Pure Data

Introduction course

Ninon Devis

ninon.devis@ircam.fr

Made with
love
for ATIAM

Content

01

Basics

Introduction and Elementary Notions in Pd

What is PD ?

- Developed by Miller Puckette @ IRCAM => Max MSP
- Open-source visual programming language
- Can even run on Raspberry Pi and smartphone
- Two major components:
  - Pd Vanilla: Manipulation of audio and MIDI
  - Pd Extended: Add video processing (obsolete)

STARTING

- Go to www.puredata.info for precompiled version (GNU/Linux, Mac & Windows compatible)

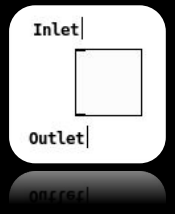
➔ Open Pd !

- The first opened window acts as a console which displays:
  - Errors of your patches
  - Messages when using the “print” object

Open a new patch

Ctrl/Cmd + N

BASICS



- *Unlock mode*: enable edition of patch
- *Lock mode*: run the graphical objects

Ctrl/Cmd + E

- Toggle: acting as a switch on/off

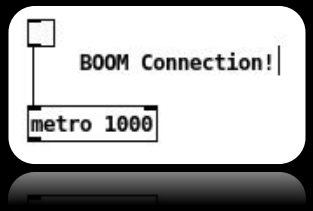
Shift + Ctrl + T

- Metronome: create an object, then write “metro 1000” inside

Shift + Ctrl + 1

- Bang: acting as a trigger

Ctrl/Cmd + B



Key Point

Always patch from up to down and right to left:
That’s how the signal flows in Pd

5 Types of Pd Boxes

Messages

Can contain any types of data and send it through its outlet. Can be clicked in lock mode to send the message

Ctrl + Shift + 2

Symbols

Similar to numbers but uses string instead of numbers

Ctrl + Shift + 4

Objects

Take data, make some changes to it and send the result

Ctrl + Shift + 1

Numbers

Store integers and floats. Value can be varied by sliding in lock mode

Ctrl + Shift + 3

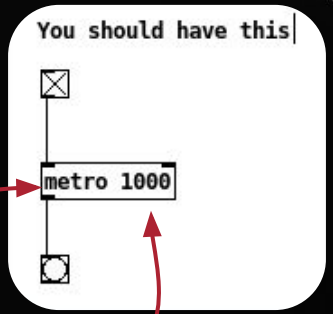
Comments

They are necessary as usual programming comment. Highly recommended to use them.

Ctrl + Shift + 5

Objects Boxes

Shift + Ctrl + 1

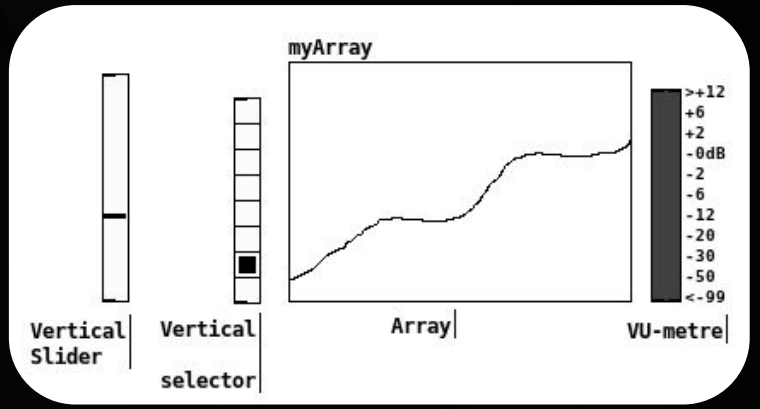


Contain:

- *Function*: defined by the first string
- *Arguments*: following item(s)
  - ➢ [name_object] by convention in this course

Examples:

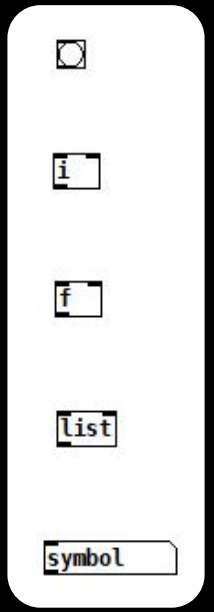
Graphical ones:



Kind of complete list of Pd objects:
http://blazicek.net/list_of_pure_data_objects.html

Data Types

- Bang: in the bang object
- Integer: in the [i] object
- Float: in the [f] object
- List: in the [list] object
- Symbol: in the symbol box

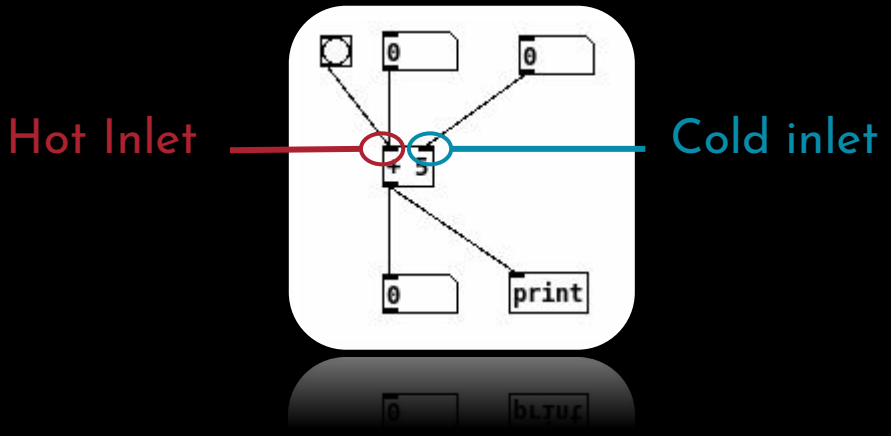


INLETS

Most important and complex concept of Pd

- Hot Inlet: the leftmost one. A message sent into a hot inlet triggered its execution.
- Cold inlet: the remaining ones. Useful to change the argument of the object, store it into the object but do not trigger a calculation.

A bang is used to triggered the cold inlet storage



Signal Flow

Up to down and right to left

Can be avoided by using:

- [trigger]
- [route]

Find Help !

Quick link

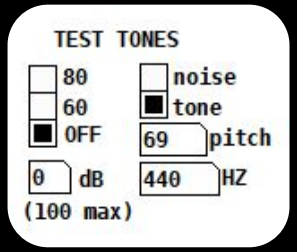
Select 2 boxes
&
Ctrl + K

Pd has been made to learn by yourself

- Objects help are in patch form
- You can try, modify, copy / paste them... till you get it !

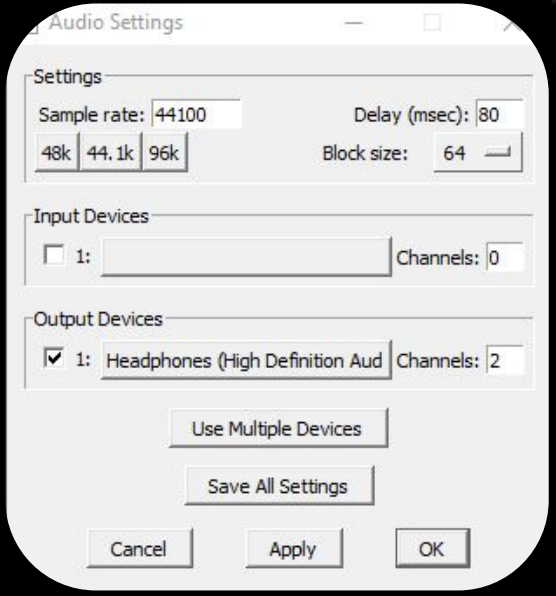
Let's Check the sound

- Media > Test Audio and Midi



If you don't have any sound...

- Media > Audio Settings



02

Synthesizers

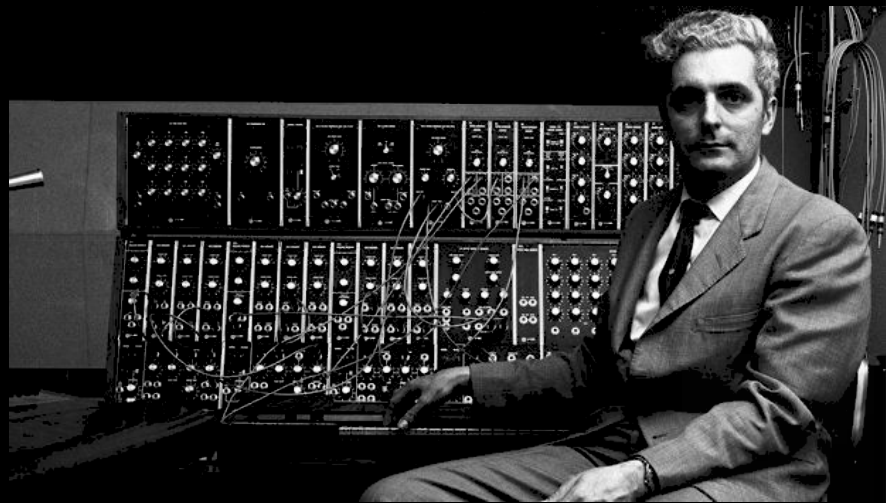
Major Components

One of the most fundamental instrument
in electronic music ❤️

.....

Originally based on a modular architecture

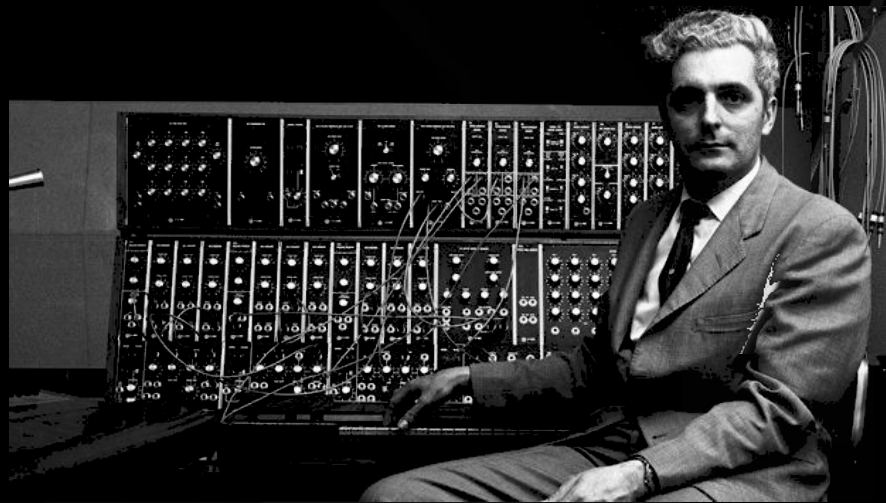
- Oscillators: generate the tone
- Filters: emphasize or remove certain frequencies
- Amplifiers: control the gain of the synth



Completed by some modulation modules:

- LFO (low frequency oscillator): modulate either the frequency or gain of the oscillator(s), or the frequency of the filters.
- Envelope Generator: control changes in frequency or gain over the note.

Let's create a simplified Minimoog with Pd



03

Mini-Moog

Process Building

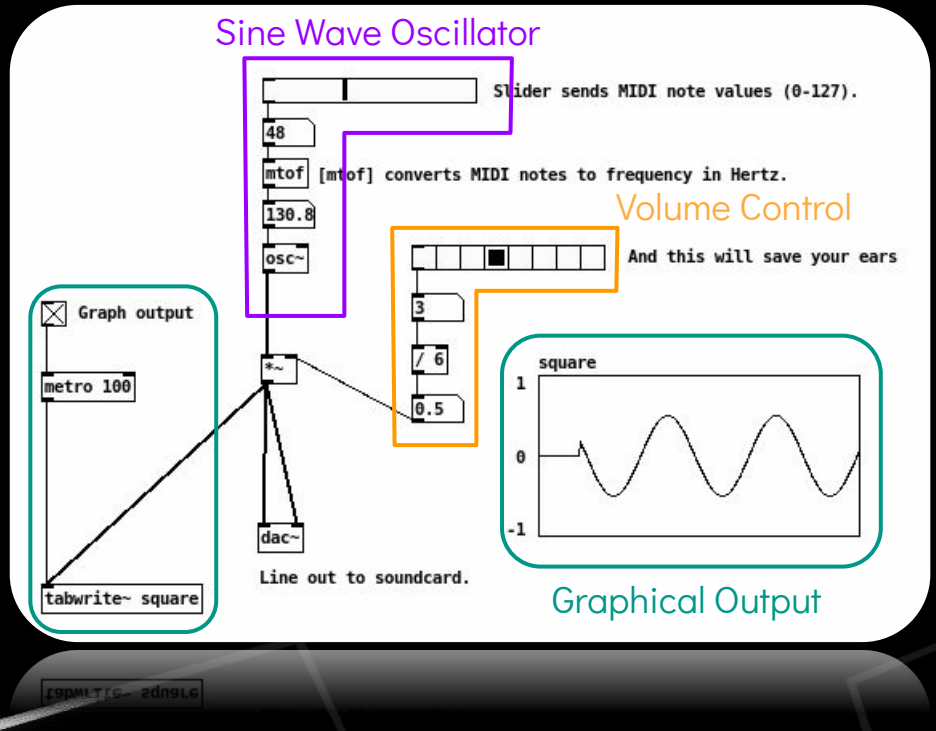
OSCILLATORS & FREQUENCIES

Audio signal
=
stream of numbers [-1 ; 1]

slider	⇧+Ctrl+J
numbers	⇧+Ctrl+3
objects	⇧+Ctrl+1
toggle	⇧+Ctrl+T
selector	⇧+Ctrl+I
comments	⇧+Ctrl+5
table	⇧+Ctrl+A

Mac users: ⌘ instead of Ctrl

Sine Wave Oscillator

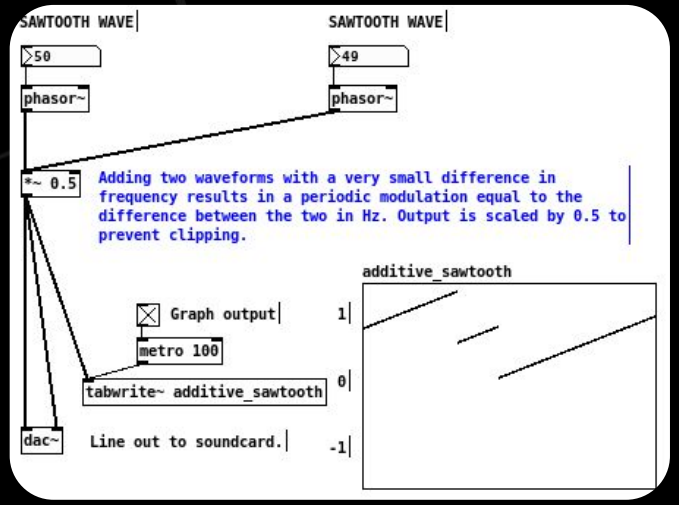


Typical signature " ~ " for audio objects

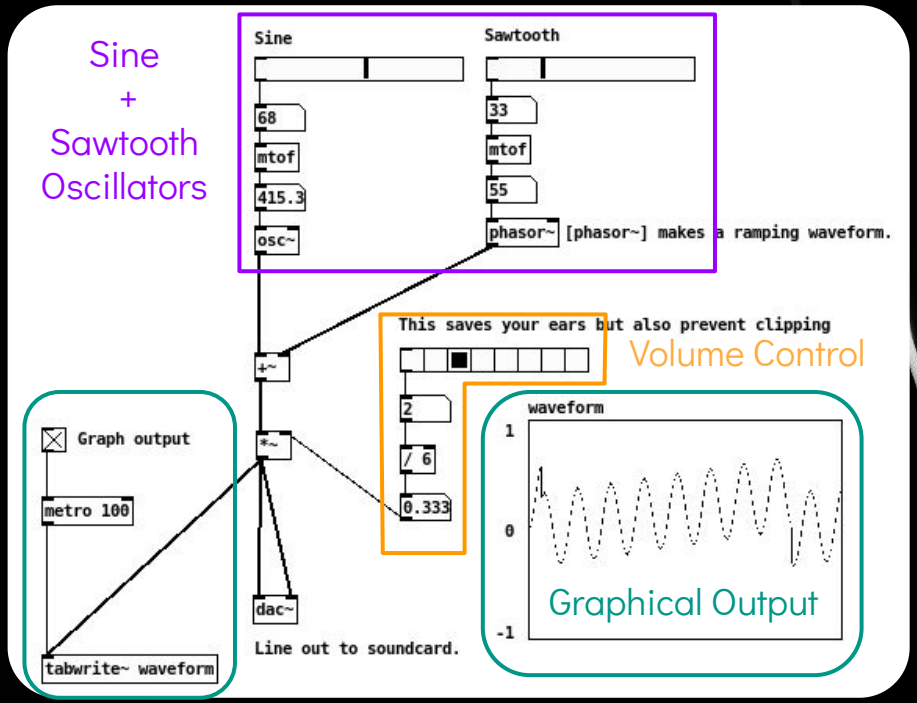
Additive Synthesis

- Notice from your patch the (slight) difference between *number/message* and *audio* cable

Beating frequency for fat bass



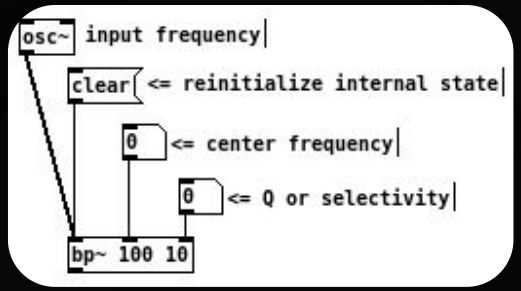
Sine + Sawtooth



FILTERS

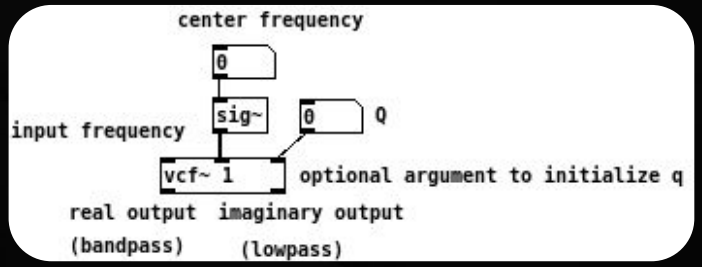
Easy to manipulate, 3 classic types:

- Low pass: [lop~]
- High pass: [hip~]
- Band pass: [bp~]

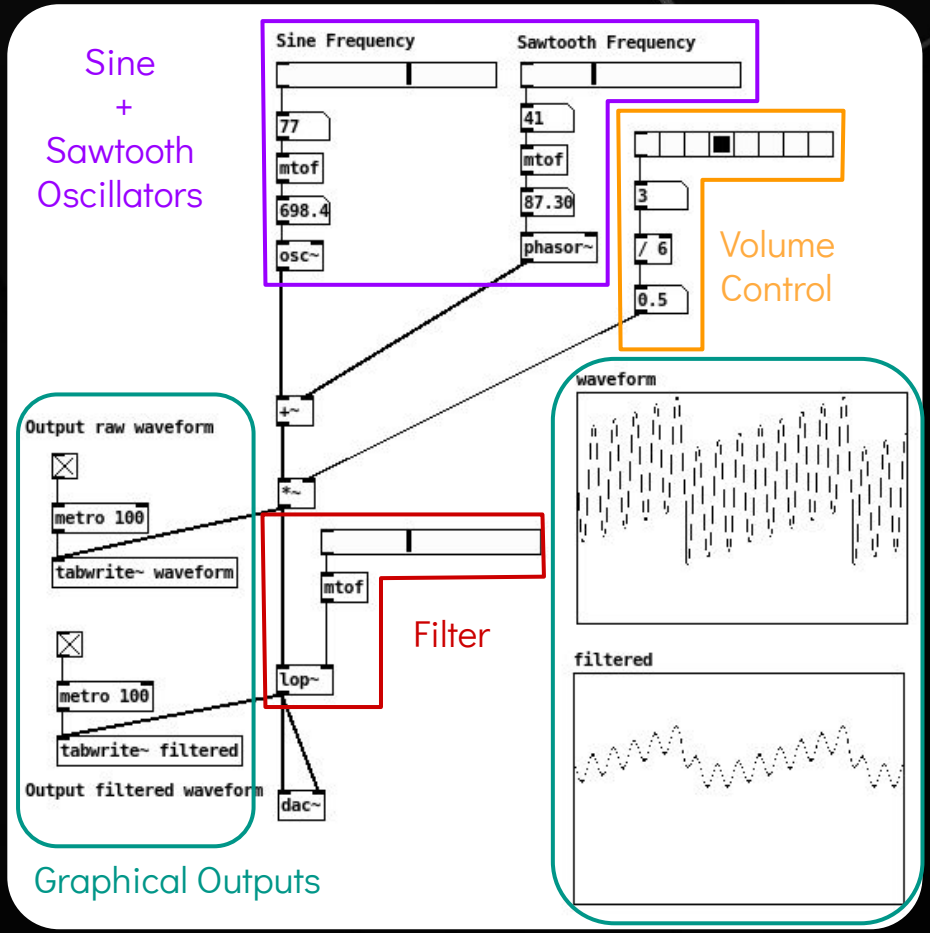


And a VCF (voltage controlled filter):

- [vcf~]: resonnant bp and lp that take audio signal to set center frequency
- Can change continuously in time !



Low Pass Filter



AMPLIFIERS

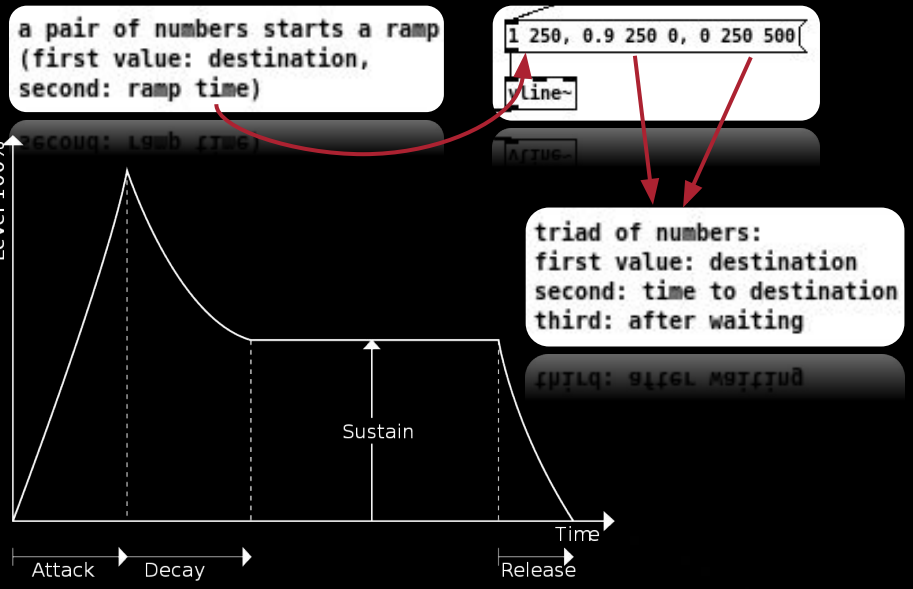
Controls the gain of the synth

You may want to have a look on:

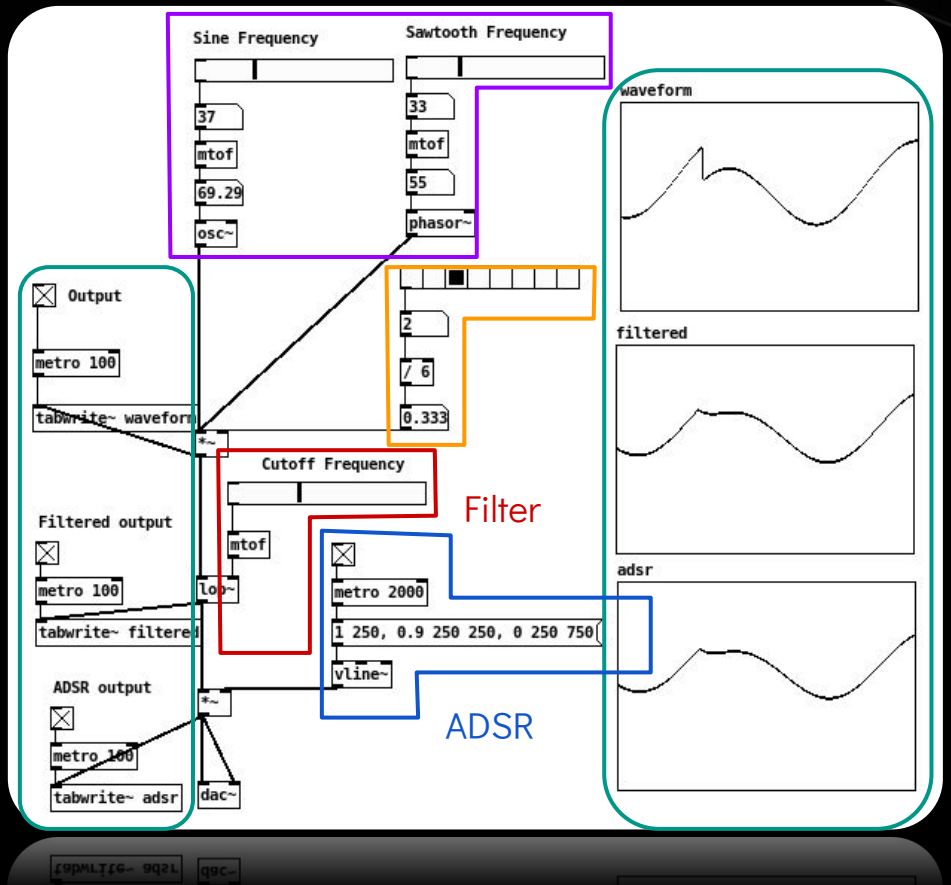
- [line~], [tabread4~], [vline~]

Let's make a classic ADSR with [vline~]

➔ Generate an audio ramp



ADSR



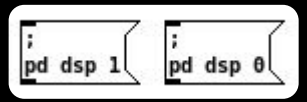
Keyboard Control

Object [key] return ASCII value

- Can be treated as MIDI note

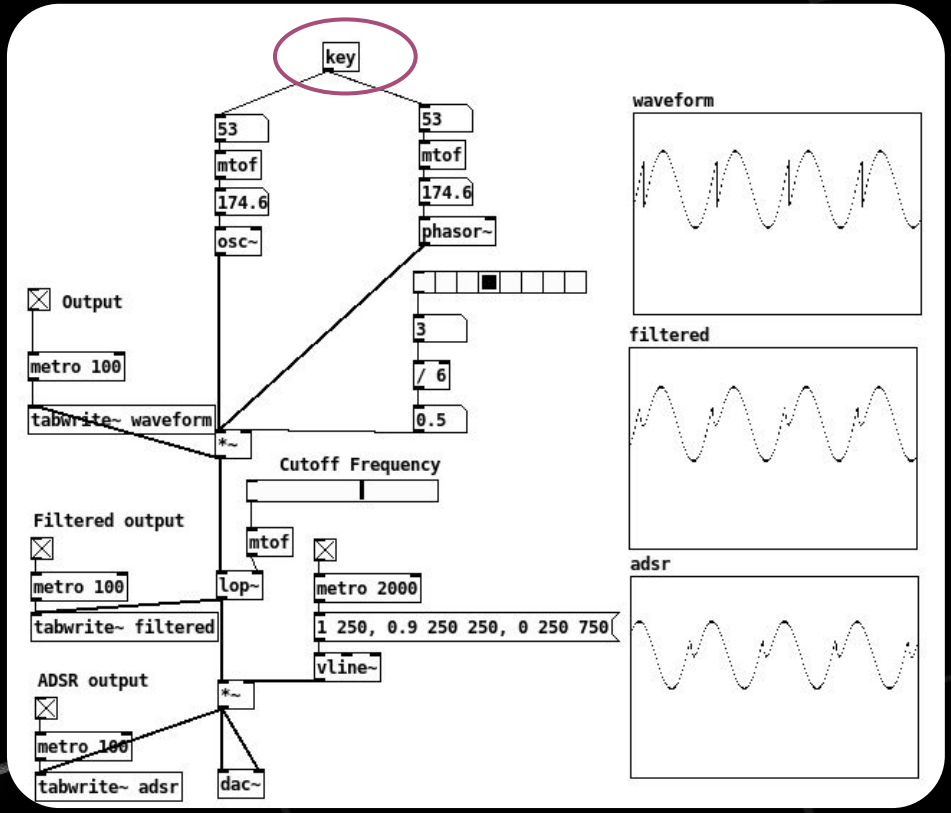
For those who want to use a MIDI keyboard look at [notein] object

DSP control on/off:



- It's a message box, not an object

Keyboard plays note

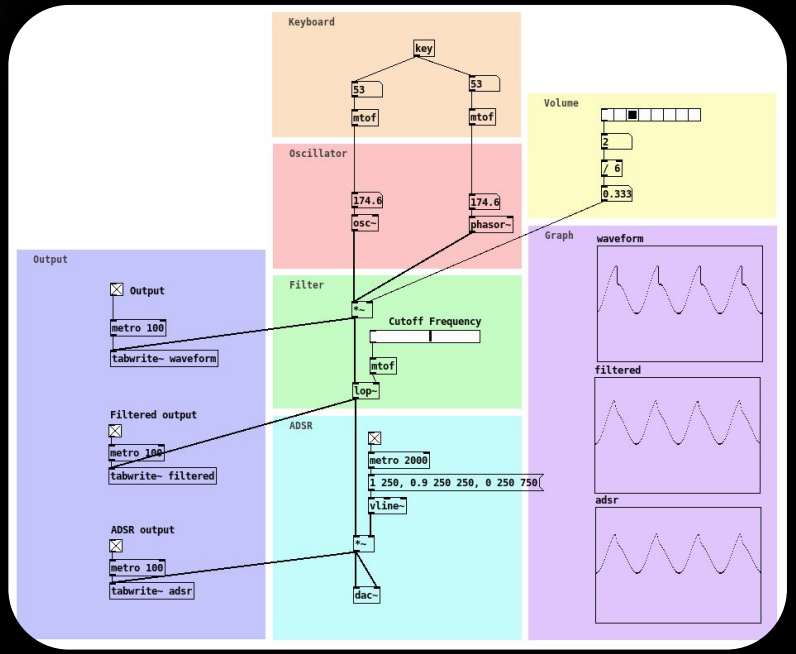


SUBPATCHES

pd name_subpatch

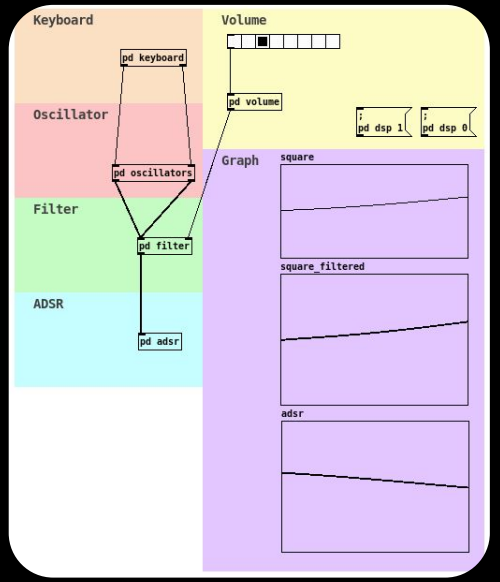
Some lightness and clarity with:

- Colors
- Subpatches/abstractions
  [pd name_subpatch]



Your subpatch appears !

- Add the piece of code you want
- Add as many [inlet] and [outlet] you need: this will add the inlet/outlet on the original [pd name_subpatch]
- Plug your subpatches together



Be aware of the difference between [inlet] and [inlet~]

04

PROJECT

Presentation

PROJECT

You can already manage half of the project

Patch me some monophonic synths with the following features:

- at least one triangle oscillator
- at least one voltage controlled filter
- one capable of playing notes with keyboard so that the note lasts only the time the key is pressed
- with a delay of 1 second on your signal
- which plays melodies from random pitch and duration

Please remember: ¼ of the final grade is on the comprehensibility of your patches: usage of abstraction, comments, organization…

You will need to look by yourself the necessary objects:
we have not seen all of them!