Pure Data

❖ Introduction course ❖

pd~

Ninon Devis

https://ninon-io.github.io/

Made with
love
for ATIAM

Table of Content

01

Basics

Starting
PD boxes
Data types
Inlets & Signal flow

02

Synthesizer

Origins
Major Components
Different types of synthesizers
Analog versus Digital

03

Mini-Moog

Additive synthesis
Keyboard Control
Subpatches

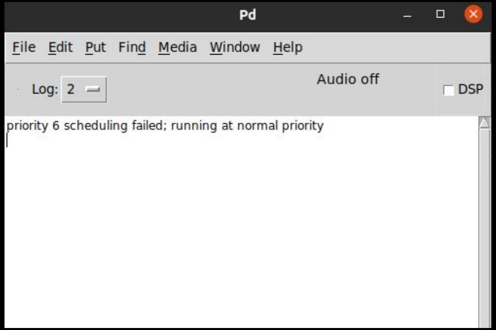
01

BASICS

Introduction and Elementary Notions in Pd

WHAT IS PD ?

- Open-source visual programming language
- Developed by Miller Puckette @ IRCAM => Max MSP
- Can even run on Raspberry Pi and smartphone
- 3 major components:
  - Pd Vanilla: Manipulation of audio and MIDI
  - Purr Data: pretty GUI and many new functions
  - Pd Extended: obsolete since 2013



```
priority 6 scheduling failed; running at normal priority
```

This is your console

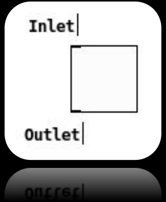
STARTING

Open a new patch

Ctrl/Cmd + N

- Go to www.puredata.info for precompiled version (GNU/Linux, Mac & Windows compatible)
- The first opened window acts as a console which displays:
  - Errors of your patches
  - Messages when using the "print" object

BASICS


Inlet

Outlet

Quick link

Select 2 boxes
&
Ctrl + K

Ctrl/Cmd + E

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

Shift + Ctrl + T

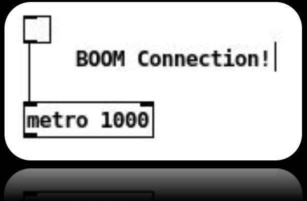
- Toggle: acting as a switch on/off

Shift + Ctrl + 1

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

Ctrl/Cmd + B

- Bang: acting as a trigger


BOOM Connection!

metro 1000

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

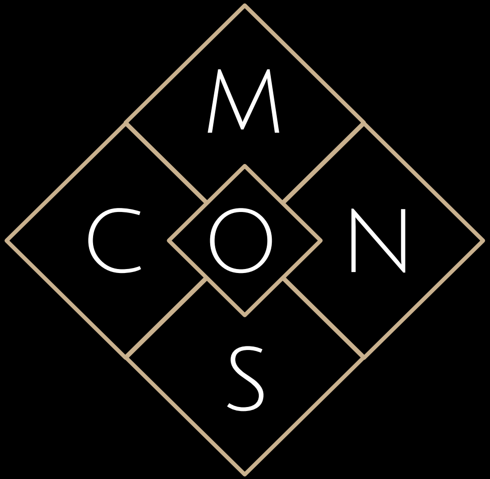
Can contain any types of data and send it through its outlet. Can be clicked in lock mode to send the message.

Ctrl + Shift + 2

Symbols

Store a symbol (i.e. string)
Often filenames

Ctrl + Shift + 4



Objects

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

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.

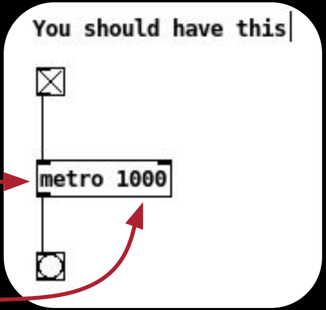
Ctrl + Shift + 5

OBJECTS

Shift + Ctrl + 1

Contain:

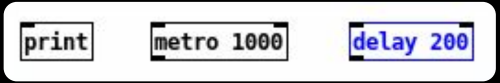
- *Function*: defined by the first string
- *Arguments*: following item(s)
  - ➢ [name_object] by convention in this course



You should have this

metro 1000

Classic examples:



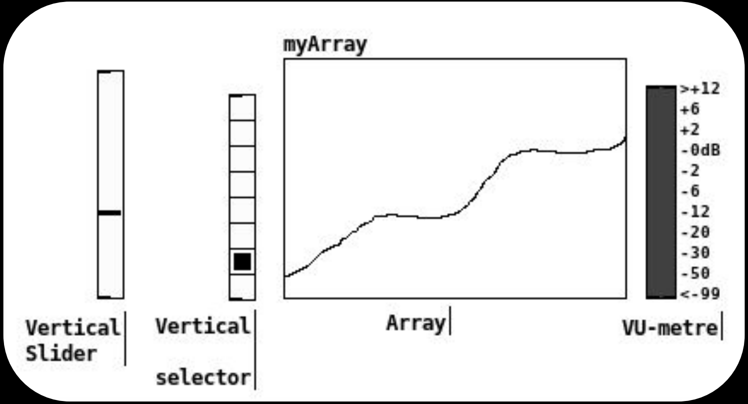
print

metro 1000

delay 200

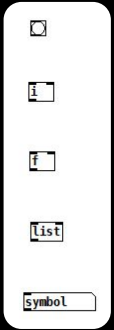
Kind of complete list of Pd objects:
http://blazicek.net/list_of_pure_data_objects.html

Graphical examples:



DATA TYPES

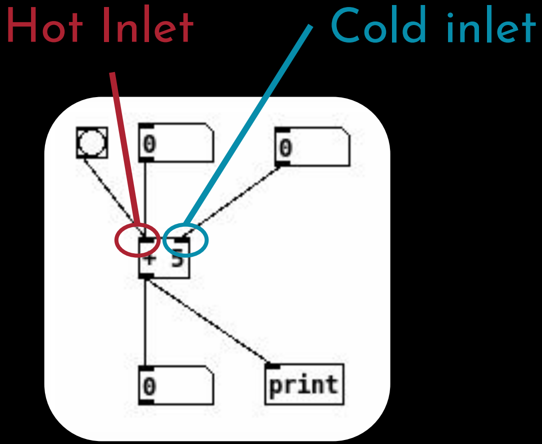
- Bang: in the bang object
- Integer: in the [i] object
- Float: in the [f] object
- List: in the [list] object
- Symbol: in the symbol box



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]

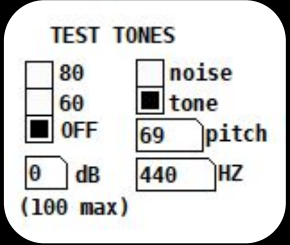
Find Help !

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

1. Media > Test Audio and Midi



TEST TONES

□ 80
□ 60
■ OFF

□ noise
■ tone

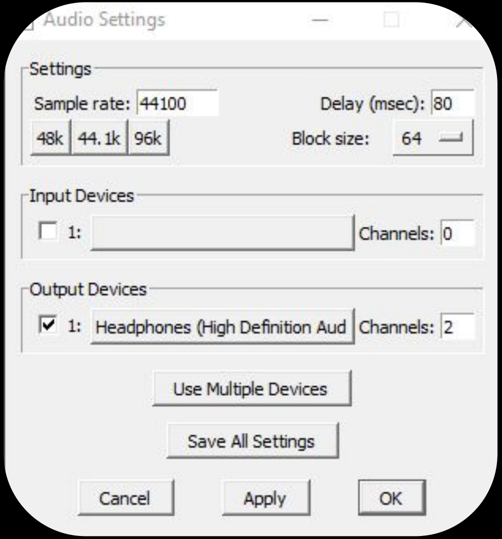
69 pitch

0 dB
(100 max)

440 HZ

If you don't have any sound…

2. Media > Audio Settings



Audio Settings

Settings
Sample rate: 44100
Delay (msec): 80
48k 44.1k 96k
Block size: 64

Input Devices
☐ 1: Channels: 0

Output Devices
☑ 1: Headphones (High Definition Aud Channels: 2

Use Multiple Devices

Save All Settings

Cancel Apply OK

02

SYNTHESIZERS

Major Components

https://learningsynths.ableton.com/en/get-started

Origins & Components

- 1800: Theremin
- 1950: Early synths
- 1970: Bob Moog introduces the Minimoog ❤

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.

OSCILLATORS

https://pudding.cool/2018/02/waveforms/

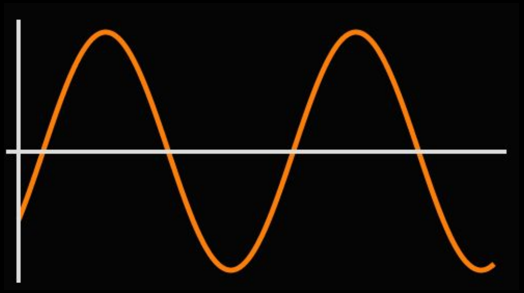
- Oscillation of voltage:
  Oscillators = VCO
  (voltage control oscillator)
- Range often: [-5 V ; +5V]

Increase of 1V = Increase 1 octave

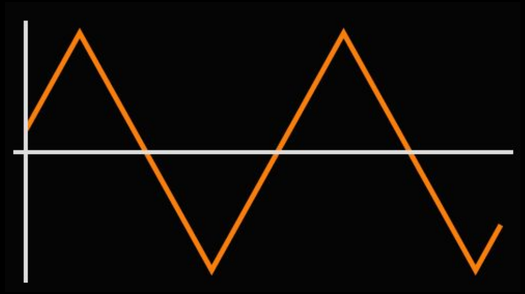
The motion creates a waveshape, and it's that wave shape that gives the tone its fundamental characteristics.

SINE

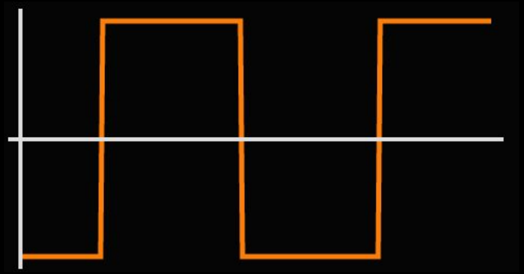
fundamental waveform



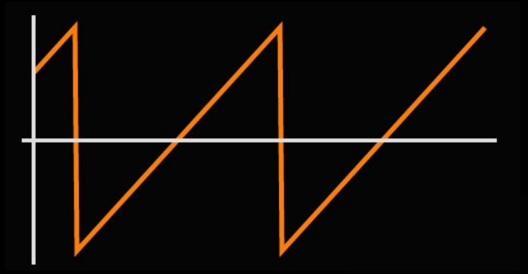
TRIANGLE



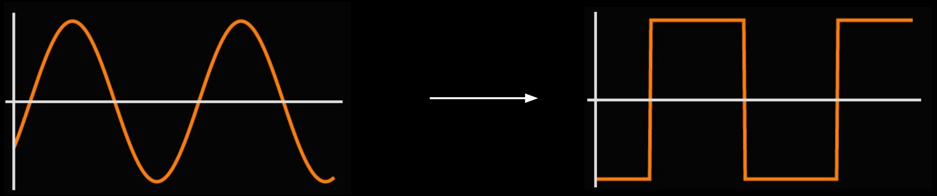
SQUARE



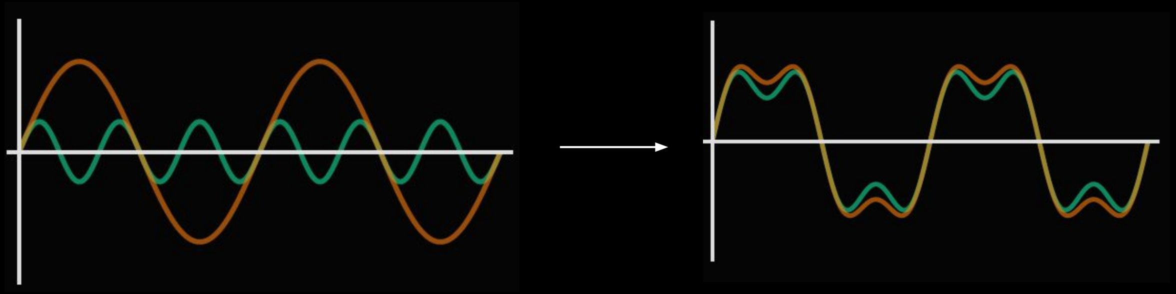
SAWTOOTH



ADDITIVE SYNTHESIS



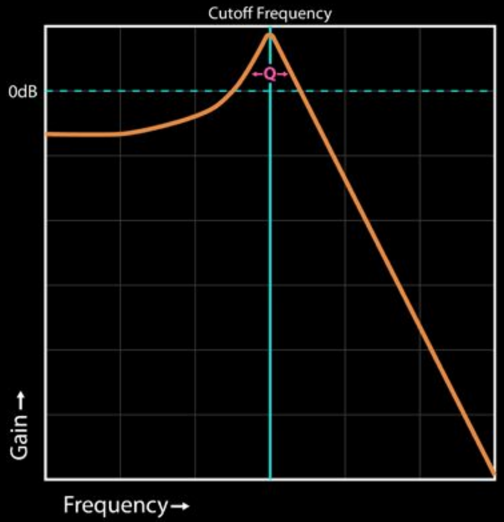
Building a sound by combining multiple sine waves of differing levels and frequencies



SUBTRACTIVE SYNTHESIS

Partials of a sound are attenuated by a filter to alter the timbre of a sound

- Types of filters: Hi-Pass, Low-Pass, Band-Pass, Band-Stop.
- VCF = Voltage control Filter
- What is a resonant filter ?



03

MINI-MOOG

Process Building

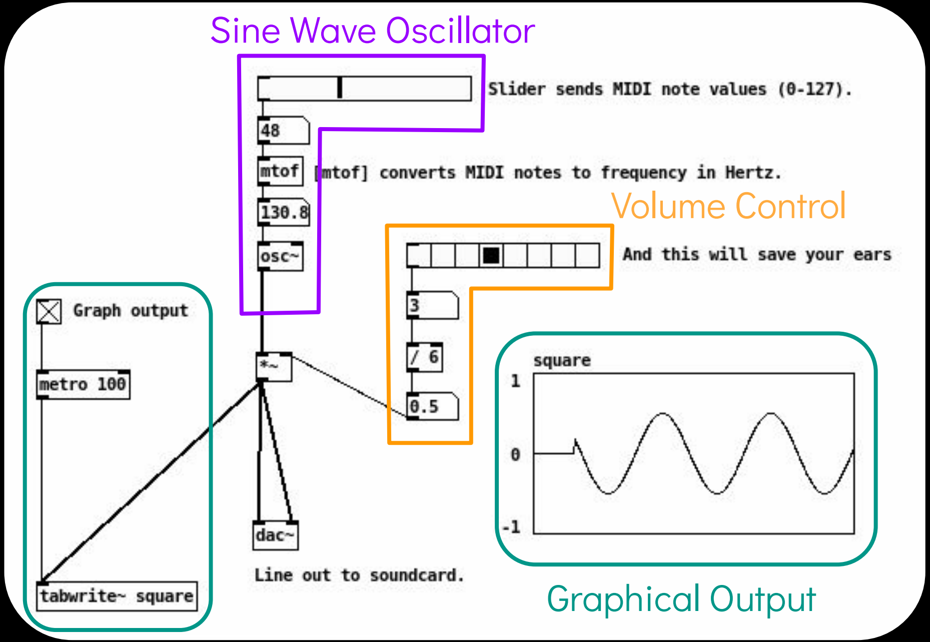
OSCILLATORS & FREQUENCIES

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



Slider sends MIDI note values (0-127).

[mtof] converts MIDI notes to frequency in Hertz.

And this will save your ears

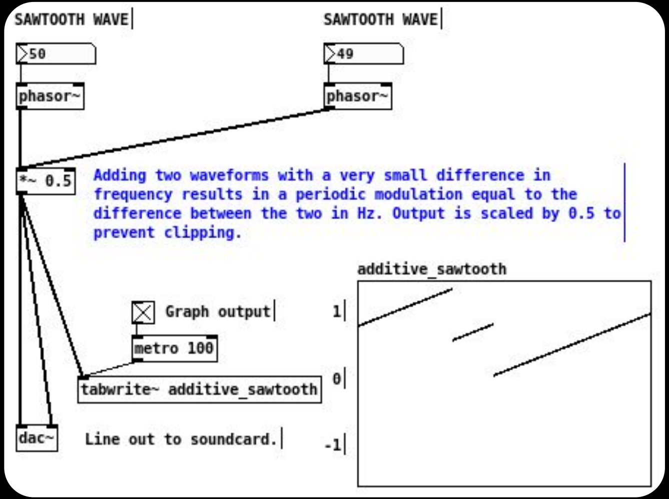
Line out to soundcard.

Typical signature “ ~ ” for audio objects

ADDITIVE SYNTHESIS

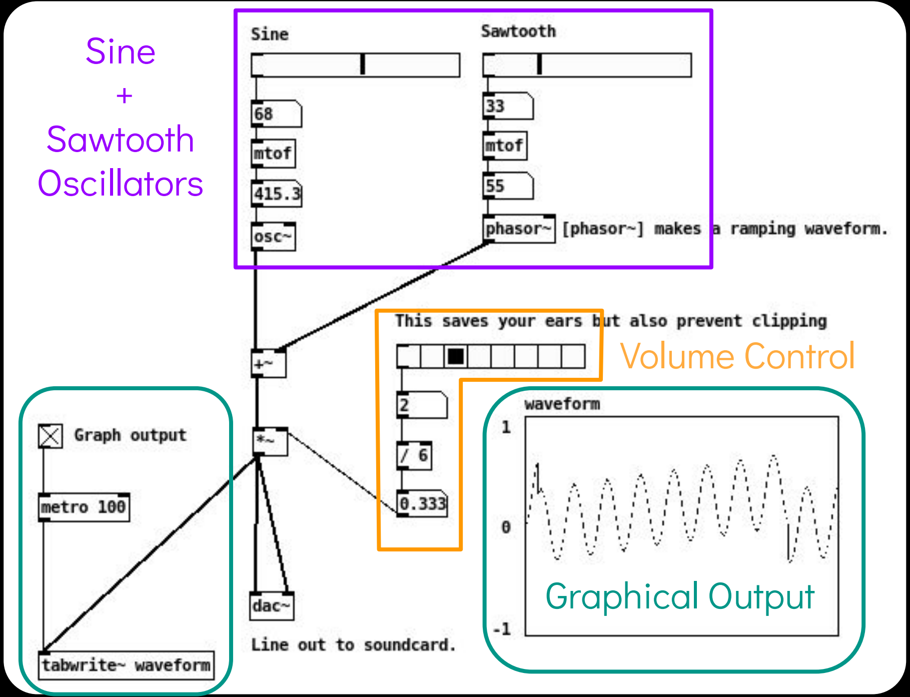
- Notice from your patch the (slight) difference between number/message and audio cable

Beating frequency for fat bass



Adding two waveforms with a very small difference in frequency results in a periodic modulation equal to the difference between the two in Hz. Output is scaled by 0.5 to prevent clipping.

Sine + Sawtooth



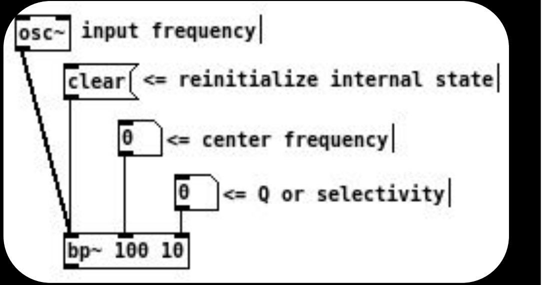
[phasor~] makes a ramping waveform.

This saves your ears but also prevent clipping

FILTERS

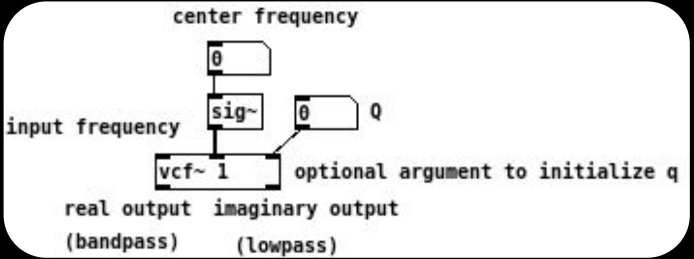
Easy to manipulate, 3 classic types:

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

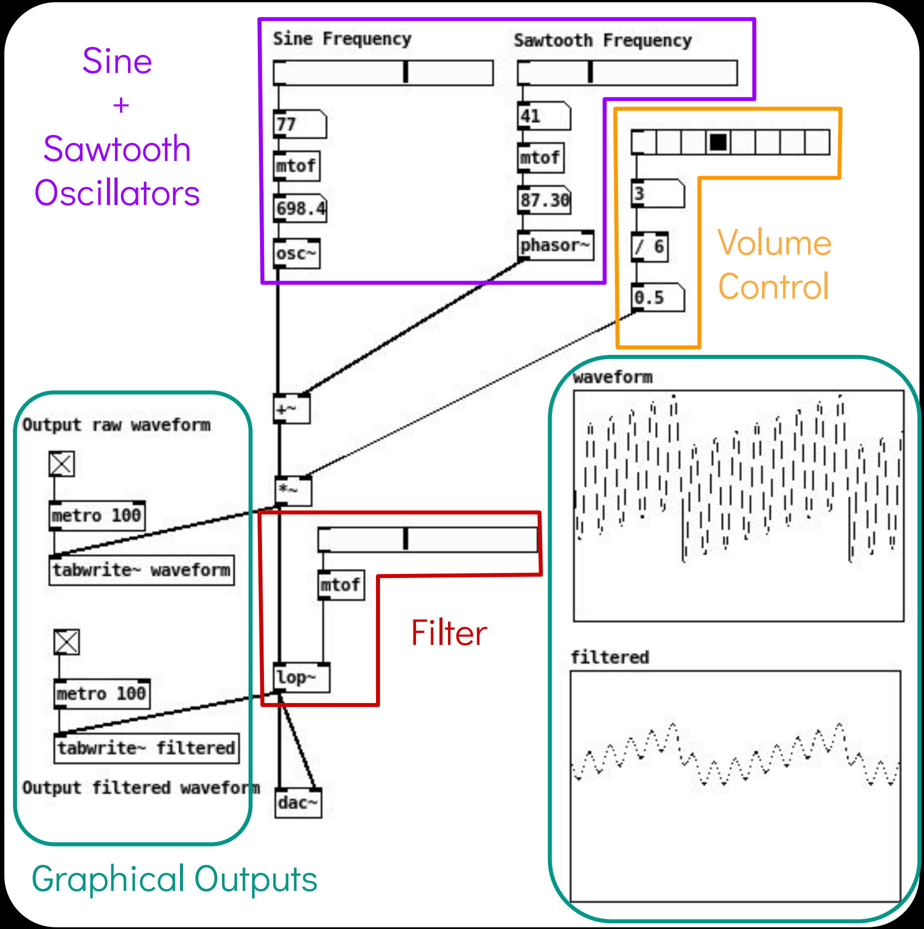


And a VCF (voltage controlled filter):

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



Low Pass Filter


Sine + Sawtooth Oscillators

Volume Control

Filter

Graphical Outputs

AMPLIFIERS

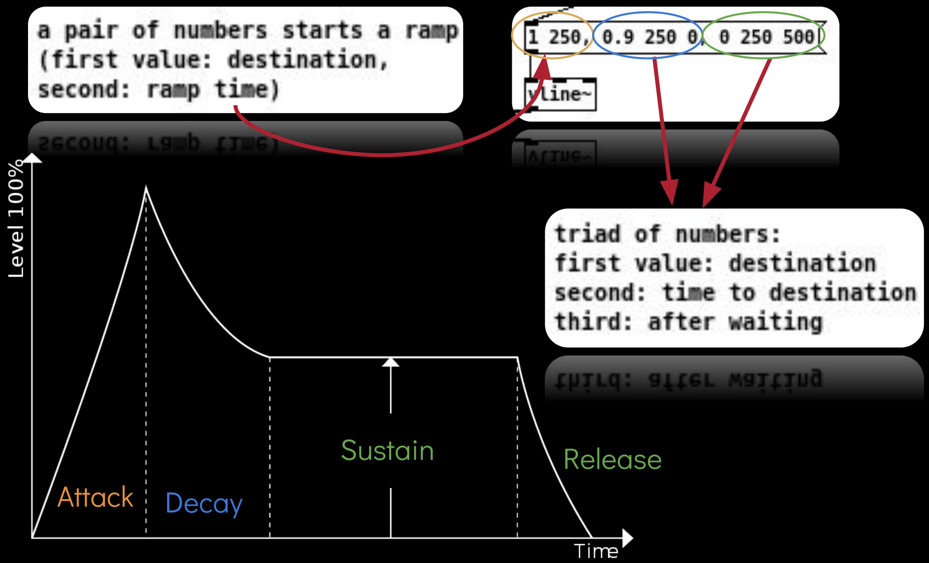
Controls the gain of the synth

You may want to have a look on:

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

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

→ Generate an audio ramp

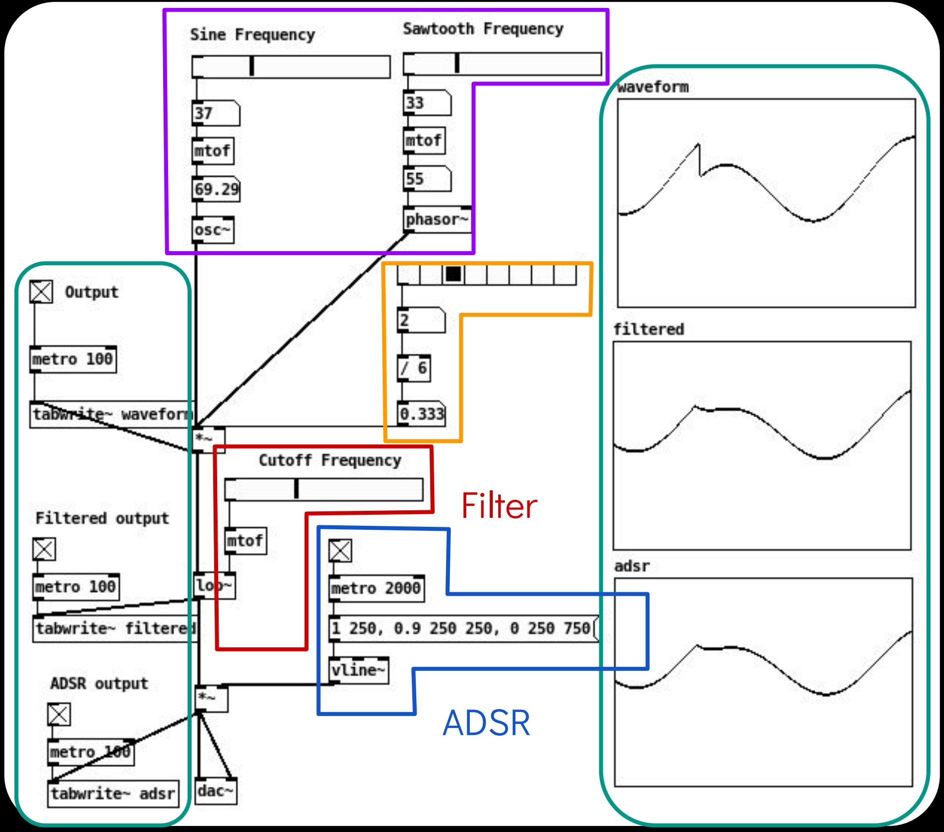


a pair of numbers starts a ramp (first value: destination, second: ramp time)

```
1 250, 0.9 250 0, 0 250 500
vline~
```

triad of numbers: first value: destination second: time to destination third: after waiting

ADSR



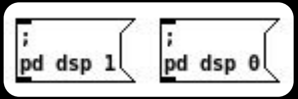
KEYBOARD CONTROL

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:

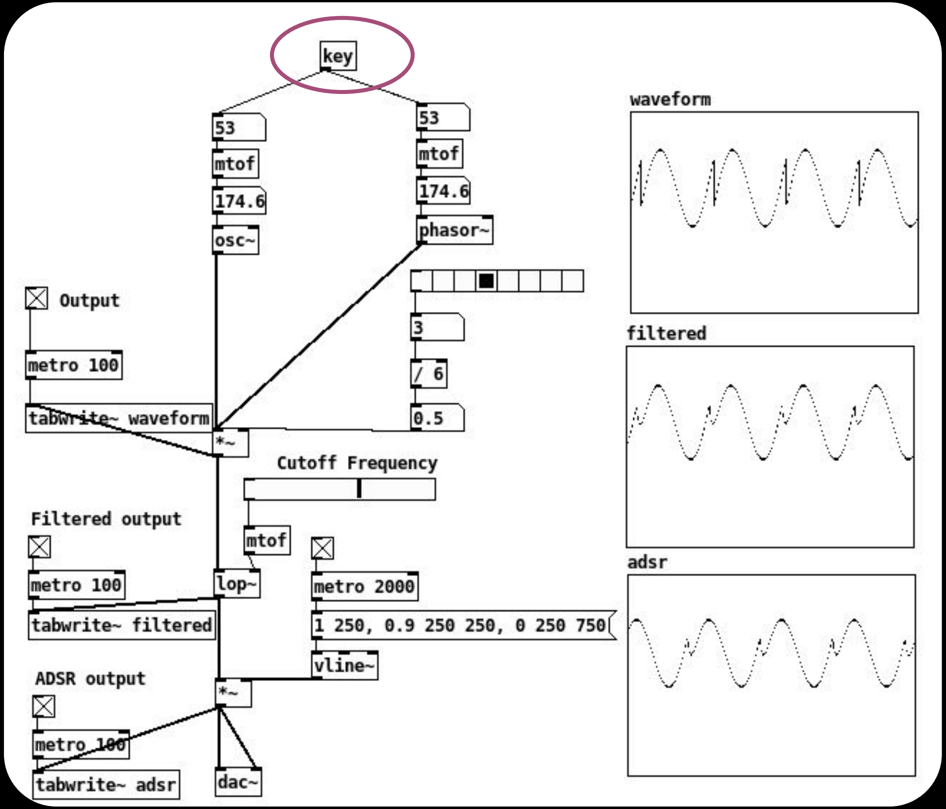


```
;
pd dsp 1
```

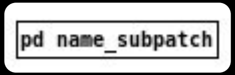
```
;
pd dsp 0
```

- It's a message box, not an object

Keyboard plays note

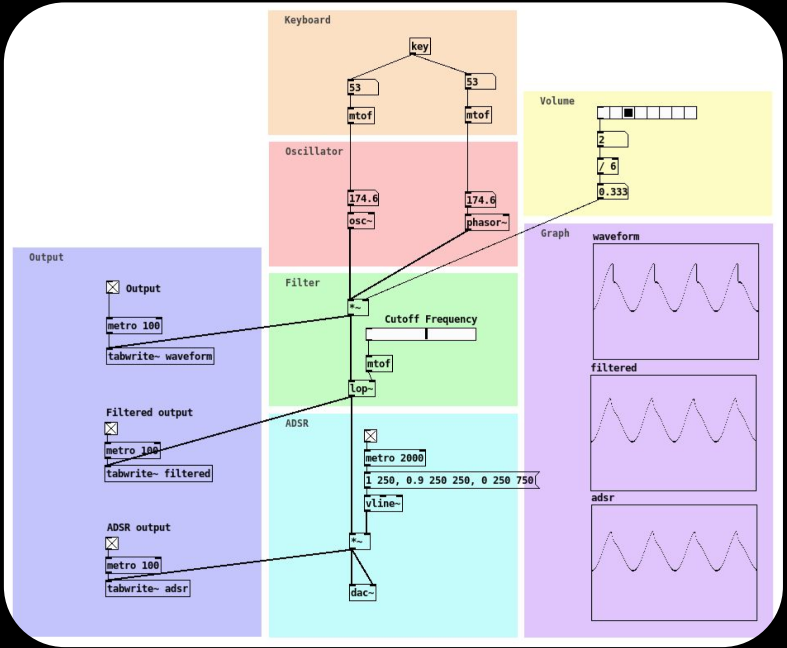


SUBPATCHES



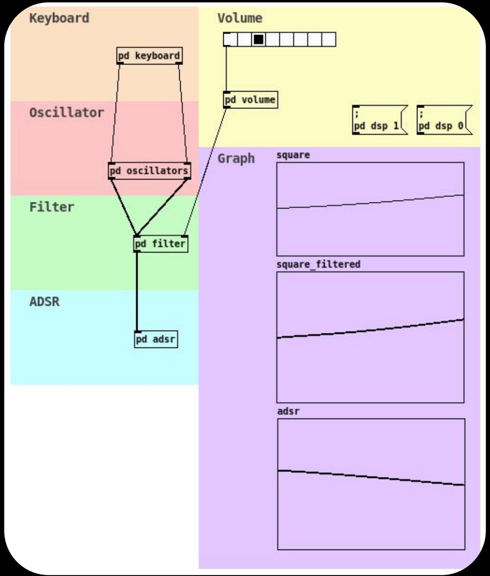
Some lightness and clarity with:

- 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~]

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!