SOUND INTERACTION WORKSHOP

Creating sound

Imanol Gómez
imanolgomez.net
6th May 2017

INTRODUCTION

Repository

https://github.com/ImanolGo/SoundInteractionWorkshop

References

- http://www.pd-tutorial.com/english/
- http://write.flossmanuals.net/pure-data/introduction2/

Background

Pure Data (or Pd) is a real-time graphical programming environment for audio, video, and graphical processing.

Written by Miller S. Puckette (previous co-developed the well known and similarly structured software Max/Msp).

Open Source! As opposed to Max/Msp.

Visual metaphor from analogue synthesizer patches.

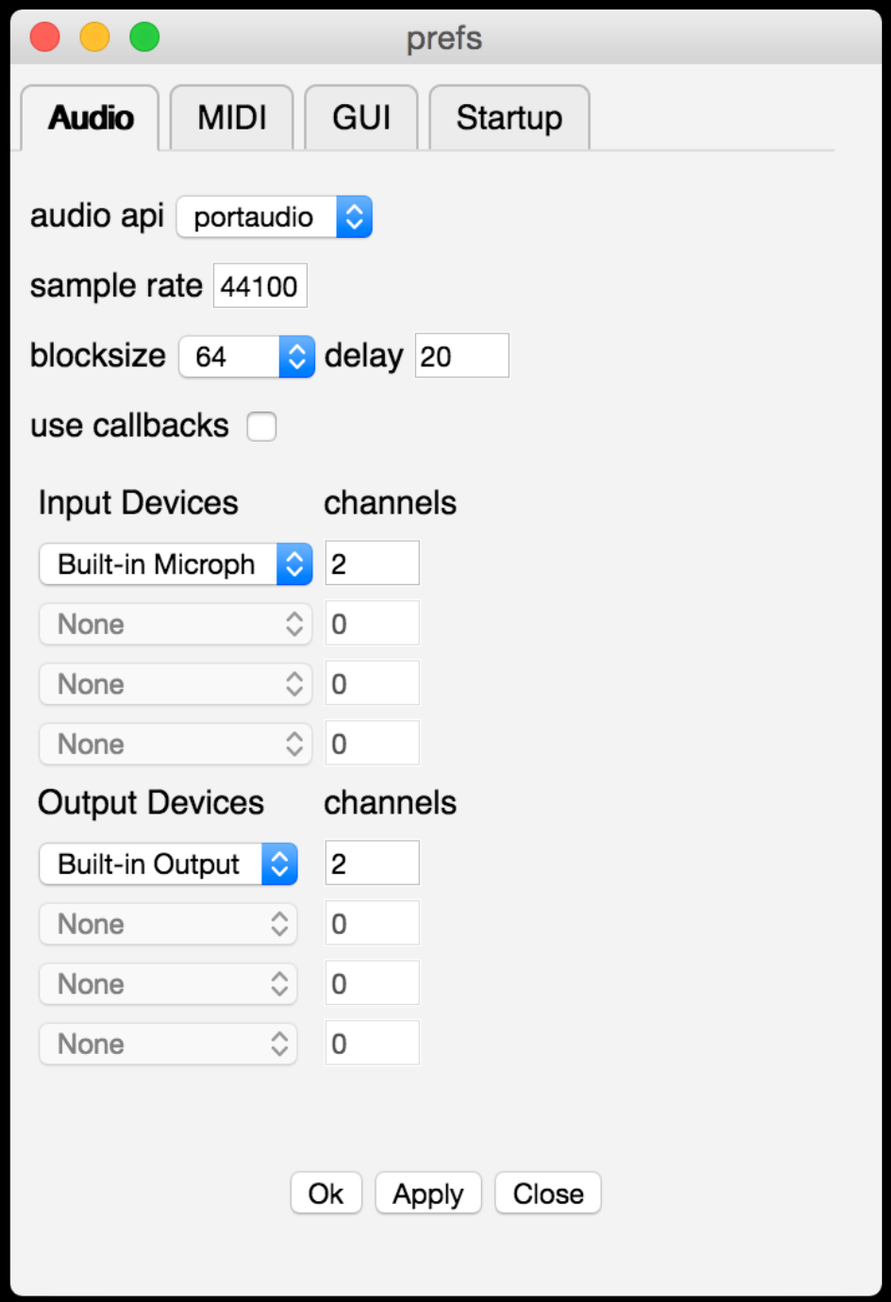
Modular Synthesizer

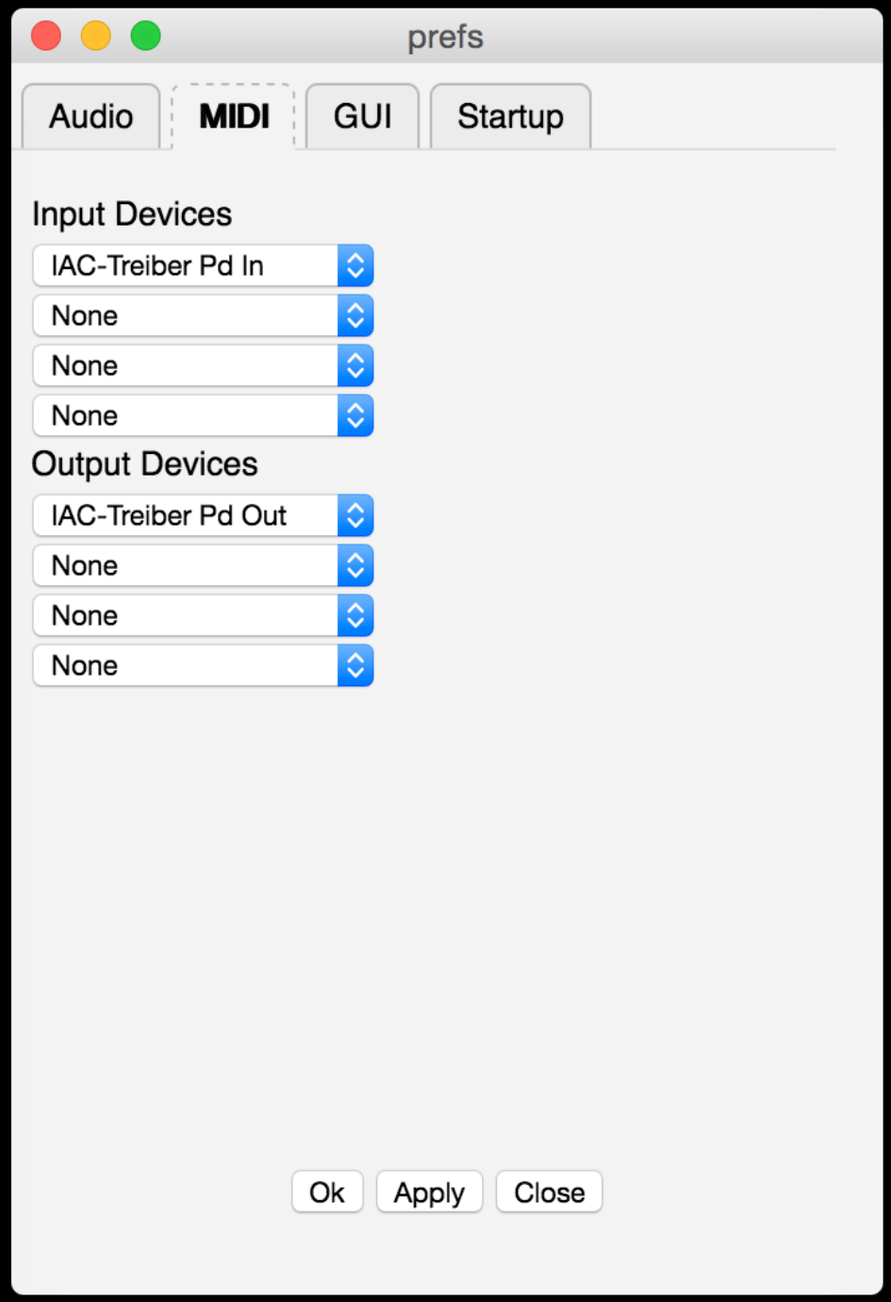
Installation

Purr Data:

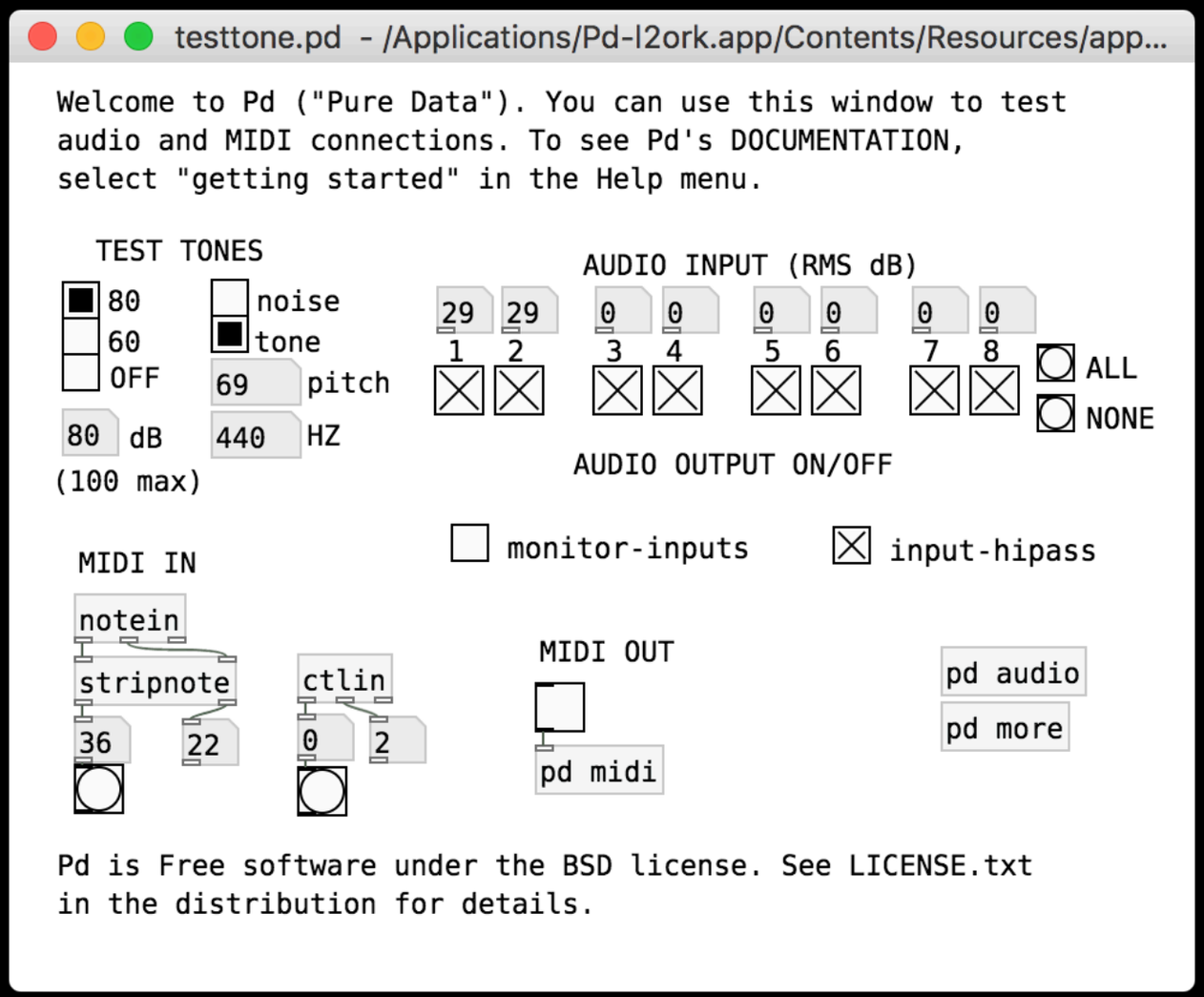
https://github.com/agraef/purr-data/releases

Configuration



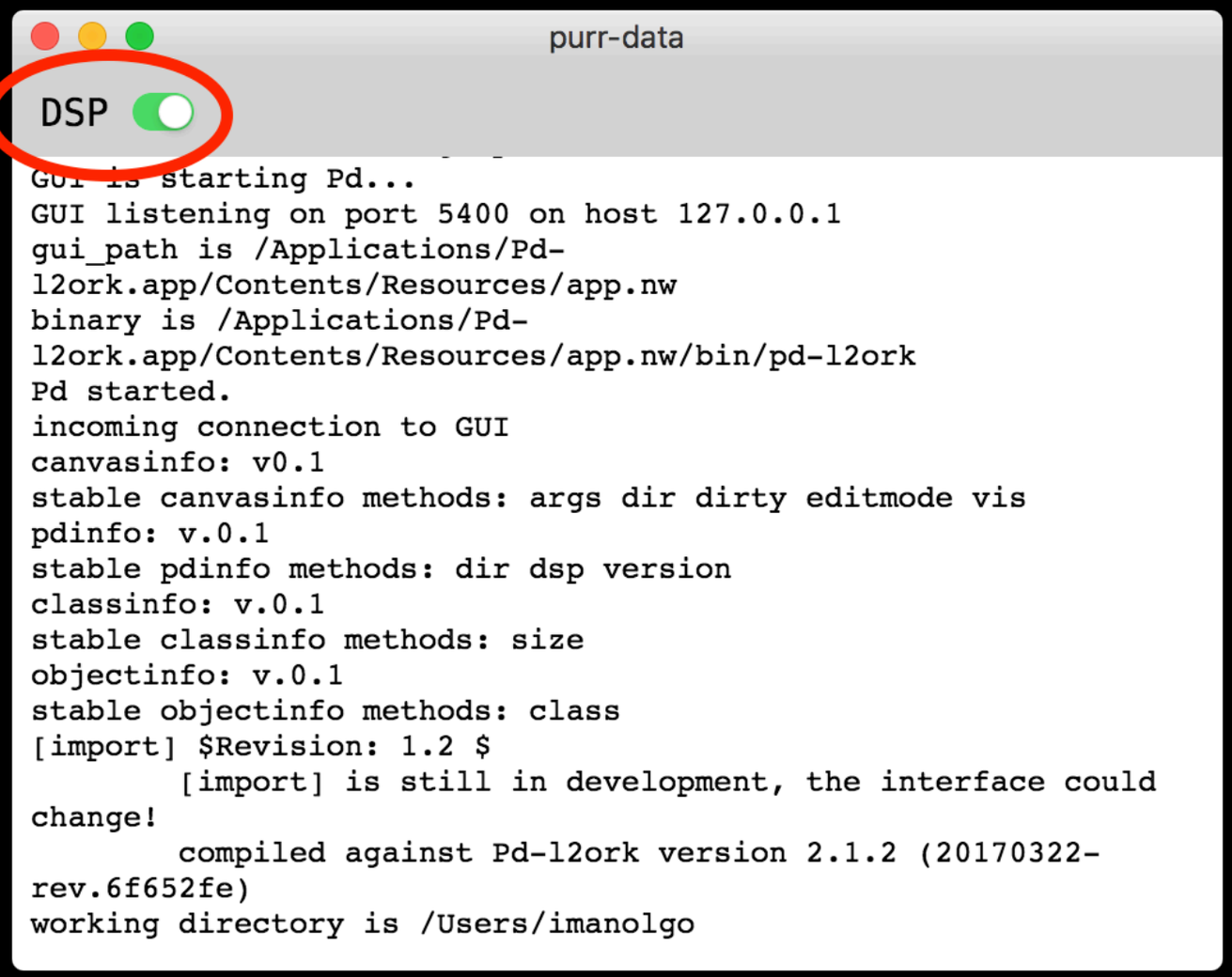


Test Audio / Midi

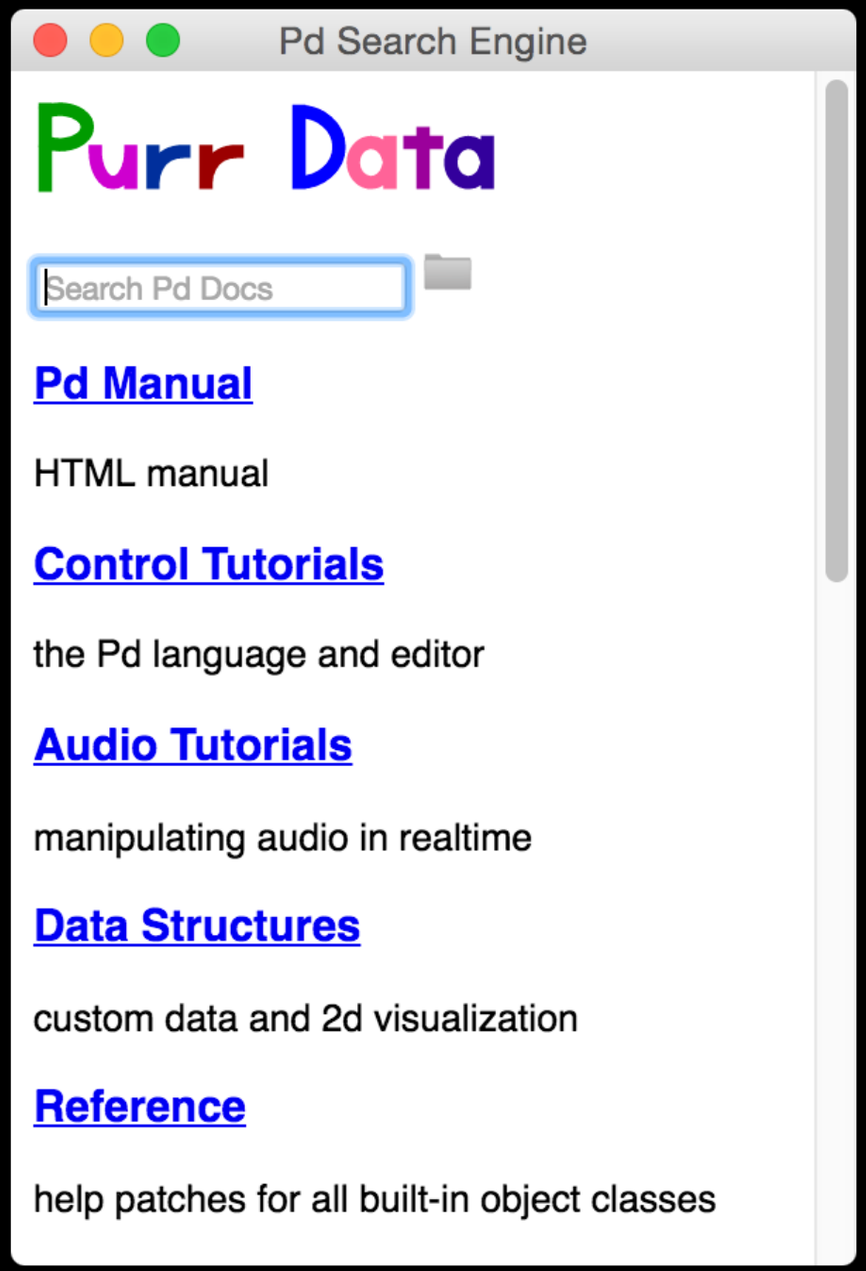


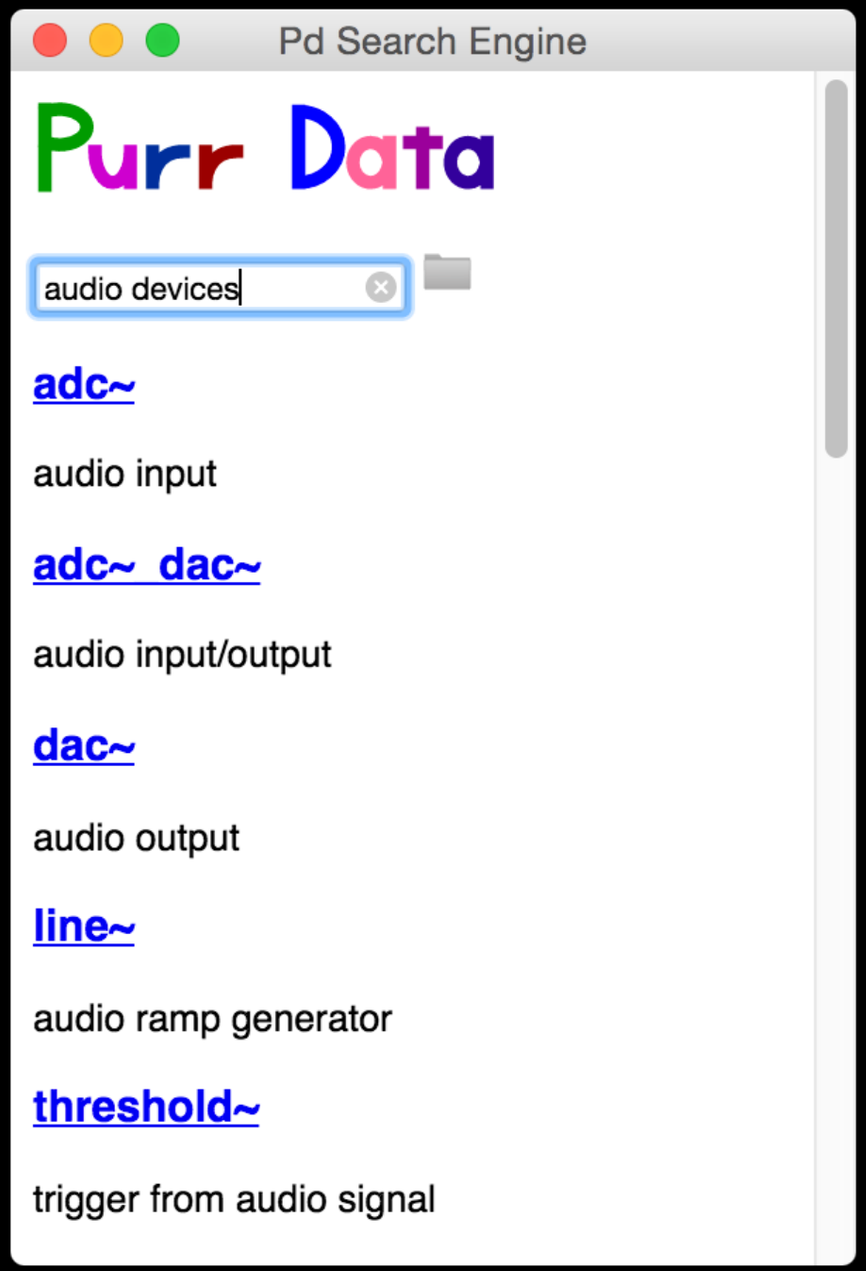
DSP

Don’t forget to turn on DSP!



Getting Help





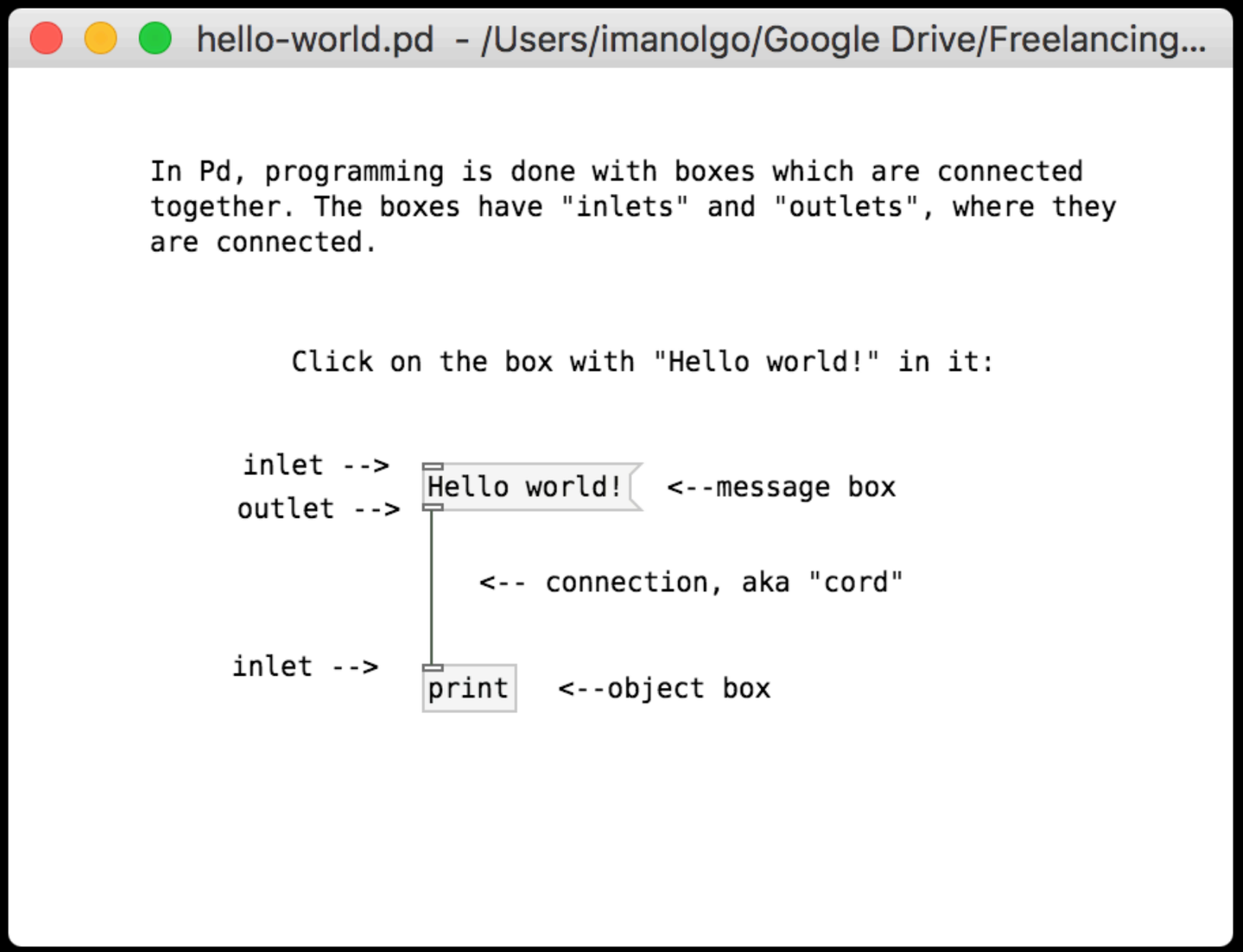
Introducing Pure Data

Programming with Pure Data - interaction that is much closer to the experience of manipulating things in the physical world

The most basic unit of functionality is a box, and the program is formed by connecting these boxes together into diagrams

Diagrams represent the flow of data but is also performing the operations mapped out in the diagram

Hello world!

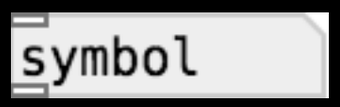

hello-world.pd - /Users/imanolgo/Google Drive/Freelancing...
In Pd, programming is done with boxes which are connected
together. The boxes have "inlets" and "outlets", where they
are connected.
Click on the box with "Hello world!" in it:
inlet -->
Hello world!
<--message box
outlet -->
<-- connection, aka "cord"
inlet -->
print
<--object box

Basic elements

Objects:
rectangular - object name and default value 

Messages:
indentation on the right side
passes data which is stored inside of them when clicked 

Numbers:
atoms 

Symbols:
atoms 

Comments:
string 

Basic Functionality

Play or Edit Mode: Patcher/canvas locked/unlocked
CMD E (Mac), CTRL E (Windows)

Output can be printed in the Terminal

Pd blocks have inlets on the topside and outlets on the bottom

To create a connection between two blocks, drag a line from the outlet of one block to the inlet of another block

Use shortcuts! (different for Windows and Mac)

Exercise

1. Modify the Hello World patch so print something else
2. Add a number, attach it to print and change it dynamically.

Getting Started

Examples and descriptions can be found in the Pd Help Browser.

Help/Pd Help Browser

You can get help with anything by right-clicking a box and select “help”

Controlling

Common Objects

`[bang]` does stuff!

`[tgl]` toggle 1 or 0

`[trigger]` sequence messages in right-to-left order

`[metro]` set bangs periodically

`[select]` bangs when received specific number `[pack]` packs lists

`[send]` `[receive]`

`[maxlib/scale]` scaling input/output ranges

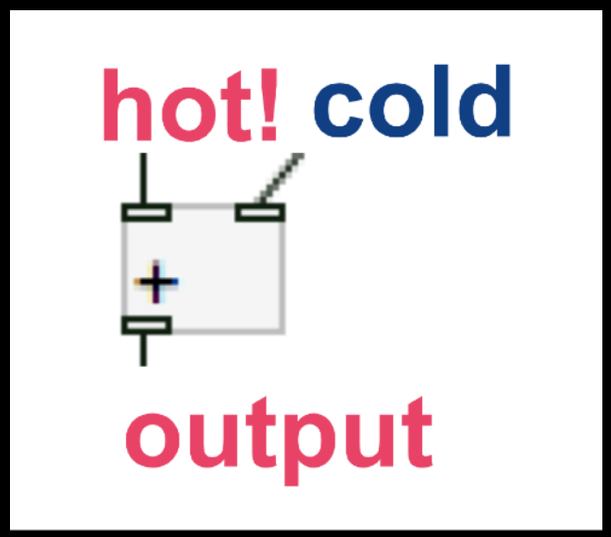
Hot and Cold Inlets

The right inlet will typically set an argument to a function. Only input to the active inlet will create an output from the outlet(s).

the left-most inlet is “hot” : it will output something whenever it receives data.

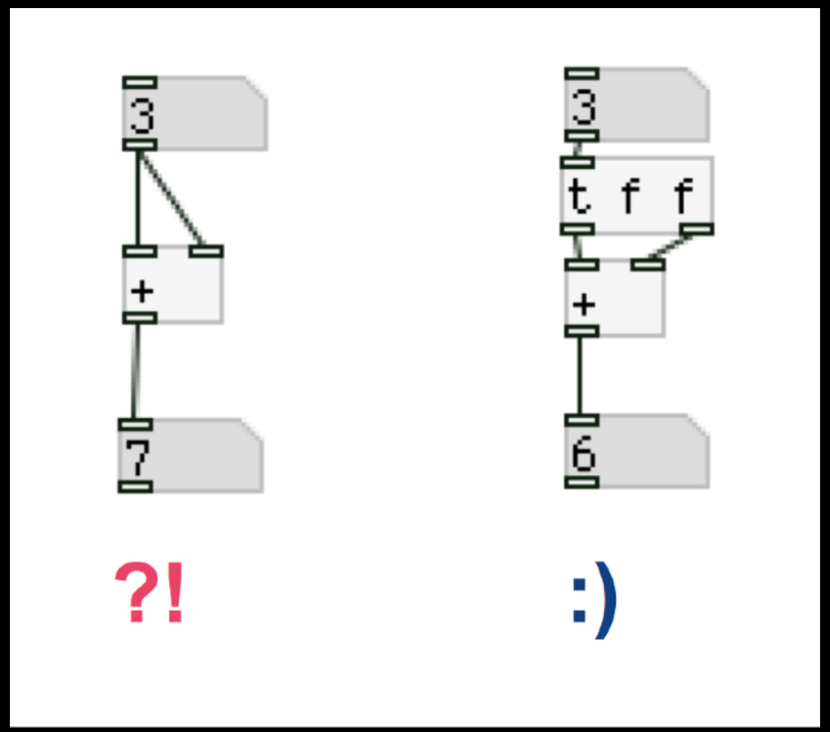
all other inlets are generally “cold”: they just store data

when the object receives input on the “hot” inlet, the object will read the stored data from all inlets and do stuff



Hot and Cold Inlets

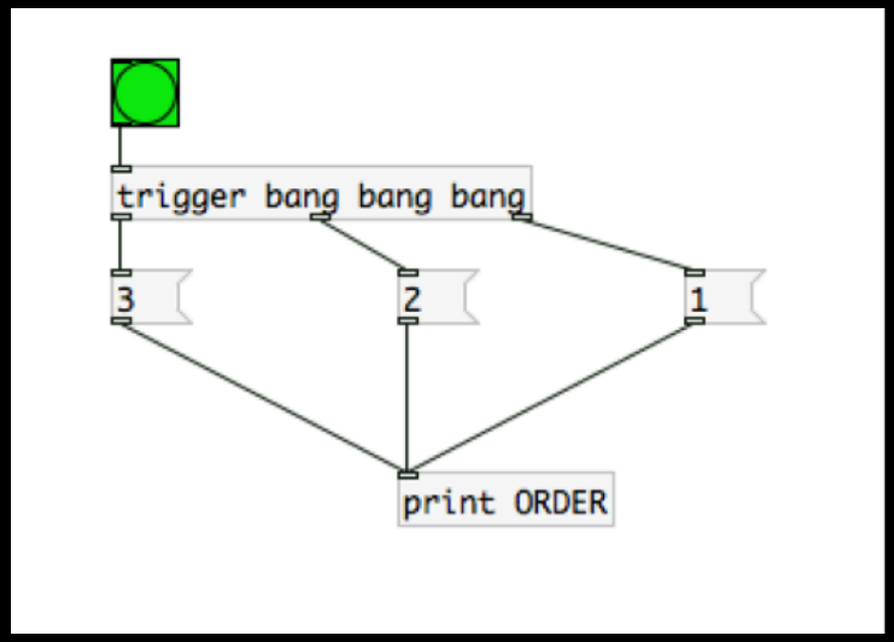
Problems can arise when a single outlet is connected (either directly or through arbitrarily long chains of message passing) to different inlets of a single object:



Right to Left Order

Patches are read from right to left, top to bottom. Objects output from right to left.

forcing a specific execution order can be done with the “trigger” object



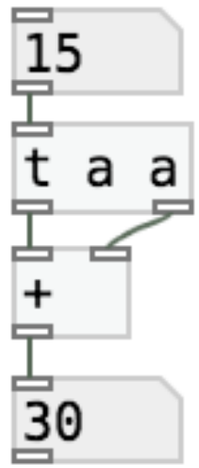
this patch will output :

```
ORDER 1
ORDER 2
ORDER 3
```

Always use [Trigger]

trigger
sequence messages in right-to-left order/ convert data types

trigger or t



arguments

1) symbol atom - for each creation argument, a new outlet is created. If no arguments are supplied, trigger defaults to two outlets that output bang messages.

The [trigger] object can be abbreviated as "t" and the creation arguments can be abbreviated as follows:

"float" = f
"bang" = b
"symbol" = s
"list" = l
"pointer" = p
"anything" = a

trigger float bang symbol list pointer anything

is the same as

t f b s l p a

Atomic messages

gatom
atom (number box and symbol box)

0 and symbol

0

A number box allows you to display a number or enter a number using the mouse and keyboard. When a number arrives at the number box's inlet, it is displayed and sent to the outlet. You can click on a number box and drag upward or downward to change the value continuously.
The number box is called "Number" in the "Put" menu.

set 45

0

symbol

symbol

A symbol box allows you to display a single symbol-atom or enter one using the mouse and keyboard. Unlike a number box you cannot change the value by clicking and dragging.
The symbol box is called "Symbol" in the "Put" menu.

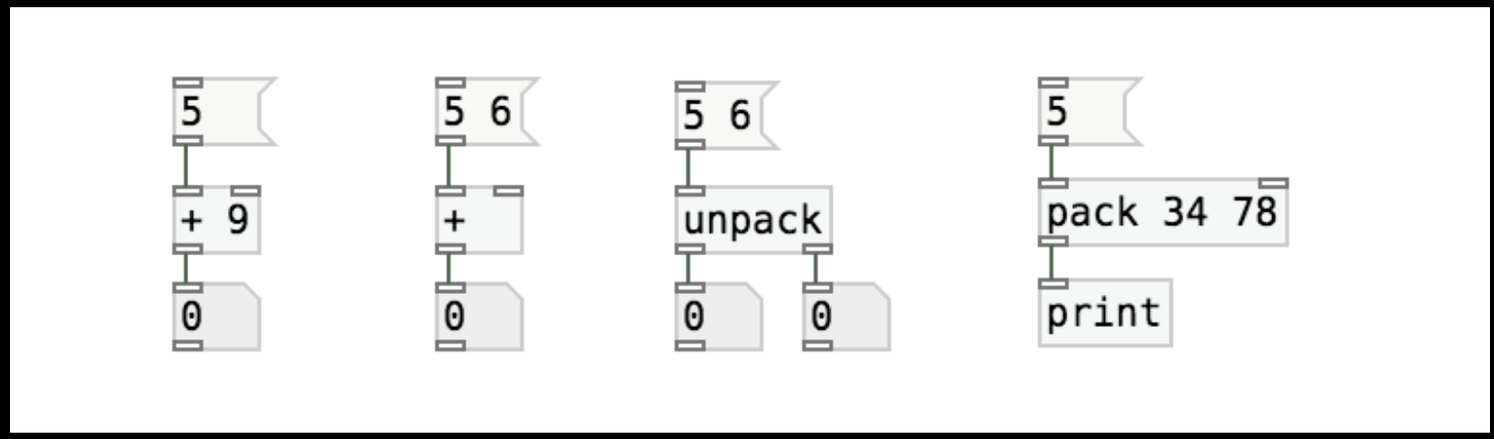
set foo

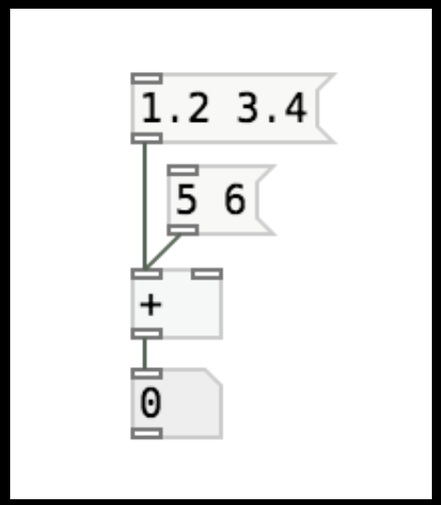
symbol

To enter data simply click a number box or symbol box and begin typing. Then click "Enter" to finish and output it.

Messages

Most Pd messages are just numbers or short lists of numbers

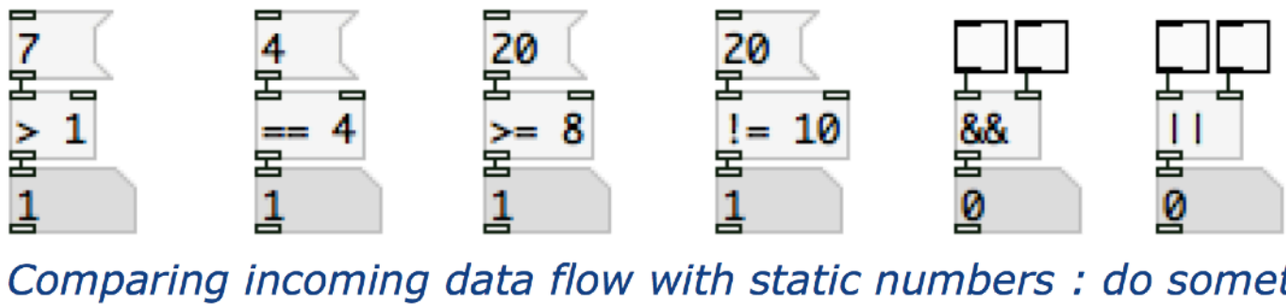




In Pd all numbers are floating point.

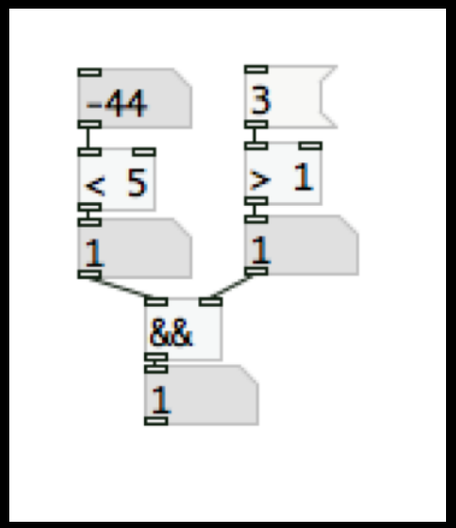
Conditional

Logic operators:



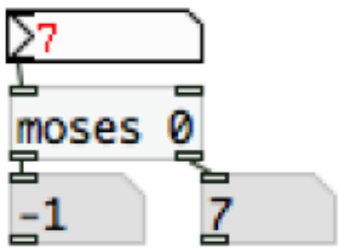
Comparing incoming data flow with static numbers : do something when true (i.e. 1)

Multiple Comparisons:



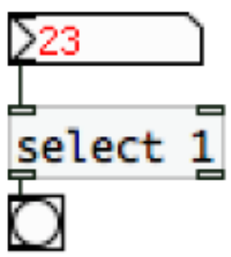
Conditional

`[moses]` - splits a range of numbers :



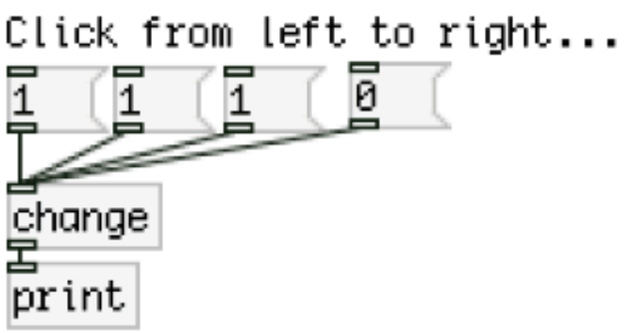
values below 0 will be outputted to the left
values equal to or larger than 0 will be outputted to the right

`[select]` - bangs when number is received

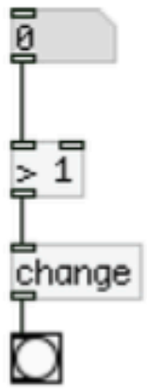


Conditional

`[change]` - eliminates redundancy in a number stream



output is only allowed when value is changed



Example: bang once when value exceeds 1

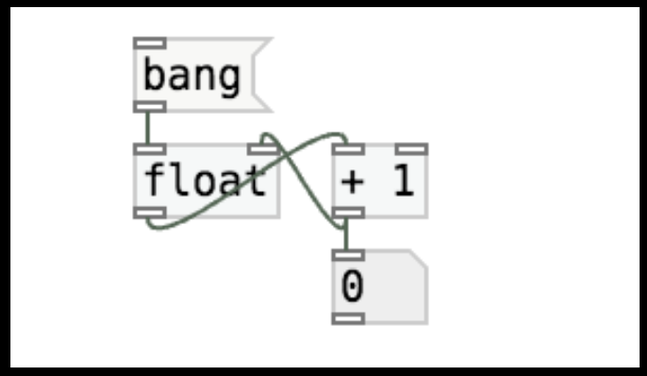
Arithmetic Objects

```
[+  ]
[-  ]
[*  ]
[/  ]
[pow]
[max]
[min]
```

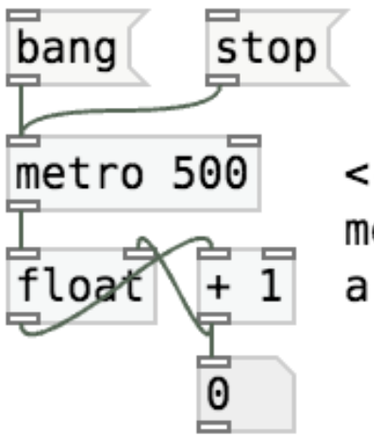
`[expr]` allows you to write mathematical formulas

Counter

Here's a simple counter. Click repeatedly on the "bang" message to see it work:



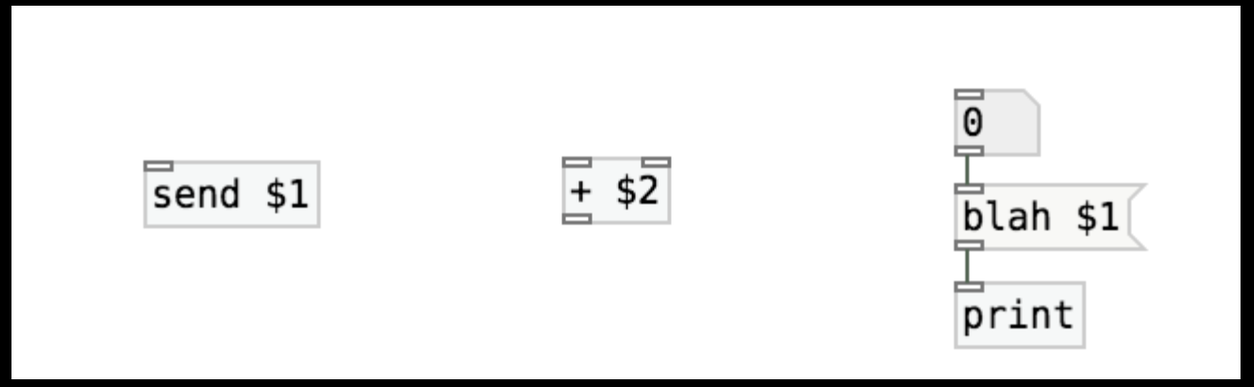
Here's a timed counter. Hit the "bang" to start it...




```
<-- new object: [metro], which is a metronome. The "500"
means it outputs a bang every 500 milliseconds-- i.e., twice
a second.
```

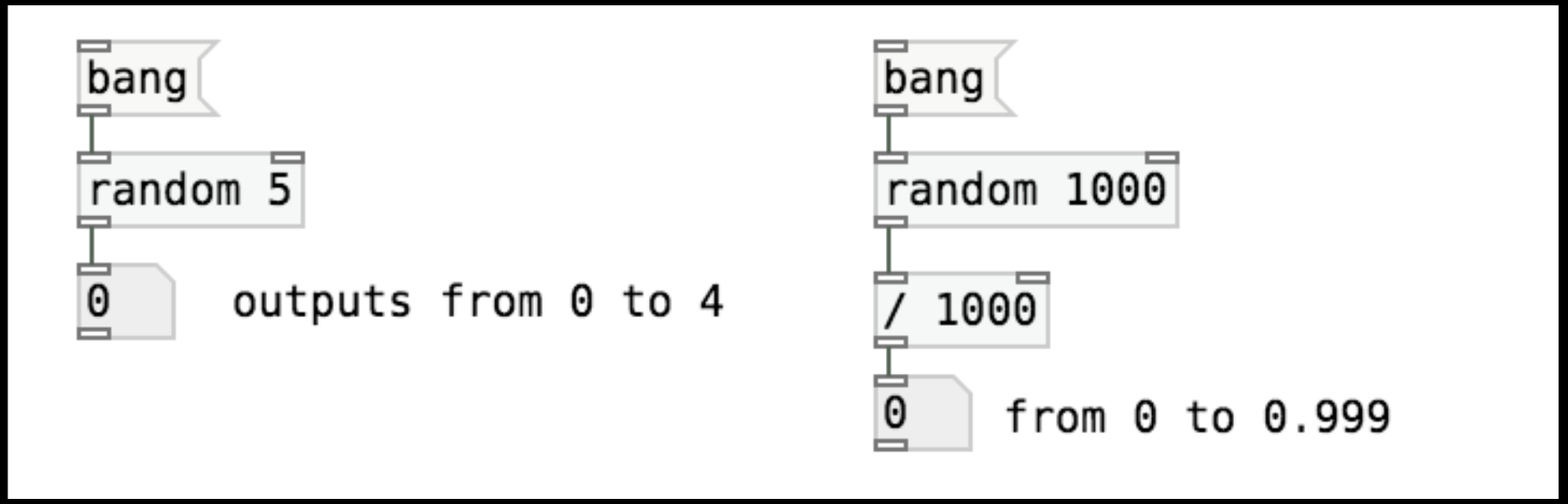
Dollar Sign: $

In object boxes, dollar signs refer to the abstraction's creation arguments. In message boxes, they change dynamically:



Random

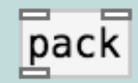
Use the `[random]` object to make pseudo-random integers. To get continuously variable random numbers, make a random number in a large range and divide:



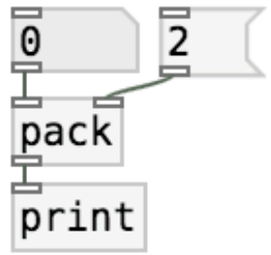
Pack

pack

combine several atoms into one message



The [pack] object takes a series of inputs and then outputs a concatenated list. By default, [pack] has two inlets, each of which will accept a float.



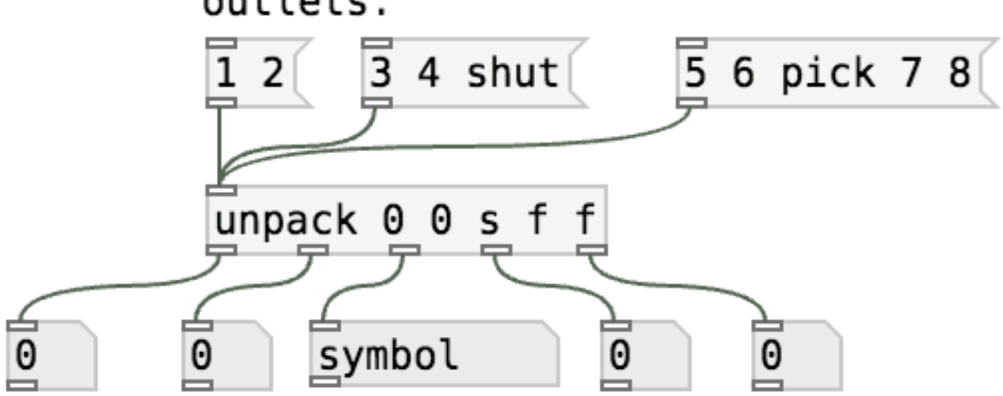
Unpack

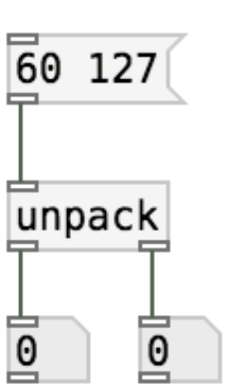
unpack

split a message into atoms

unpack

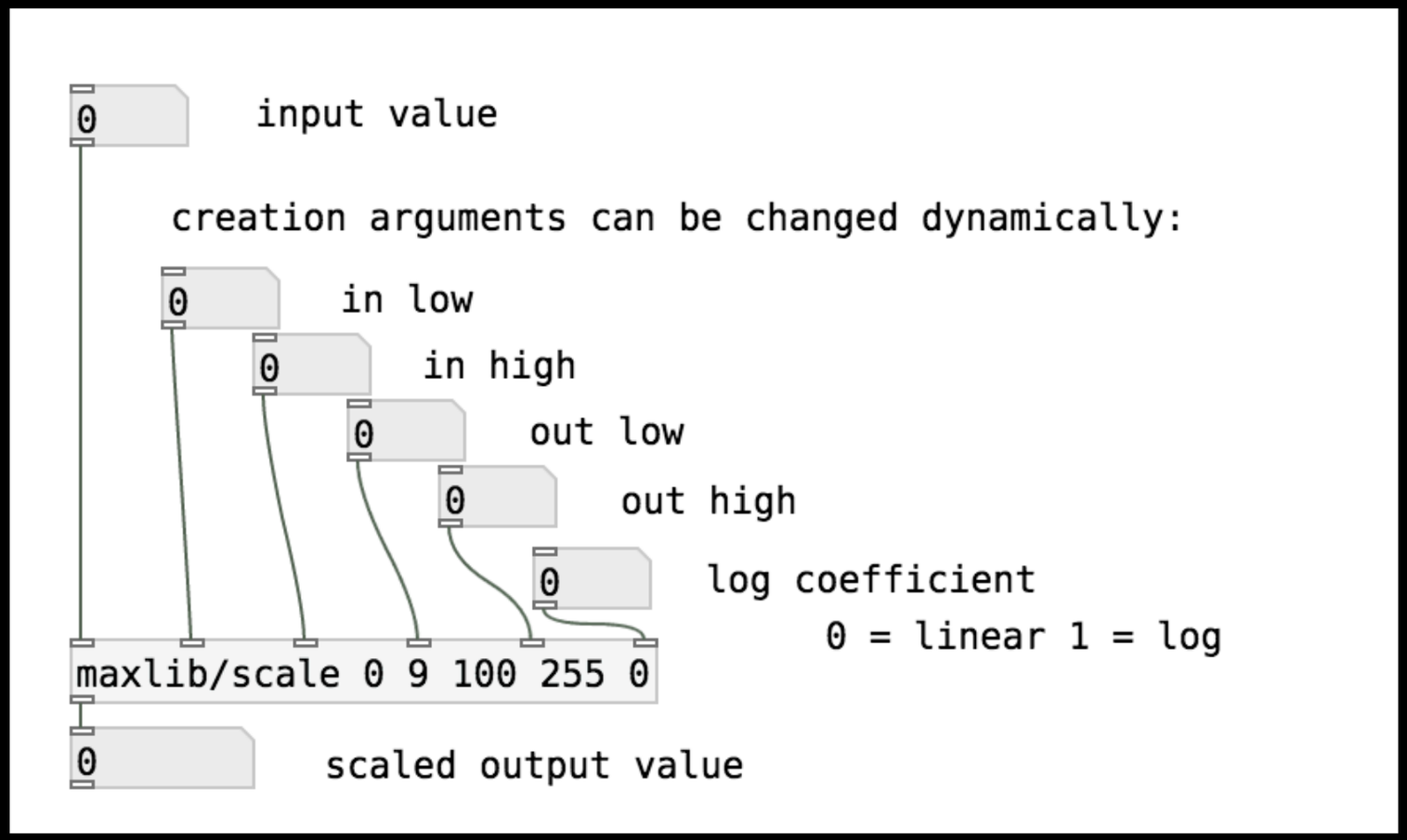
[unpack] takes a list and distributes the elements to its outlets.





maxlib/scale

`[scale]` scale input from a certain input range to lie between output boundaries

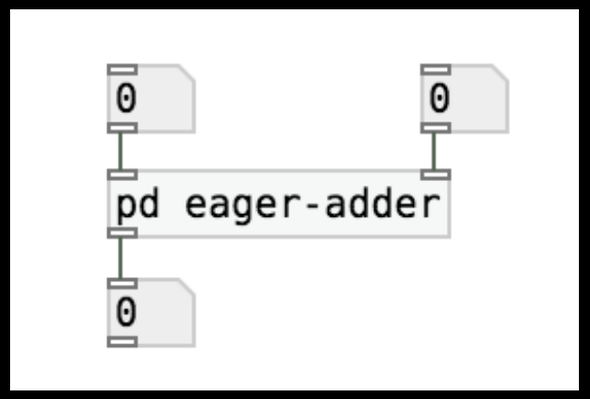


Subpatch

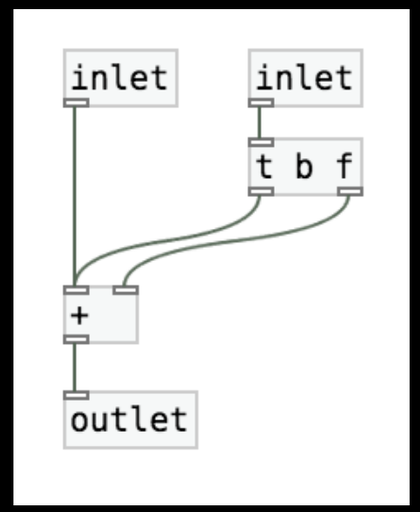
You can nest entire windows inside Pd boxes (and so on, as deep as you wish.) There are two different ways to do it. First, if you just want to add a room to the house, so to speak, type

```
pd sample-subpatch <-- you can give the window a name as an argument
```

If you click on the box (in run mode) the subwindow appears. Click on the one below to see how you give a subpatch inlets and outlets.

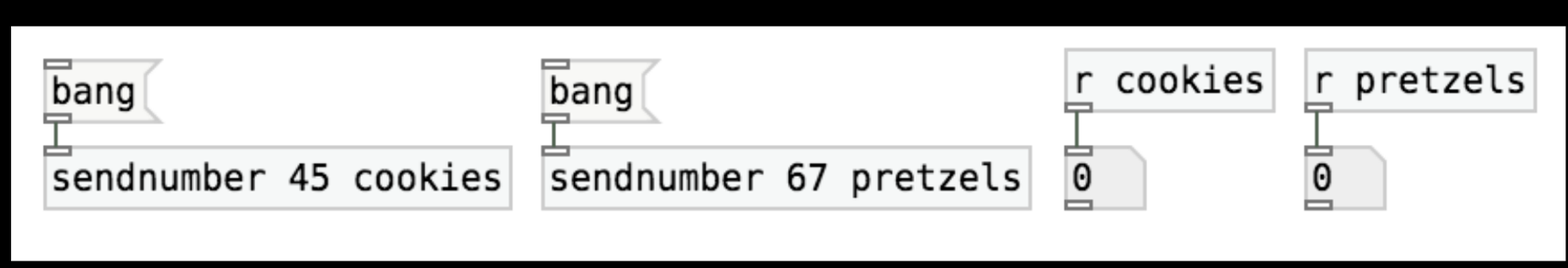


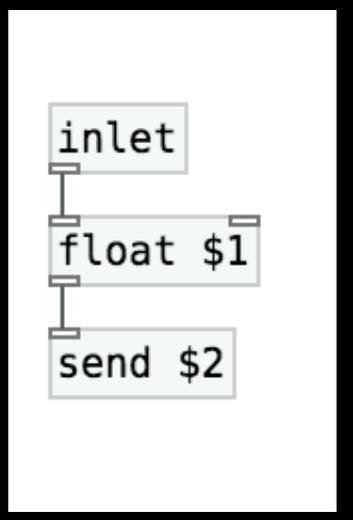
—>



Subpatch

There is also a facility for making many copies of a patch which track any changes you make in the original. The subpatches are called abstractions.





There is a separate file in this directory named "sendnumber.pd" which is loaded every time you type "sendnumber" in an object box. Click on a "sendnumber" box above to see it.

Audio

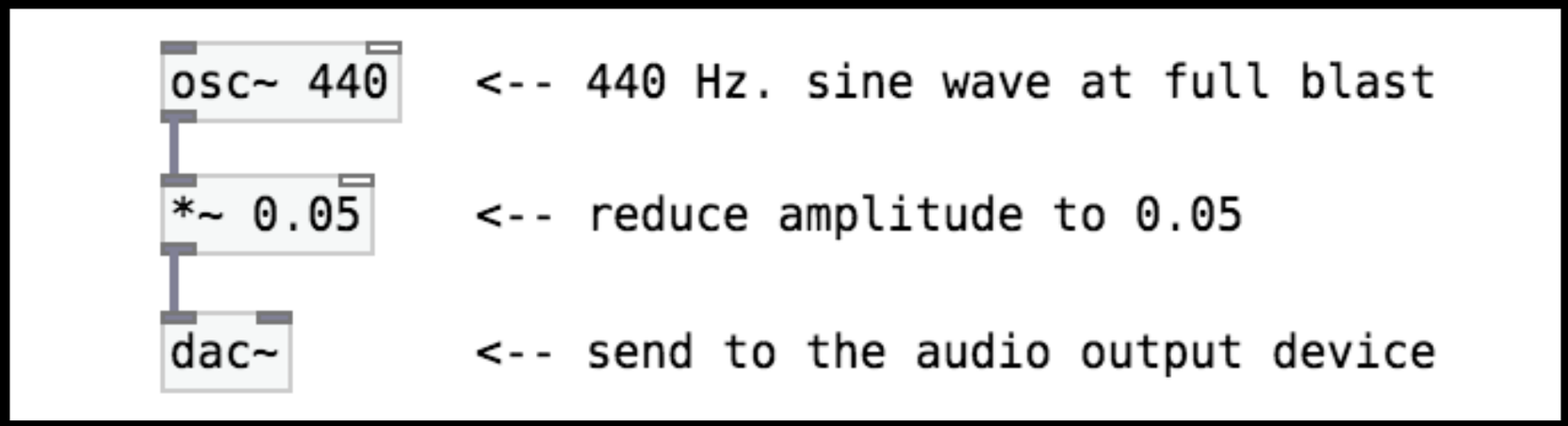
Common Objects

[dac~]
[adc~]
[osc~]
[snapshot~]
[vline~]
[sig~]
[clip~]

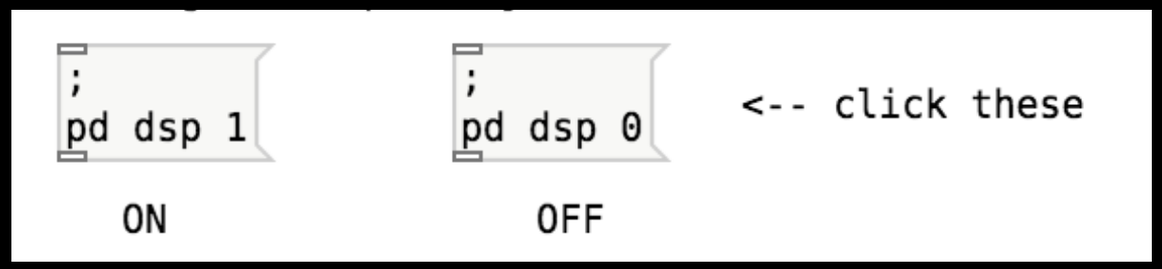
[+~] [-~] [*~] [/~]

Sinewave

Audio computation in Pd is done using "tilde objects" such as the three below. They use continuous audio streams to intercommunicate, as well as communicating with other ("control") Pd objects using messages.



Make sure DSP is on!

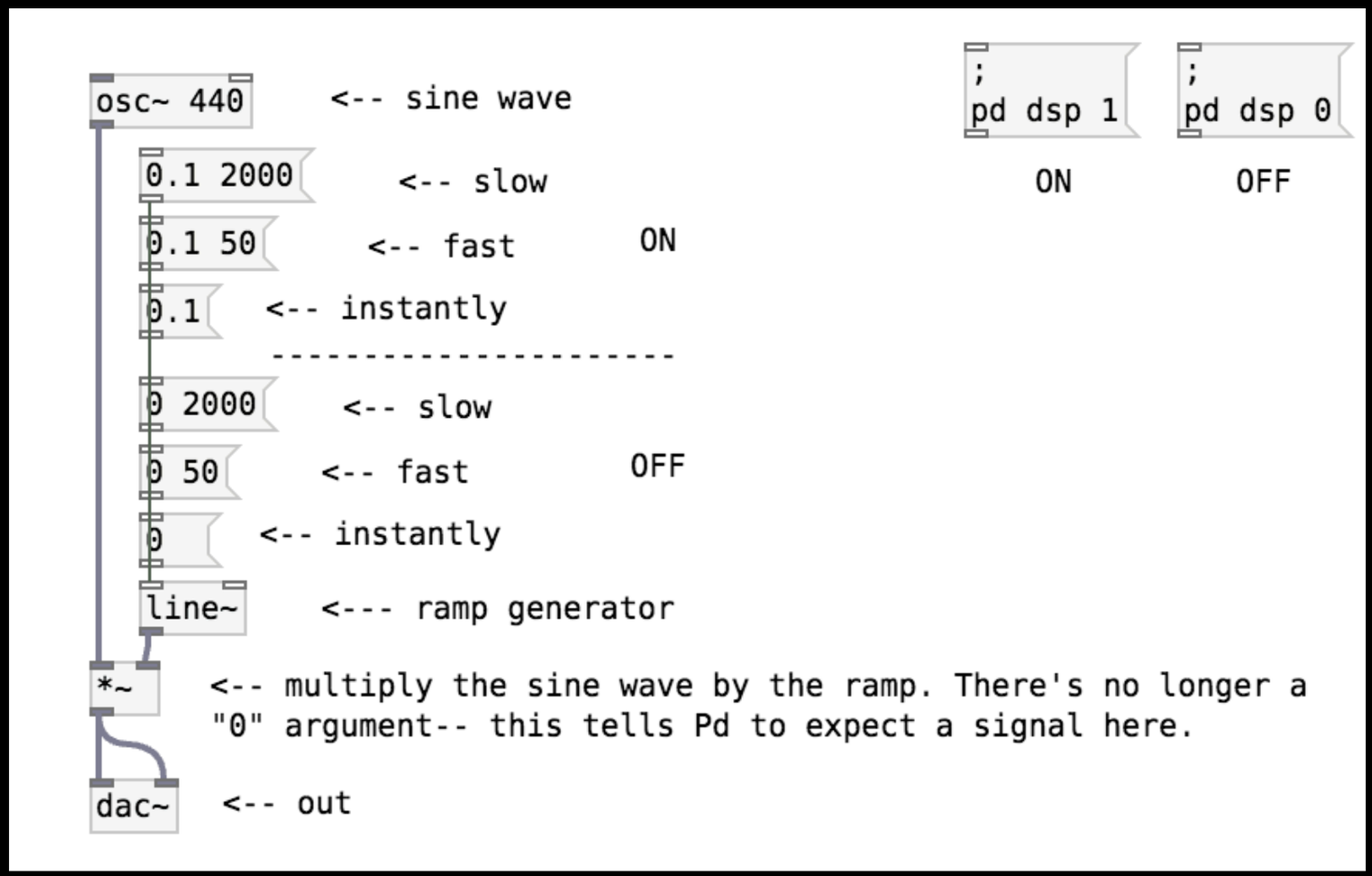


Exercise

1. Change dynamically the frequency value
2. Change dynamically the amplitude
3. Add a `[slider]` to control the parameters
4. Use `[phasor~] and hear the difference`
5. Don’t loose hearing!

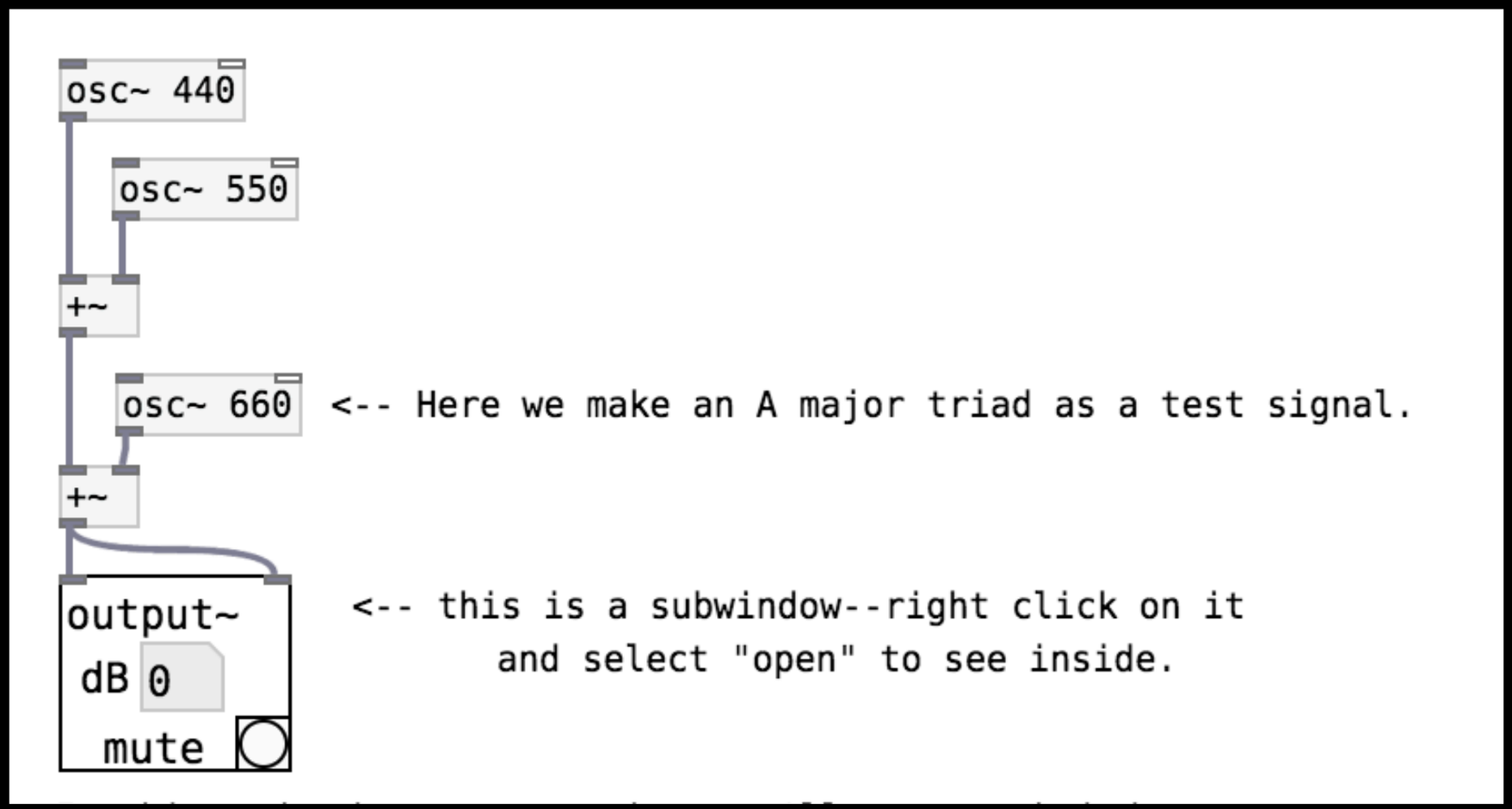
Line

In this patch, the multiplier is configured to multiply two signals.
The amplitude is now a signal computed by the `[line~]` object.



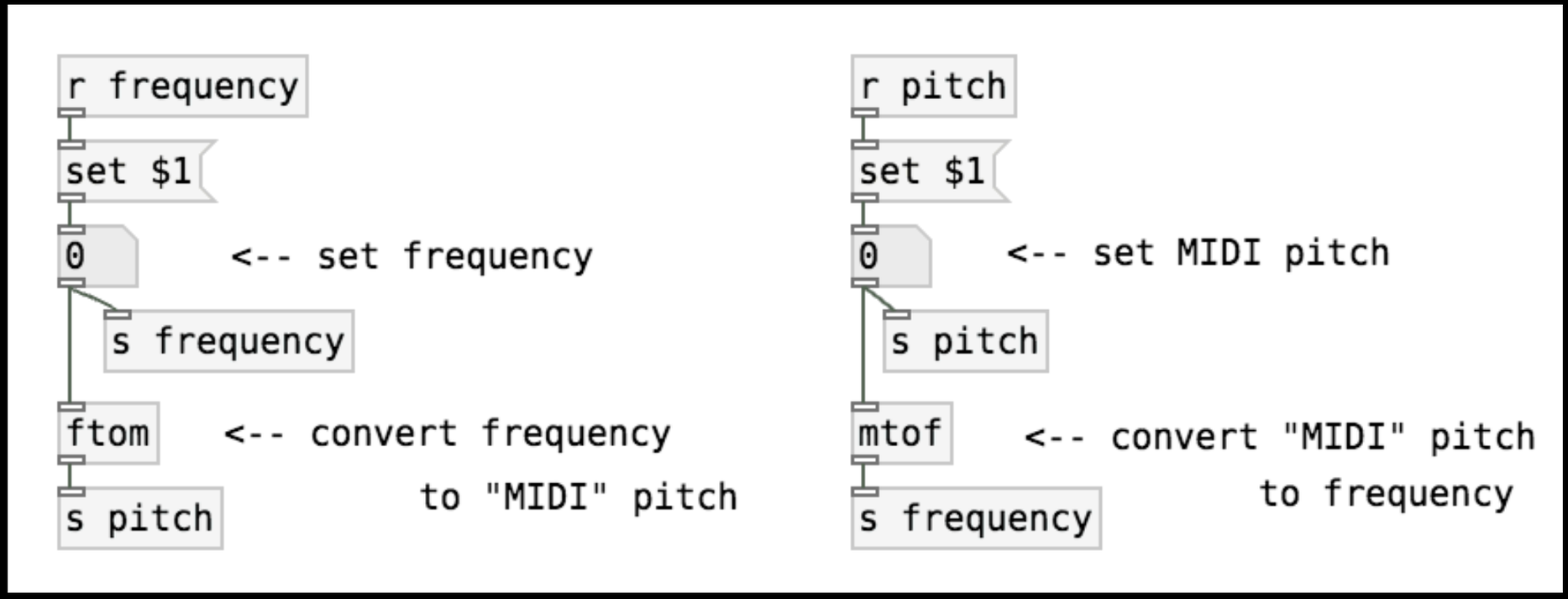
Output Amplitude

In this and subsequent patches, we'll use a subwindow, "output", to control overall amplitude.



Frequency and Pitch

Frequency and pitch are converted using the `[ftom]` and `[mtof]` objects. Frequency refers to the number of cycles per second. Pitch is "60" for Middle C, 61 for C sharp, 72 for the next C up, and so on.



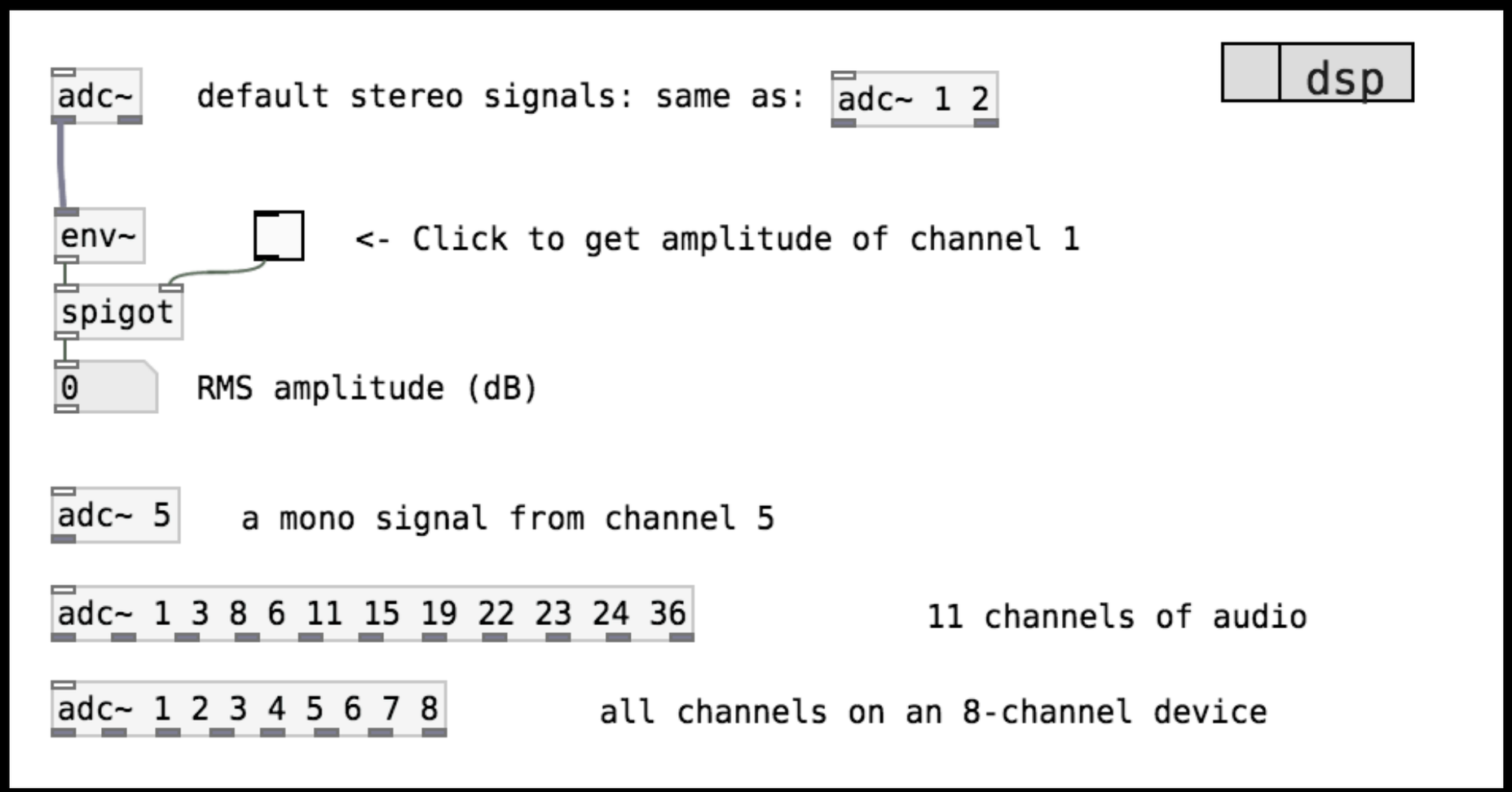
MIDI and Frequency

One can imagine the MIDI scale as a piano keyboard with 128 keys on it, and each key has been marked with a frequency in Hertz which represents that musical note.

MIDI Note	Frequency	MIDI Note	Frequency	MIDI Note	Frequency
C	36	65.4063913251	48	130.8127826503	60	261.6255653006
Db	37	69.2956577442	49	138.5913154884	61	277.1826309769
D	38	73.4161919794	50	146.8323839587	62	293.6647679174
Eb	39	77.7817459305	51	155.5634918610	63	311.1269837221
E	40	82.4068892282	52	164.8137784564	64	329.6275569129
F	41	87.3070578583	53	174.6141157165	65	349.2282314330
Gb	42	92.4986056779	54	184.9972113558	66	369.9944227116
G	43	97.9988589954	55	195.9977179909	67	391.9954359817
Ab	44	103.8261743950	56	207.6523487900	68	415.3046975799
A	45	110.0000000000	57	220.0000000000	69	440.0000000000
Bb	46	116.5409403795	58	233.0818807590	70	466.1637615181
B	47	123.4708253140	59	246.9416506281	71	493.8833012561

Audio Input

`[adc~]` provides real-time audio input for Pd.



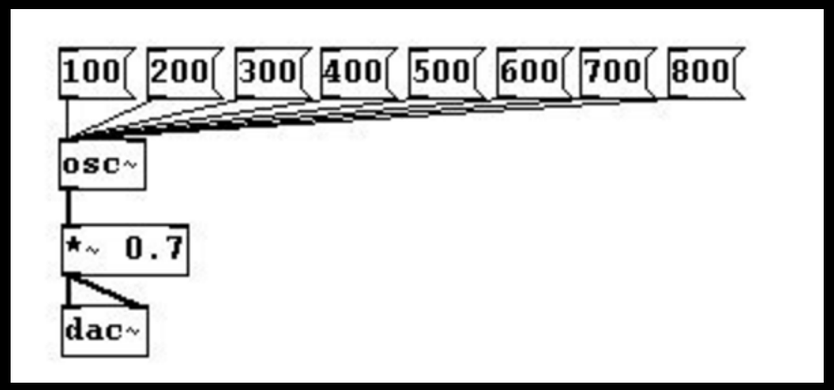
Synthesis

Mini Moog

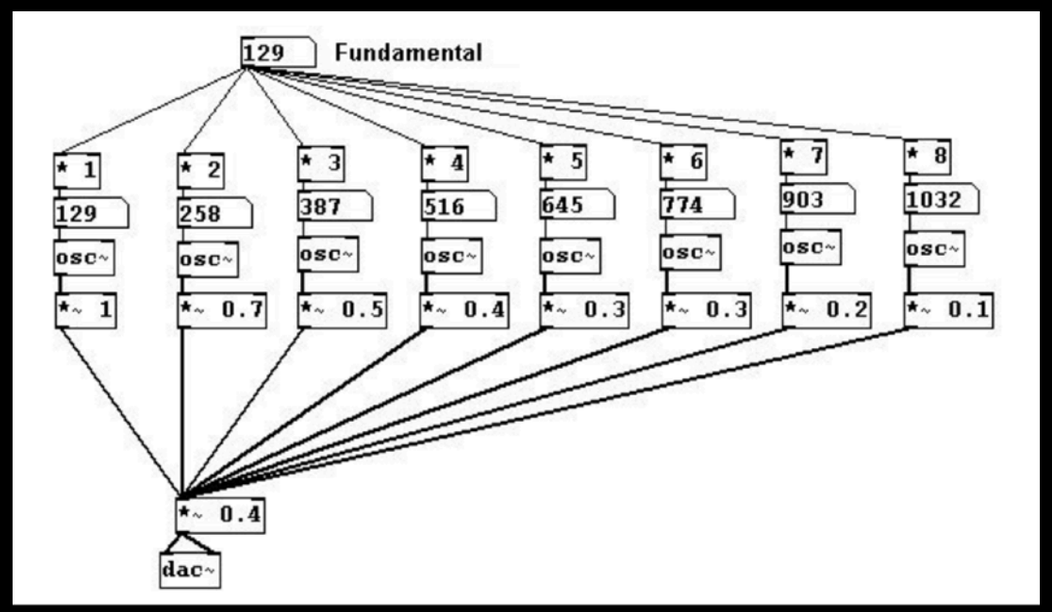

CONTROLLERS
OSCILLATOR BANK
MIXER
MODIFIERS
OUTPUT
moog
minimoog

Additive Synthesis

The additive series of frequencies, which results in a string of intervals of decreasing size, is called the harmonic series:

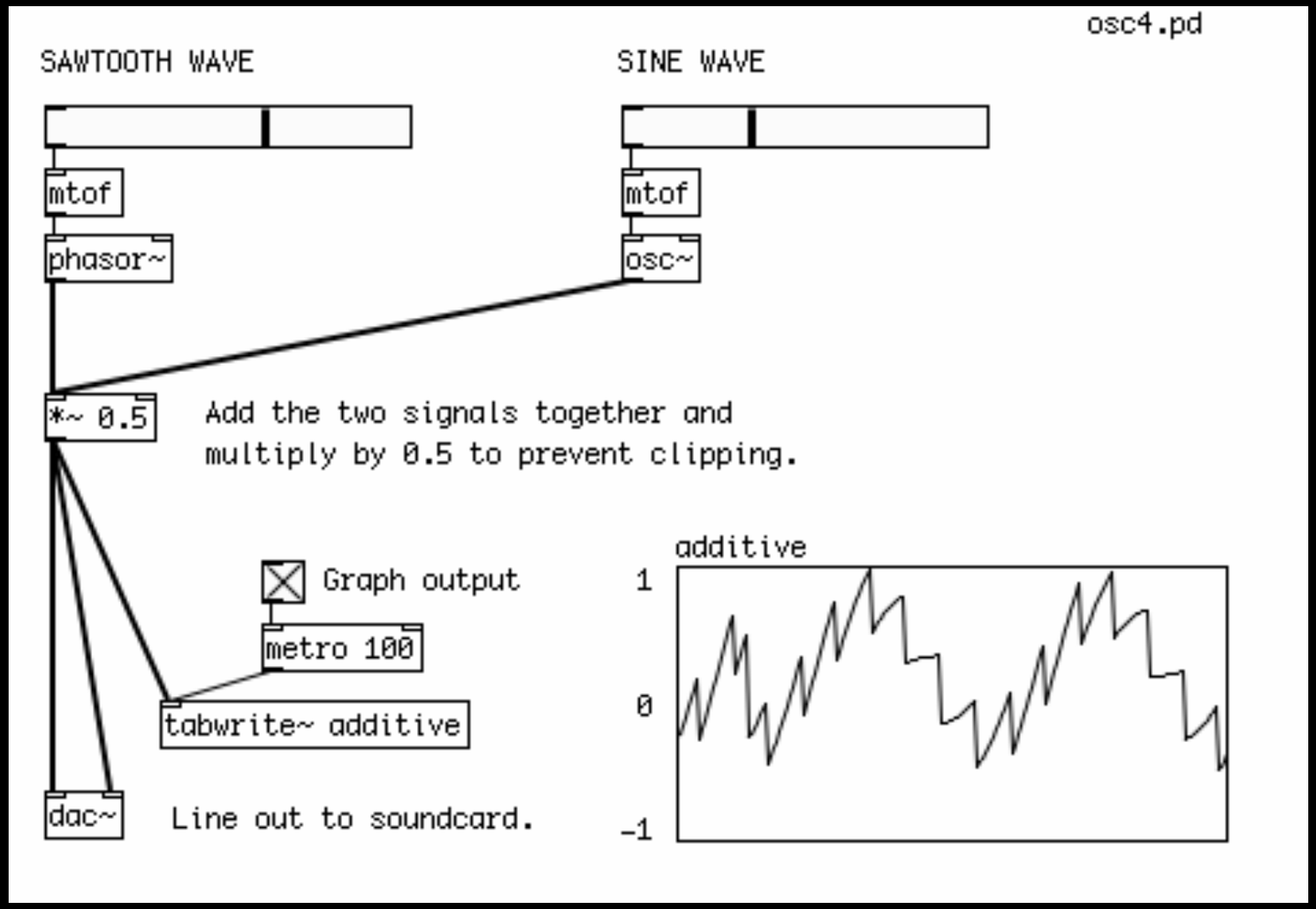


Our ears blend the overtones together becomes clear when you change the fundamental frequency:



Additive Synthesis

Combining two or more signals into a single waveform is simple



Exercise

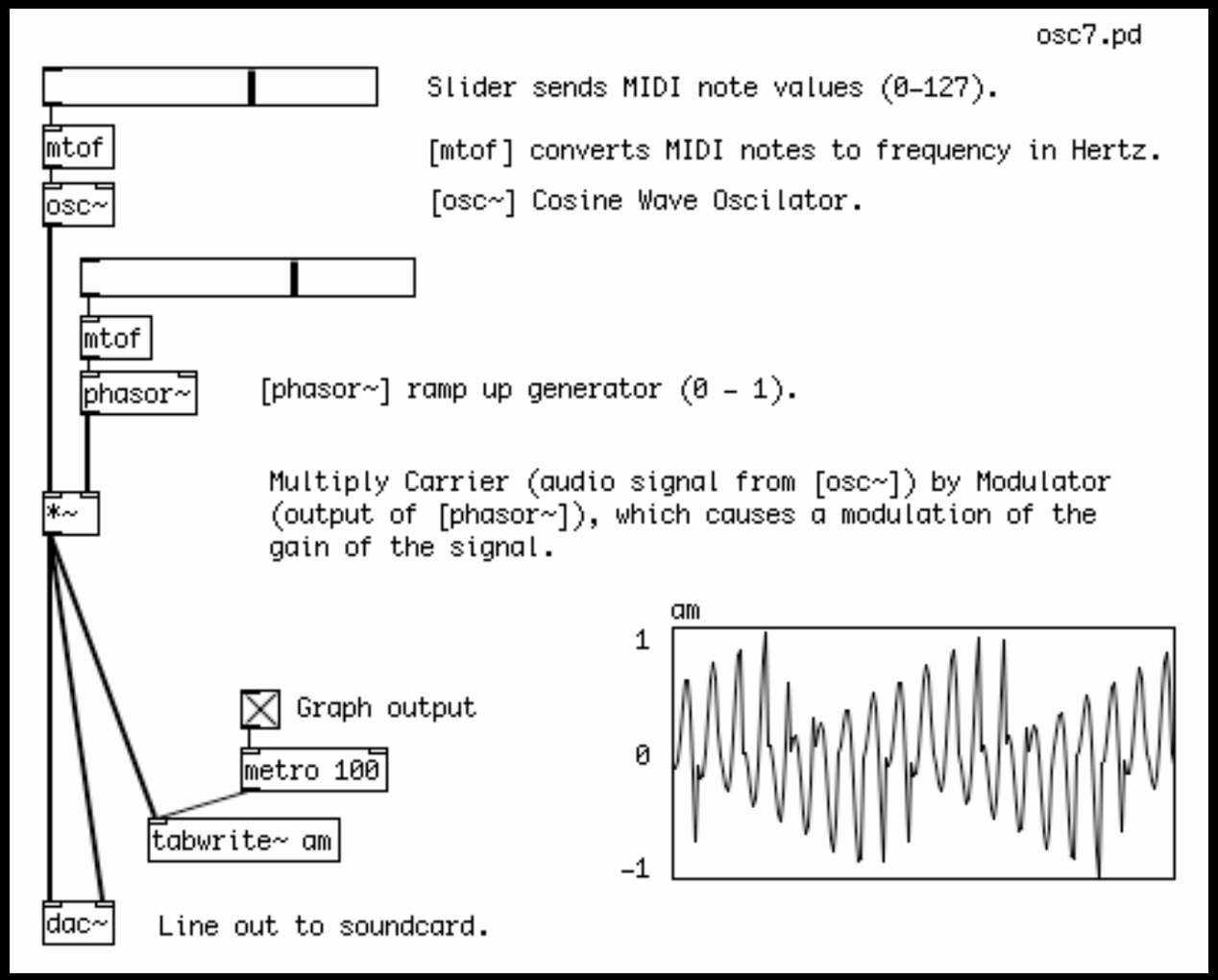
1. Create your own additive synth patch as in the first example
2. Open and play around osc4.pd, osc5.pd and osc6.pd
3. Open 3-2-2-1-random-color.pd and play around

Exercise

1. Create your own additive synth patch as in the first example
2. Open and play around osc4.pd, osc5.pd and osc6.pd
3. Open 3-2-2-1-random-color.pd and play around

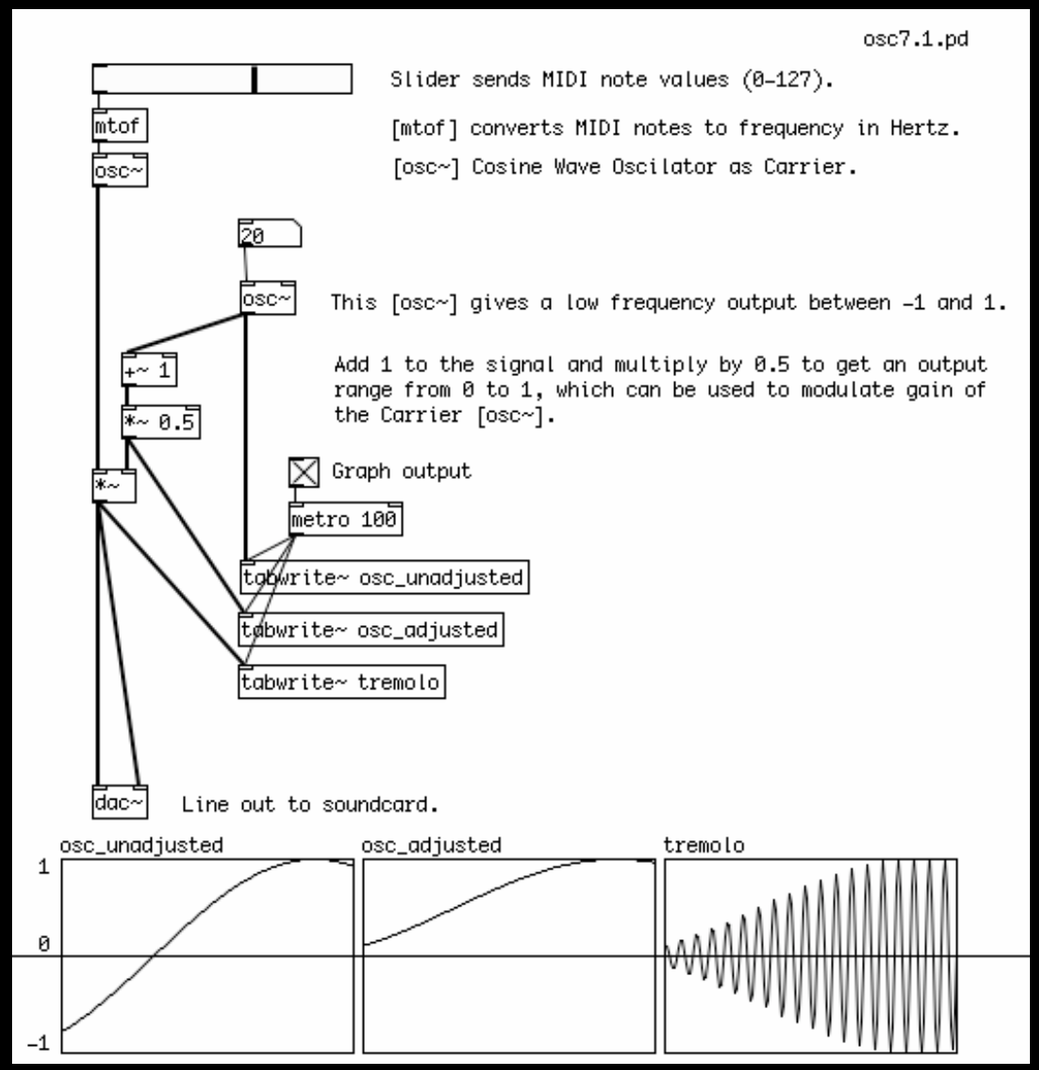
Amplitude Modulation

Amplitude Modulation Synthesis is a type of sound synthesis where the gain of one signal is controlled, or modulated, by the gain of another signal.



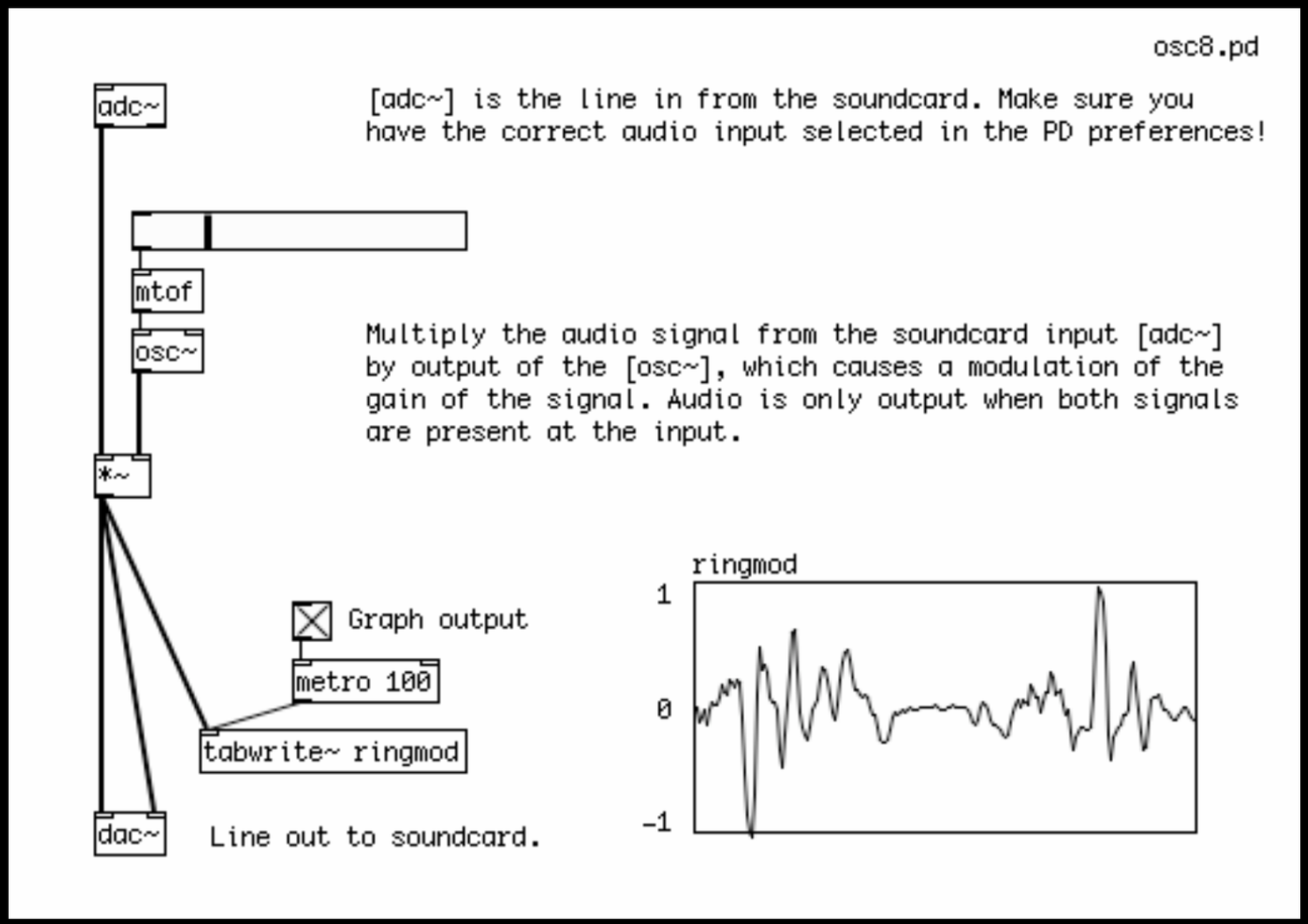
Tremolo

Tremolo is a form of Amplitude Modulation where the gain of an audio signal is changed at a very slow rate



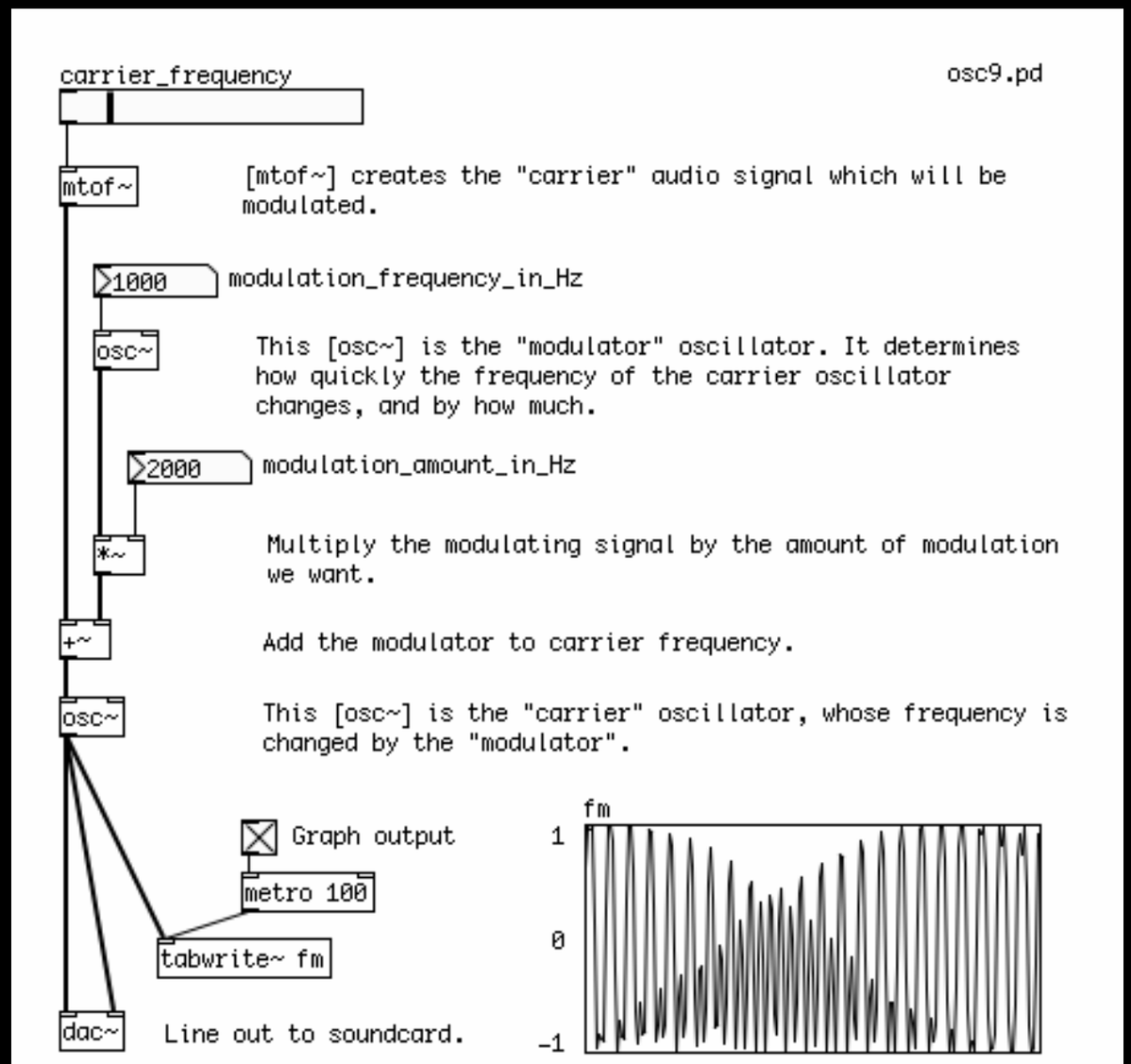
Ring Modulation

Tremolo is a form of Amplitude Modulation where the gain of an audio signal is changed at a very slow rate



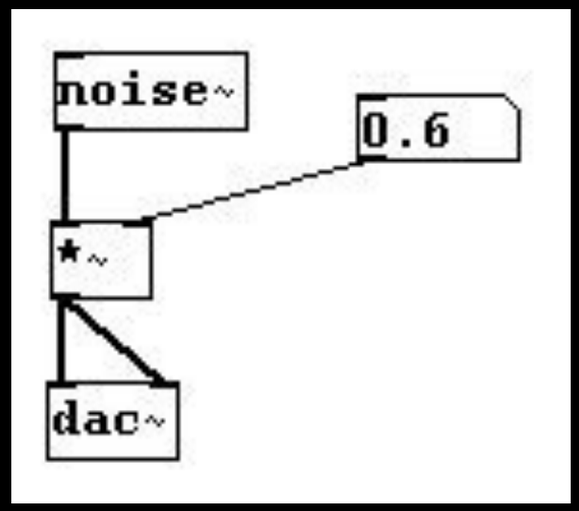
Frequency Modulation

FM Synthesis, is used to make periodic changes to the frequency of an oscillator.



Subtractive synthesis

In contrast to additive synthesis - which uses what might be considered the 'atom' of sound, the sine tone, as a starting point - subtractive synthesis begins with all sound and reduces

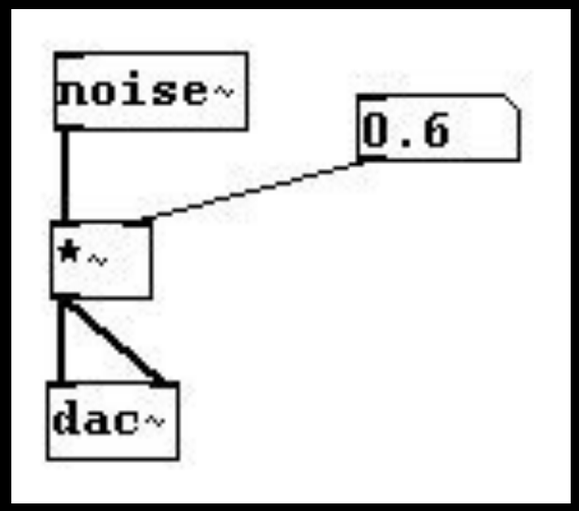


Causing a speaker membrane to vibrate completely chaotically and randomly will produce all audible frequencies simultaneously.

- White Noise -

Subtractive synthesis

In contrast to additive synthesis - which uses what might be considered the 'atom' of sound, the sine tone, as a starting point - subtractive synthesis begins with all sound and reduces

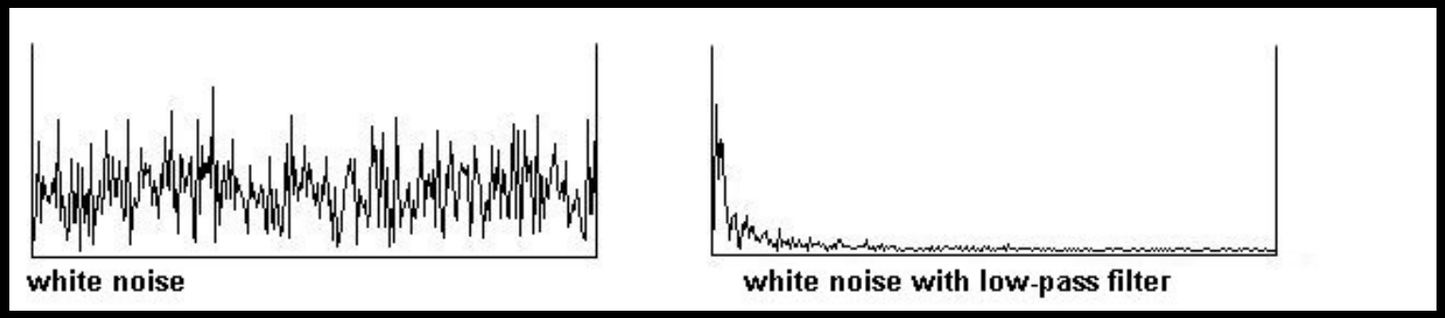


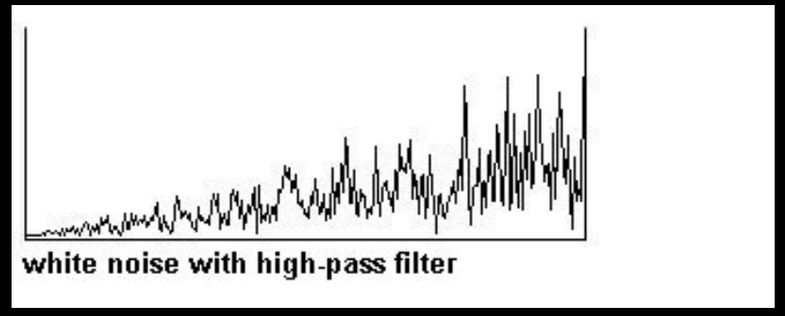
Causing a speaker membrane to vibrate completely chaotically and randomly will produce all audible frequencies simultaneously.

- White Noise -

Filters

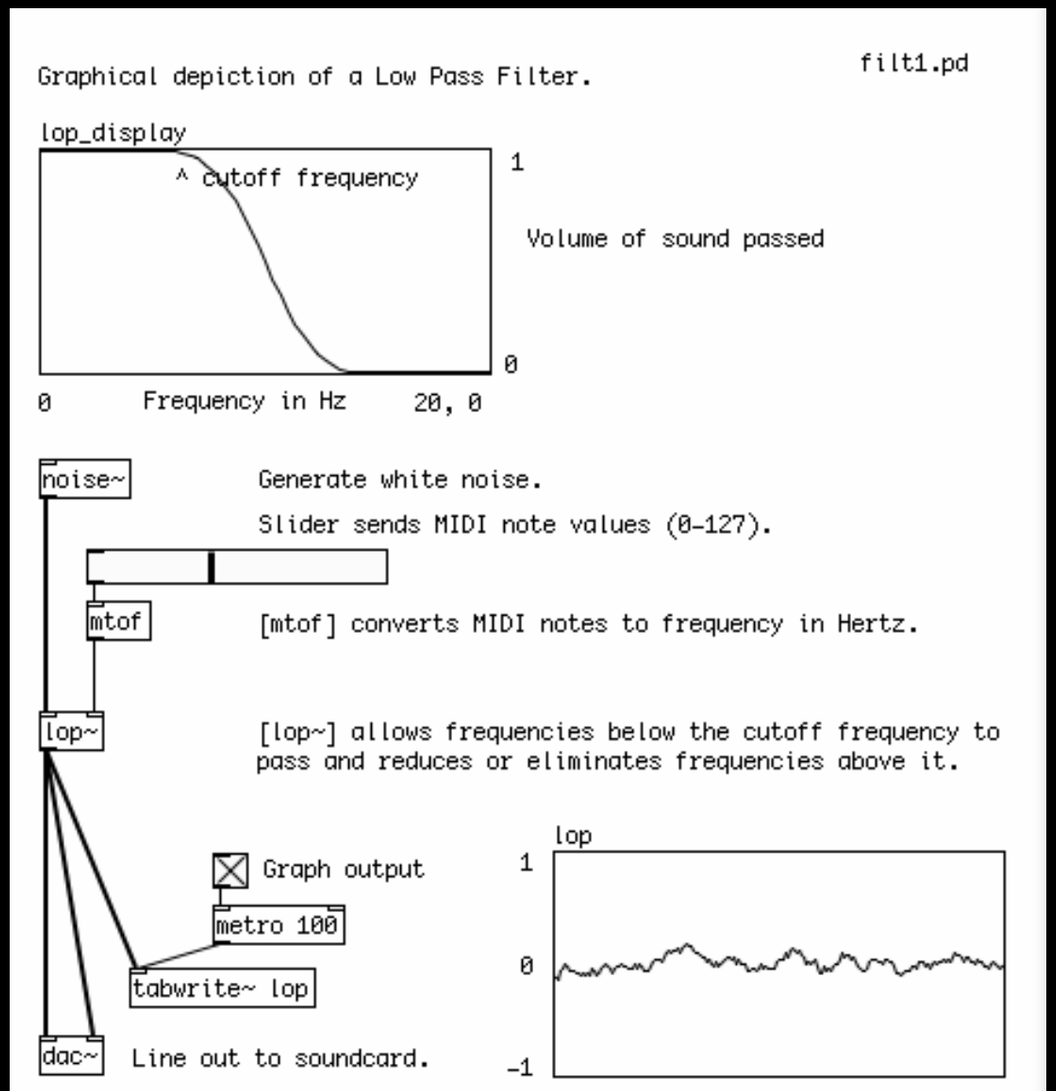
A filter works by allowing some frequencies through, while reducing or eliminating others.





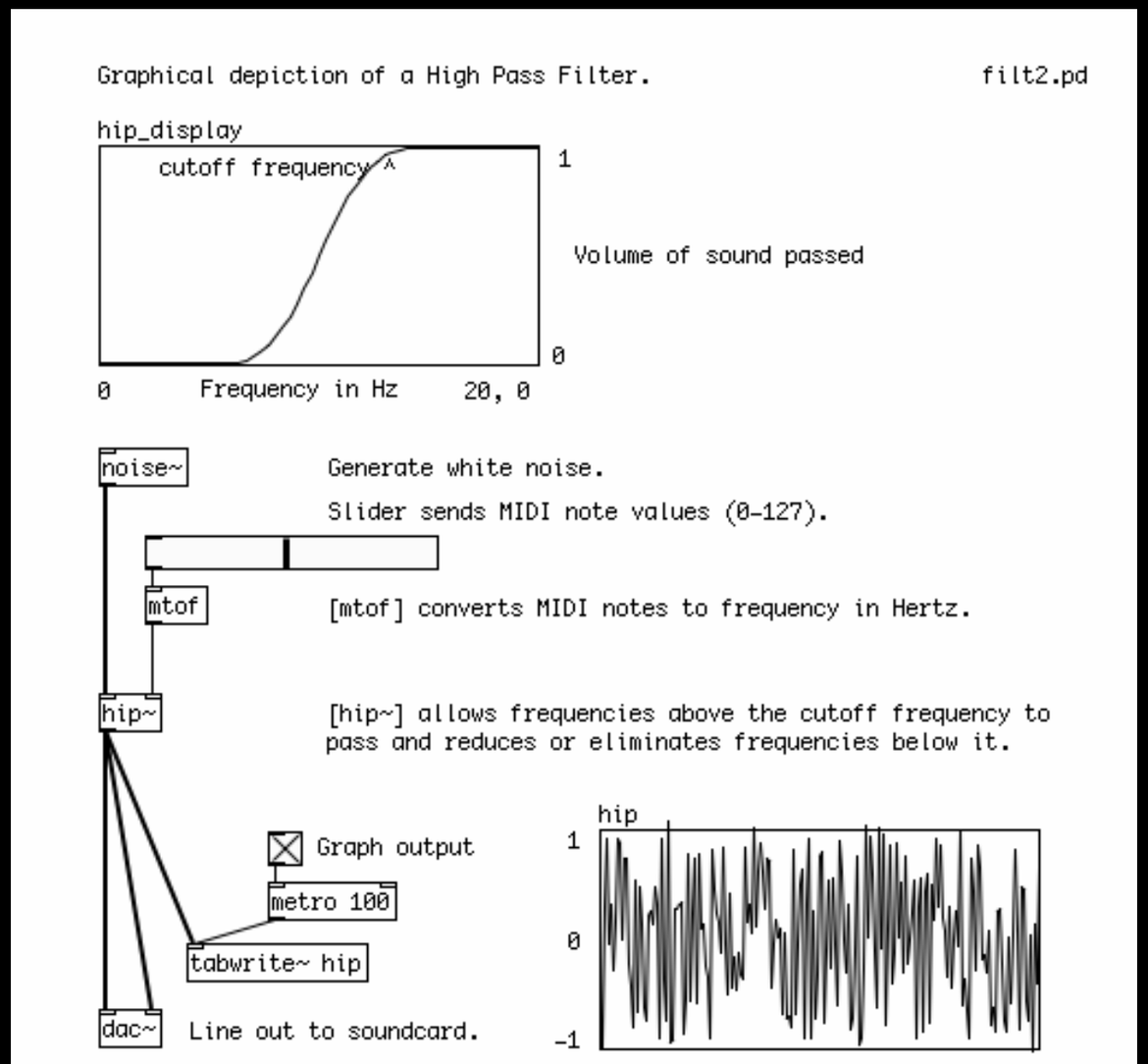
Low Pass Filter

A filter which allows only low frequencies to pass is called a **Low Pass Filter**.



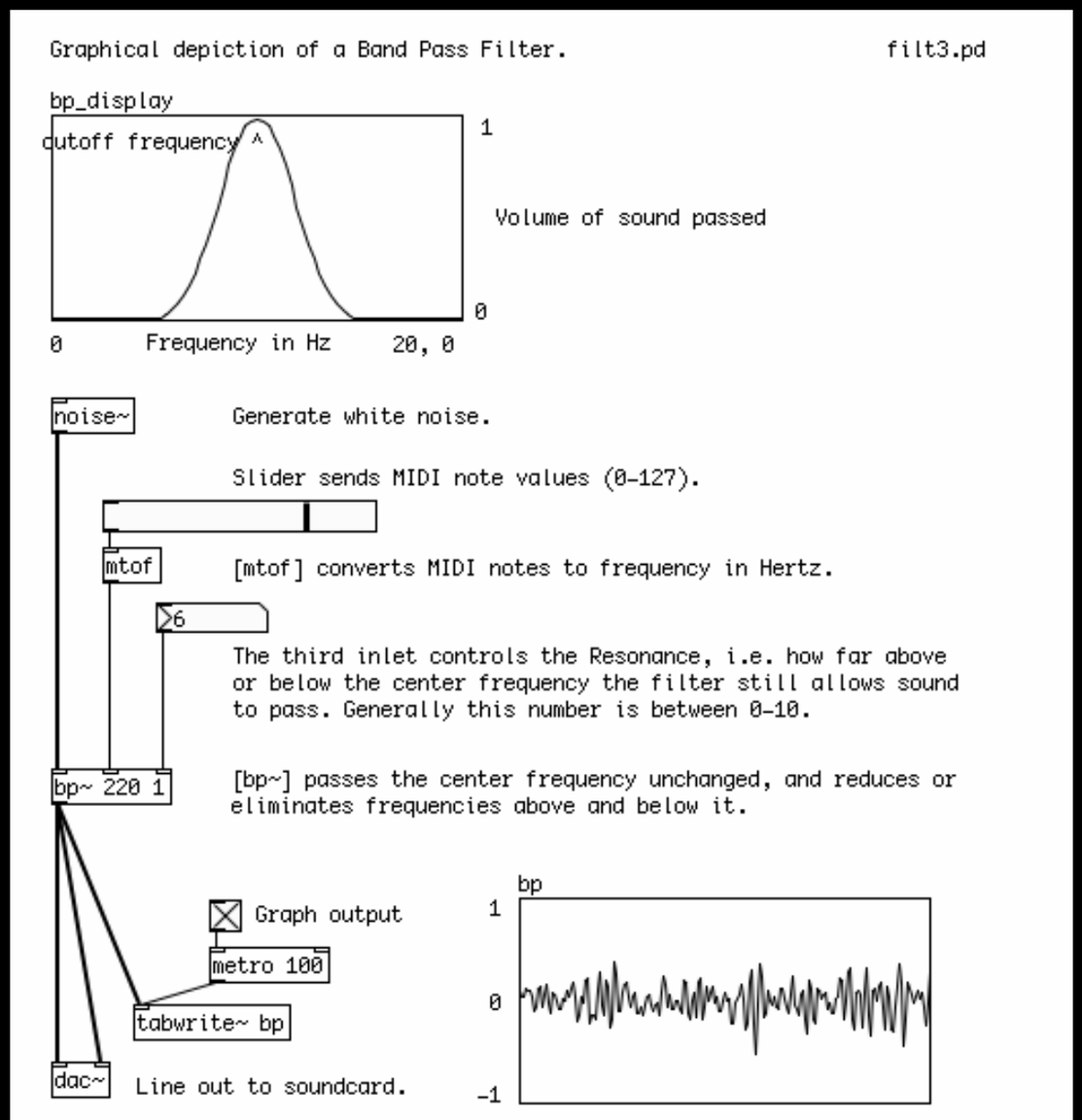
High Pass Filter

While one which allows only high frequencies is called a **High Pass Filter**.



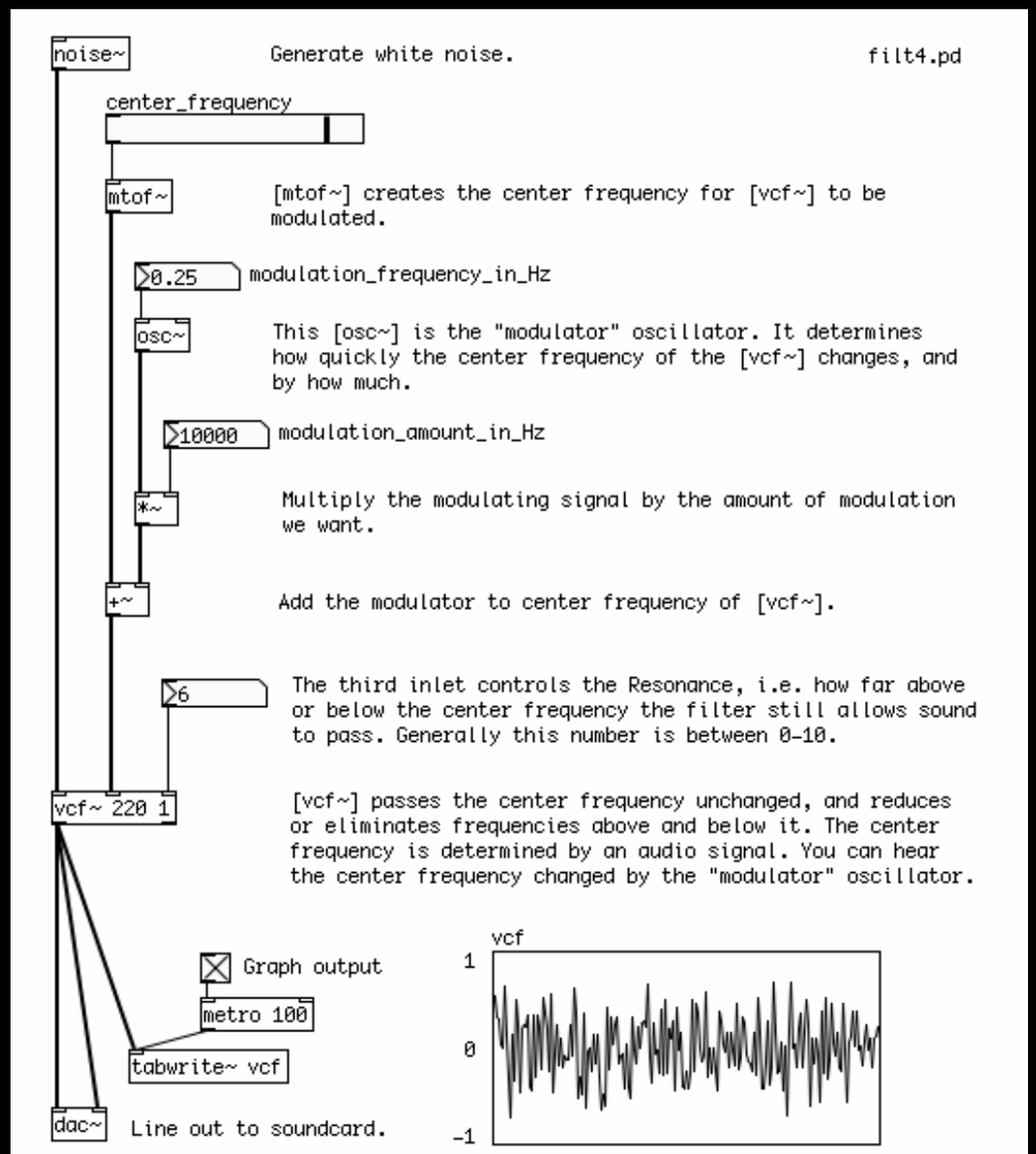
Band Pass Filter

A filter which allows some range of frequencies between highest and lowest is called a **Band Pass Filter.**



Voltage Controlled Filter

Voltage Controlled Filter) is a filter whose Center Frequency and Resonance can be controlled by audio signals.



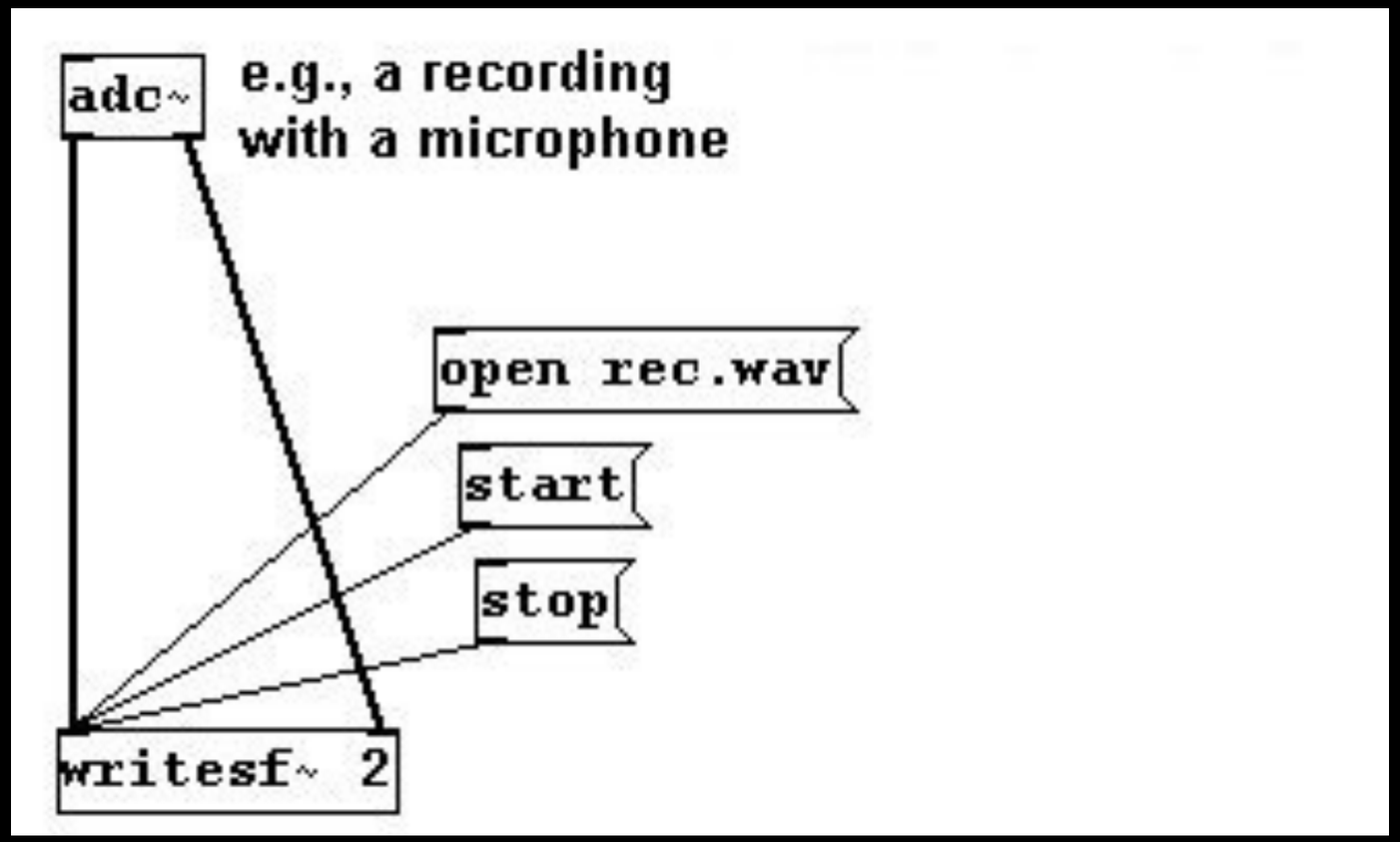
Exercise

1. Open and play around osc7.pd, osc8.pd, osc9.pd .. osc12.pd
2. Open and play around filt1.pd, filt2.pd, filt3.pd and filt4.pd
3. Try to understand 3-3-2-1-filtercolors

Sampling

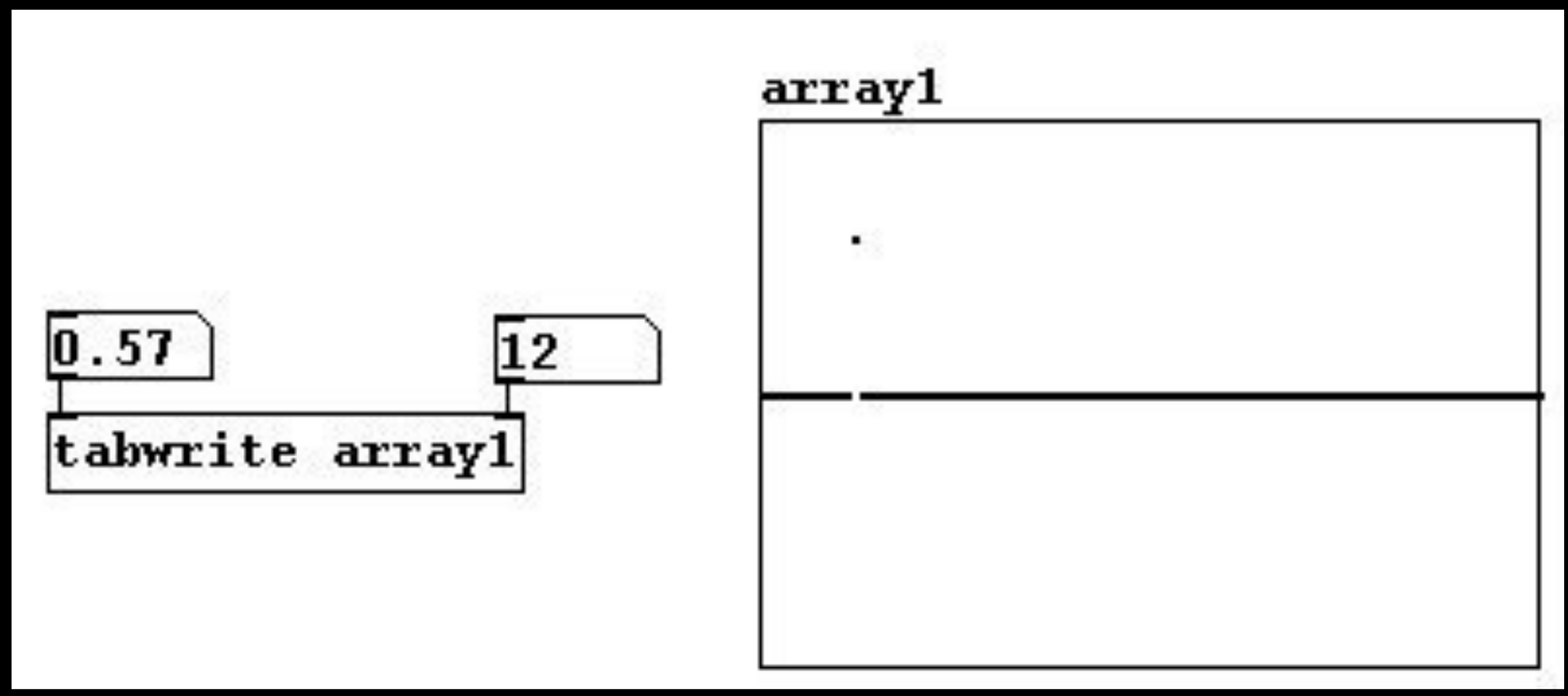
Sound files

First you have to use the message "open [name]" to choose the name of the file you want to create. Start recording using "start" and stop it using "stop".



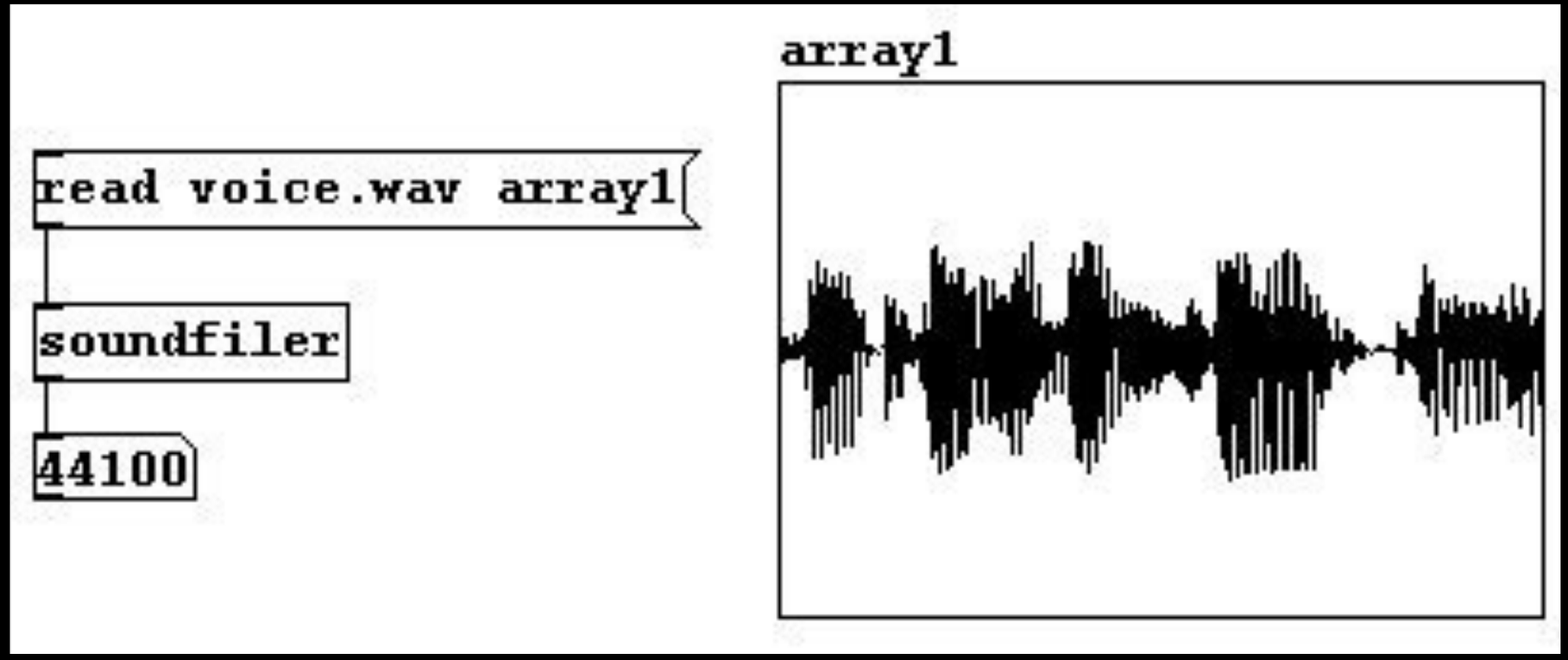
Buffers

Create one place for one sound using “array”. Using [tabwrite]



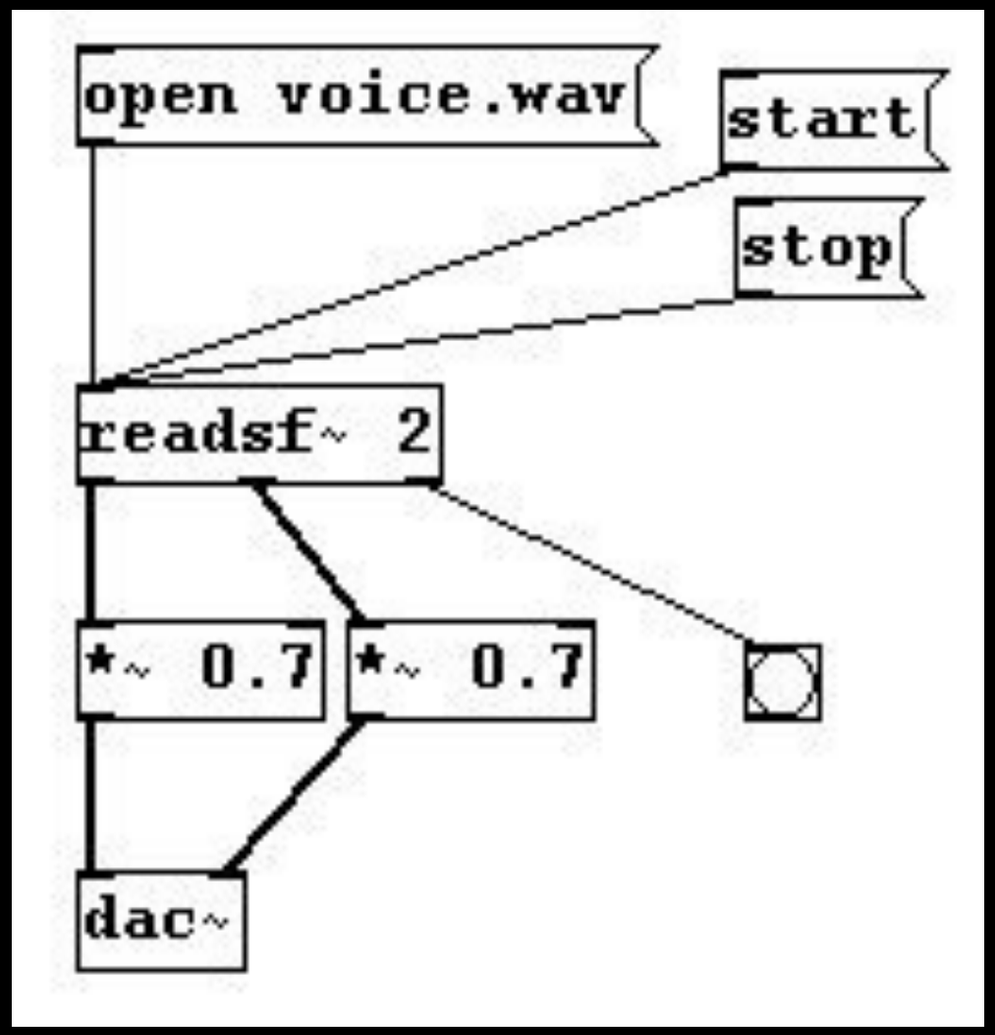
Soundfiler

You can also make a connection between sound files located in the main memory and those located on the hard disk in array. Using [soundfiler]



Playback of saved sound

Sound files that are on an external storage device like a hard disk can be read - that is, played back - in Pd with [readsf~]



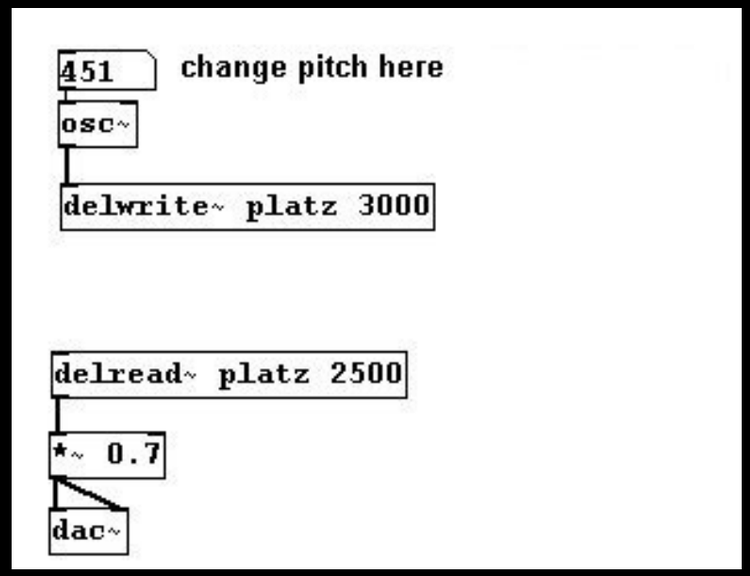
Exercise

1. Open the examples in the sampling folder.
2. Try to understand them so you can reuse it in the future.

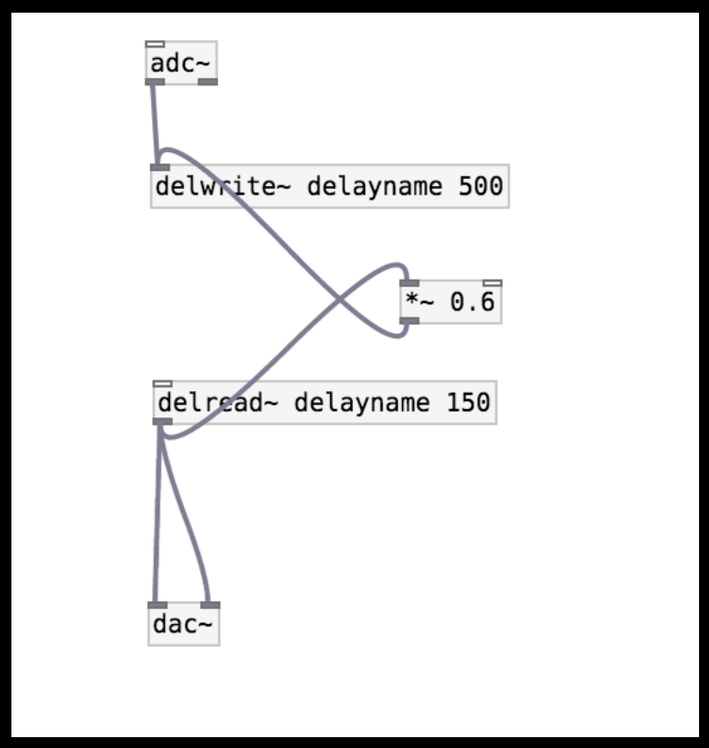
Effects

Audio Delay

You can also delay signals!

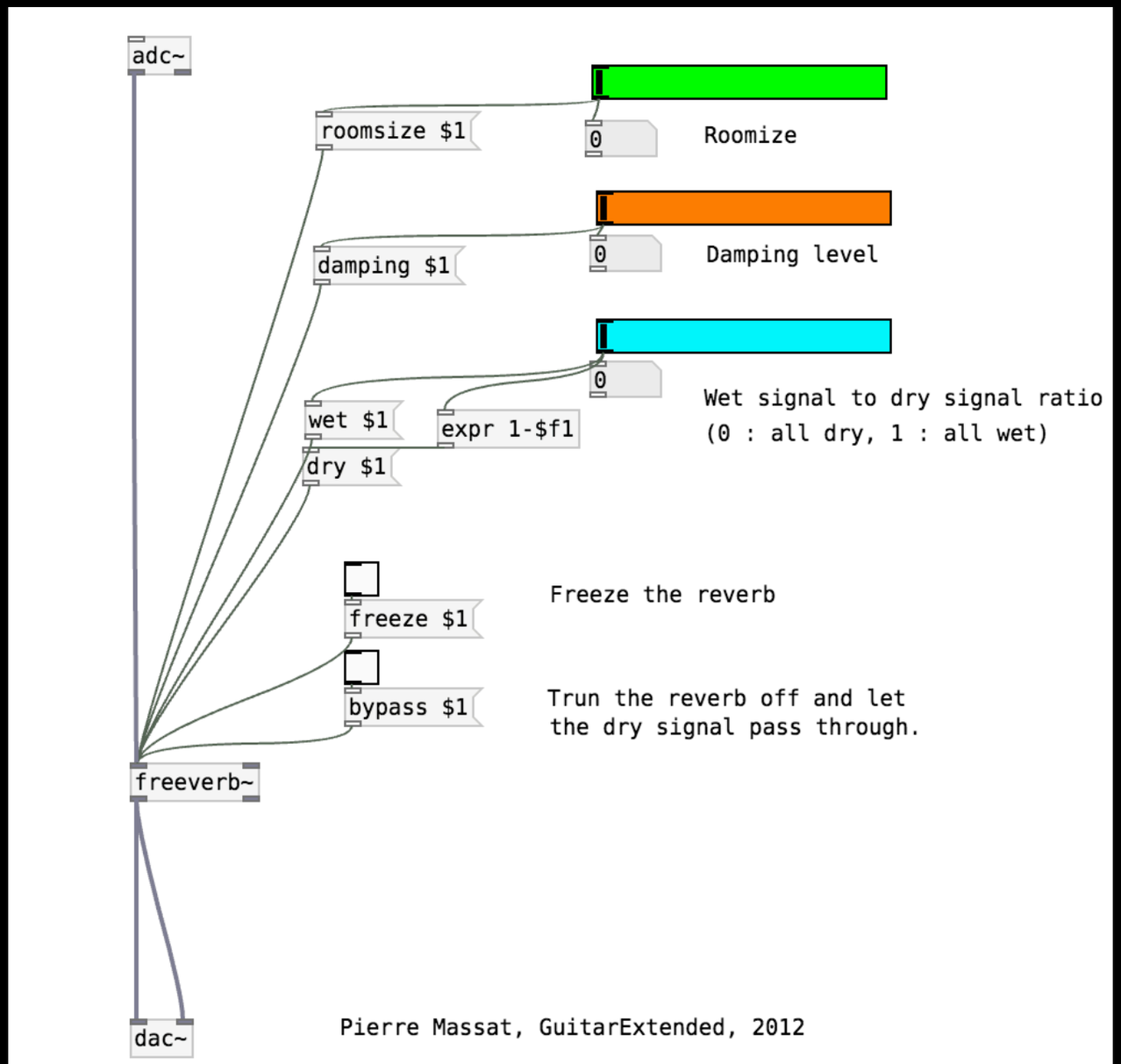


And add it to the signal



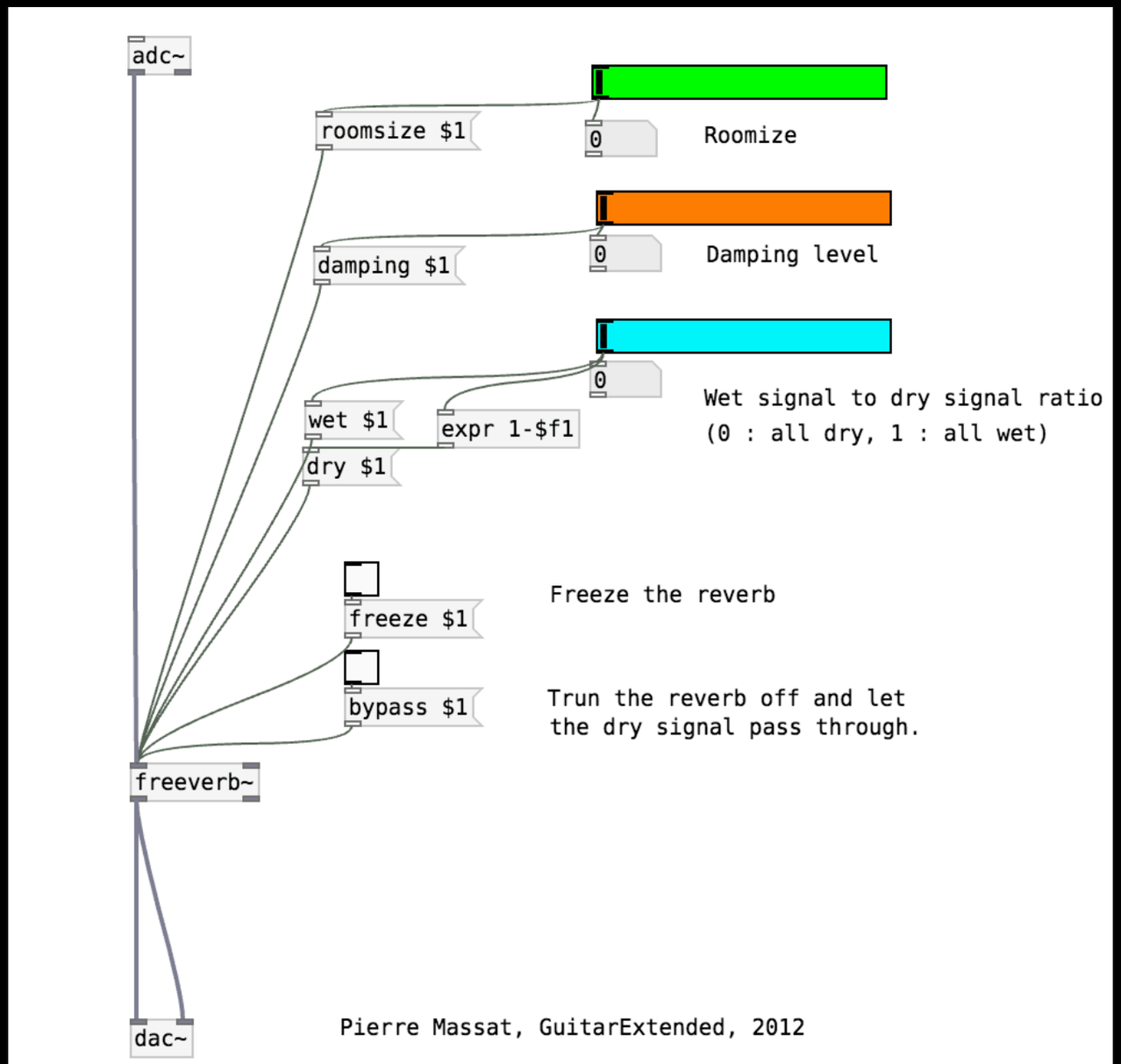
Reverb

It is very simple to get a nice reverb effect using [freeverb~].



Reverb

It is very simple to get a nice reverb effect using [freeverb~].



Other effects

Some other common audio effects are:

- **Echo:** The echo patch takes an input signal and outputs the same signal repeatedly with lower amplitude and a time delay.
- **Flanger:** The flanger audio effect is created by mixing two identical audio signals together, time-shifting one back and forth by a few milliseconds.
- **Phaser:** Instead of time-shifting the signal, the phaser uses a series of all-pass filters that modify the phase of the input signal.
- **Chorus:** The chorus effect is created by making multiple copies of the original signal and pitch-modulating them with an LFO

Exercise

1. Open the examples in the effects folder.
2. Try to understand them so you can reuse it in the future.

Questions?

Imanol Gómez
imanolgomez.net
yo@imanolgomez.net