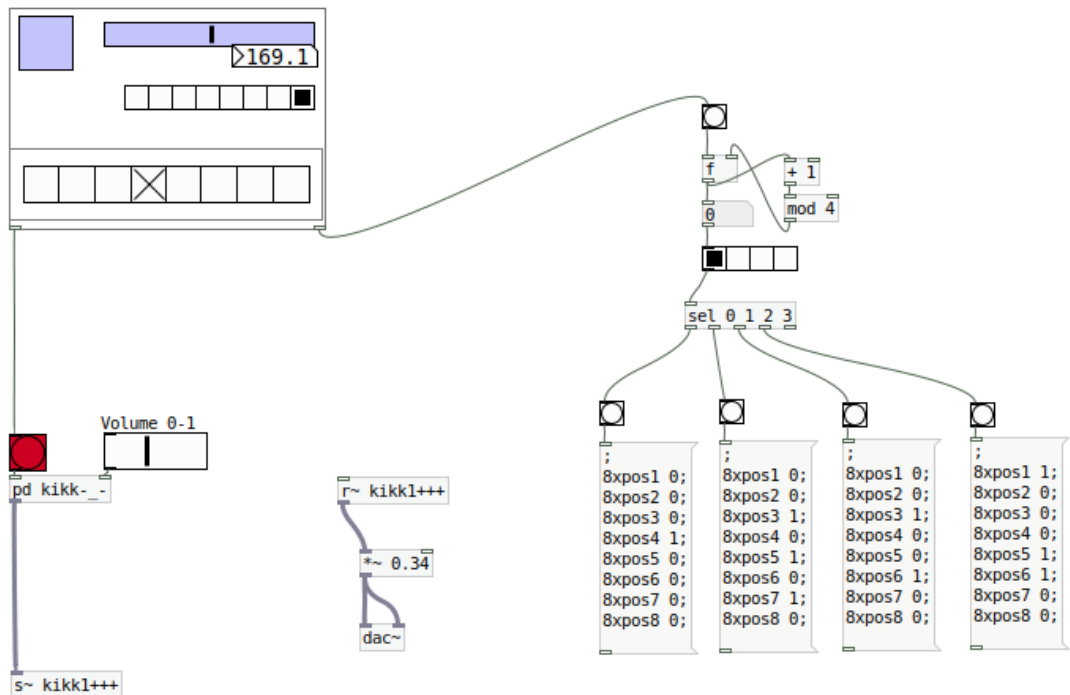
```

is possible to set a particular combination of the whole 8 steps.

Therefore we can make a tree of patterns that are triggered every time the main loop starts.

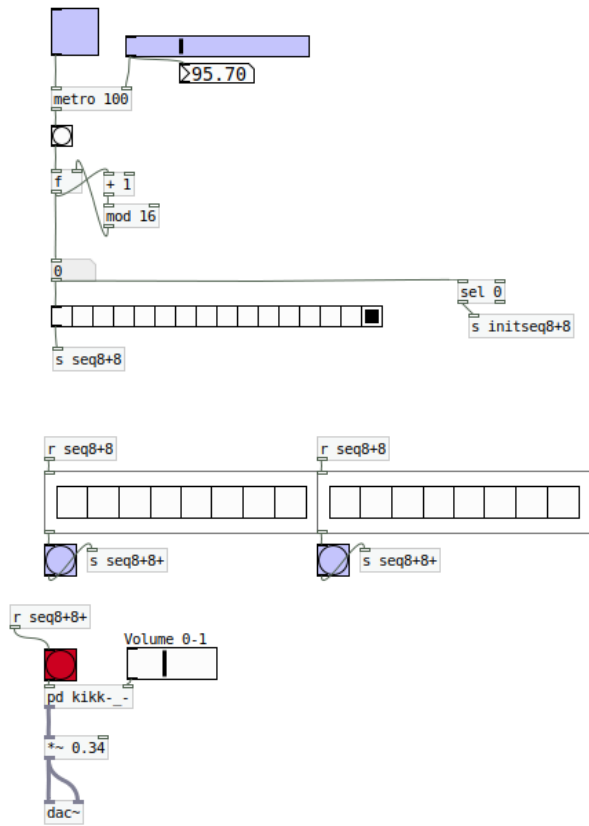
In this case we have a tree of 4 different patterns but we can build some more complex structures in time in a pretty easy way.

Linear Sequencer : 8steps Phrases

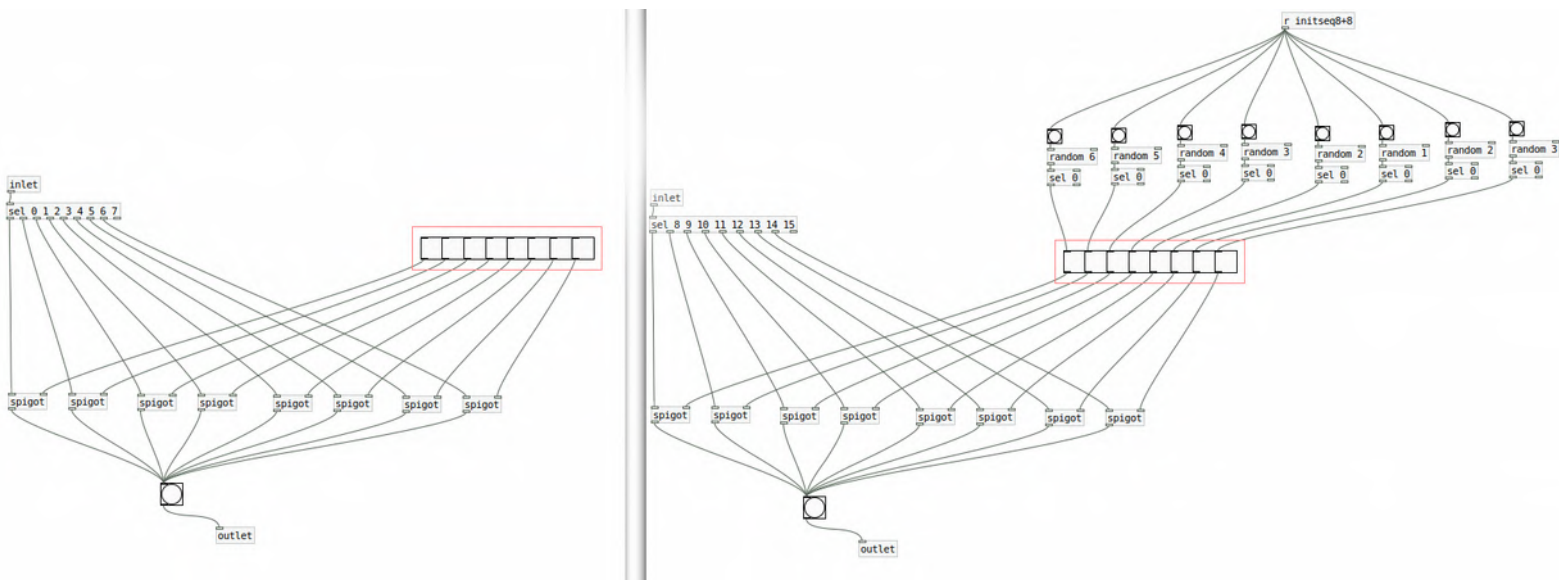


8.1.4 8 steps SEQ Fixed + 8 steps SEQ Random

Linear Sequencer : 8steps FIX + 8Steps random



This example combines a 8step fixed sequencer where we can manually select the desired active step mixed up with a random 8 step sequencer with different proportions of random that can be tweaked or cancelled for a certain and desired randomization.



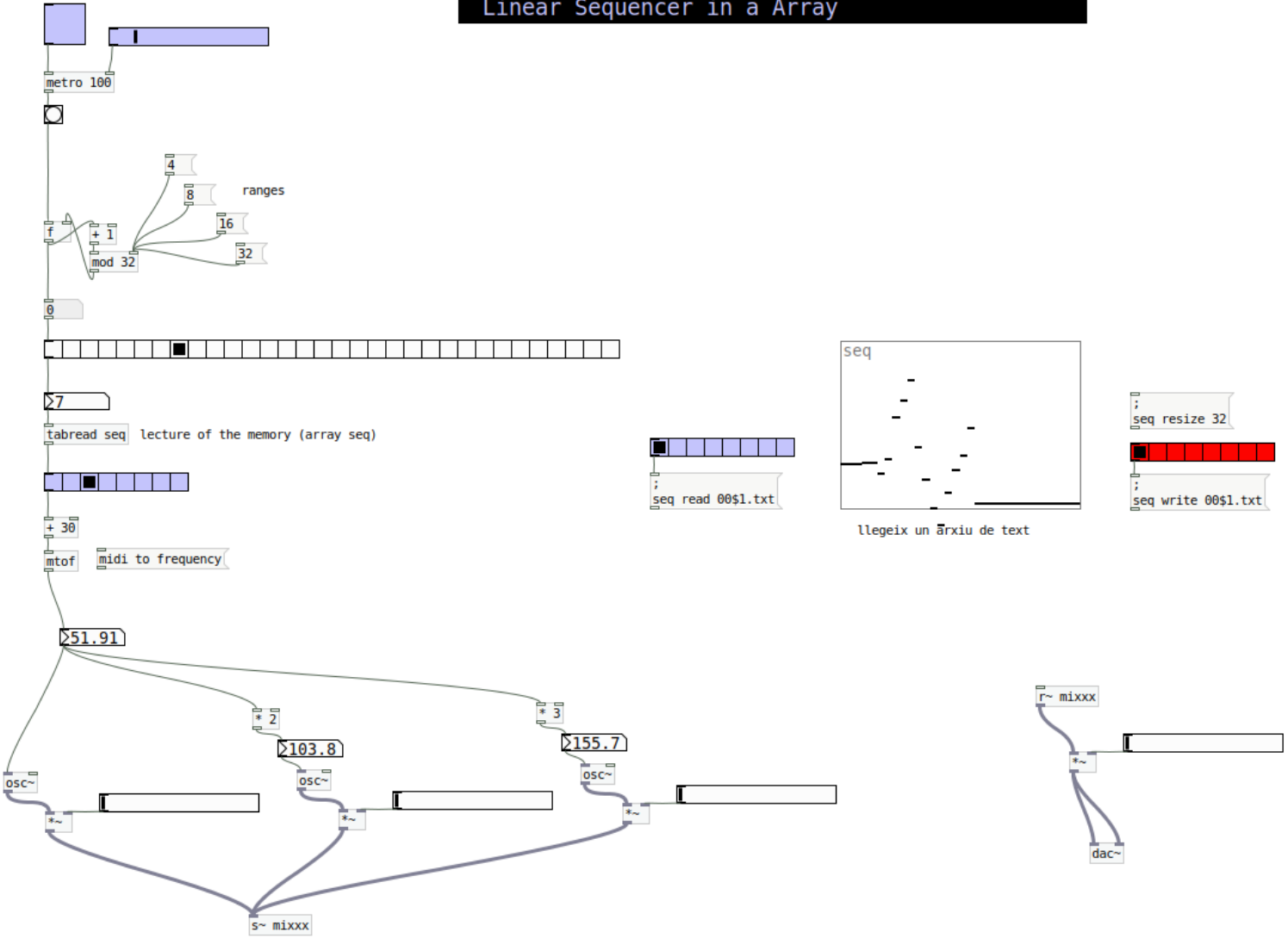
8.2 Sequencers in Arrays

Another method to define sequences is with Arrays.

In this case a looper is counting the content of an array with [tabread seq] that in this case trigs the stored value as a main frequency of a synth with 3 superior harmonic oscillators.

Notice that the values of the array in this case are refered to a midi values. Therefore with the object [mtof] miditofrequency we can translate values to properly income in [osc~] objects

Linear Sequencer in a Array



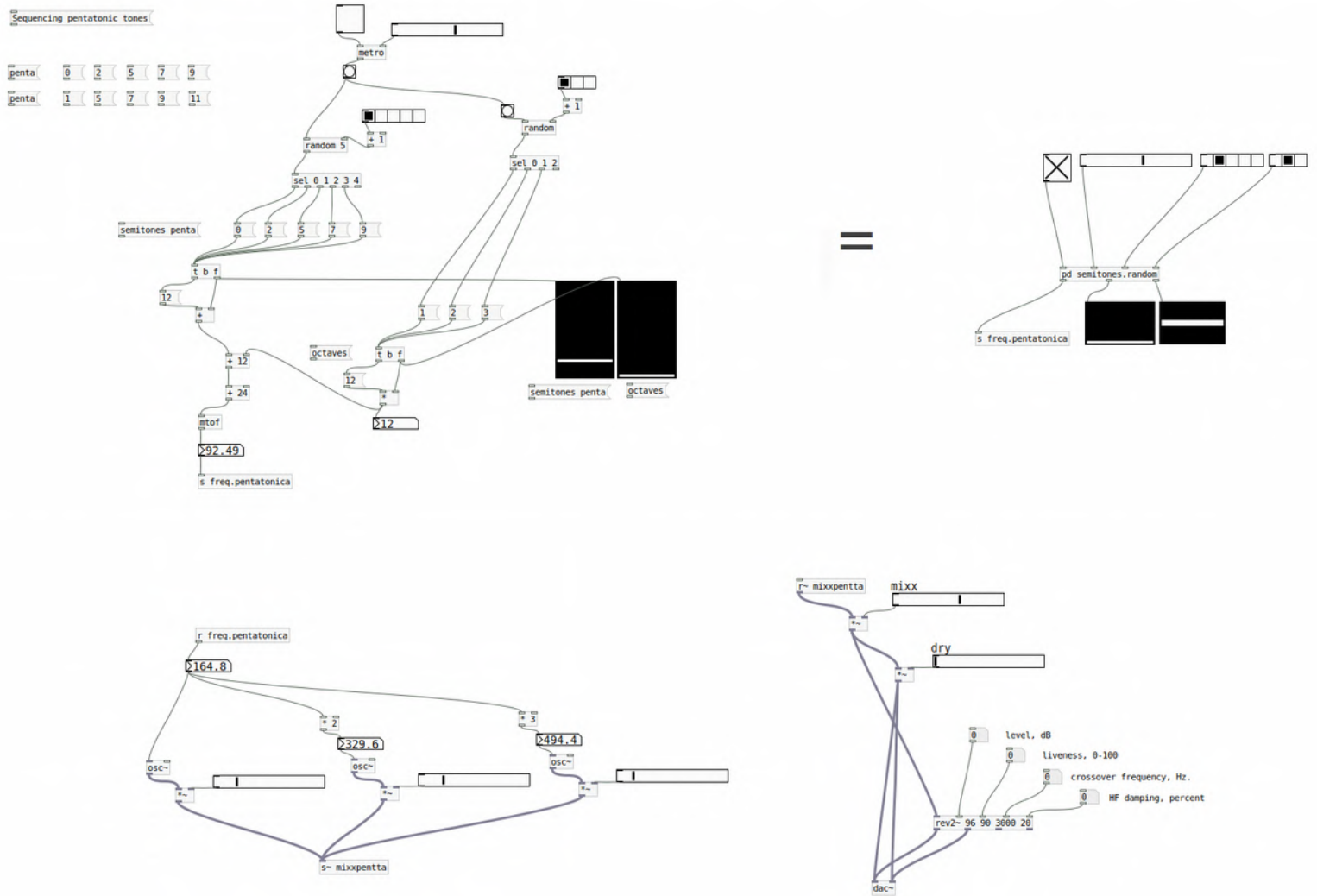
8.3 Random Sequencer with Pentatonics

When we are programming we can build nonconventional methods to build sequences.

For example, in this case a defined metronome is triggering a random value between those values that corresponds to a pentatonic scale combination. In the code the semitones relation [0 [2 [5 [7 [9 [

Usually playing with random is not a straight forward task for nice sonic designs, but in this case we have the advantage that between pentatonic tones every tone matches ‘harmonic’ with the others. Therefore any random combination between those semitones will be ‘audible-comfortable’.

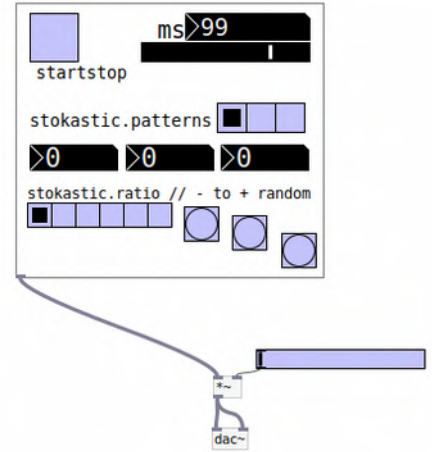
Random Sequencer with Pentatonics



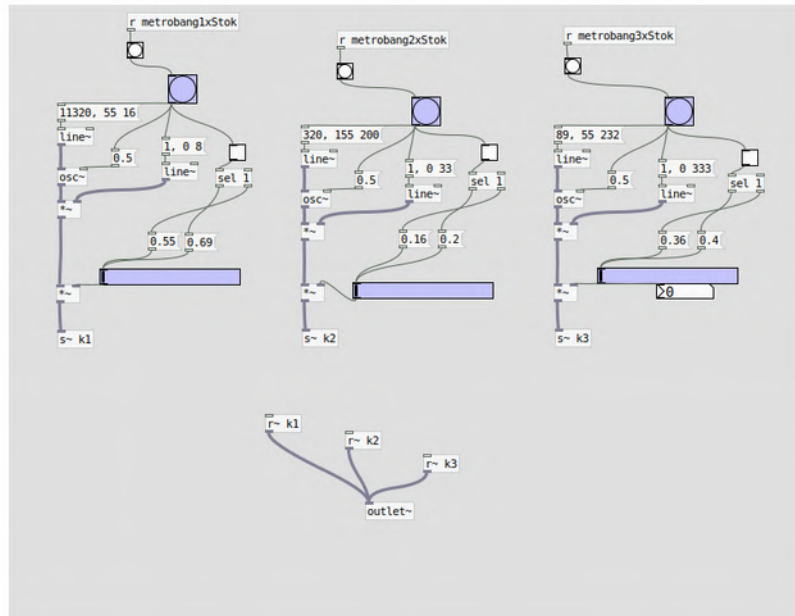
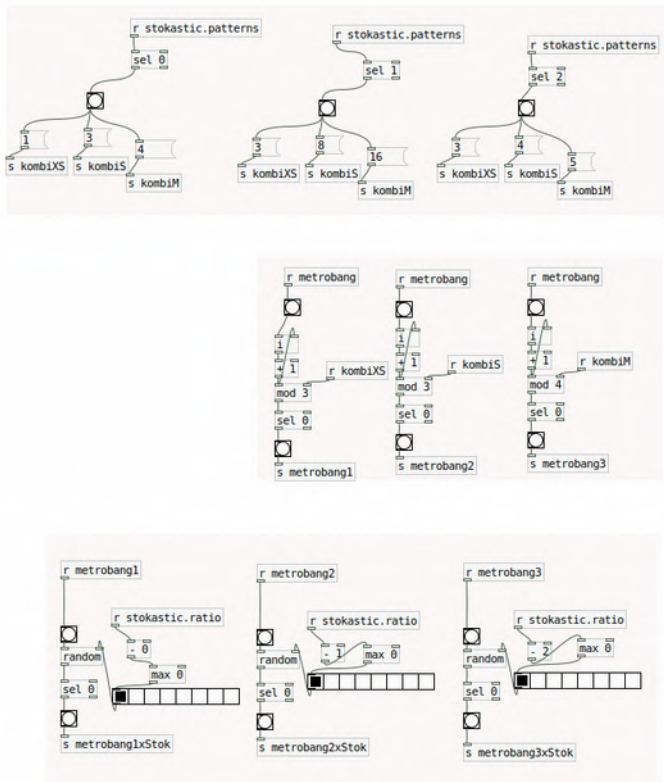
8.4 Stokastic & Probability Sequencers

Probabilistic (Stokastik) Sequencer

Another non conventional method, borrowed from Xenakis researches decades ago in probabilistic sequencers / synths, is this tiny example of 3 triggers (with 3 embed percussive sytnhs) that can be both triggered in 3 types of combinations and amount of stokastic ratio (from less random to more random probabilities that triggers are activated or not.)



Here the internal code with all probability operations.

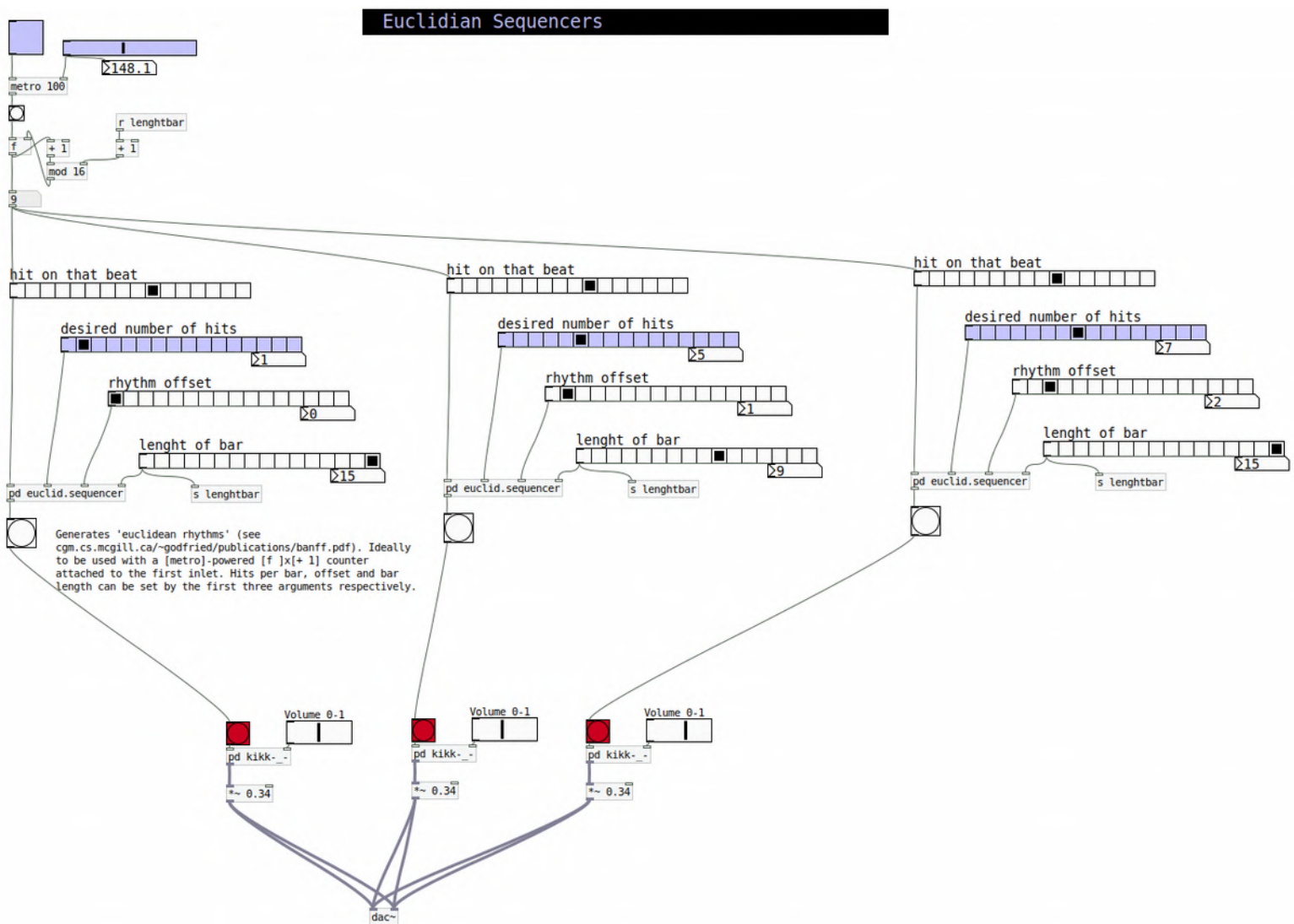


8.5 Polyrhythms : Euclidian Sequencers

A very interesting method for build polyryhtms is using Euclidian Sequencers.

As a reminder, Euclidean rhythms have their roots in Greek mathematician Euclid's algorithm and involve using the greatest common divisor of two numbers to place hits in a sequence as evenly as possible across a set timing division.

This technique allows us to build patterned and organic sequences due to the fact to being manage different ranges of sequences 'geometrically'. Therefore unusual beat combinations like 5/4 7/8 8/9 can be easily combined as well with the most common static patterns 4/4 2/4 etc, in order to build unconventional but interesting beats.



HIDs

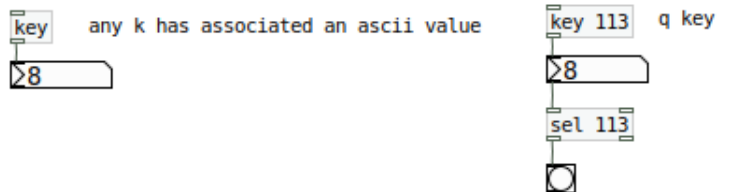
10.Human Interface Devices

Useful functions and methods for Sonic Interaction

10.1 keyboard control

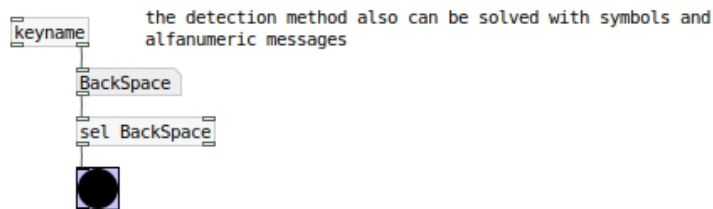
In pd we can easily manage laptop's keyboard as a triggers or switches.

We can use the object **[key]** which corresponds to a numeric value codified in Ascii. Every key has a certain number or alphanumeric description. In the first case we are using **[key ascii number]**.

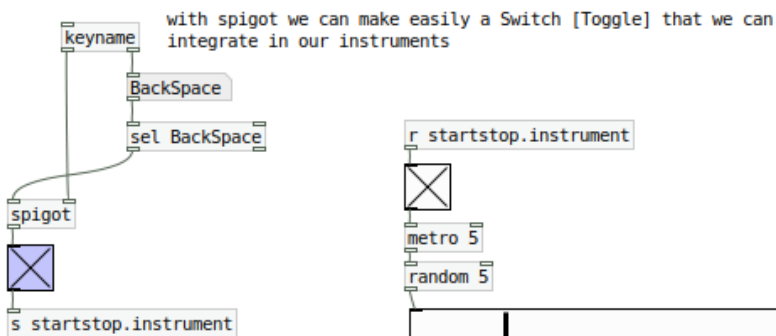


In the second we are using **[keyname]** attached to a symbol.

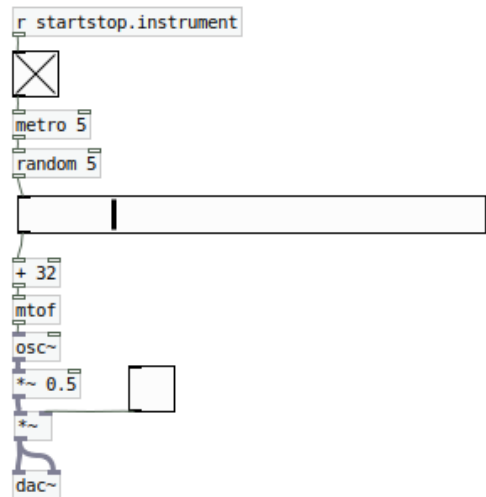
Adding a [sel] with the incoming specific number or symbol we want to control we can easily have a trigger.



Also with spigot we can transform the simple trigger into a switch. See on the right >>



This is useful to switch on/off certain algorithms we want to launch in time, like the most on the right section in this image >>>>



Another feature using keyboard is [keyup]. That's not so efficient for triggers, but maybe for some special functions we need to control as soon as certain key is released can be ok.

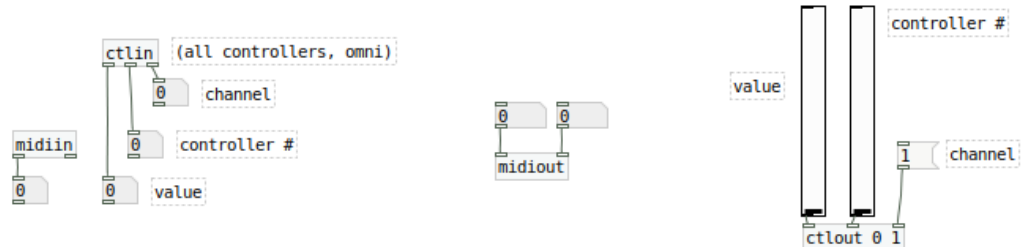
Midi

11 Midi controllers

MIDI protocol is obviously integrated in Pd.

Usually is used for Midi in messages from a midi controller, but also can be used in both directions sending also midi messages to another devices.

For input methods we have **[ctlin]** which extracts three outlets : the current midi value, the controller ID of a certain key knob or slider, and the channel (default 1).



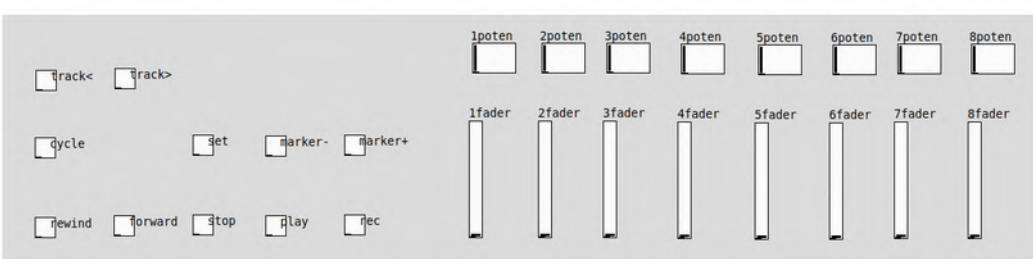
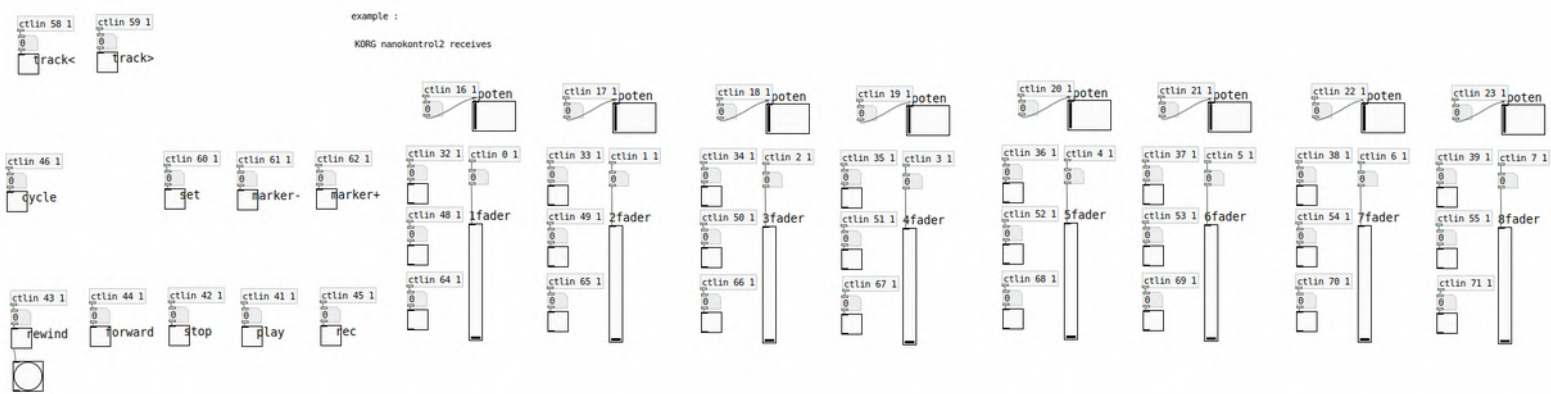
Also this method can be written with the syntax **[ctlin ID 1]** that directly extracts the current midi value in its outlet (check next example).

For output methods we have **[ctlout]** which in an opposite way as **ctlin** integers three inlets : the current midi value, the controller ID of a certain key knob or slider, and the channel (default 1).

Note that to manage different midi IO devices those have to be set up in the preferences menú (menú Edit / Preferences).

As an example in this tutorial we have the midi mapping for a korg nanokontrol 2 device. In case you have onther controller is very useful to make a mapping patch of it, in order to use with several instruments we want to develop.

Ex korg nanokontrol2 mapping



3

HOW 2 Load and Build Pd Patches into LICH

Tutorial by Xavi Manzanaras
by-sa // 2021

*LICH Module by ://
Befaco & Rebel Technologies*

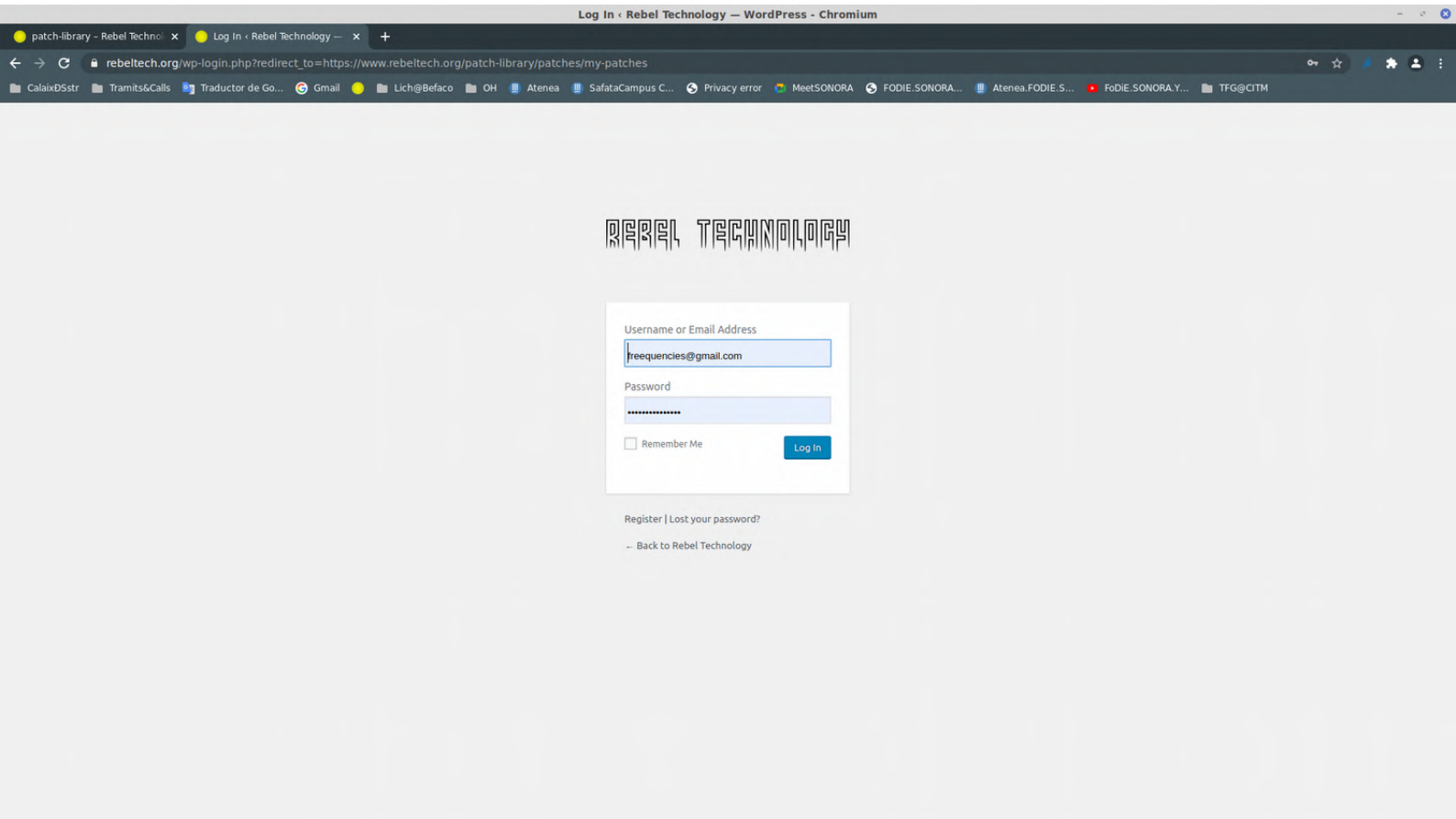
PD
FOR
LICH
A
PATCH
WITH
pd~



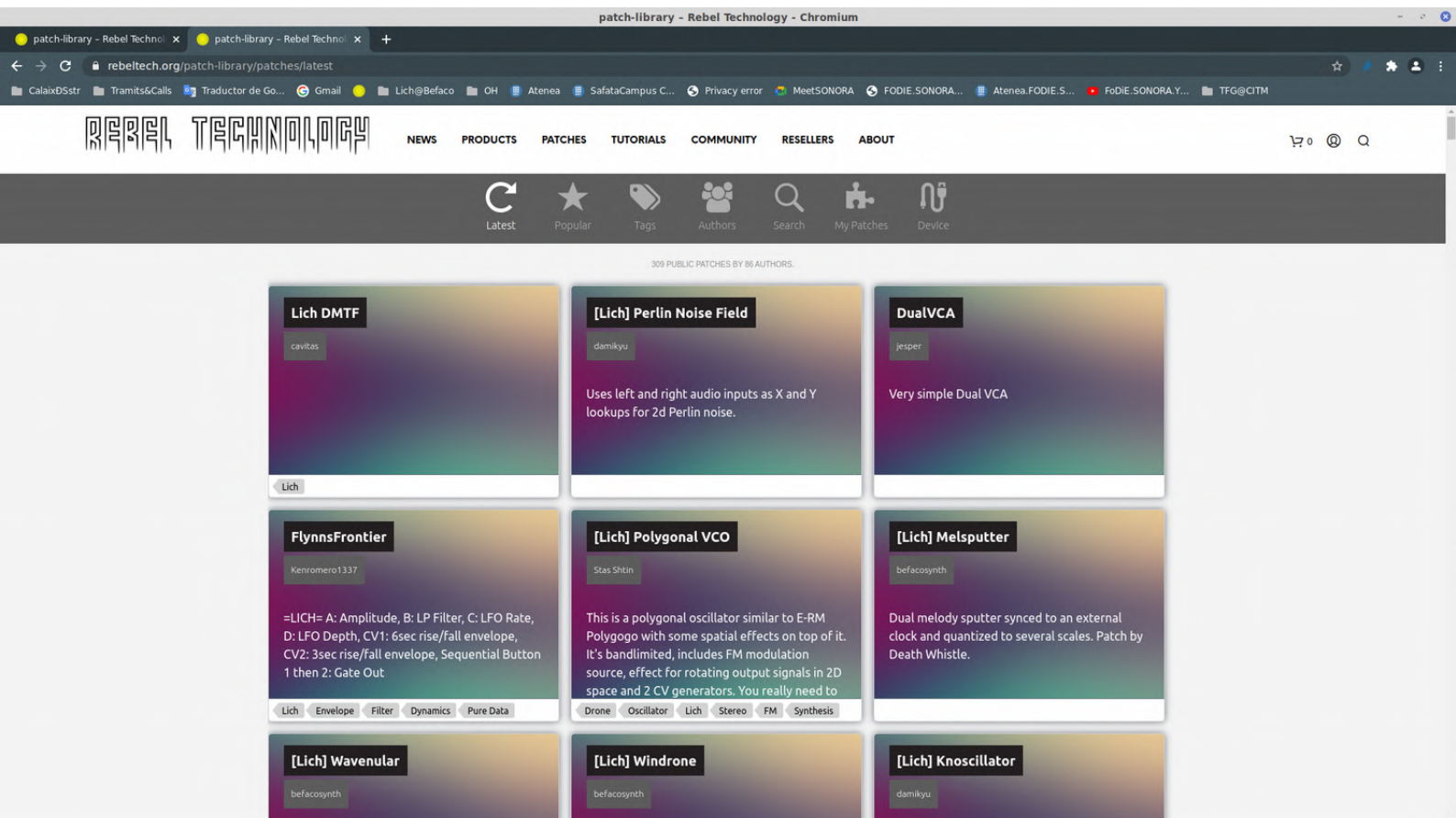
HOW 2 Load and Build Pd Patches into LICH Firmware

Enter at <https://www.rebeltech.org/patch-library/patches/latest>

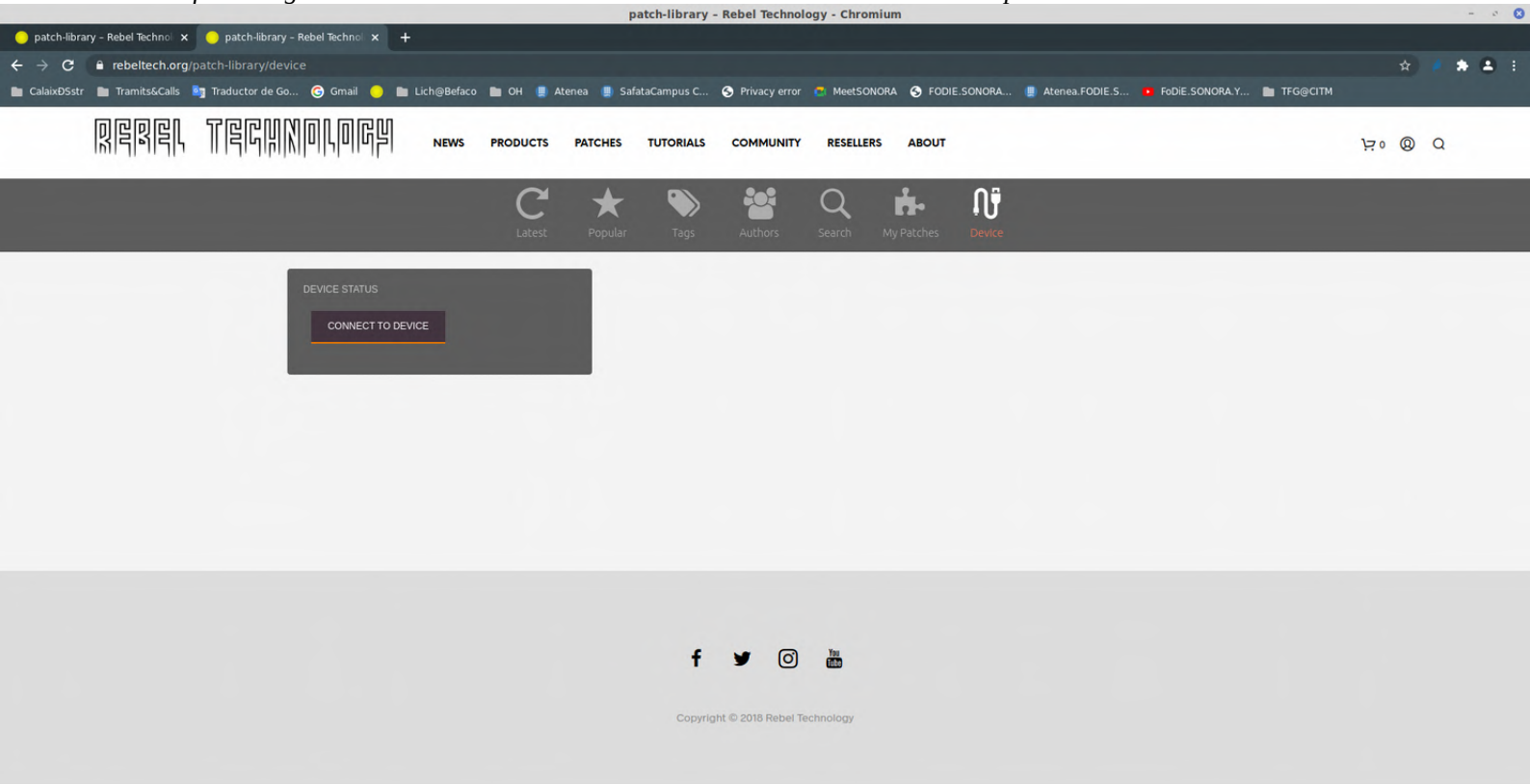
Will ask you for your login in order to contribute to the public repos and developments for Lich.



Once registered you'll enter into this interface showing the last developments done by the community.

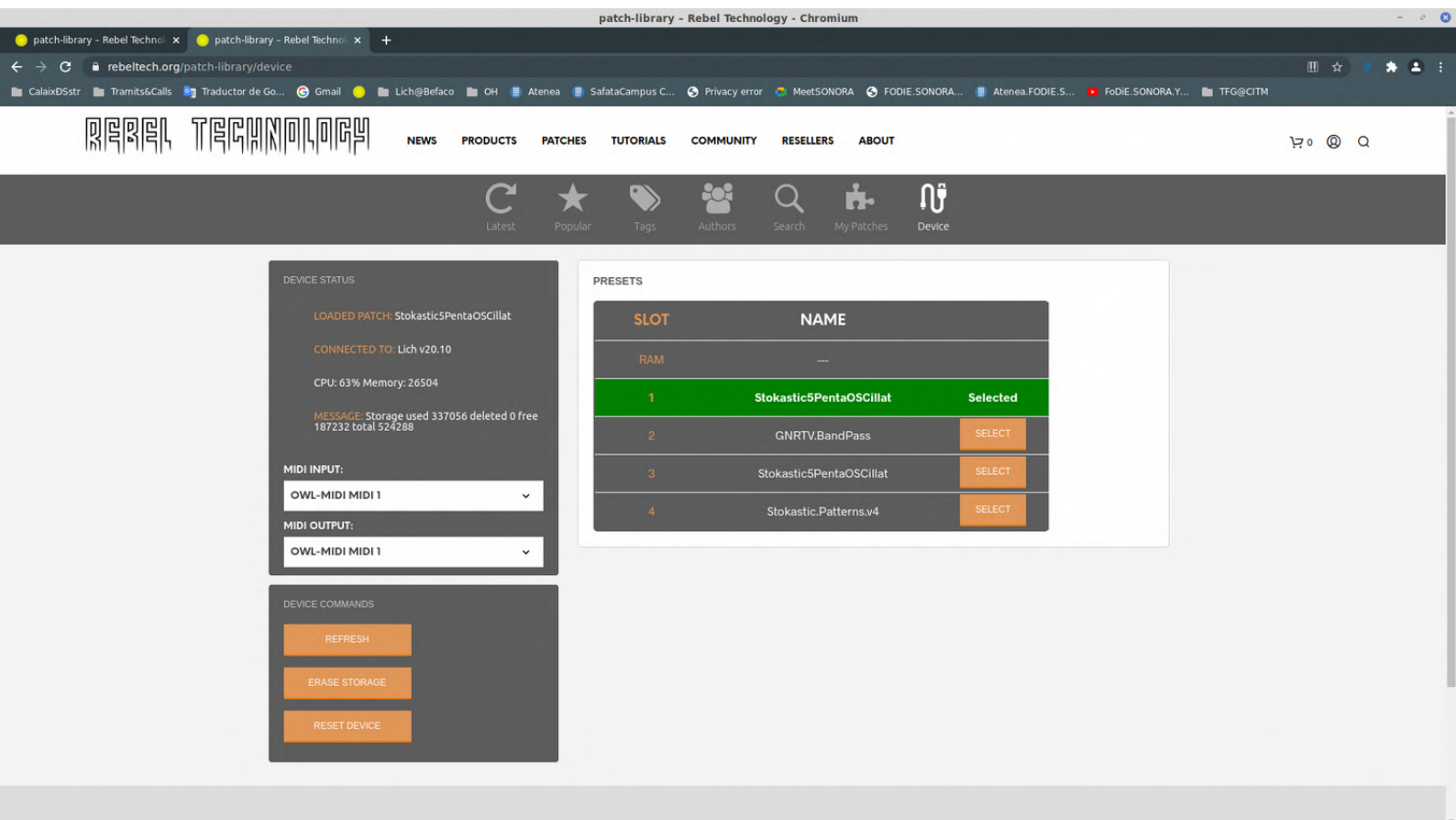


Let's imagine we have build a patch in the computer and we want to upload it into the LICH firmware.
The first thing is to turn on the LICH module and connect it via USB to the computer.

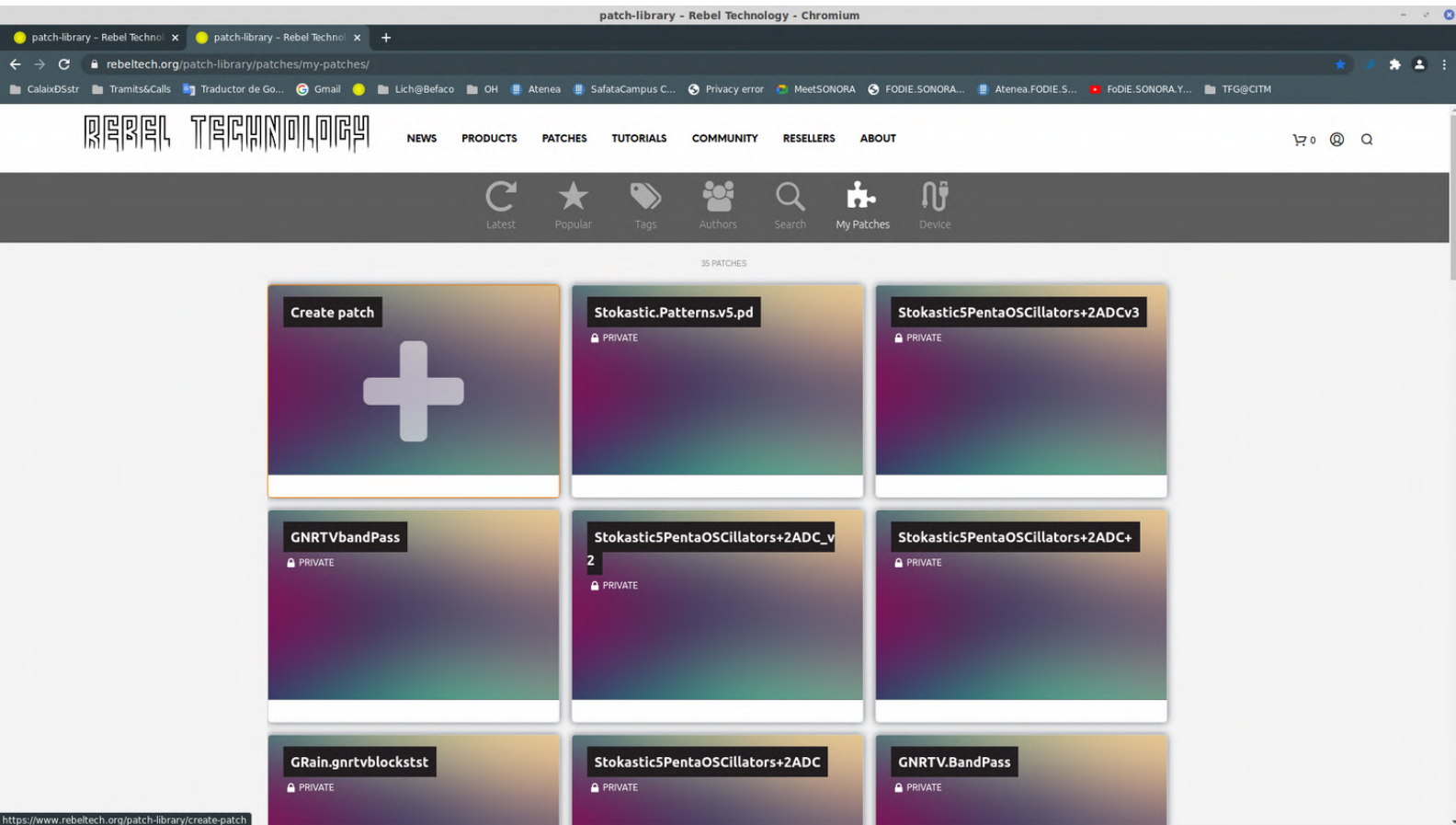


Note : You'll need a browser that supports Web MIDI (for ex. Google Chrome) due to the fact that the RebelTechnology's Browser tool requires the MIDI protocol to talk with Lich firmware. In case you still don't have set it up, a pop-up will ask you if you want to allow communication permissions of Midi devices within the browser.

Once connected, there appears the algorithms already loaded in LICH firmware.

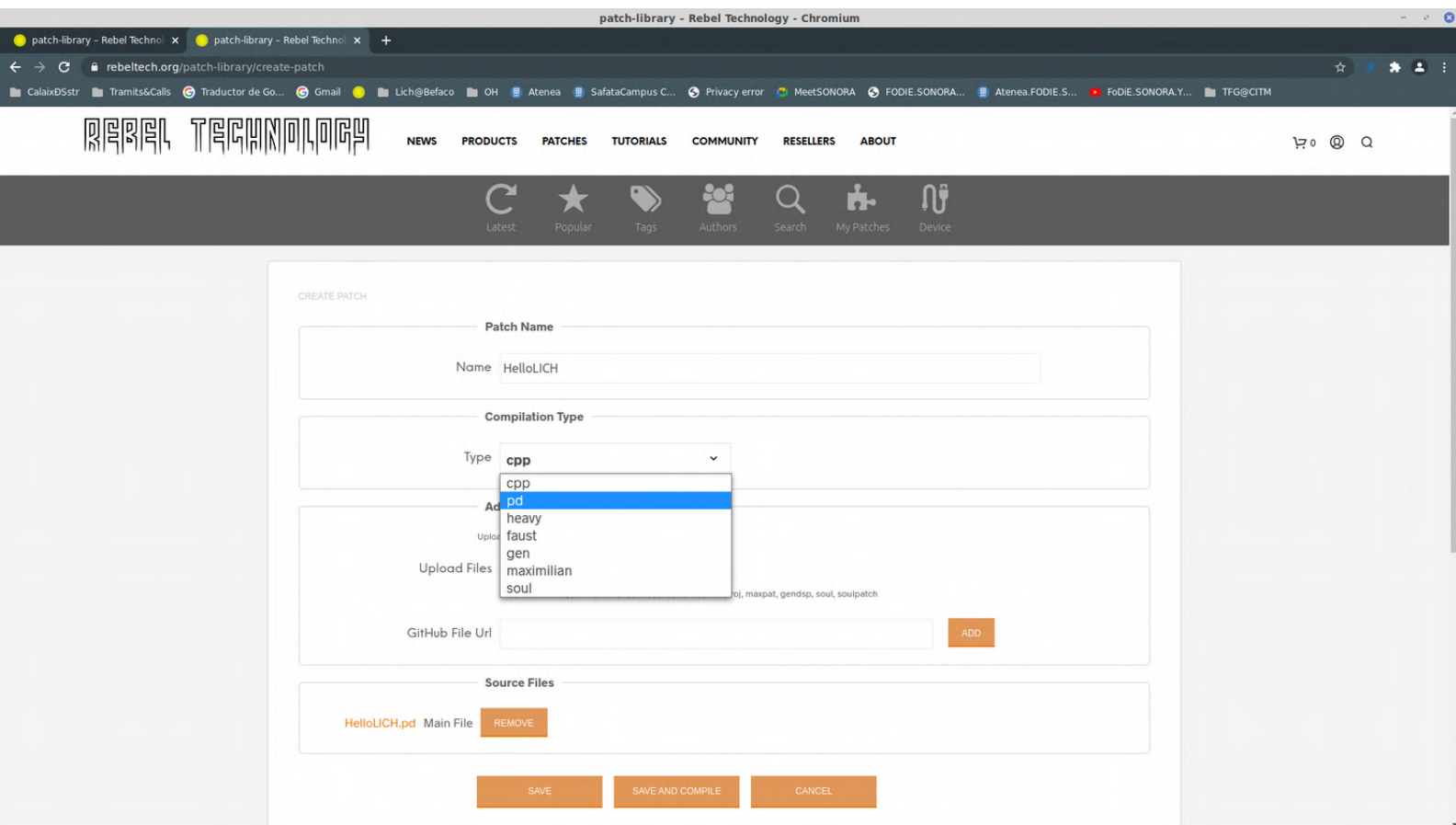


In order to load your patch from the computer, click **MyPatches** section, and **Create Patch**

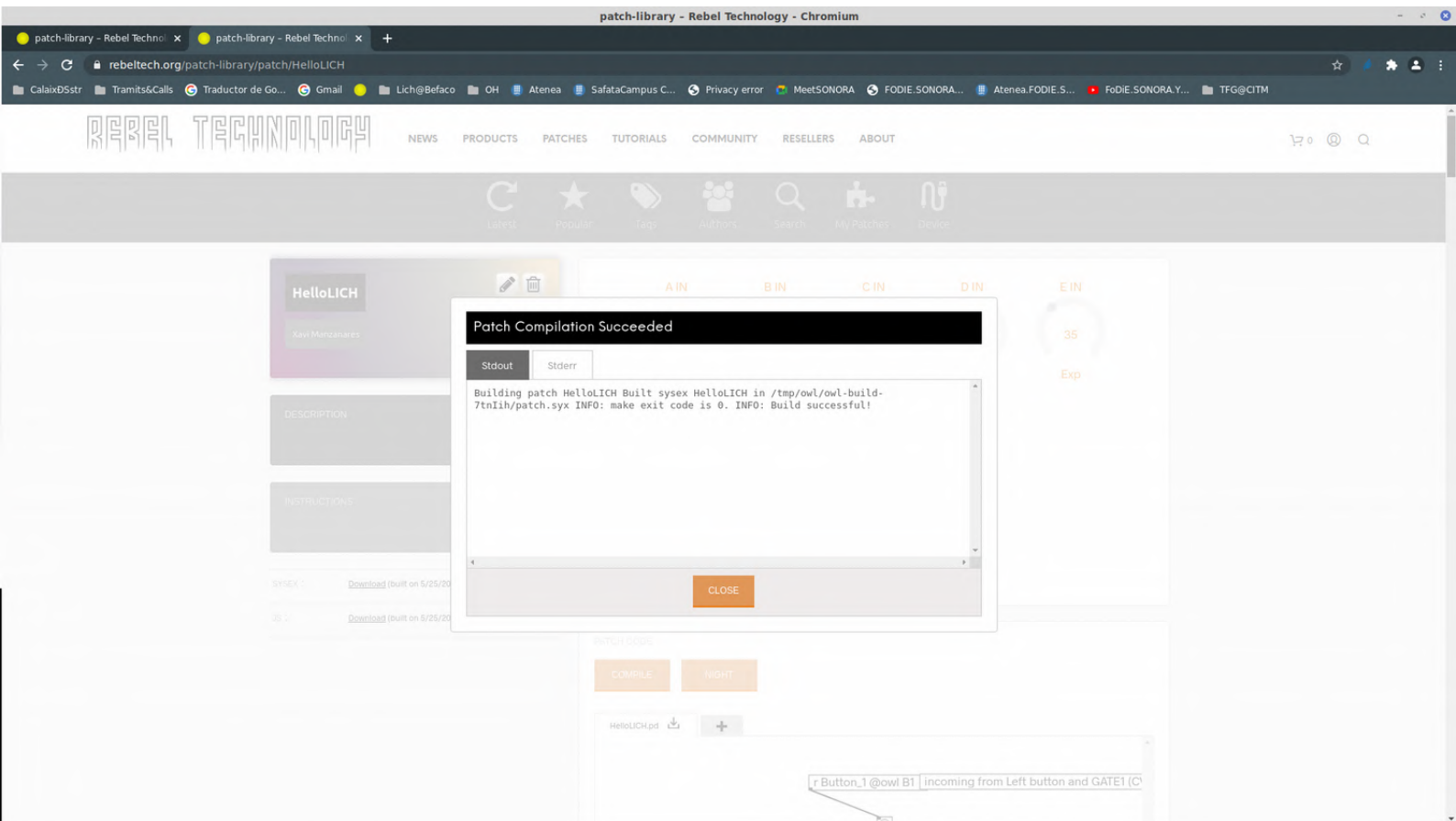


It appears an interface to Upload the Alorythm* (in our case Pd). We can write the name we want, click type **pd** in the **compilation type** after we'll browse the patch in our computer in the **upload files** section. Alternatively we can add the **github's file url** of the patch instead of upload it from our computer. Then we can click **'Save and Compile'** button in the bottom.

* can be different codes programmed in C++, pd, heavy, faust, gen, maximilian or soul.



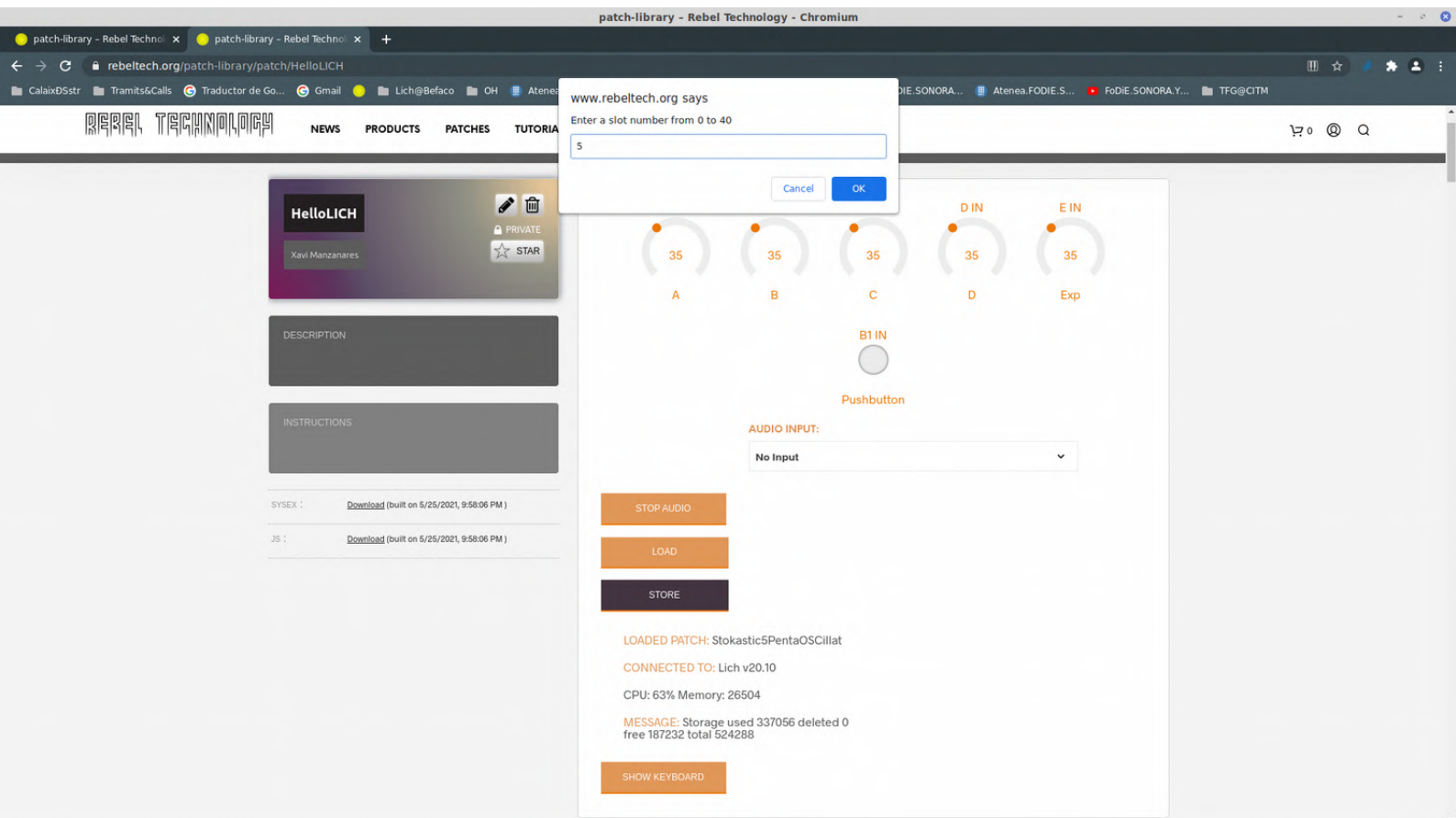
Few seconds later of being pressed **'Save and Compile'** button, will appear a pop-up informing about the compiling process.



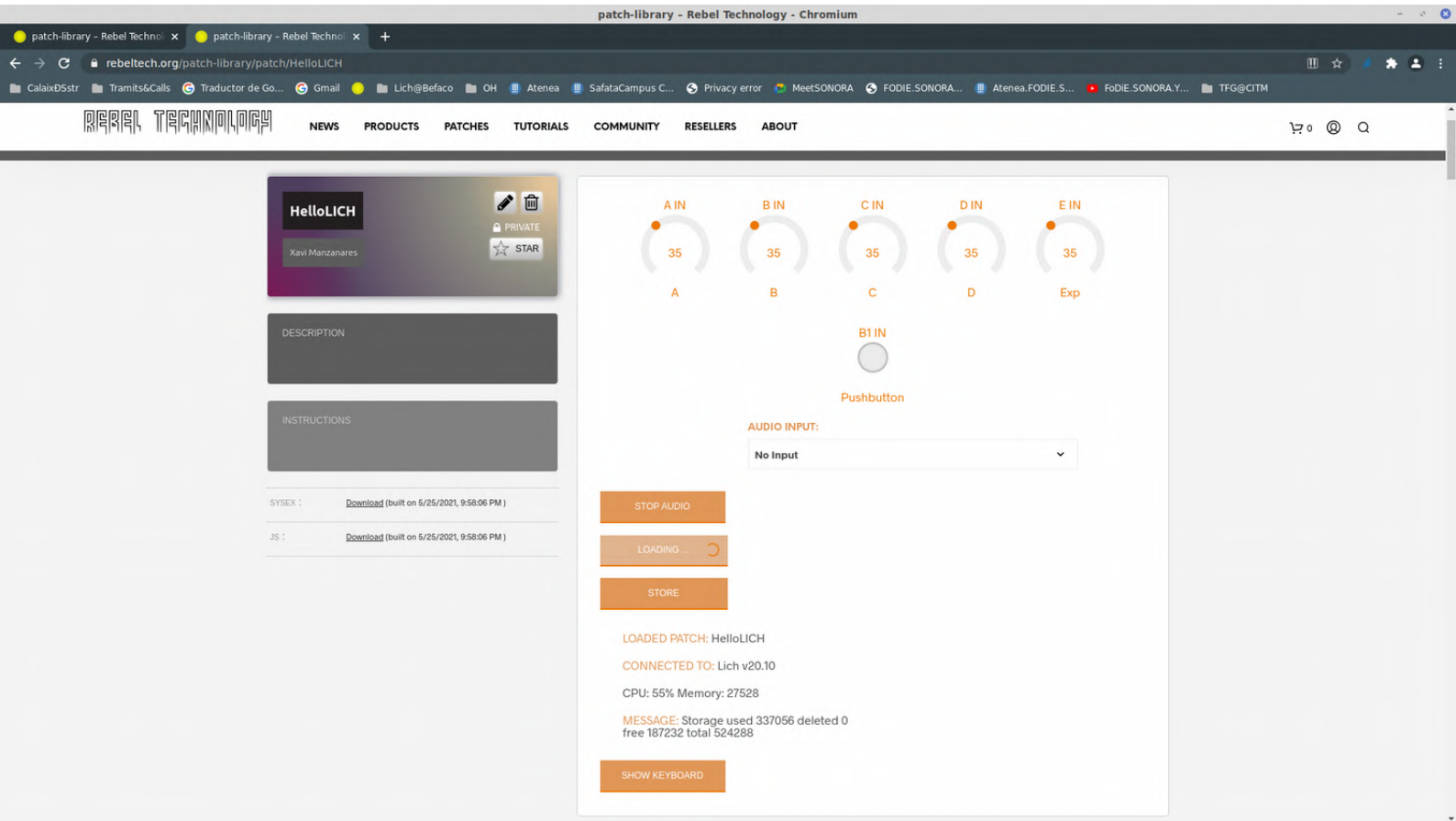
In case there is an error, pd patch will not be loaded into LICH and we have to re-program and debug our patch in the computer, following the clues and errors described in the shell.

*Otherwise, in case it appears **'Build Successful!'** in the pop-up terminal, we have to Close the pop-up, and complete another tweaks : Click **Store** > It will appear another popup asking us for the slot we want to store the new patch (or code) into the LICH's firmware.*

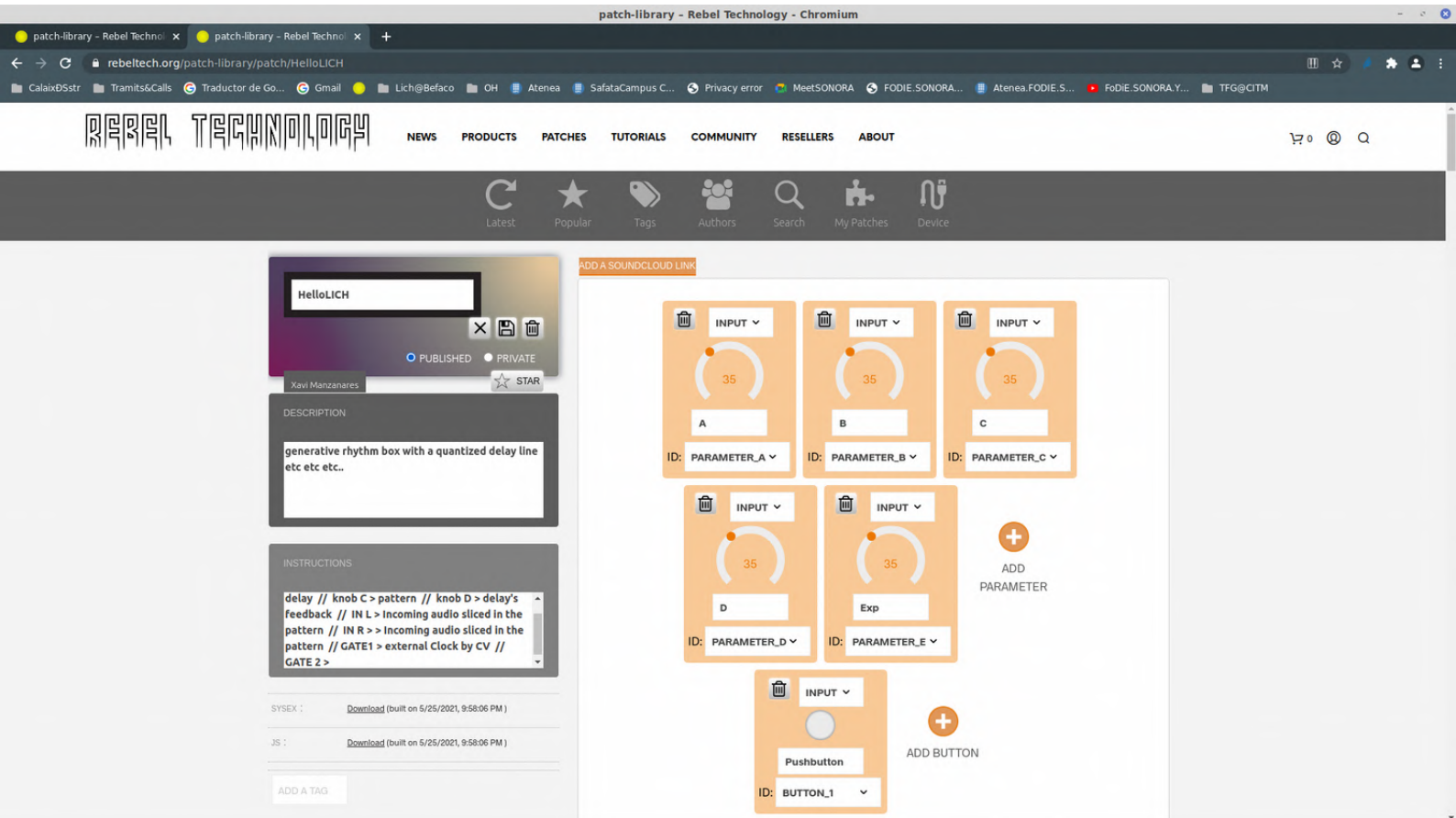
Important note: *in the popup appears slot number from 0 to 40, but I strongly recommend you just store from slots 0 to 9.*



After STORE the number, press LOAD, and if there is not an unexpected error, your pd patch will be running in your LICH



If you like your development and want to publish it and contribute with the community, you can add Concept Description of your Pd patch and Instructions that describes the parameters associated with the LICH's Hardware (therefore Knobs, buttons, Signal Ins, CV Ins, CV outs, Signal Outs).



Once edited the info you can save and publish it your patch.
Don't forget to add some tags in the Description, wich will define more details for other users searches.

The screenshot shows a web browser window displaying the Rebel Technology patch library. The browser's address bar shows the URL `rebeltech.org/patch-library/patch/HelloLICH`. The website's navigation menu includes **NEWS**, **PRODUCTS**, **PATCHES**, **TUTORIALS**, **COMMUNITY**, **RESELLERS**, and **ABOUT**. The main content area is divided into three columns:

- Left Column:** Features a patch card for "HelloLICH" by Xavi Manzanares. It includes a "DESCRIPTION" section with the text "generative rhythm box with a quantized delay line etc etc..." and an "INSTRUCTIONS" section with detailed patch parameters. At the bottom, there are download links for SYSEX and JS files.
- Middle Column:** Contains a "DESCRIPTION" section with the text "generative rhythm box with a quantized delay line etc etc..." and an "INSTRUCTIONS" section with detailed patch parameters.
- Right Column:** Displays the patch's control interface. It features five knobs labeled "A IN", "B IN", "C IN", "D IN", and "E IN", each set to a value of 35. Below these is a "B1 IN" knob and a "Pushbutton". An "AUDIO INPUT" dropdown menu is set to "No Input". At the bottom of this column are buttons for "STOP AUDIO", "LOAD", "STORE", and "SHOW KEYBOARD".

Additional information at the bottom of the right column includes: "LOADED PATCH: HelloLICH", "CONNECTED TO: Lich v20.10", "CPU: 55% Memory: 27528", and a "MESSAGE: Storage used 337056 deleted 0 free 187232 total 524288".

Enjoy the power of building your algorithms and link them with your modular system!





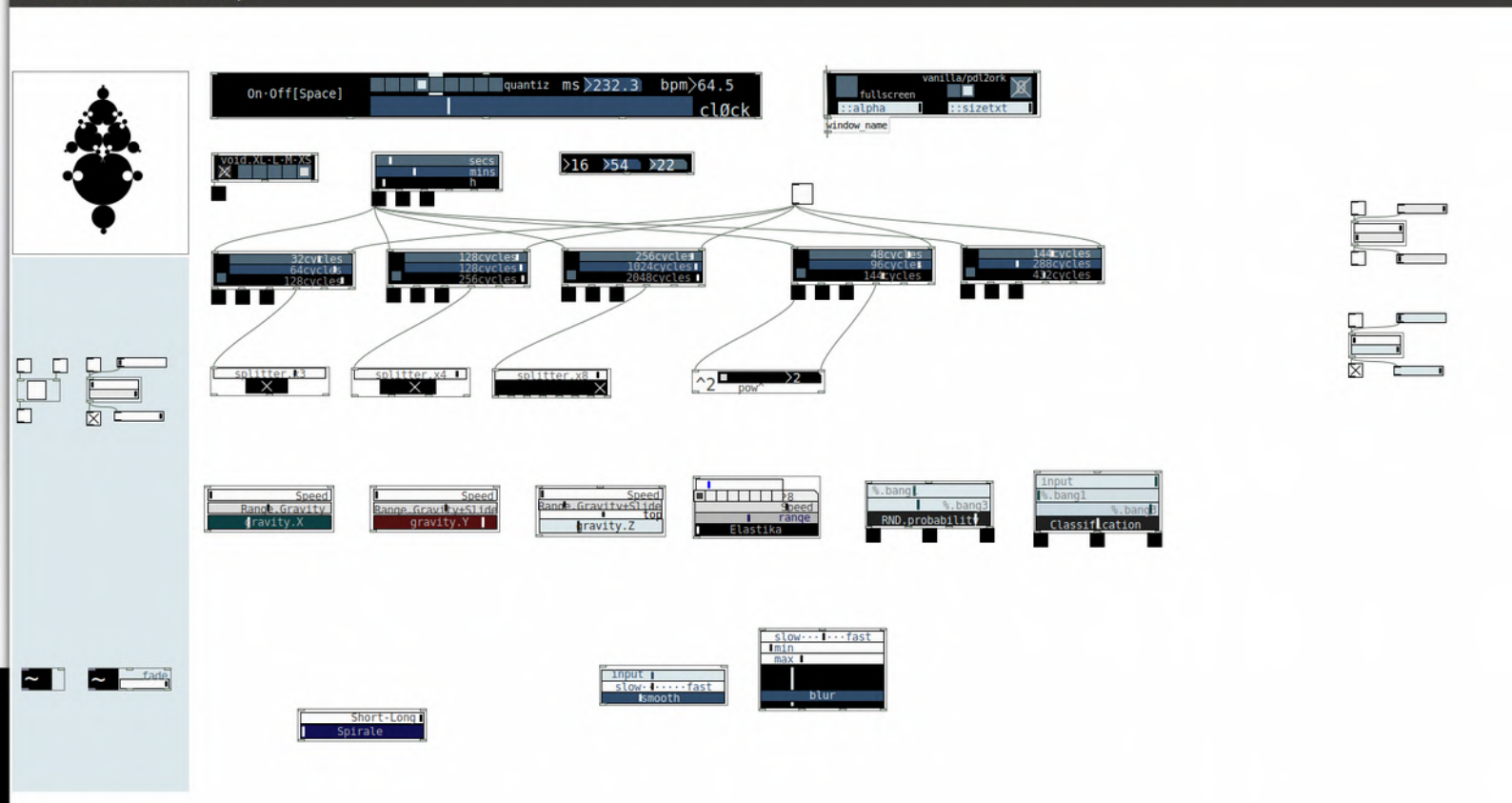
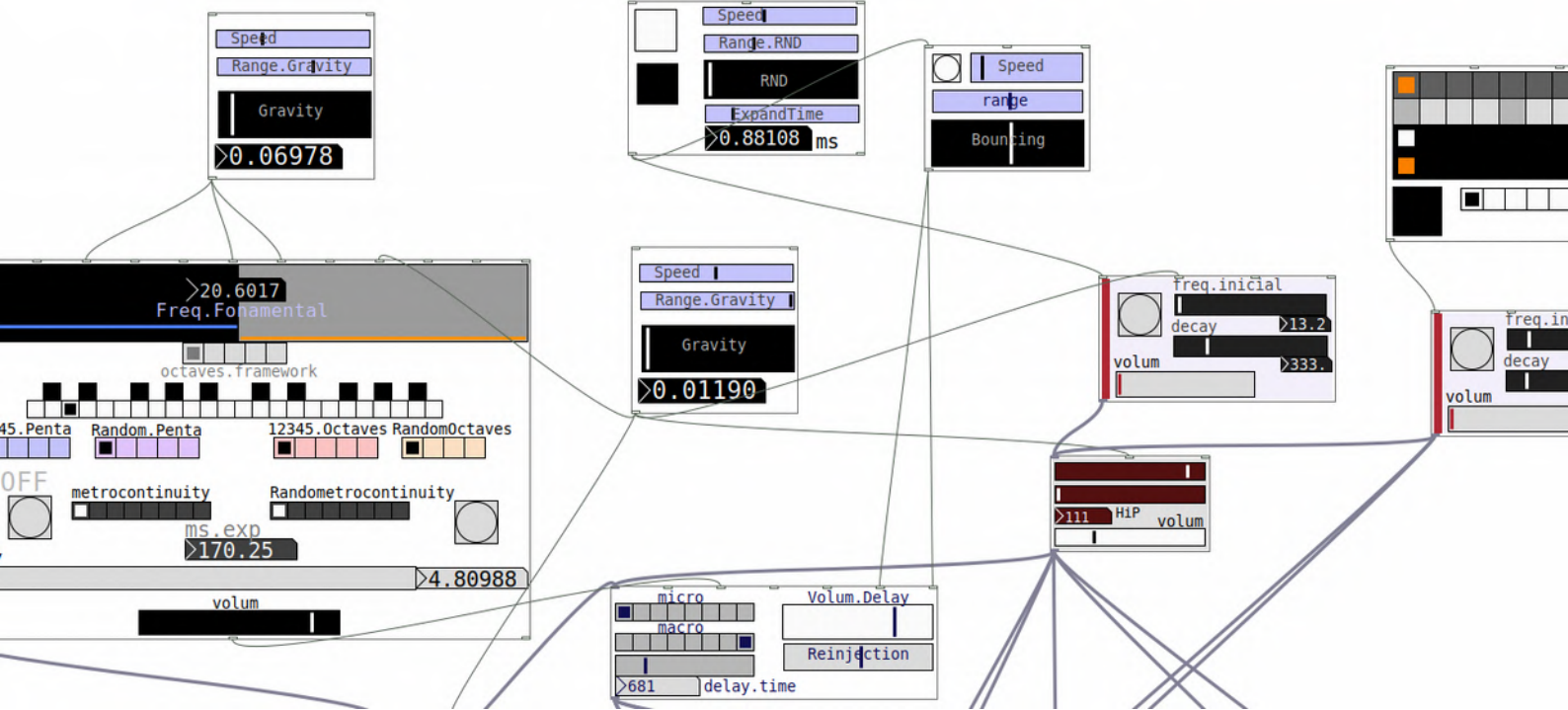
GNRTV.BLOCKS

<https://oneshaptiques.space/musIA/code/GNRTV.BLOCKS.v.1.0.zip>

&

<https://github.com/xamanza/GNRTV.BLOCKS>

مجمع المسكن



APP
FULLSCREEN
ALPHA
REC
FONT SIZE
HIDE FONTS

TIME
CLOCK
DRY
3CYCLES
TIME DISTRIBUTOR
CLOCK
DRY
3CYCLES

DRNTV
GRAVITY.X
GRAVITY.Y
GRAVITY.Z
ELASTIKA
TREE (T)
MARKOV

DISTRIBUTE FLOW
SMOOTH SPLITTER.X3
SPLITTER.X4
SPLITTER.X8
SLIDE.TIME PATTERNS
DATE DATE
DATE SIGNAL

PATTERNS
PATTERNS.X3
PATTERNS.X4
PATTERNS.X8
EUCLIDIAN LINEAR.SEQ
POLYGON

SIGNAL
SAMPLER.X3
SLICER
READXL-FILES

SIGNAL
OSC.SYNTH
OCEAN
RAPEBDD

PROCESS.SIGNAL
LDP
HDP
DELAY
REVERB
MIC.IN
DISTO
RACK MIXER
MICRO.DELAY
MACRO.DELAY

MORPHING
WAVESHAPER
BRAINER
SIDECHAIN
ATTACK.DECAY
LFO



SUMA DE ÁNGULOS INTERIORES DE UN POLÍGONO.
SI N ES EL NÚMERO DE LADOS DE UN POLÍGONO: $S = (N - 2) \cdot 180^\circ$
SUMA DE ÁNGULOS DE UN TRIÁNGULO = $(3 - 2) \cdot 180^\circ = 180^\circ$



META ML
OSC.CONNECT.FROWS
AUDIO ANALYSIS
ENV
SPECTRAL
PROBABILISTIC
CLASSIFICATION
SPECTRAL ANALYSIS
STRUCY ANALYSIS
PROBABILISTIC
SYN
SUPPORT VECTOR MACHINES
SVM & SVM KERNEL

SONIFICATION
FROM TXT FILE

SPACE
BINAURAL
TRIAD
QUAD



GNRTV://
blake
by-sa:OnesHaptiques.&.XN.2021

On-Off [Space] quantiz >17 >88

Speed
Range.RND
RND
ExpandTime
>0.88108 ms

GENERATIVE AI

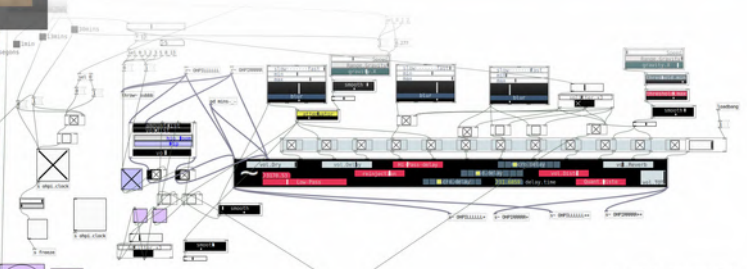
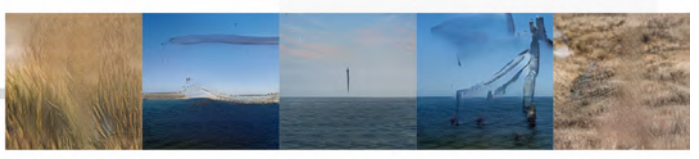
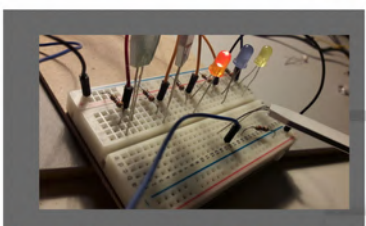
BASED ON NATURE
GROWTH
PHYSICS
COSMOS
...

BASED ON HUMAN COGNITION
BRAIN

Gravity
Gravity
01190

TATAMI HAPTIC
SOON

Volum.Delay
macro
Reinjection
delay.time



GNRTV BLOCKS

TIME FLOW GNRTV PATT SONIF SIGNAL SYNTHS SIGNAL SAMPLE MIXER MORPH META META SPACE fullscreen vanilla/pd2ork

void.XL-L-M-XS /2 On-Off[Space] quantiz ms >232.3 bpm >64.5 cl0ck

19 >42 >51

5- mix- r- mix-

vol.Dry vol.Delay Hi-Pass-delay macro.delay vol.Reverb

>8732.77 Low-Pass rejection micro.delay >154.889 delay.time Quant.Dist vol.TOT

threshold.min threshold.max smooth attenuator xpander xpander.SUM attenuat.DIA top vol.Oceans env~ random.komb1

dac-

GNRTV BLOCKS

TIME FLOW GNRTV PATT SONIF SIGNAL SYNTHS SIGNAL SAMPLE MIXER MORPH META META SPACE fullscreen vanilla

void.XL-L-M-XS /2 On-Off[Space] quantiz ms >232.3 bpm >64.5 cl0ck

5- mix- r- mix-

vol.Dry vol.Delay delay.time vol.Disto vol.Reverb

Low-Pass rejection >7.26842 Quant.Dist vol.TOT

dac-

GNRTV

Speed Range.Gray.Ity gravity.X

threshold.min threshold.max smooth

attenuator xpander xpander.SUM

attenuat.DIA top vol.Oceans env~ random.komb1 range dac-

GNRTV

%.band %.band input %.band vol.mph Classification

HELP.Blocks.Signal.pd - /home/oh/Esriptori/GNRTV.BLOCKS.vx/GNRTV.BLOCKS.DEVELV12/code

File Edit Put Windows Media Help

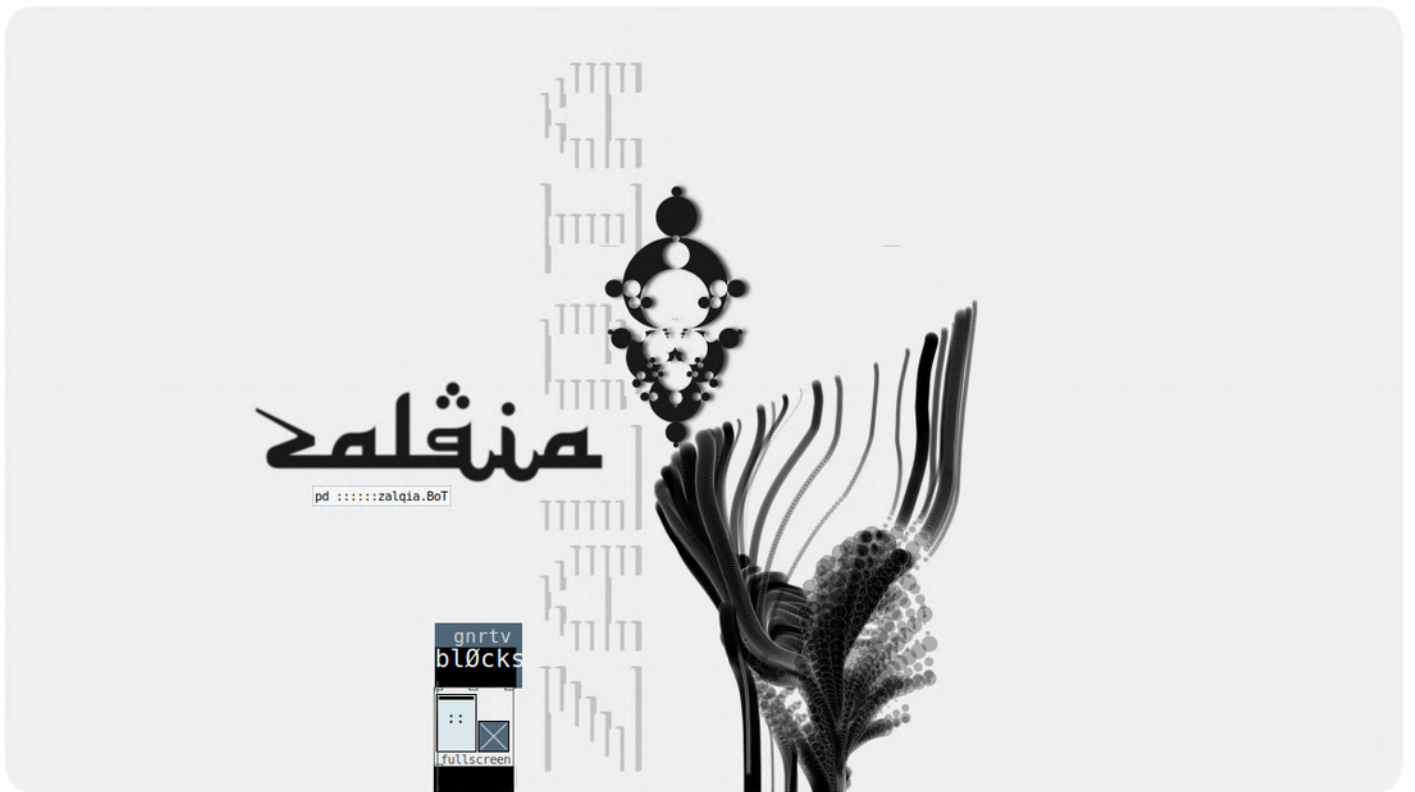
speed freq.foamont vol.Drone.A vol.Drone.B vol.Bass.3x1

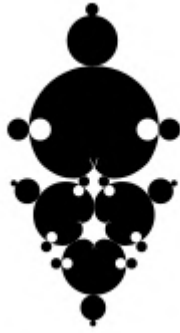
halftones vol.halftones lmb vol.chords timbr vol.open.tonic

freq.attack vol.attack vol.osc.HI vol.osc.LO vol.osc.MI

top vol.Oceans env~ random.komb1 range tipus vol.particles

Lop.zoom Hip.zoom Vol.Dist micro.delay macro.delay rejection vol.delay Liveness crossover Freq Amplif vol.Reverb







Context

Once we have learn the fundamentals of Pd and its syntax and methods with this tutorial
url > https://onshaptiques.space/musIA/pdfs/musIA_PdTutorialFundamentals.pdf
its time to turn up for another level.

Pd is a very nice tool to build complex and customized sonic algorithms, but sometimes it can be difficult to build large structures.

In this sense, i developed **GNRTV.BLOCKS** a kit of abstractions that works with Pd vanilla, and therefore with the rest of Pd versions (Pd-Extended and Pd-l2ork, and Purr-data).

Remember that you can download and install pd versions in the following urls

<https://puredata.info/downloads>

<https://puredata.info/downloads/pure-data>

<http://l2ork.music.vt.edu/main/make-your-own-l2ork/software/>

<https://github.com/pd-l2ork/pd>

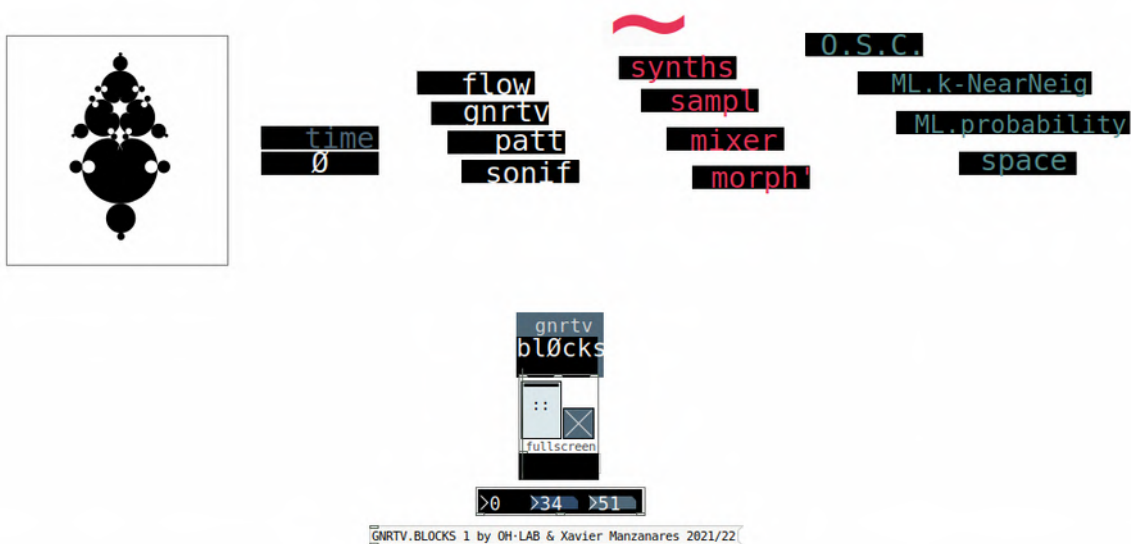
There are repositories of GNRTV.BLOCKS in the following urls >

<https://onshaptiques.space/musIA/>

<https://onshaptiques.space/musIA/code/GNRTV.BLOCKS.v.1.0.zip>

<https://github.com/xamanza/>

ENGLISH VERSION



GNRTV . BLOCKS is a toolkit library programmed with Pd visual programming environment useful to build sonic generative applications in a straight-forward way. The idea inspired by the modular synth cosmos and other modular tools features a big amount of different 'BLOCKS' that can be interconnected one from each other, therefore a toolkit for building custom projects oriented for live performances and installations.

BLOCKS has a big large of elements constituted as abstractions that features different functionalities related to the next domains :

Time [includes a master clock, a GMT clock, and several combinations of large cycles in time]

Data Flow [includes data flow splitters, gates, crossfades, pipes (delayed messages), and selectors]

Generativity [includes generative algorithms like gravity, blur, elastika, Random probability and Classification]

Patterns[includes a dual step trigger, a 16 step linear/randomic sequencer, an euclidian sequencer and also pattern storages]

Sonifications[includes a module for sonify and store patterns in txt files easily]

Synths~[includes several synthesizers and in addition basic signal filtering elements like lowpass hipass distortion quantized delay and reverb]

Samplers [includes XL files samplers, a rhythm box of 3 samples and a slicer of a desired audio sample]

Mixers~ [includes a compact and basic mixer and a bigger one with more details in tweak values]

Signal Morphing~ [includes those elements related to envelopes (attack decay, sidechain, waveshaper) and also other elements which features plasticity not only in envelopes but in signal expressivity such a granular filter or a AutoRessonant filter

Meta-OSC [includes OSC communication with other applications such Processing or Wekinator which is a pretty nice tool for MachineLearning Interaction Design]

Meta-ML [includes a prototyping approach of several Machine Learning methods like k-nearest neighbour algorithm or Probabilistic sequencers]

Space [includes modules for binaural and quadraphonic purposes]


GNRTV . BLOCKS works 100% with *pd-l2ork* and *pd-extended* versions, and about 90% with *pd-vanilla* core environments. Check your version for your system here >

pd-l2ork

pd-extended

pd-vanilla

Due to the fact that we are inside Pd environment, we have to consider the two classes of elements among the environment : Those which manages ***data*** (*according to the speed of your processor*) and those which manages ***audio signal*** (*Therefore by default 44100 samples of information per second*).

In Blocks we can see the difference with the elements that features a  which means that are *blocks, elements or abstractions* that are managing or producing audio signal.

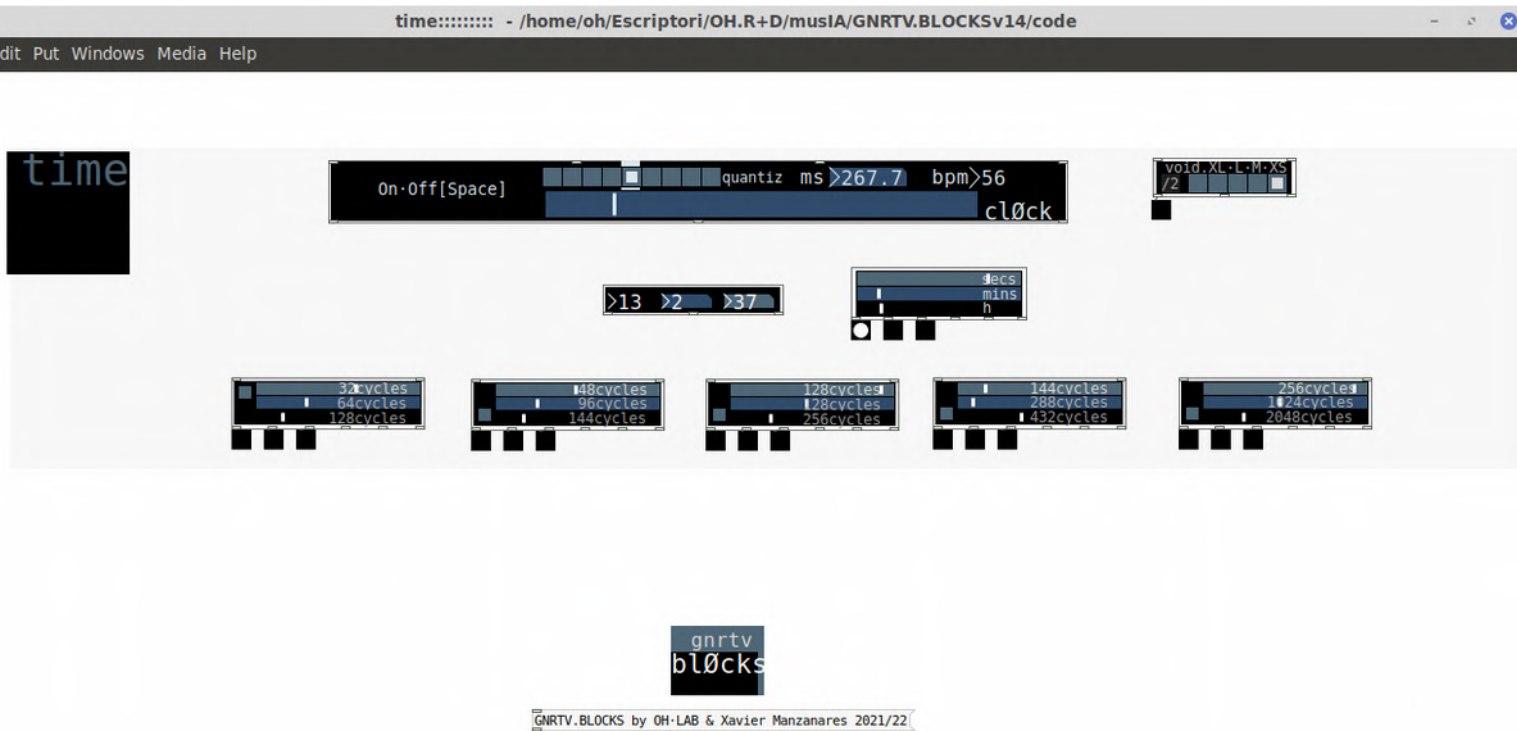
If you are not familiar with pd environment check out this tutorial which explains the fundamentals of the language > url

As a brief approach any **block** has inlets or inputs in the top and outlets or outputs on the bottom. The basic concept is that a block can have different amount of inlets according to the subelements included in the block (sliders, radio buttons or numbers that we want to interconnect]. The order of inlets or inputs is from left to right and from top to bottom.

Blocks has outlets or outputs that are depending on the functionality of each block. In the main menu bar you'll find the instructions of those connections in the directory ***code.HELP*** of the whole software package. In this directory you'll find the description of each block and its connections.

GNRTV.BLOCKS will work in several objects and blocks with ***normalized values (therefore between 0 and 1)***. This is a project decision in order to increase the connectivity between elements and also, make this library much more updated and compatible with ML external applications like Wekinator or others.

As it was presented before, **GNRTV . BLOCKS** is structured in several sublevels or groups of abstractions. Let's see in detail :



Time In this level we'll find

[clØck] a main clock with bpm, and ms translation. Also has a selector allowing quantizing time from a main timeframework.

[void·XL·L·M·XS] is the block attached internally with clock in order to make different groups of triggers for different instruments we want to control. Notice that this block without clØck does not work.

[GMT Time] features Hours Minutes and seconds in real time, therefore is a nice block to use in installations to control events according certain daytime

[secs mins h] similar to the previous is useful to sequence or trigger events in Large Time Spans.

Cycles

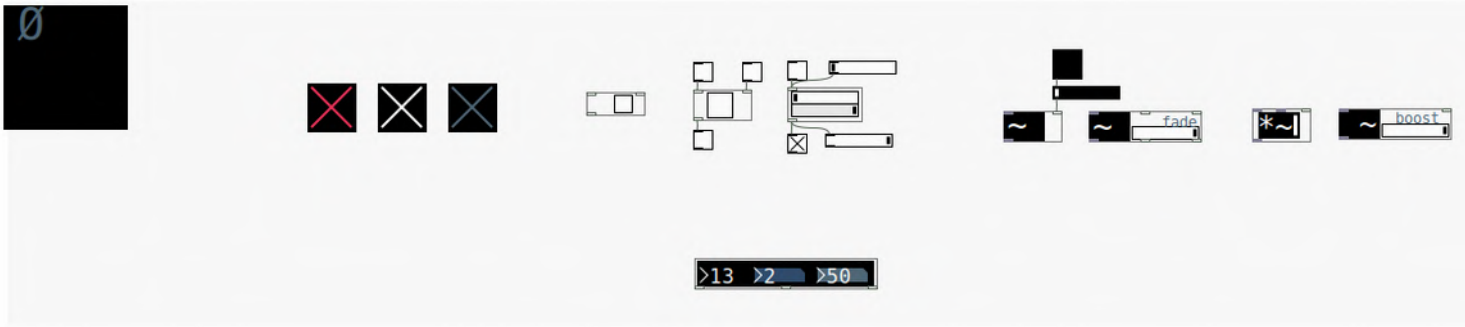
[32·64·128 cycles] creates loops of 32·64·128 distances depending on the incoming clock on the top left inlet

[48·96·144 cycles] creates loops of 48·96·144 distances depending on the incoming clock on the top left inlet

[128·256·512 cycles] creates loops of 128·256·512 distances depending on the incoming clock on the top left inlet

[144·288·432 cycles] creates loops of 144·288·432 distances depending on the incoming clock on the top left inlet

[256·1024·2048 cycles] creates loops of 256·1024·2048 distances depending on the incoming clock on the top left inlet



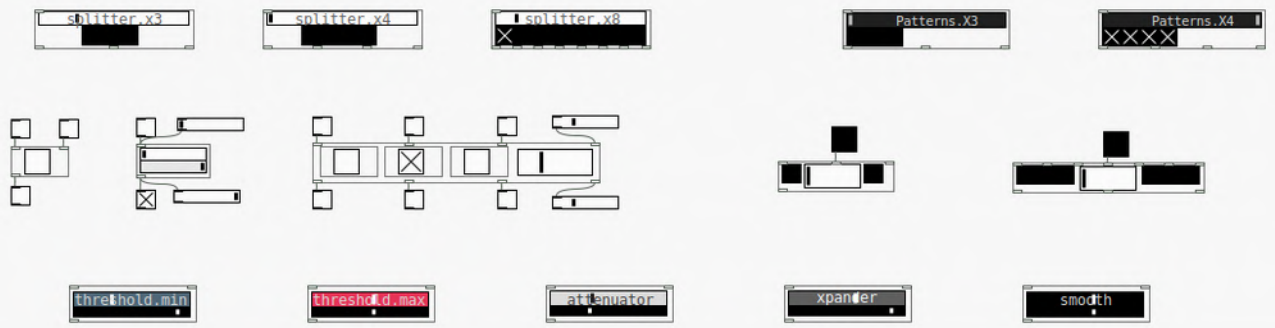
gnrtv
bløcks

GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

In this level **0** we'll find modules that will be useful to interconnect nodes and groups of nodes and maybe we are going to clone several times in our project.

Somehow like Gates or Switch Buses that Turns on or off a piece of code we want to control (both for dataflow and for audio signal) > In this sense blocks which features a **~** symbol, are compatible to manage and interconnect **other signal blocks**. Otherwise if blocks does not have ~ symbol, are **directly data blocks** which manages numbers and alphanumeric messages.

flow



GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

flow menú features difeferent blocks to manage data flow in ouir algyryhtm,ws

[splitter.x3] Splitter controls 3 different and independent outputs through a normalized slider (from 0 to 1 values). Those output will be featured as a Toggle message (message 0 or 1 value) with it correspondent outlet.

[splitter.x4] Splitter controls 4 different and independent outputs through a normalized slider (from 0 to 1 values). Those output will be featured as a Toggle message (message 0 or 1 value) with it correspondent outlet.

[splitter.x8] Splitter controls 8 different and independent outputs through a normalized slider (from 0 to 1 values). Those output will be featured as a Toggle message (message 0 or 1 value) with it correspondent outlet.

[patterns.x3] controls different combinations of 3 different patterns (can be superposed different outputs at once) through a normalized slider (from 0 to 1 values). Those outputs will be featured as a Toggle message (message 0 or 1 value) with it correspondent outlet.

[patterns.x4] controls different combinations of 4 different patterns (can be superposed different outputs at once) through a normalized slider (from 0 to 1 values). Those outputs will be featured as a Toggle message (message 0 or 1 value) with it correspondent outlet.

[patterns.x8] controls different combinations of 8 different patterns (can be superposed different outputs at once) through a normalized slider (from 0 to 1 values). Those outputs will be featured as a Toggle message (message 0 or 1 value) with it correspondent outlet.

[white gate] Let pass the data flow if the toggle is activated

[reverse gate] Inverts the Toogle or number from 0 to 1 value.

[3x gate] Let pass the data flow if the toggle is activated for each of the 3 independent channels. In addition a normalized slider controls the sequence of tyhe patterns.

[1value·2channels·XCrossfade] A 2-channel crossfade between a normalized value (or bang)

[3bangs·2channels·XCrossfade] A 2-channel crossfade between 3 normalized values (or 3 bangs)

[threshold.min] fixes a minimun threshold of an incoming normalized value from 0 to 1

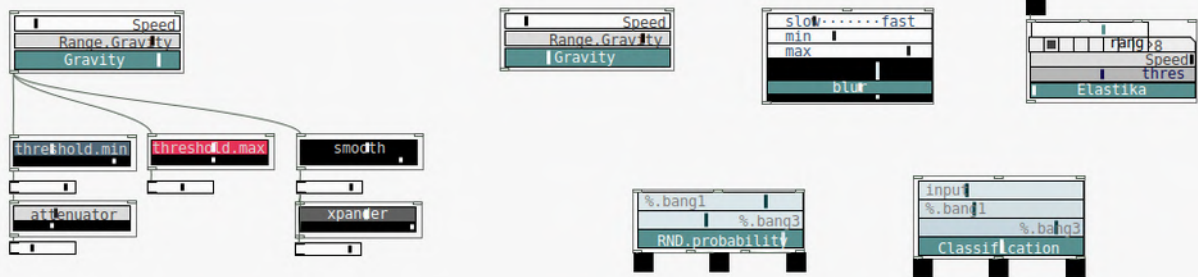
[threshold.max] fixes a maximum threshold of an incoming normalized value from 0 to 1

[attenuator] attenuates an incoming normalized value from 0 to 1

[xpander] expands an incoming normalized value from 0 to 1

[smooth] creates a smooth and slow dynamic response from an incoming normalized value from 0 to 1

gnrtv

gnrtv
bløcks

GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

gnrtv menú features different blocks that features generative algorithms.

Maybe one of the most uniques in those library that features as well the name of this project :)

[gravity] creates a string of randomic numbers according an initial speed and initial range. It behaves somehow similar to attractors (lorenz etc).

[blur] creates a string of randomic numbers defined between a superior and inferior limits, and also related to a certain speed.

[elastika] creates a string of values performing somehow as a 'bouncing ball' or elastic bouncing.

[RND.Probability] According to an incoming bang or trigger this block generates proportions or probabilities of triggered randoms according certain thresholds defined by the sliders.

[classification] -According to an incoming normalized value (from 0 to 1), this block creates the output classification, depending on the position and combination of the sliders.



gnrtv
blocks

GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

patt menú features different blocks in order to build generative patterns. Also some block to store those patterns.

[colored ones] creates a dual pattern between the featured numbers. Those combination of numbers can be selected through a normalized slider.

[x16] 16 step sequencer that can act in a linear way or randomly.

[pow²] creates a power of 2 sequence (with ranges like 2, 4, 8,16, 32 and 64) which triggers when it starts again.

[pow⁴] creates a power of 4 sequence (with ranges like 4, 16, 64, 256, 1024, 4096) which triggers when it starts again.

[ran.lin] creates a linear sequence (lin position) or a random combination between the selected index through the slider. When the linear sequence or the random string is value 0, it makes a trigger.

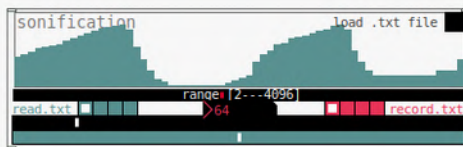
[euclidian] Euclidian sequencer of 3 different patterns.

Borrowed code from cgm.cs.mcgill.ca/~godfried/publications/banff.pdf

[trainer] stores a combination between 3 different patterns (rec) that after can be played again (read)

[random.kombi] creates a random combination of three different values which is stored in a memory that after we can call it to create sequences.

sonif

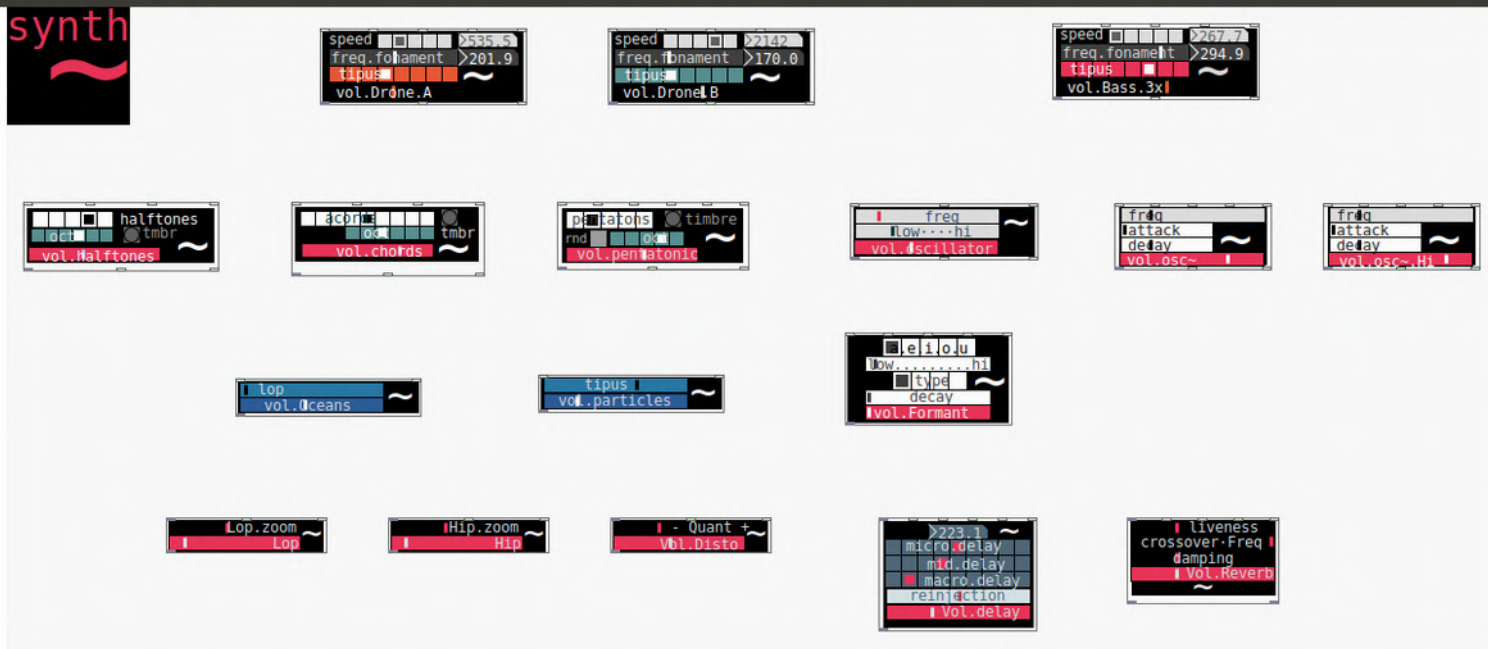
gnrtv
blocks

GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

Sonif menú features a couple of blocks to perform **data sonifications**

[sonification] block useful to read txt files and store its values in a memory that can be translated into sequences, basslines or other sonification process.

[mouse] block that reads the mouse position into the screen. Useful to interconnect the mouse movement as a Human interaction device within other blocks.



GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

synth~

[drone.A] block that creates a polyphonic drone (with 8 voices) in which we can control the fundamental frequency (freq.fonament) and the timbric components depending on the harmonics (tipus hradio). In addition speed controls the velocity of the generative harmonics changes.

[drone.B] block that creates a polyphonic drone (with 8 voices) in which we can control the fundamental frequency (freq.fonament) and the timbric components depending on the harmonics (tipus hradio). In addition speed controls the velocity of the generative harmonics changes. Is basically the same as Drone.A in order to make superposition of voices or harmonic combinations

[Bass.3x] Similarly to the previous, this block that creates a bassline with 3 voices in which we can control the fundamental frequency (freq.fonament) and the timbric components depending on the harmonics (tipus hradio). In addition speed controls the velocity of the generative harmonics changes.

[halftones] creates random combinations of the 5 halftones (black piano keys) of an octave (white hradio selector). In addition we can change as well different octaves and bang different proportions of harmonics therefore its different timbric result.[tmbr bang]

[chords] creates random combinations of chords (white hradio selector). In addition we can change as well different octaves and bang different proportions of harmonics therefore its different timbric result. [tmbr bang]

[pentatonic] creates random combinations of pentatonic tones (white hradio selector). In addition we can change between different octaves and bang different proportions of harmonics therefore its different timbric result.[tmbr bang]. Also the toggle [rnd] can change randomically between octaves

[oscillator] simple oscillator in which we can control its frequencies and range of them with [low-hi]

[osc~] simple oscillator in which we can control its frequency (for low and mid ranges) and attack decay envelope integrated

[osc~ hi] simple oscillator in which we can control its frequency (for high ranges) and attack decay envelope integrated

[oceans] white noise bank filtered with lowpasses that creates some wavy effect

[particles] white noise bank clusters filtered with bandpasses that creates some glitchy and textures effect

[formant] block that creates a formant synth which emulates vowels and pitches as a synthesized and plastic voice

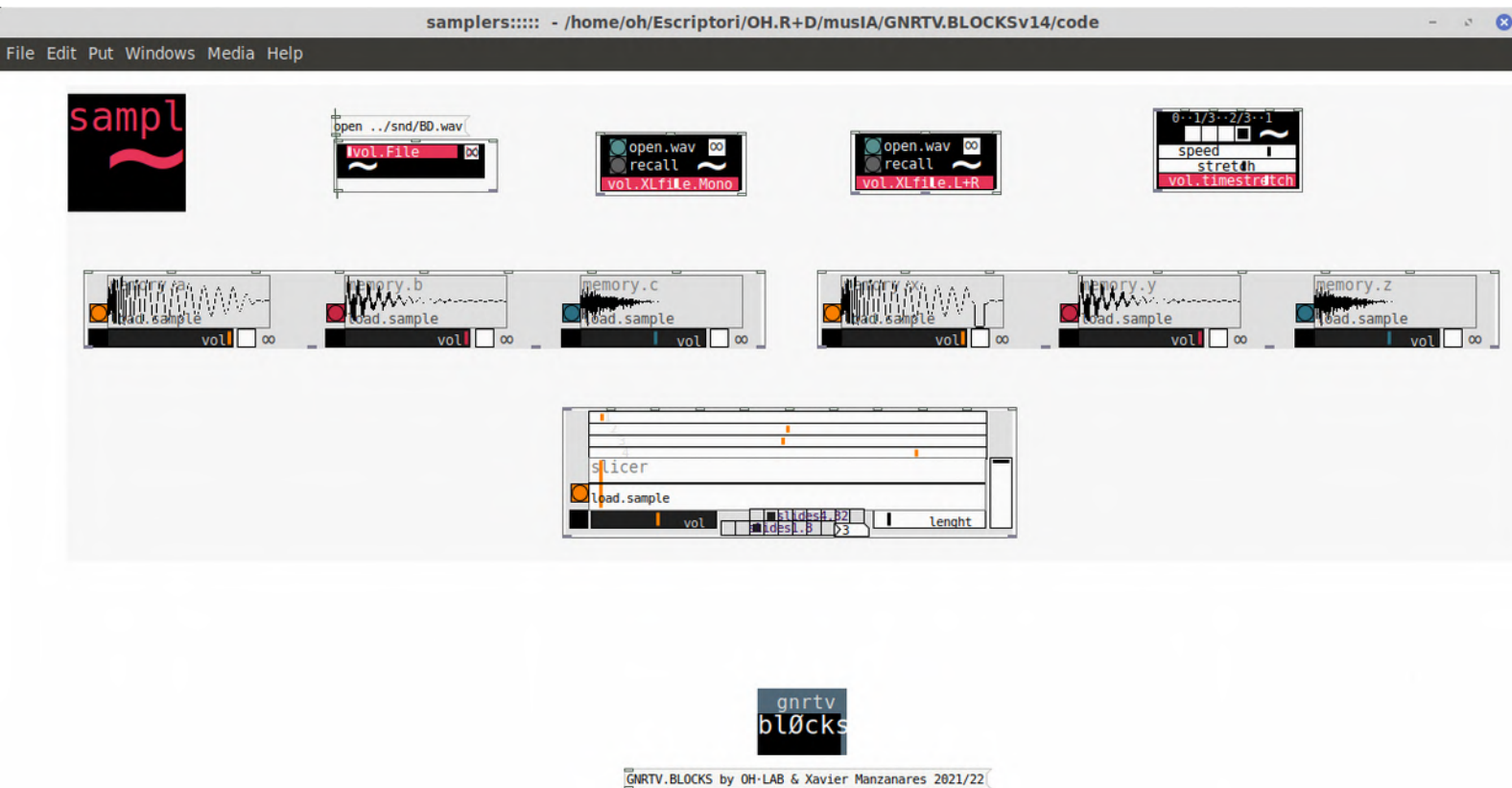
[lop~] low-pass filter

[hip~] hi-pass filter

[disto~] bit distortion filter

[delay~] quantized delay rack

[reverb~] reverb rack



sampler ~

[file~] block that plays a sample in which we can edit the path. By default samples of Generative.Blocks are stored within the 'snd' directory. The toggle ∞ loops the sample.

[XLfile~ mono] block that plays a sample in which we can open the file through a prompt. The toggle ∞ loops the sample. Useful for mono files.

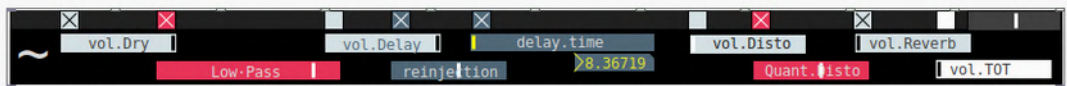
[XLfile~ L+R] block that plays a sample in which we can open the file through a prompt. The toggle ∞ loops the sample. Useful for stereo files.

[timestretch] block that stretches the incoming sound of a sample. Notice that this block works with any of the previous otherwise will not work. It features selector of sample stretch 0 1/3 2/3 1, speed and stretch slider.

[sampler.3x.abc] rhythm box composed by 3 samples that can be triggered, self looped and controlled the volume

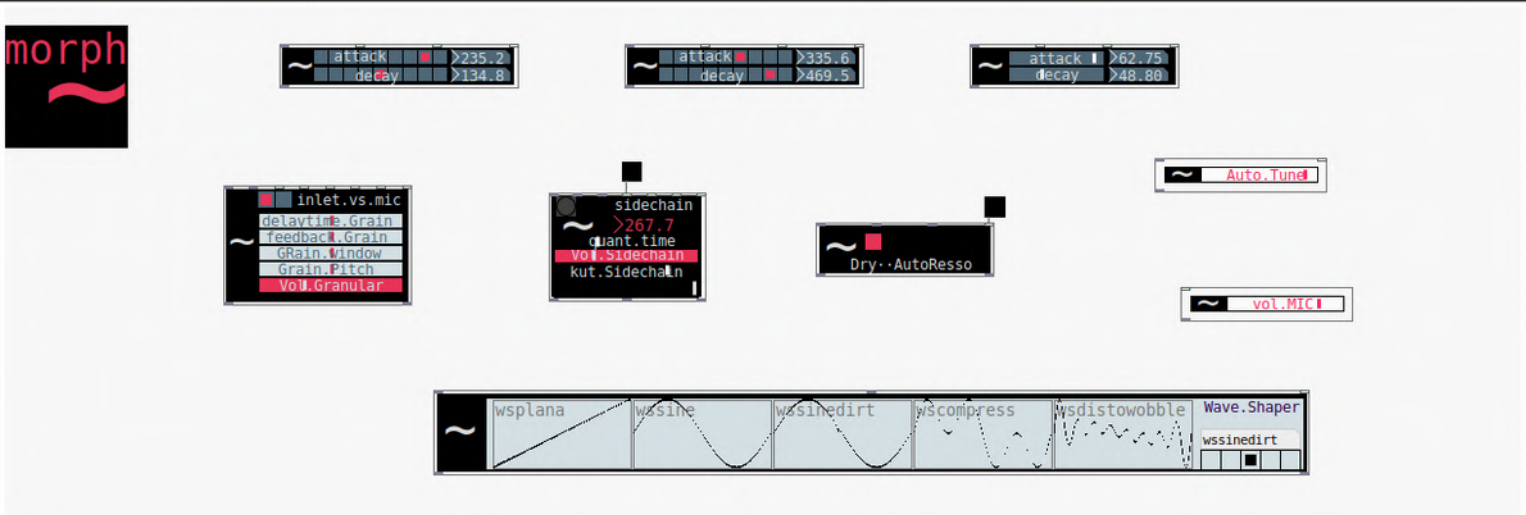
[sampler.3x.xyz] rhythm box composed by 3 samples that can be triggered, self looped and controlled the volume

[slicer] block that can create rhythms between sample slices. As options we can select different cue points for the loops, volume, steps in the sequence (slides), and length of the slices



GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

mixer menú features a final mixer that can be automatized and is featuring a LowPass filter, a Quantized delay a Distorsion and a Reverb.



Morph

[attack / decay quantized values 1] block that creates an attack and decay time for any incoming signal.
Short values

[attack / decay quantized values 2] block that creates an attack and decay time for any incoming signal
mid/long values

[attack / decay powered values] block that creates an attack and decay time for any incoming signal
exponential behaviour values

[Granular synthesizer filter / processor] granular synth that can granulate an incoming signal or the microphone line. Parameters : delaytime.grain , feedback.grain, grain.window, grain.pitch and volume.

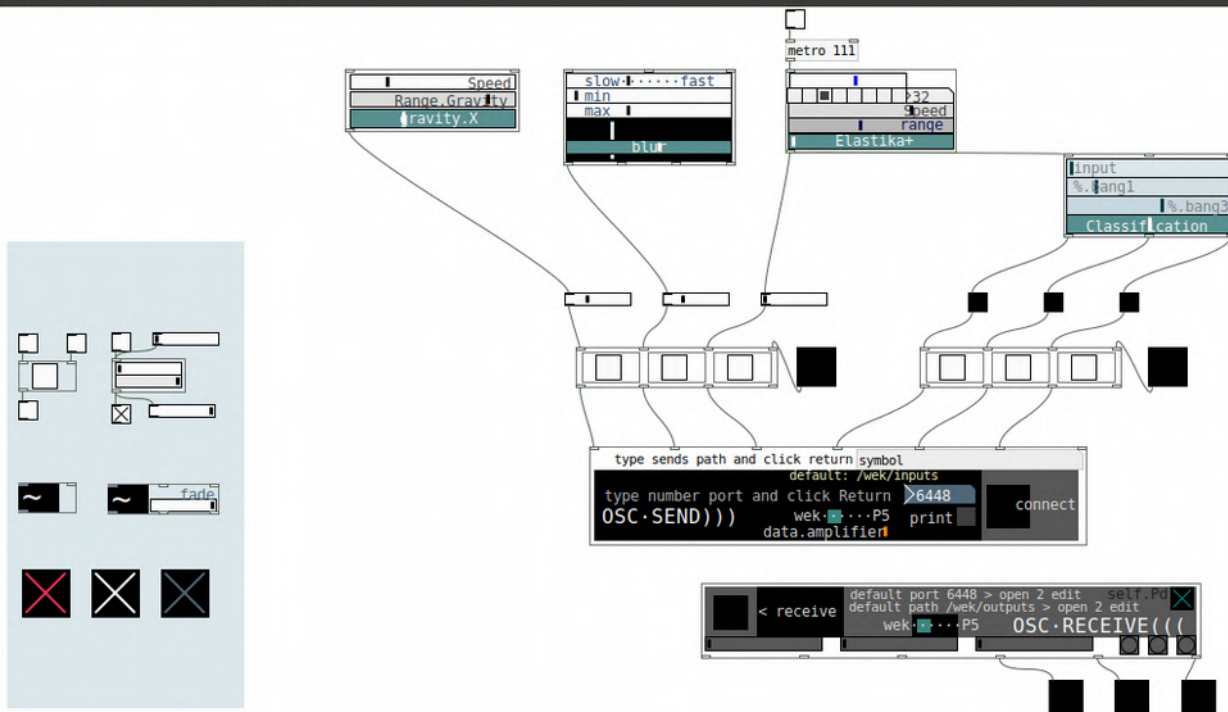
[sidechain] sidechain envelope effect from an incoming signal. Time scales are related to thje m,ain clock in order to make more precise rhythm effects. Parameters : quantity of time, volume and kut (envelope value in which signal is descending)

[autoressonance] Autoressonance filter block conformed by a signal which is modulated thugh a bandpass with the envelope of a secondary signal. (notice that this secondary can be the initial and duplicated incoming signal)

[autotune] autotune algorithy block that creates some kind of basic autotune effect from an incoming signal.

[microphone] Microphone or linein signal that we can interconnect with other blocks.

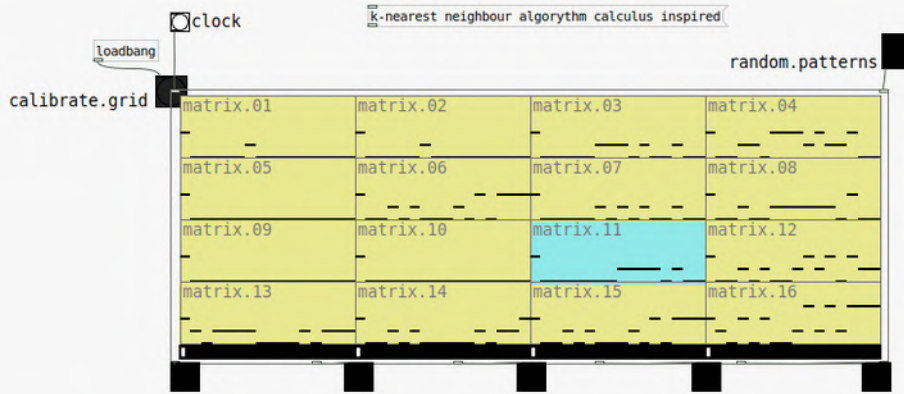
[waveshaper] Waveshaper block with 5 different shapes



OSC

[OSC · SEND]] block to send OSC messages both normalized (from 0 to 1 values) or triggers (bangs)

[OSC · RECEIVE]] block to receive OSC messages both normalized (from 0 to 1 values) or triggers (bangs)

meta
MLgnrtv
bløcks**ML-knearest neighbour**

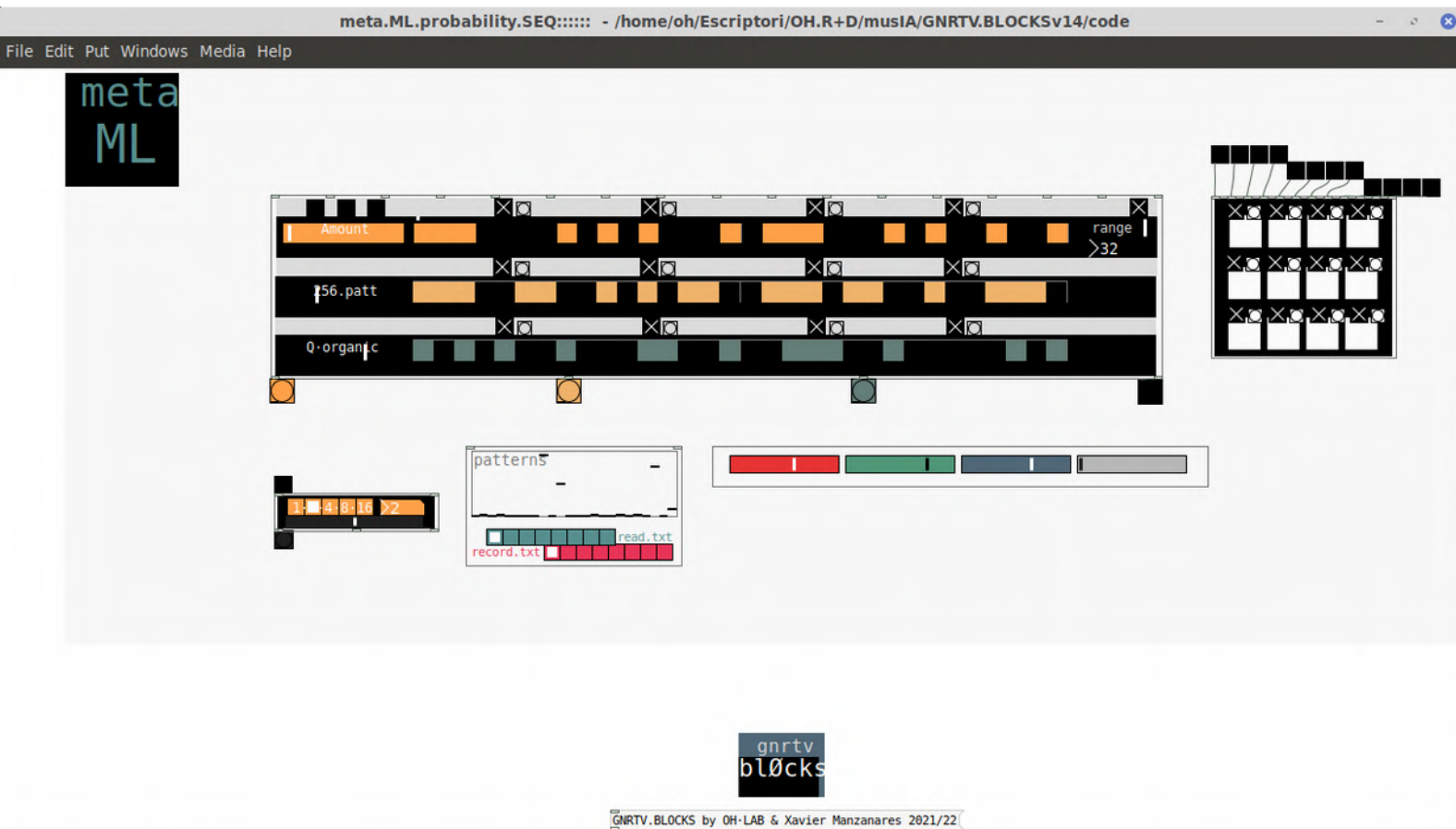
[matrix.16] block inspired in ML projects like *latent loops* * or *melody mixer***

It creates a 16-cell matrix we creates randomically patterns on the corners. Then an algorithym calculus based on the k-nearest neighbour algorithym calculates the patterns in the in-between positions from the corners.

The result is a combination of patterns that later we can associate to a rhythm box for example and create easy diunamic changes from similar but unique patterns.

*<https://teampieshop.github.io/latent-loops/>

**<https://experiments.withgoogle.com/ai/melody-mixer/view/>



ML.probability

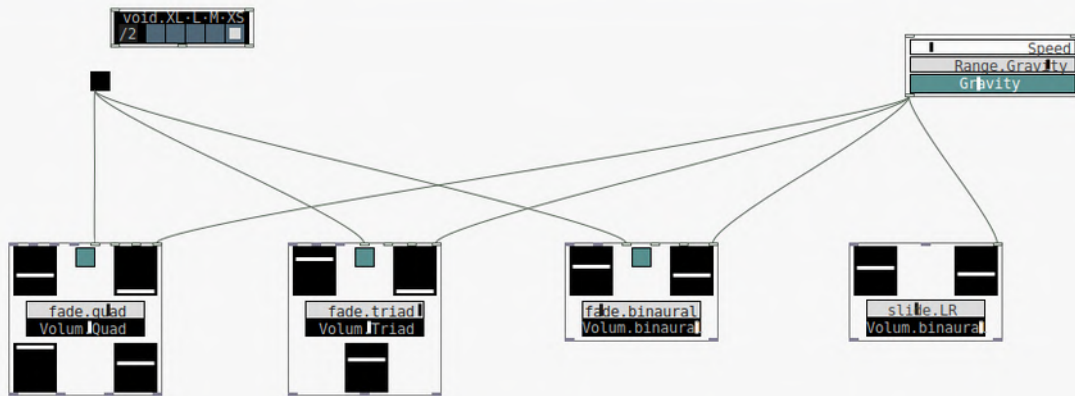
[3x.SEQ.32] 32 step sequencer that works with probability numbers associated to patterns. Some kind of stokastik or NaiveBayes ML algorithms of probability. Therefore we can create patterned and 'rigid' structures and also more organic ones.

[1.2.4.8.16 phrases] useful to create phrases associated with the previous sequencer

[patterns.store] stores the patterns created in the 3x.SEQ.32 in order to recall them later.

[RGB Color Selector] RGB sliders to change the colors theme

space

gnrtv
bløcks

GNRTV.BLOCKS by OH-LAB & Xavier Manzanares 2021/22

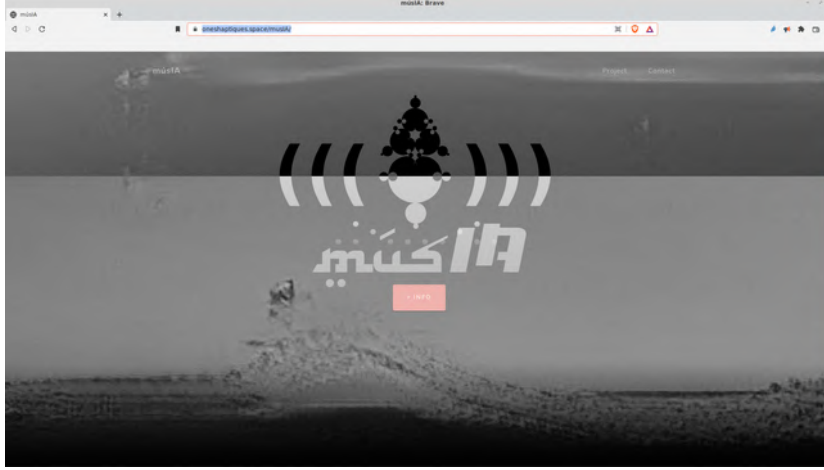
space

[quad] block to create a quadraphony from an incoming signal. We can associate the rotations to a certain clock or even connect with generative algorithms blocks like 'gravity' to create more fluid transitions and movements. In addition 'fade' slider controls the openness/contrast of the oscillation in the move.

[triad] block to create a triphony from an incoming signal. We can associate the rotations to a certain clock or even connect with generative algorithms blocks like 'gravity' to create more fluid transitions and movements. In addition 'fade' slider controls the openness/contrast of the oscillation in the move.

[binaural] block to create a binaural effect from an incoming signal. We can associate the rotations to a certain clock or even connect with generative algorithms blocks like 'gravity' to create more fluid transitions and movements. In addition 'fade' slider controls the openness/contrast of the oscillation in the move. In this block the result of channel L is not necessary the reverse of the R.

[binaural.creator] block to create a binaural effect from an incoming signal. Similar to the previous one but useful if we want to build a binaural sound from a mono incoming signal. In this block the result of channel L is the reverse of the R.



musIA . Urls

Repositori de la recerca

<https://oneshaptiques.space/musIA>

[pdfs]

musIA :// papers & counterpapers

https://oneshaptiques.space/musIA/pdfs/musIA_papers+counterpapers.pdf

musIA :// Intro al Machine Learning

https://oneshaptiques.space/musIA/pdfs/musIA_Intro_al_Machine_Learning.pdf

tutorial GNRTVBLOCKS ENG v.

https://oneshaptiques.space/musIA/pdfs/Tuto_GNRTVBLOCKS_ANG.pdf

tutorial Pd Language fundamentals ENG v.

https://oneshaptiques.space/musIA/pdfs/musIA_PdTutorialFundamentals.pdf

tutorial Pd Language fundamentals for LICH and WITCH

https://github.com/xamanza/LICH.Pd.Patches/raw/main/Pd4LICH.Part1.PdTutorialEssentials_PdLangFundamentals.pdf

tutorial compiling algorithms for Lich & Witch

https://github.com/xamanza/LICH.Pd.Patches/raw/main/Pd4LICH%26WITCH_Part3.RebelTech.BrowserCompilingTool.pdf

linkografia AI Music & ARTS

https://oneshaptiques.space/musIA/pdfs/musIA_RelevantResources+Linkography_Music+ARTS.pdf

FULL.researchDoc.musIA

https://oneshaptiques.space/musIA/pdfs/musIA_full.research.doc.pdf

[codi]

LICH & WITCH Pd Patches

<https://github.com/xamanza/LICH.Pd.Patches>

GNRTV.BLOCKS

<https://oneshaptiques.space/musIA/code/GNRTV.BLOCKS.v.1.0.zip>

&

<https://github.com/xamanza/GNRTV.BLOCKS>

R!FFFS

<https://github.com/xamanza/R!FFFS>





SOON

