



Context

Once we have learned the fundamentals of Pd and its syntax and methods with this tutorial
url > https://oneshaptiques.space/musIA/pdfs/musIA_PdTutorialFundamentals.pdf
it's 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

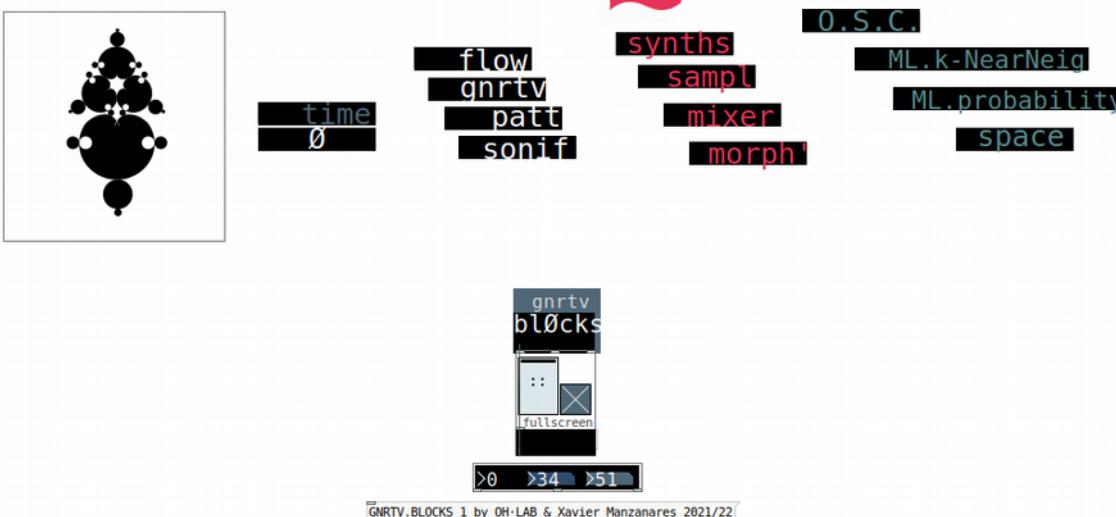
<https://puredata.info/downloads>
<https://puredata.info/downloads/pure-data>
<http://l2ork.music.vt.edu/main/make-your-own-l2ork/software/>
<https://github.com/pd-l2ork/pd>

There are repositories of GNRTV.BLOCKS in the following urls >
<https://oneshaptiques.space/musIA/>

<https://oneshaptiques.space/musIA/code/GNRTV.BLOCKS.v1.0.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 algoryhtms 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 algorythm 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*.

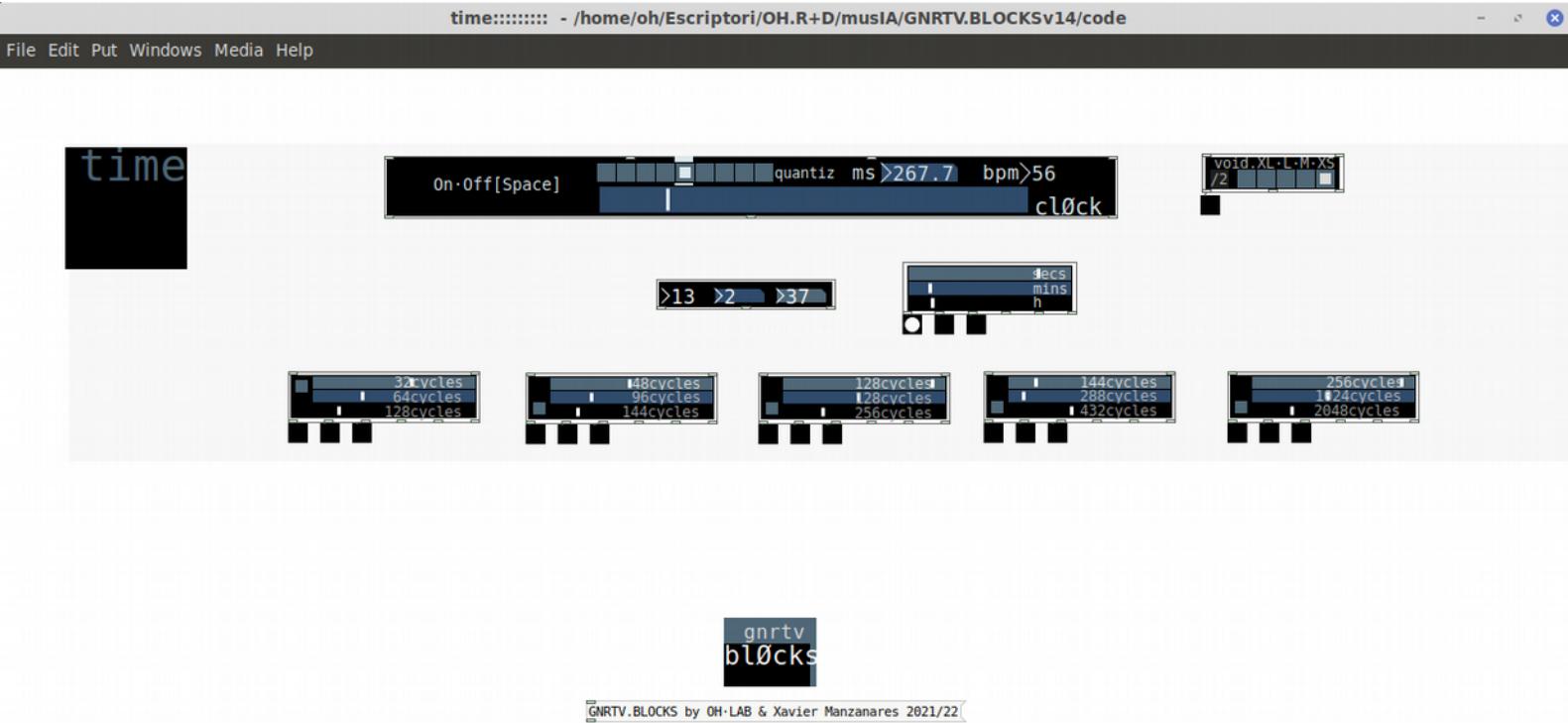
If you are not familiar with pd environment check out this tutorial which explains the fundamentals of the language > url

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 youll 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

[cl0ck] a main clock with bpm, and ms translation. Also has a selector allowing quantizing time from a main timeframe.

[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 cl0ck 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

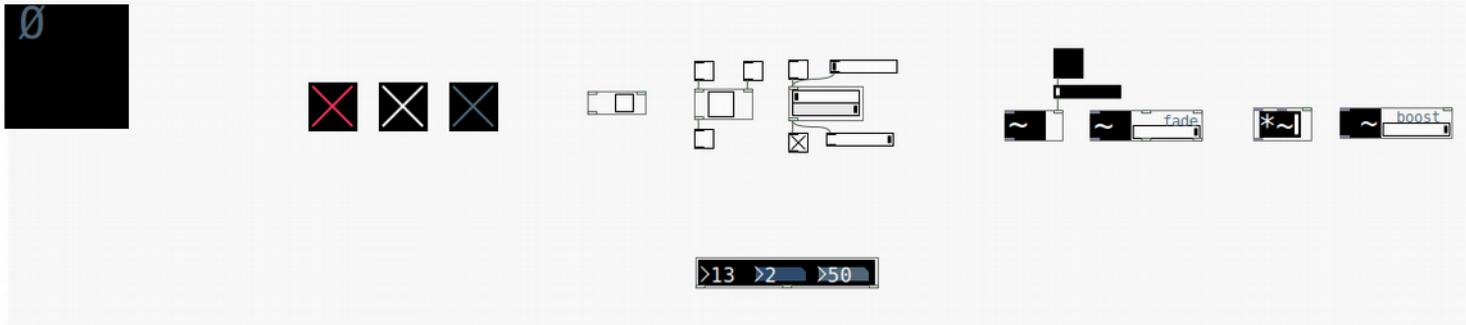
[32·64·128 cycles] creates loops of 32·64·128 distances depending on the incoming clock on the top left inlet

[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

[256·512·1024·2048 cycles] creates loops of 256·512·1024·2048 distances depending on the incoming clock on the top left inlet

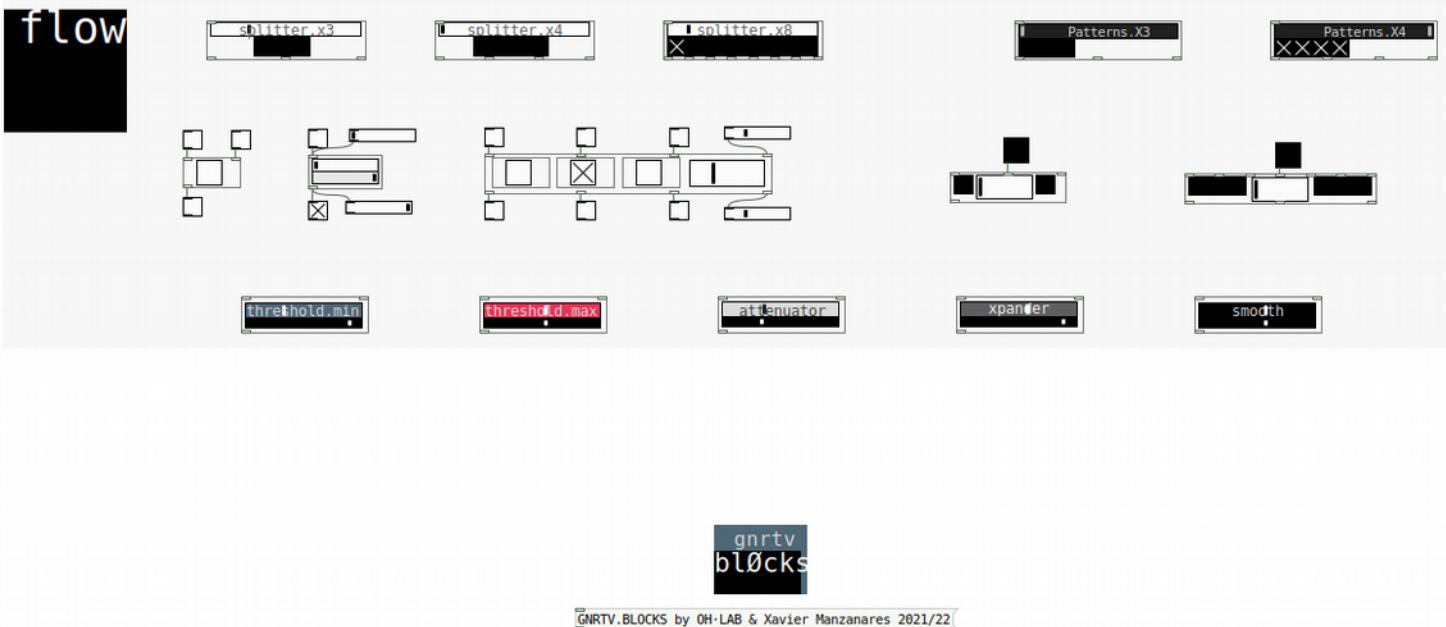


gnrtv
blocks

gnrtv.blocks by OH-LAB & Xavier Manzanares 2021/22

In this level **Ø** 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 menú features difeferent blocks to manage data flow in ouir algoryhmt,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

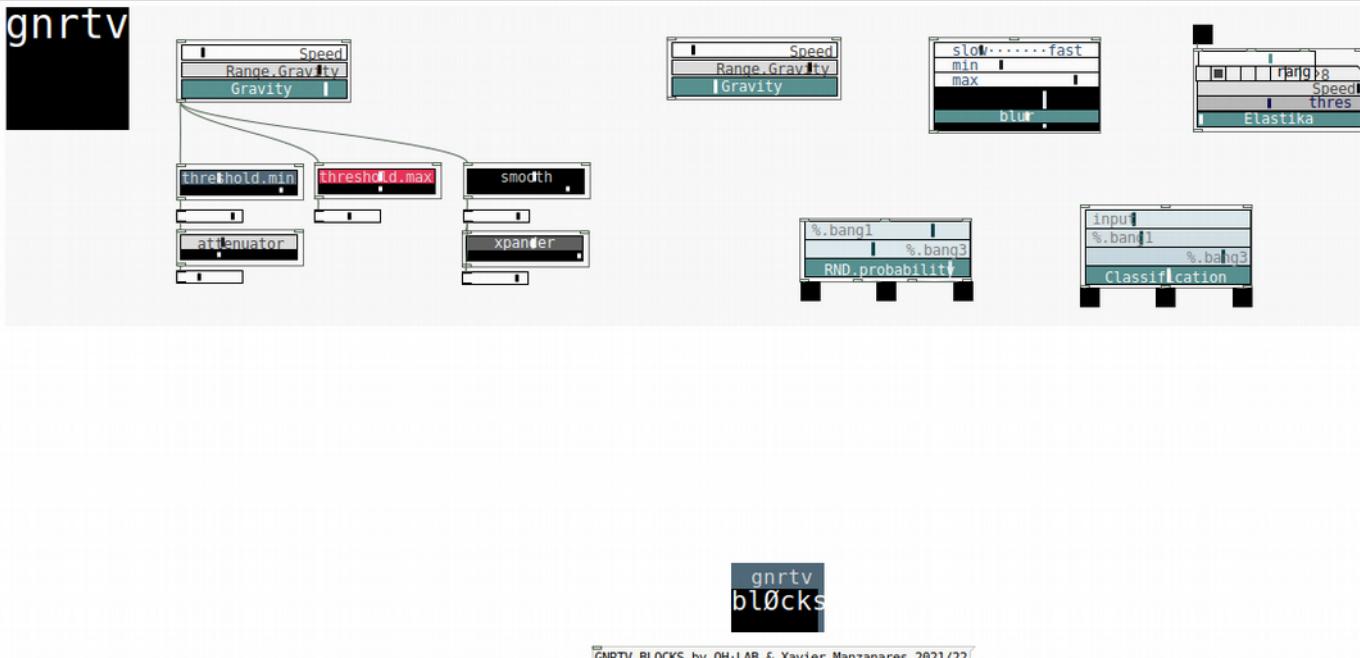
[threshold.max] fixes a maximum threshold of an incoming normalized value from 0 to 1

[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 menú features different blocks that features generative algorythms.

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

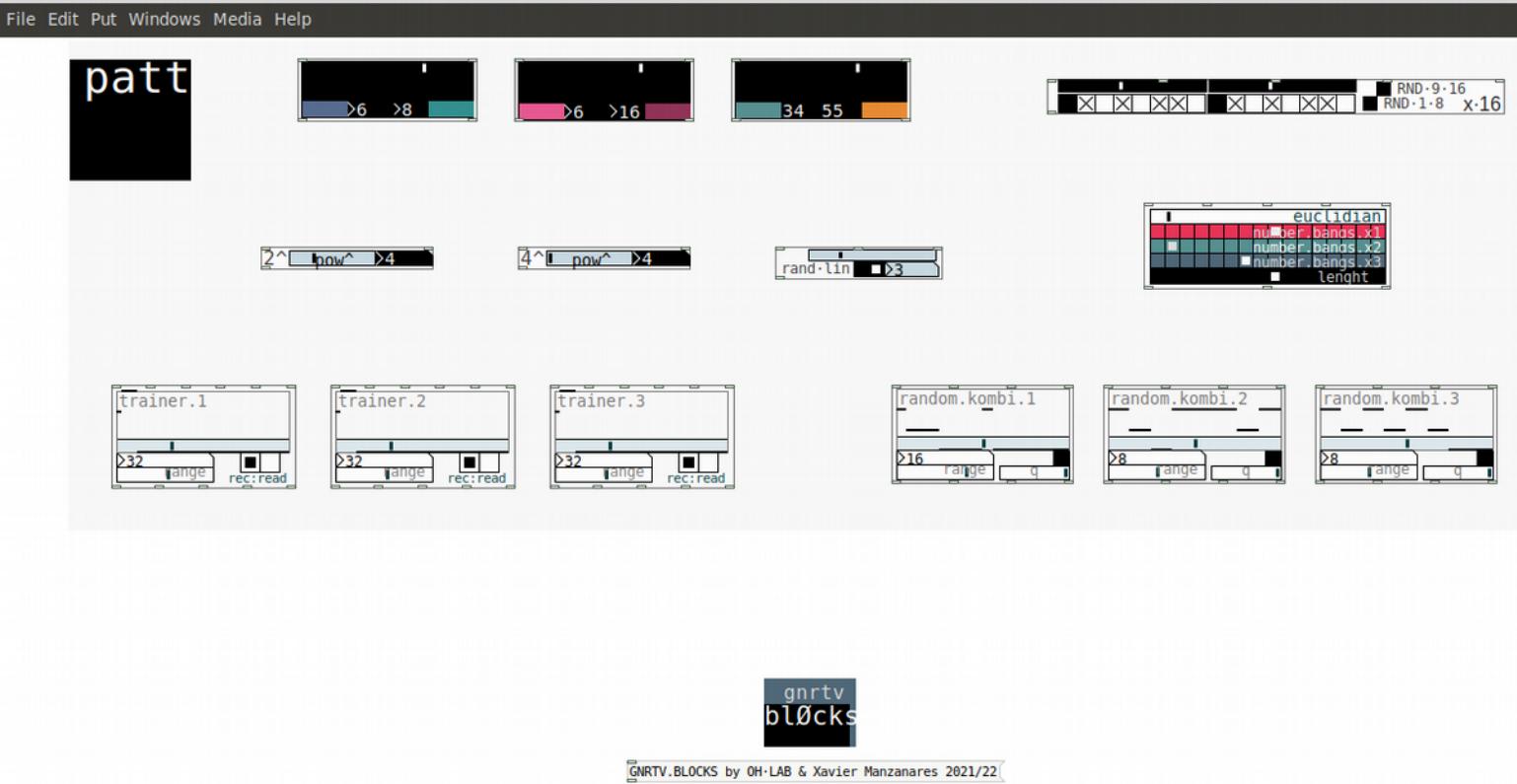
[gravity] creates a string of randomic numbers according an initial speed and initial range. It behaves somehow similar to attractors (lorenz etc).

[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.



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 trigg when it starts again.

[pow⁴] creates a power of 4 sequence (with ranges like 4, 16, 64, 256, 1024, 4096) which trigg 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 randomic 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.

sonifgnrtv
bl0cks

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.

synthgnrtv
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 timbral 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 timbral 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 timbral 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 timbral 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 timbral 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 timbral result.[tmbr bang]. Also the toggle [rnd] can change randomly between octaves

[oscillator] simple oscillator in which we can control its frequencies and range of them with [low-hi]

[osc~ hi] simple oscillator in which we can control its frequency (for high 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

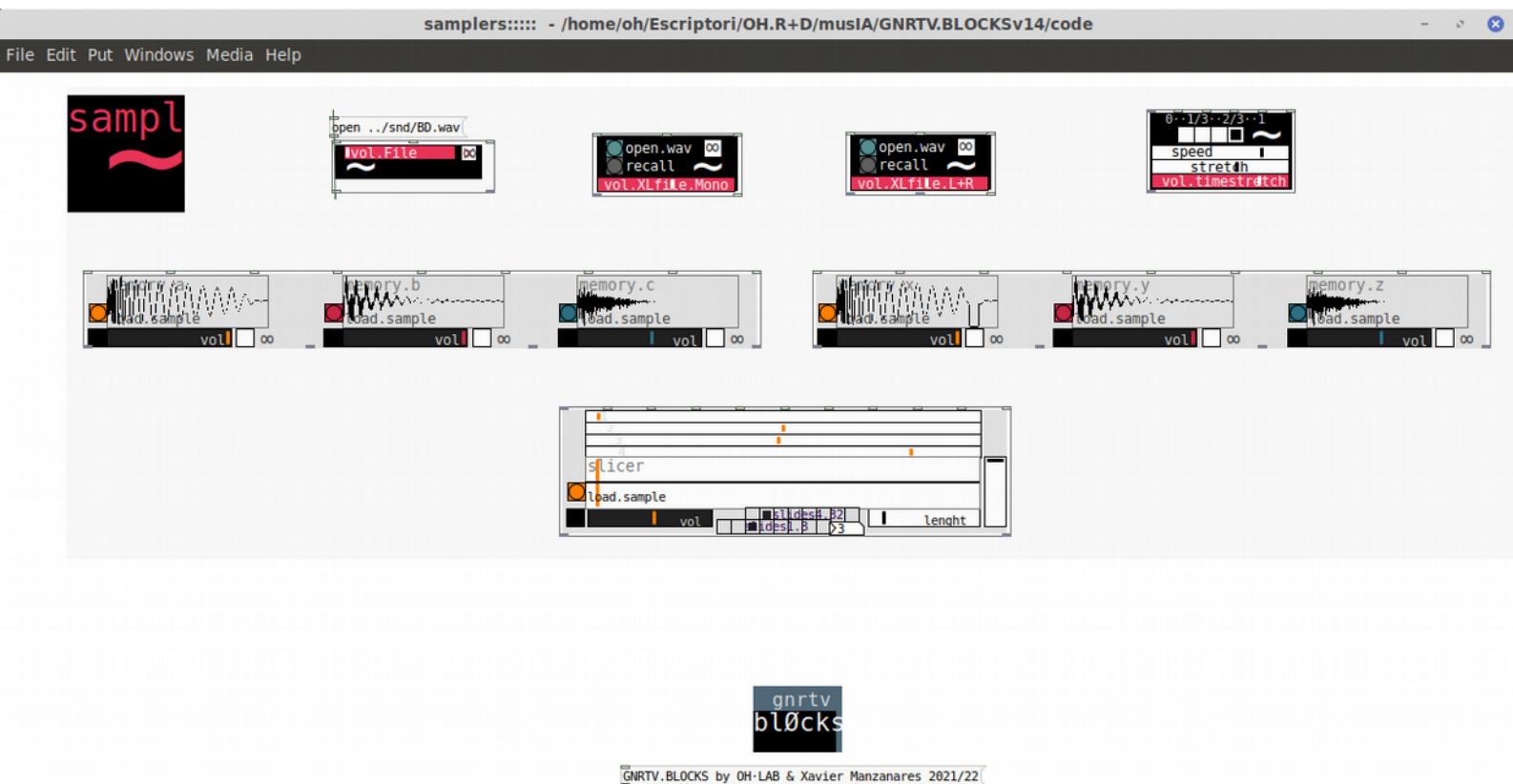
[lop~] low-pass filter

[hip~] hi-pass filter

[disto~] bit distortion filter

[delay~] quantized delay rack

[reverb~] reverb rack



sampl ~

[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.

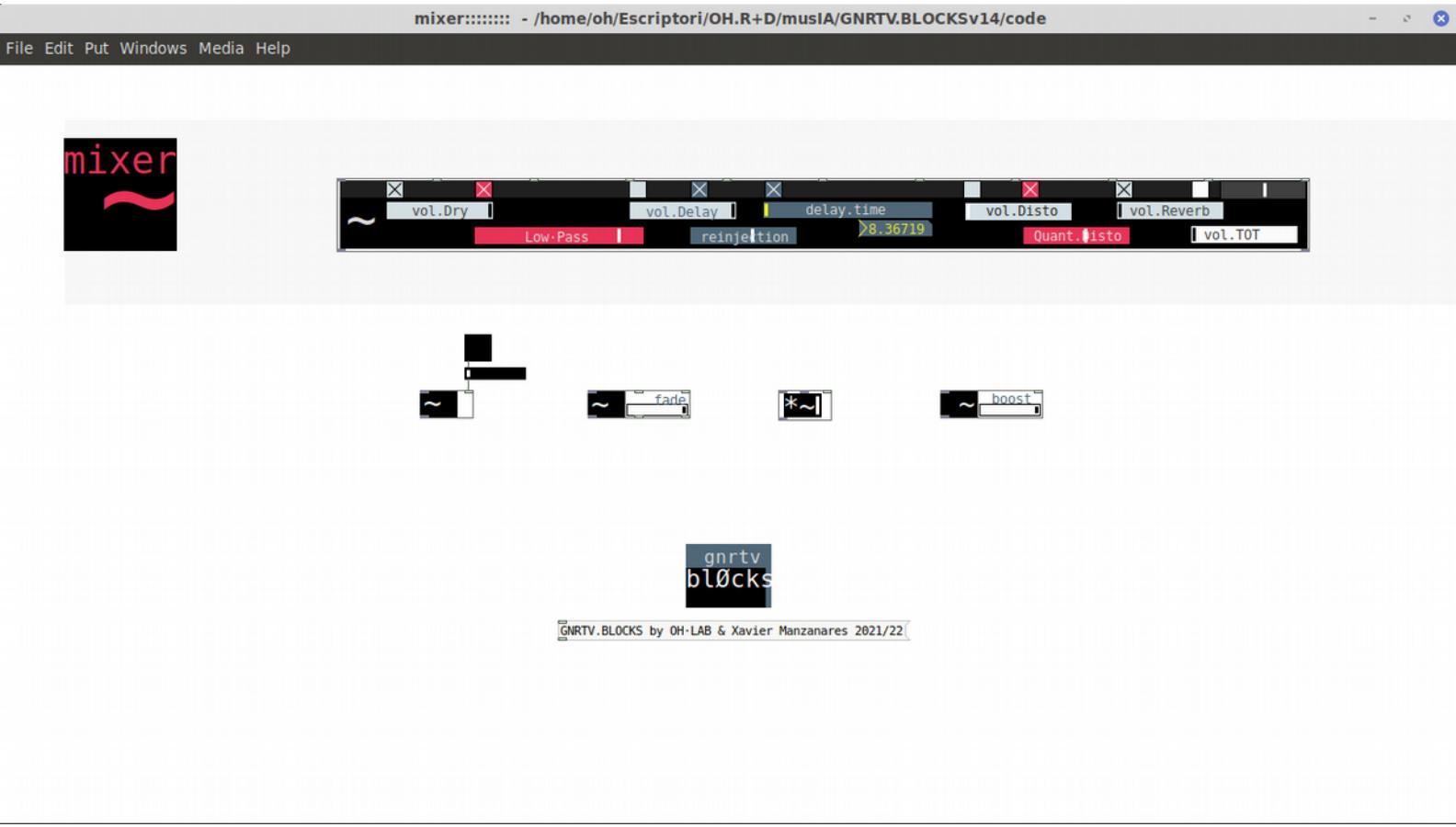
[XLfile~ L+R] block that plays a sample in which we can open the file through a prompt. The toggle ∞ loops the sample. Useful for stereo files.

[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 anmd stretch slider.

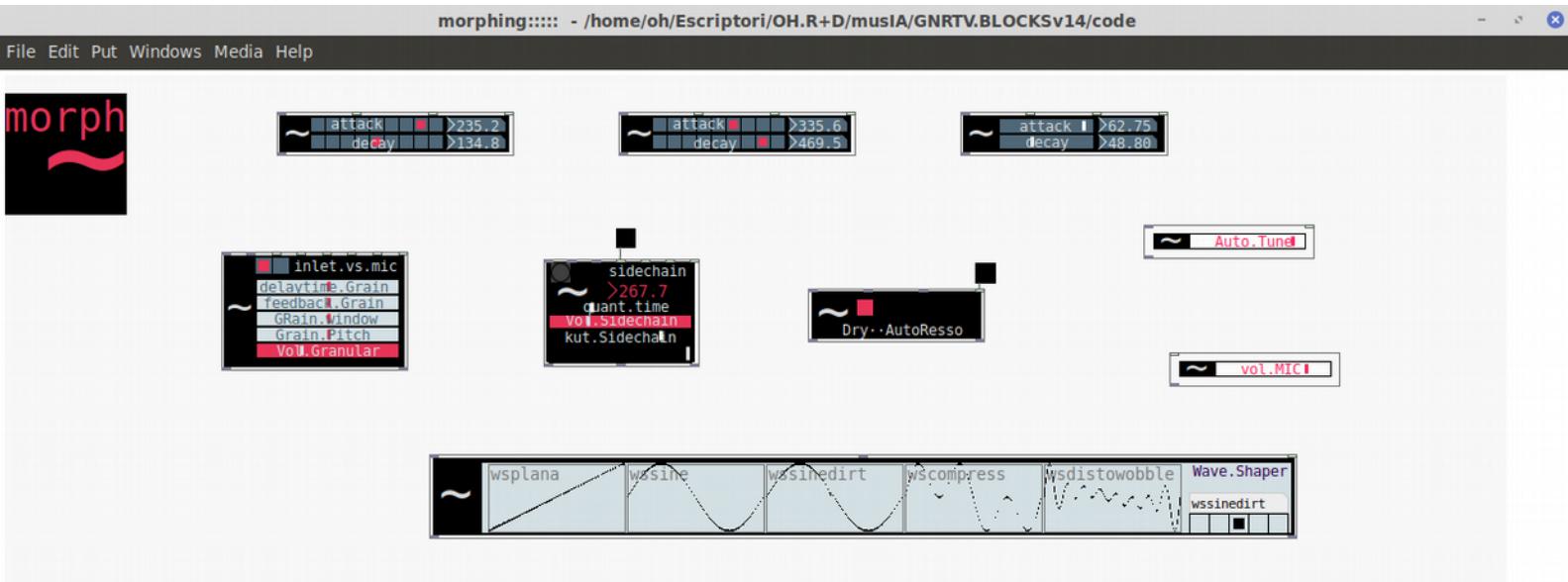
[sample.3x.abc] rhythm box composed by 3 samples that can be triggered, self looped and controlled the volume

[sample.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 lenght of the slices



mixer menú features a final mixer that can be automatized and is featuring a LowPass filter, a Quantized delay a Distortion 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.

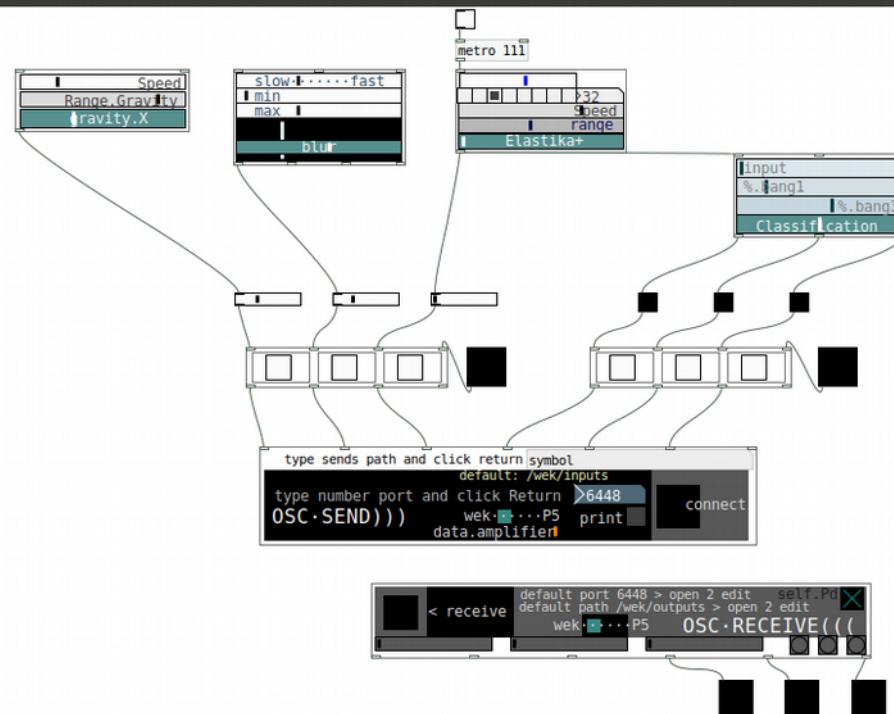
[sidechain] sidechain envelope effect from an incoming signal. Time scales are related to the main clock in order to make more precise rhythm effects. Parameters : quantity of time, volume and kut (envelope value in which signal is descending)

[autoresonance] Autoresonance filter block conformed by a signal which is modulated through a bandpass with the envelope of a secondary signal. (notice that this secondary can be the initial and duplicated incoming signal)

[autotune] autotune algorithm 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.

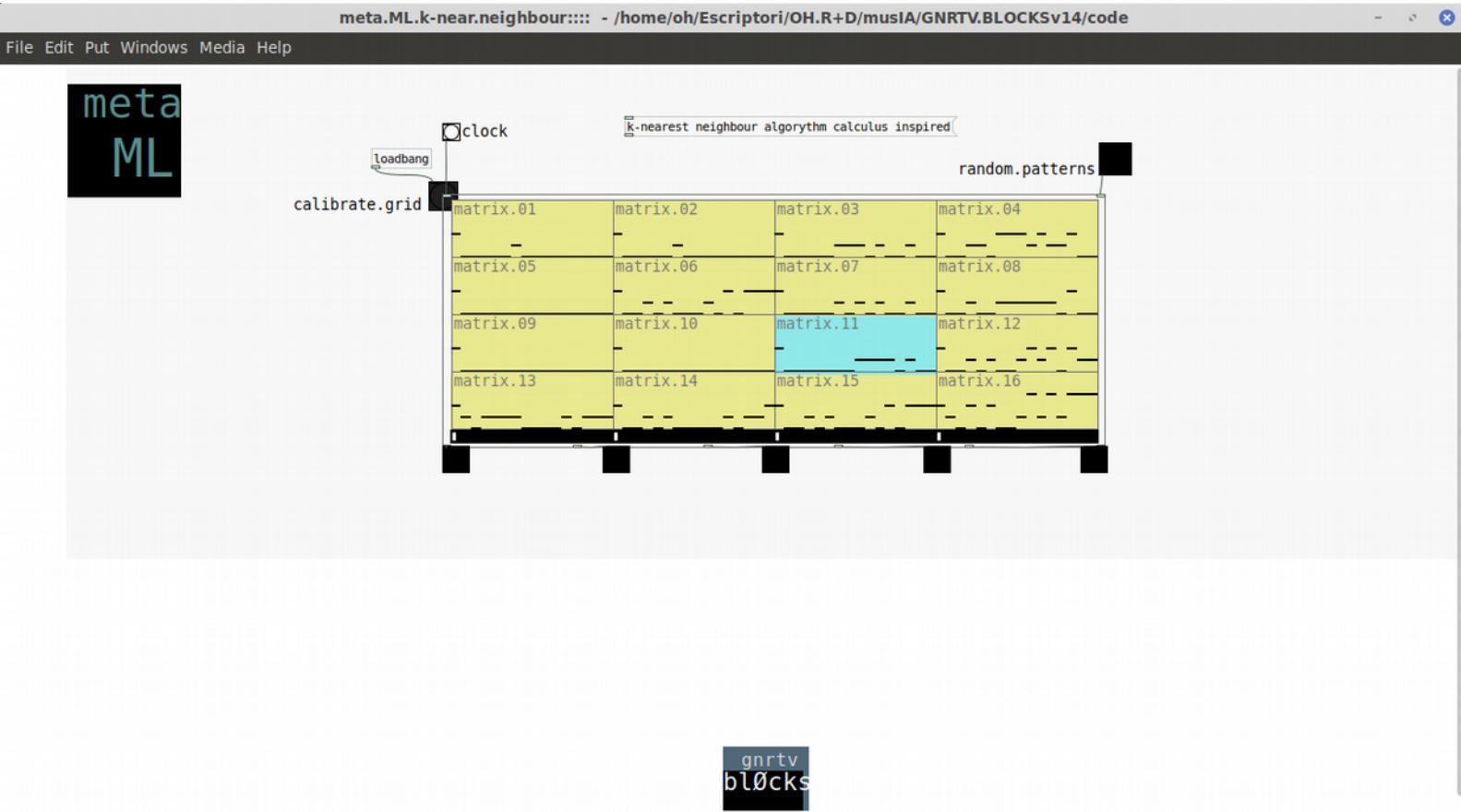
[waveshaper] Waveshaper block with 5 different shapes



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)



ML-knearest neighbour

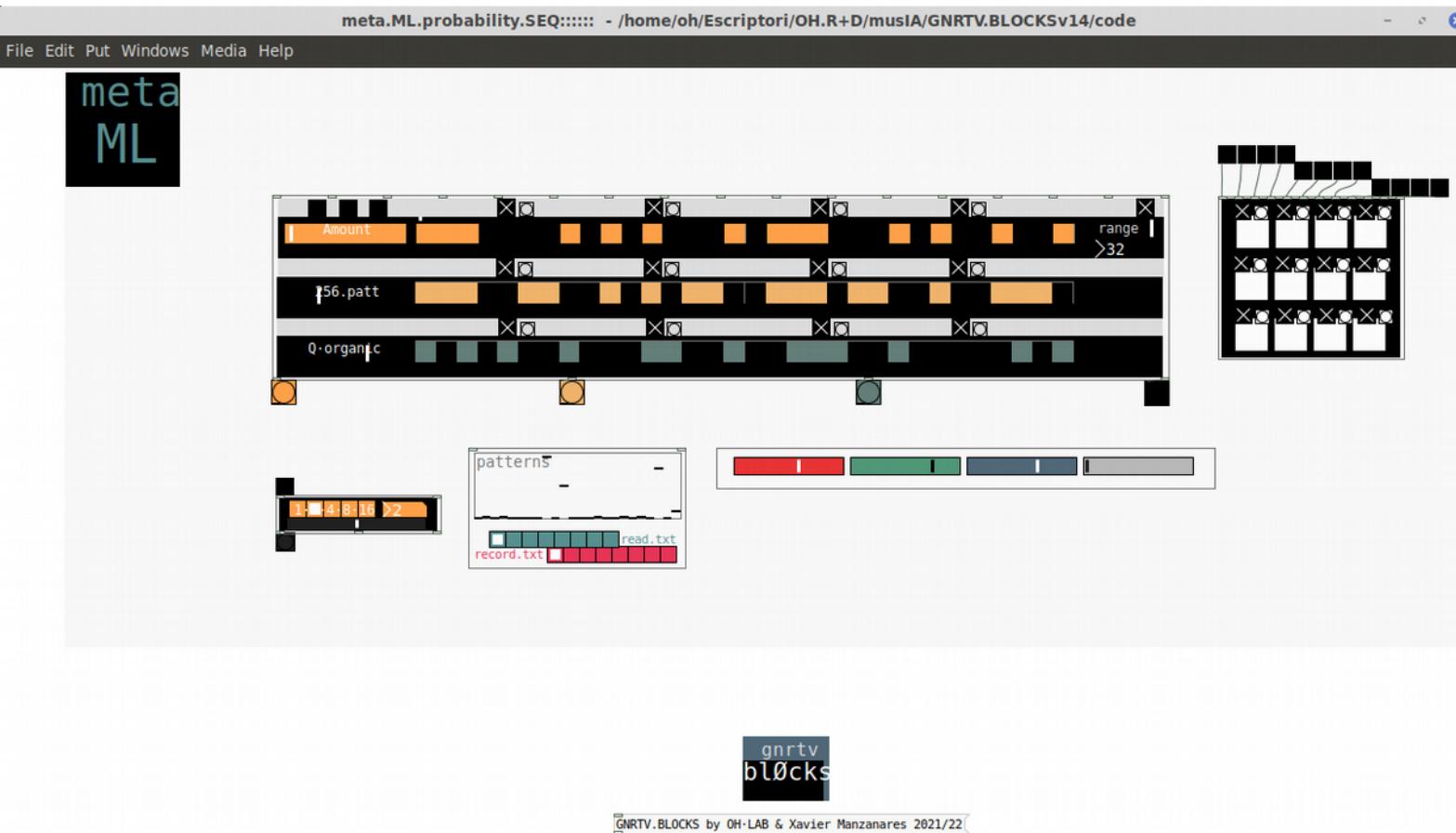
[matrix.16] block inspired in ML projects like latent loops * or melody mixer**

It creates a 16-cell matrix we creates randomly patterns on the corners. Then an algorythm calculus based on the k-nearest neighbour algorythm 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 ryhthm 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

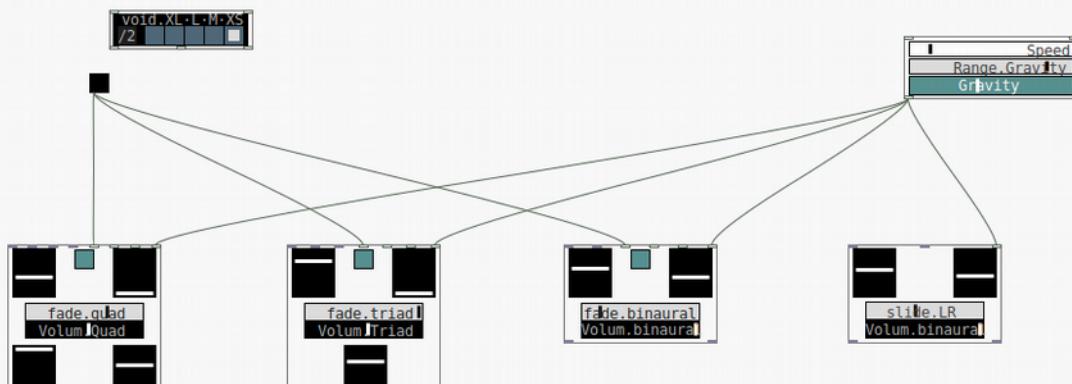
[3x.SEQ.32] 32 step sequencer that works with probability numbers associated to patterns.
Some kind of stokastik or NaiveBayes ML algorythms of probability. Therefore we can create patterned and ‘rigid’ structures and also more organic ones.

[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
bl0cks

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 algoryths 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 algoryths 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 algoryths 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.