Intro to Pure Data

Mikael Fernström

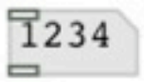
Pure Data

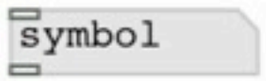
- http://puredata.info/
- visual programming language
- originally by Miller Puckette, 1990s
- for creating interactive multimedia works.
- open source project
- similar to Puckette's original Max program

The basic pd building blocks

object

number





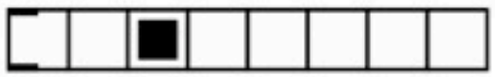
comment

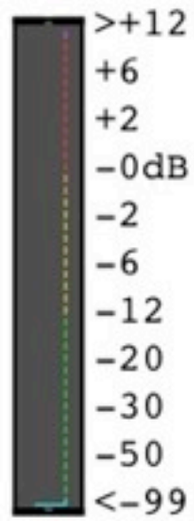
bang

checkbox

sliders

radiobuttons





Pure Data Reference Card

Karim Barkati – December 12, 2010

Modes

`ctl-e` (or `cmd-e`) toggle between *run* mode (performance) and *edit* mode (programming); this affects how mouse clicks affect the patch.

Glue

`bang`	output a bang message
`float`	store and recall a number
`symbol`	store and recall a symbol
`int`	store and recall an integer
`send`	send a message to a named object
`receive`	catch "sent" messages
`select`	test for matching numbers or symbols
`route`	route messages according to first element
`pack`	make compound messages
`unpack`	get elements of compound messages
`trigger`	sequence and convert messages
`spigot`	interruptible message connection
`moses`	part a numeric stream
`until`	looping mechanism
`print`	print out messages
`makefilename`	format a symbol with a variable field
`change`	remove repeated numbers from a stream
`swap`	swap two numbers
`value`	shared numeric value

Time

`delay`	send a message after a time delay
`metro`	send a message periodically
`line`	send a series of linearly stepped numbers
`timer`	measure time intervals
`cputime`	measure CPU time
`realtime`	measure real time
`pipe`	dynamically growable delay line for messages

Math

`+ - * / pow`	arithmetic
`== != > < >= <=`	relational tests
`& && | || %`	bit twiddling
`mtof ftom powtodb rmstodb`	convert acoustical units
`dbtopow dbtorms`	
`mod div sin cos tan atan`	higher math
`atan2 sqrt log exp abs`	
`random expr`	lower math
`max min`	greater or lesser of 2 numbers
`clip`	force a number into a range

Midi

`notein ctlin pgmin bendin touchin`	MIDI input
`polytouchin midiin sysexin`	
`noteout ctlout pgmout bendout touchout`	MIDI output
`polytouchout midiout`	
`makenote`	send note-on messages and schedule note-off for later
`stripnote`	strip note-off messages

Tables

`tabread`	read a number from a table
`tabread4`	read with 4 point interpolation
`tabwrite`	write a number to a table
`soundfiler`	read and write tables to soundfiles

Misc

`loadbang`	bang on load
`serial`	serial device control for NT only
`netsend`	send messages over the internet
`netreceive`	receive them
`qlist`	text-based message sequencer
`textfile`	file to message converter
`openpanel`	"Open" dialog
`savepanel`	"Save as" dialog
`bag`	set of numbers
`poly`	polyphonic voice allocation
`key, keyup`	numeric key values from keyboard
`keyname`	symbolic key name

Audio Math

`+~ -~ *~ /~`	arithmetic on audio signals
`max~ min~`	maximum or minimum of 2 inputs
`clip~`	constrict signal to lie between two bounds
`q8_rsqrt~`	cheap reciprocal square root (beware 8 bits!)
`q8_sqrt~`	cheap square root (beware 8 bits!)
`wrap~`	wraparound (fractional part, sort of)
`fft~`	complex forward discrete Fourier transform
`ifft~`	complex inverse discrete Fourier transform
`rfft~`	real forward discrete Fourier transform
`rifft~`	real inverse discrete Fourier transform
`framp~`	estimate frequency and amplitude of FFT components
`mtof~ ftom~ rmstodb~ dbtorms~ rmstopow~ powtorms~`	acoustic conversions

Audio Glue

`dac~`	audio output
`adc~`	audio input
`sig~`	convert numbers to audio signals
`line~`	generate audio ramps
`vline~`	deluxe `line~`
`threshold~`	detect signal thresholds
`snapshot~`	sample a signal (convert it back to a number)
`vsnapshot~`	deluxe `snapshot~`
`bang~`	send a bang message after each DSP block
`samplerate~`	get the sample rate
`send~`	nonlocal signal connection with fanout
`receive~`	get signal from `send~`
`throw~`	add to a summing bus
`catch~`	define and read a summing bus
`block~`	specify block size and overlap
`switch~`	switch DSP computation on and off
`readsf~`	soundfile playback from disk
`writesf~`	record sound to disk

Audio Oscillators and Tables

`phasor~`	sawtooth oscillator
`cos~`	cosine
`osc~`	cosine oscillator
`tabwrite~`	write to a table
`tabplay~`	play back from a table (non-transposing)
`tabread~`	non-interpolating table read
`tabread4~`	four-point interpolating table read
`tabosc4~`	wavetable oscillator
`tabsend~`	write one block continuously to a table
`tabreceive~`	read one block continuously from a table

Audio Filters

`vcf~`	voltage controlled filter
`noise~`	white noise generator
`env~`	envelope follower (RMS amplitude in dB)
`hip~`	high pass filter
`lop~`	low pass filter
`bp~`	band pass filter
`biquad~`	raw filter (2 poles and 2 zeros)
`samphold~`	sample and hold unit
`print~`	print out one or more "blocks"
`rpole~`	raw real-valued one-pole filter
`rzero~`	raw real-valued one-zero filter
`rzero_rev~`	time-reversed `rzero~`
`cpole~ czero~`	`czero_rev` corresponding complex-valued filters

Audio Delay

`delwrite~`	write to a delay line
`delread~`	read from a delay line
`vd~`	read from a delay line at a variable delay time

Subwindows

`pd`	define a subwindow
`table`	array of numbers in a subwindow
`inlet`	add an inlet to a pd
`outlet`	add an outlet to a pd
`inlet~ outlet~`	signal versions of `inlet` and `outlet`

Data Templates

`struct`	define a data structure
`drawcurve, filledcurve`	draw a curve
`drawpolygon, filledpolygon`	draw a polygon
`plot`	plot an array field
`drawnumber`	print a numeric value

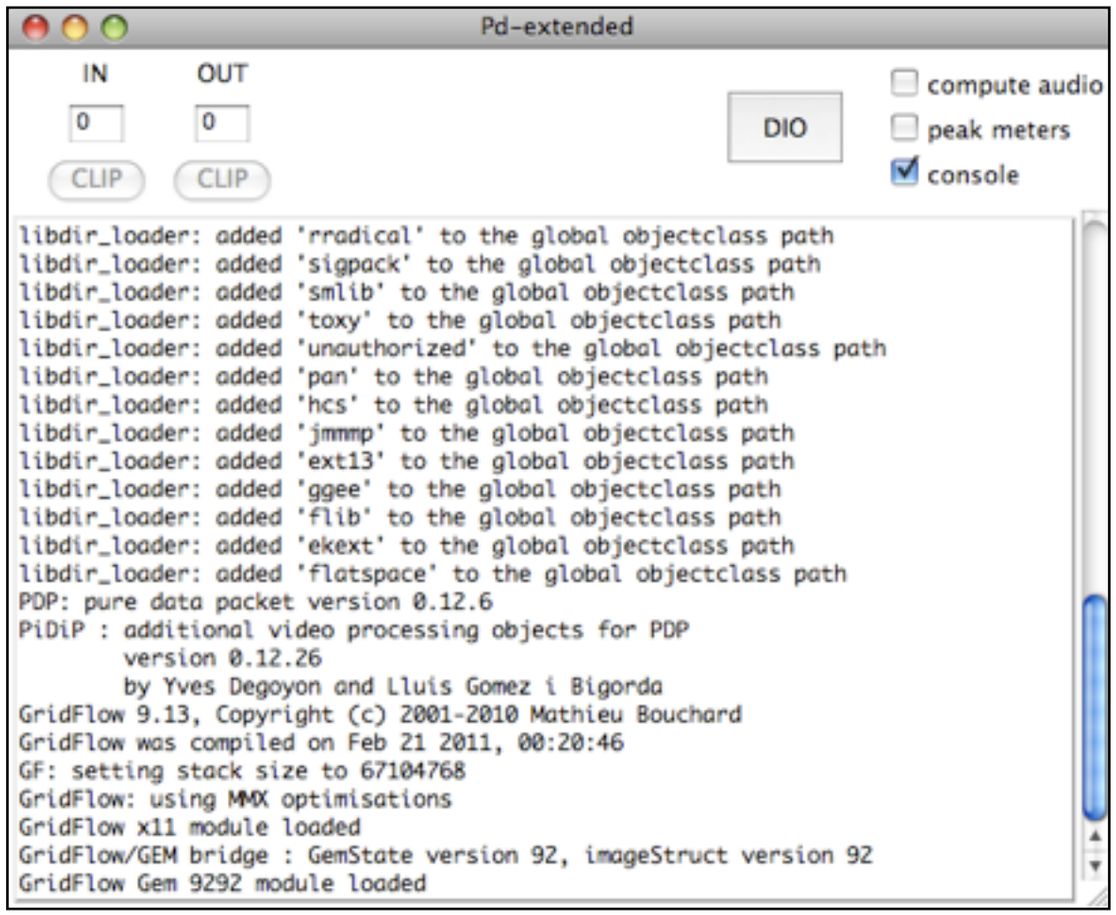
Accessing Data

`pointer`	point to an object belonging to a template
`get`	get numeric fields
`set`	change numeric fields
`element`	get an array element
`getsize`	get the size of an array
`setsize`	change the size of an array
`append`	add an element to a list
`sublist`	get a ptr into a list which is an elemt of another scalar

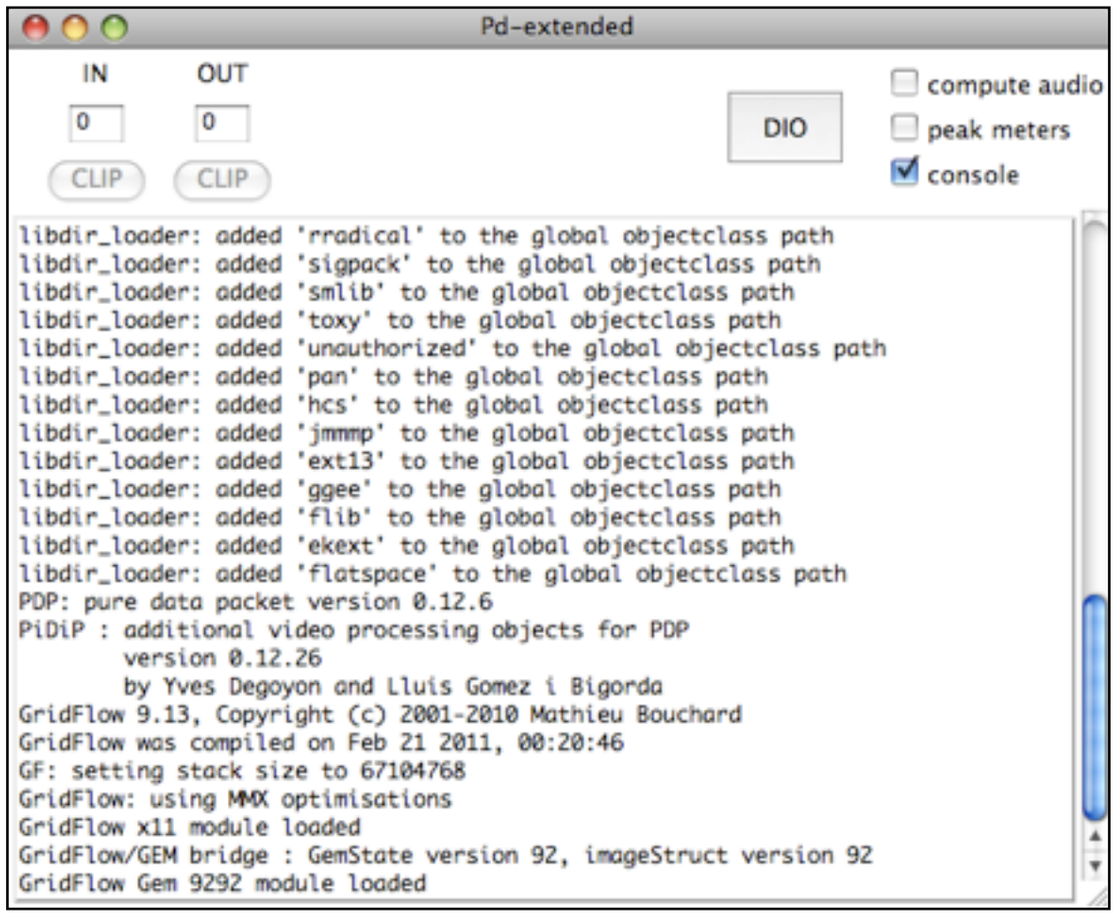
Getting started

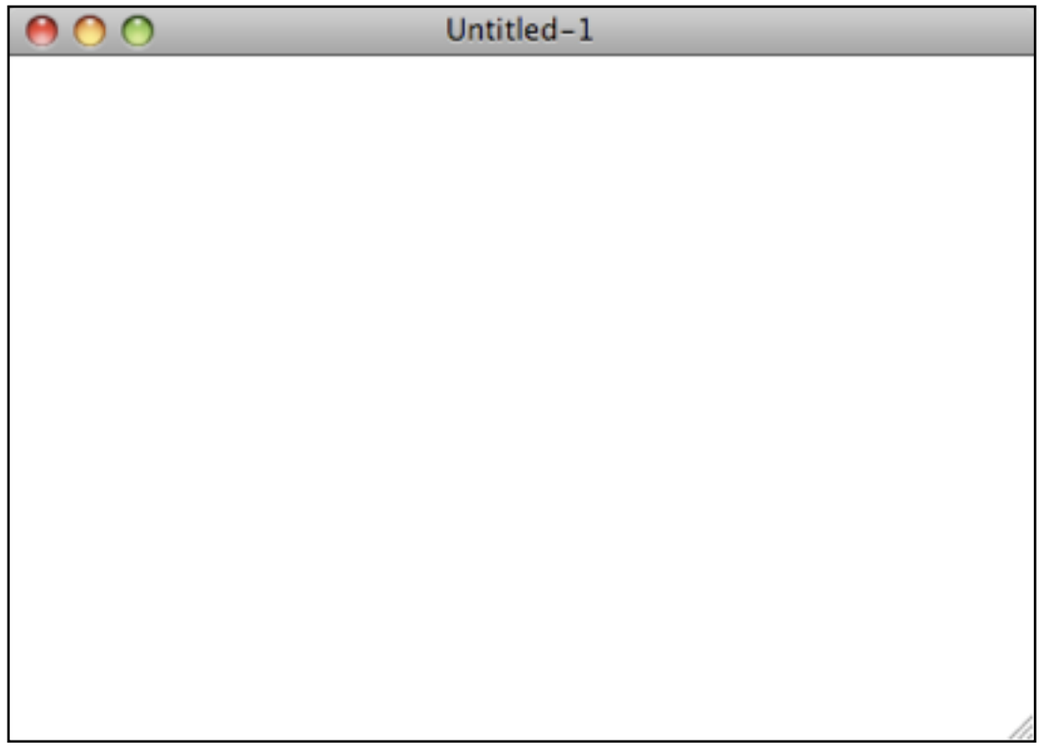
- Start PD
- File -> New

Getting started

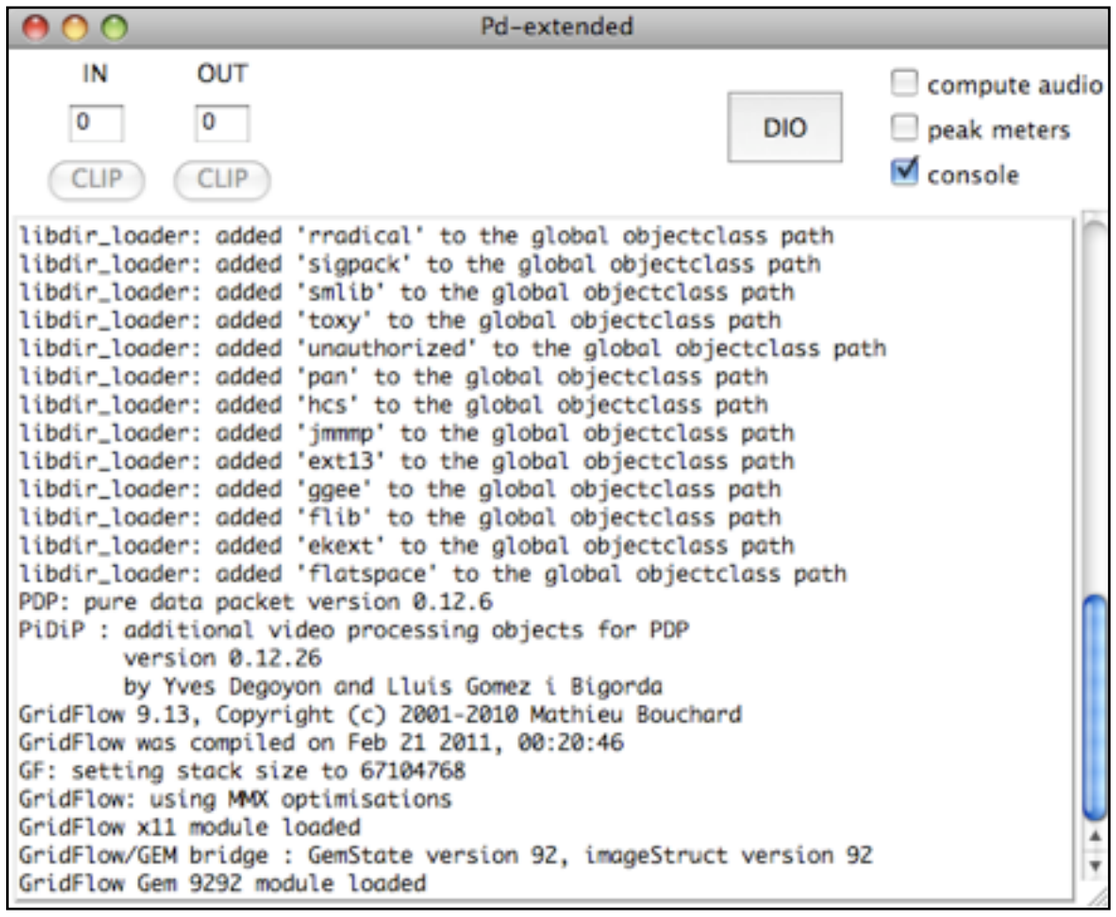


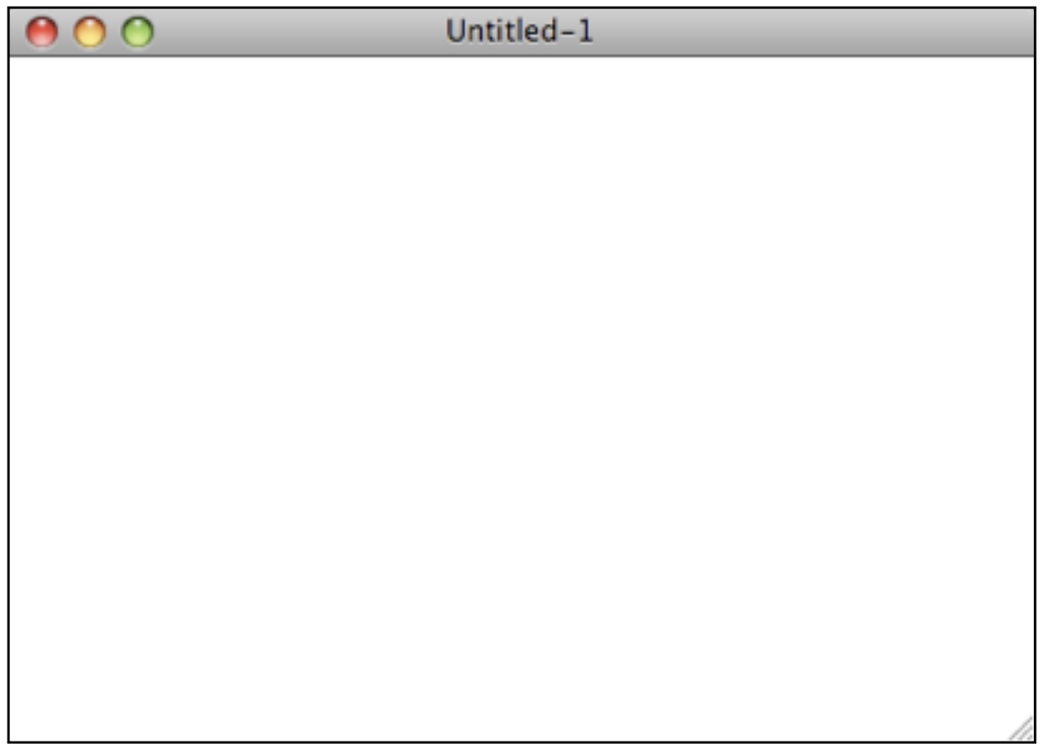
Getting started





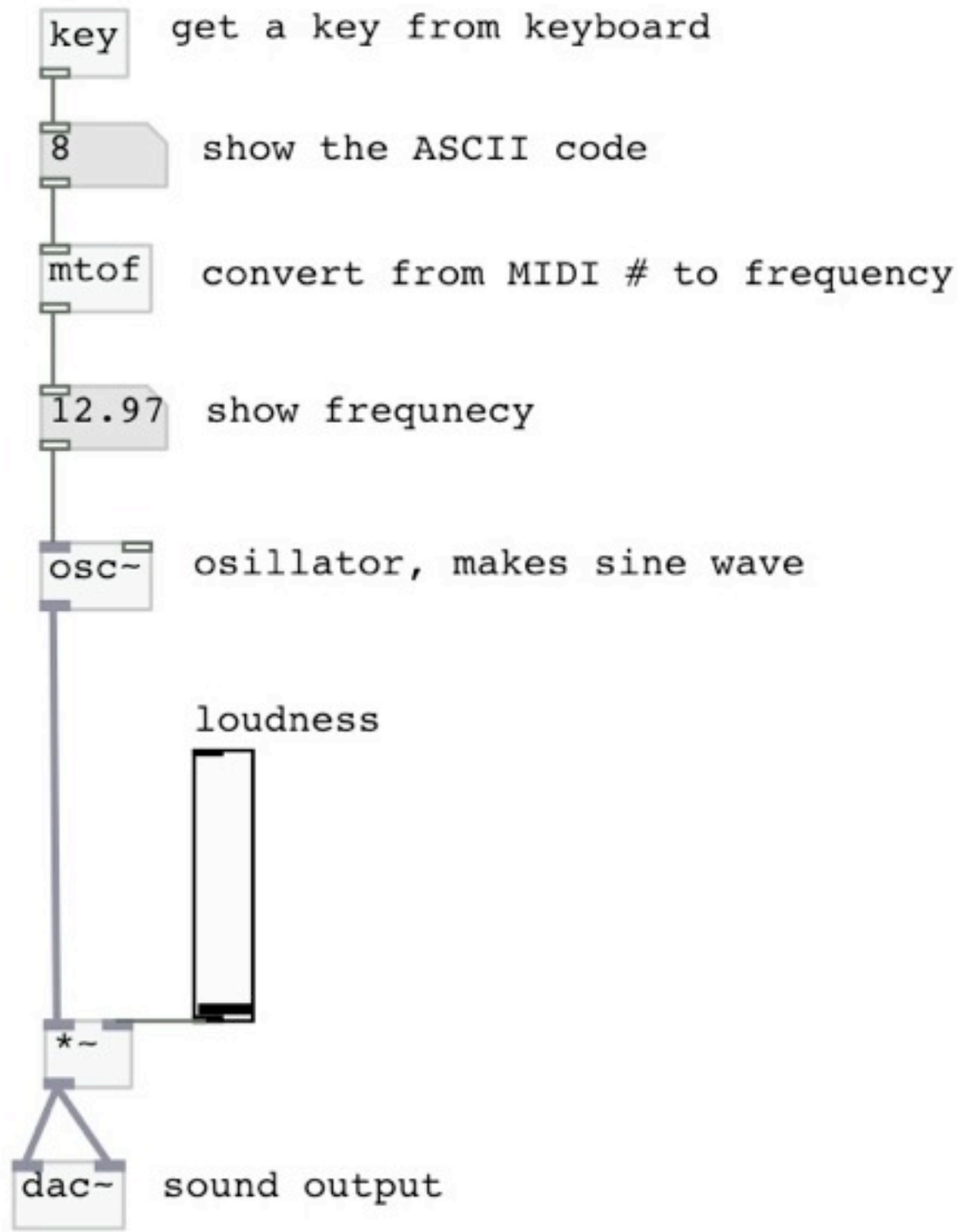
Getting started



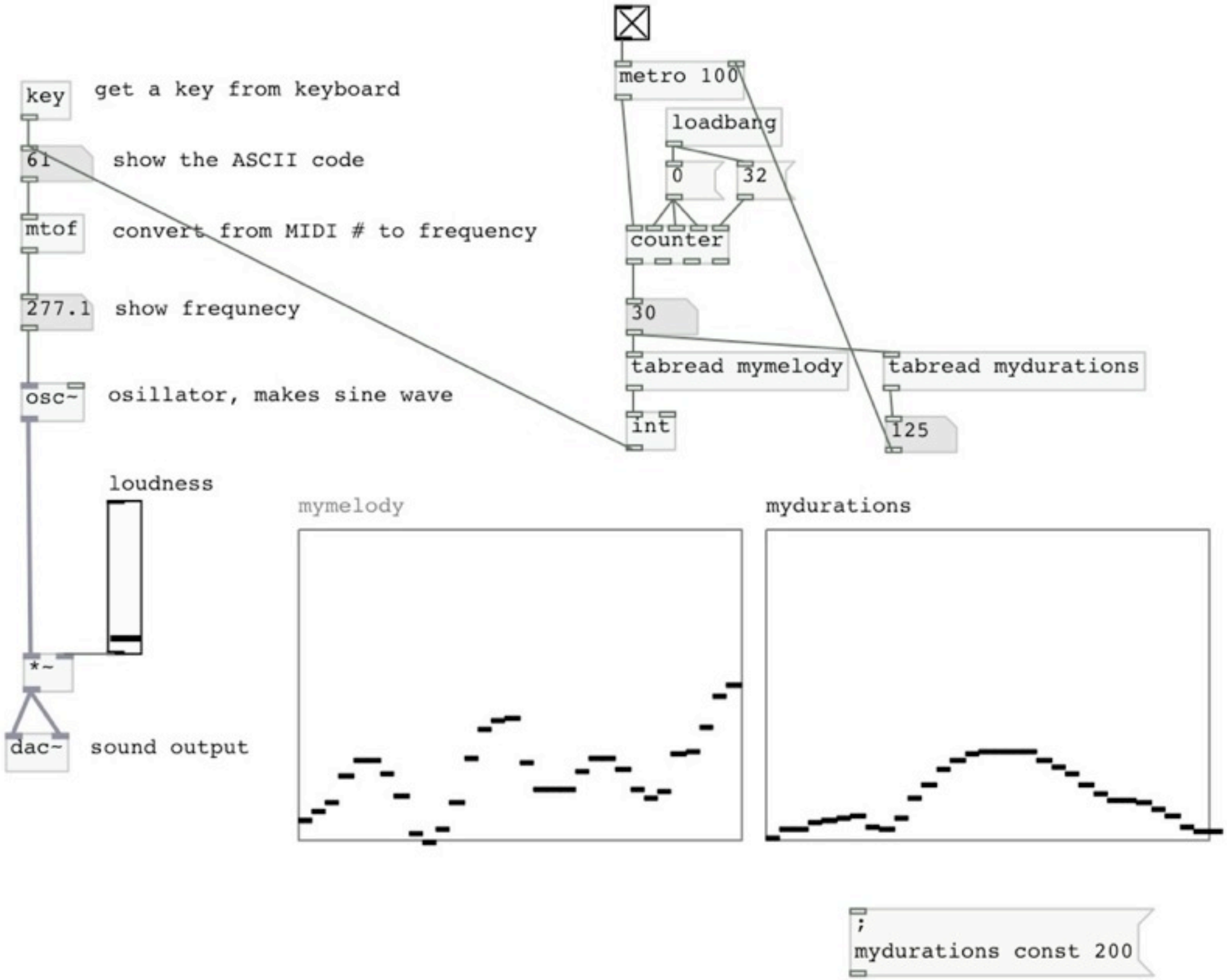


switch edit mode / run mode: **cmd** + **e**

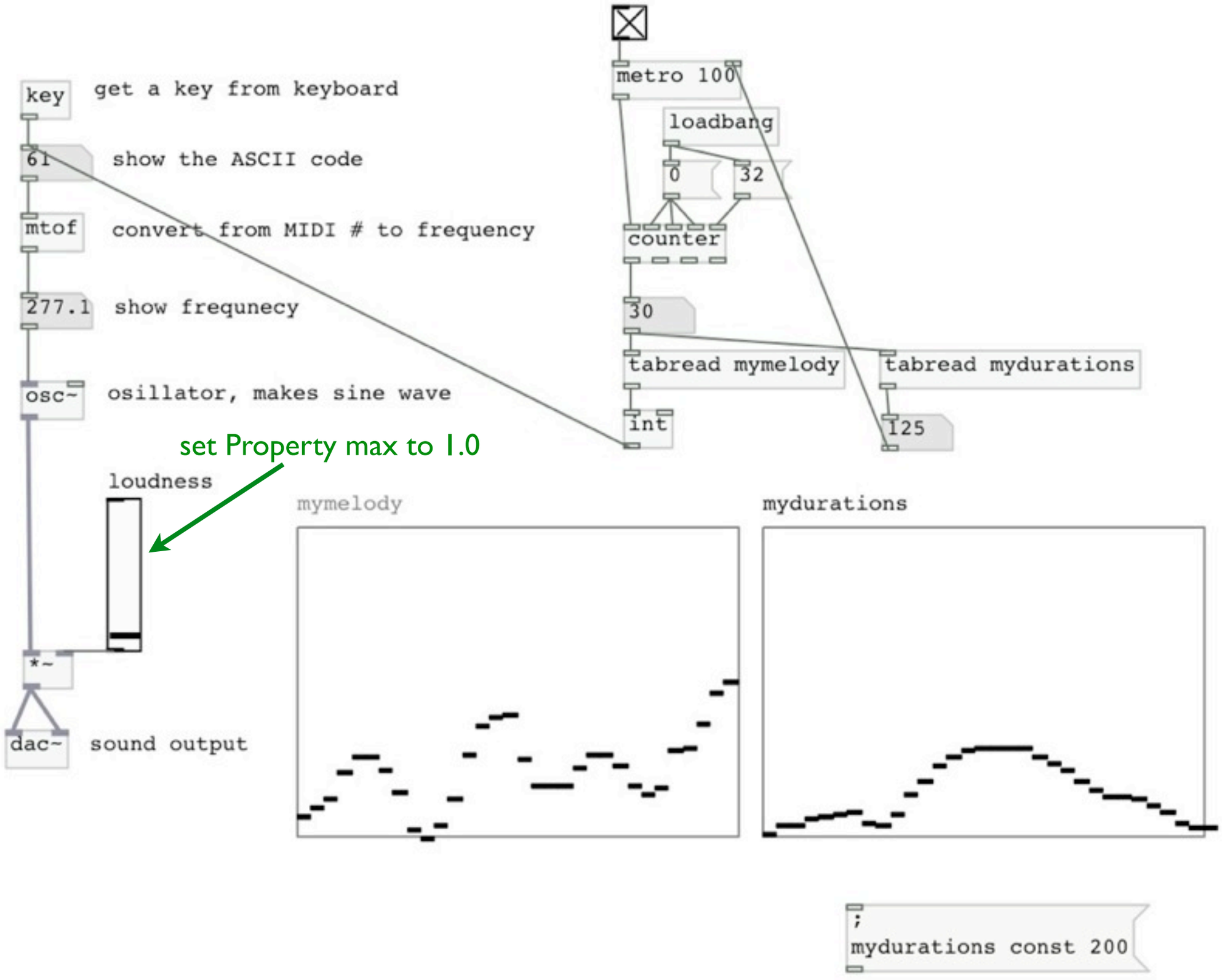
Your first pd patch



Your second pd patch: draw a melody



Your second pd patch: draw a melody



To continue your pd exploration

http://puredata.info/docs/StartHere/

My examples

(also including an extra 8 channel, 32 step sequencer):

www.idc.ul.ie/mikael/sounds/CAO-2012Examples.zip

Questions?