

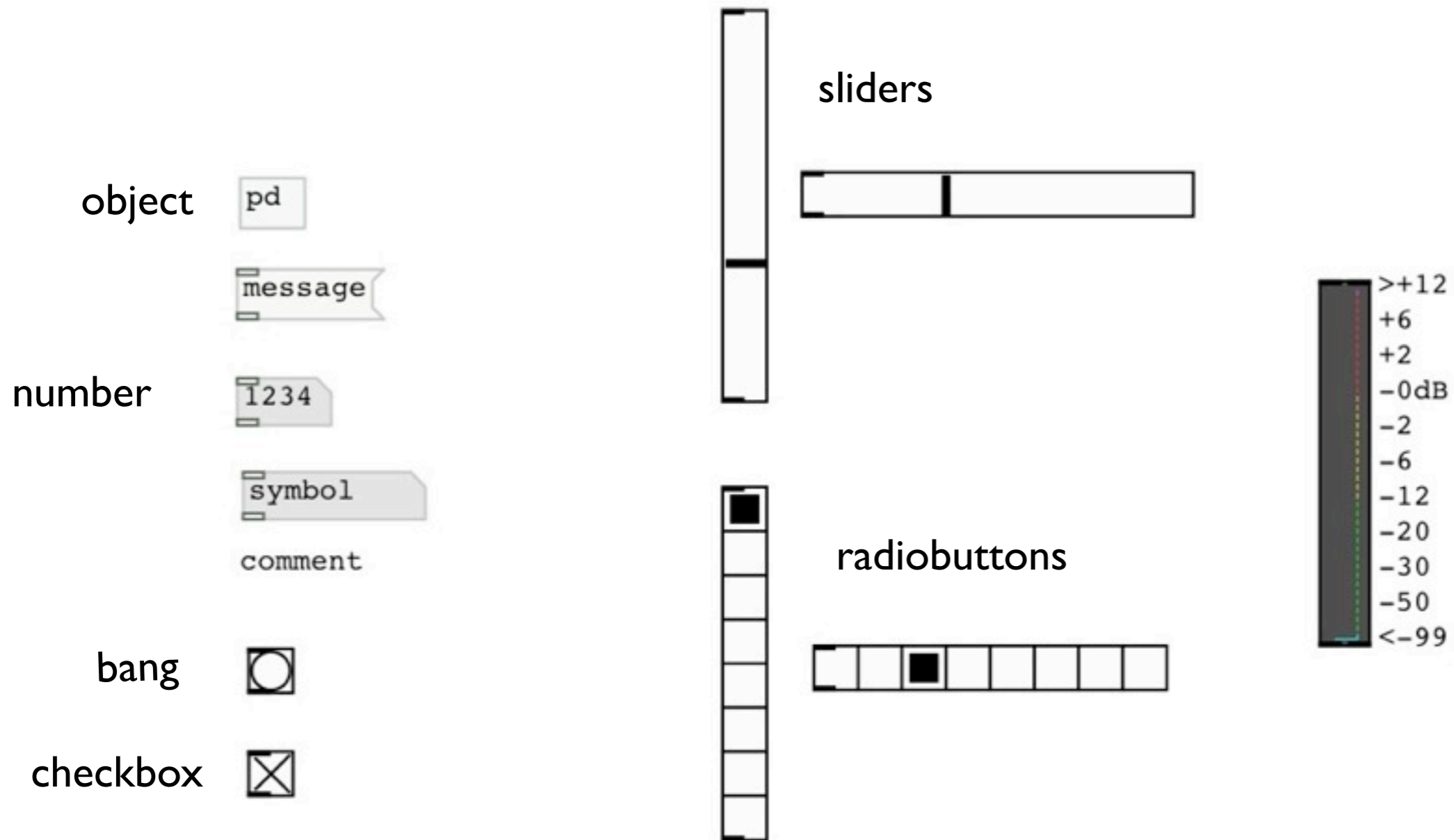
# Intro to Pure Data

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



# 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

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

## Time

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

## Math

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

## Midi

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

## Tables

<code>tabread</code>	read a number from a table
<code>tabread4</code>	read with 4 point interpolation
<code>tabwrite</code>	write a number to a table
<code>soundfiler</code>	read and write tables to soundfiles

## Misc

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

## Audio Math

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

## Audio Glue

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

Copyright © 2010 Karim BARKATI <karim.barkati@gmail.com>. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

## Audio Oscillators and Tables

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

## Audio Filters

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

## Audio Delay

<code>delwrite^-</code>	write to a delay line
<code>delread^-</code>	read from a delay line
<code>vd^-</code>	read from a delay line at a variable delay time

## Subwindows

<code>pd</code>	define a subwindow
<code>table</code>	array of numbers in a subwindow
<code>inlet</code>	add an inlet to a pd
<code>outlet</code>	add an outlet to a pd
<code>inlet^- outlet^-</code>	signal versions of inlet and outlet

## Data Templates

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

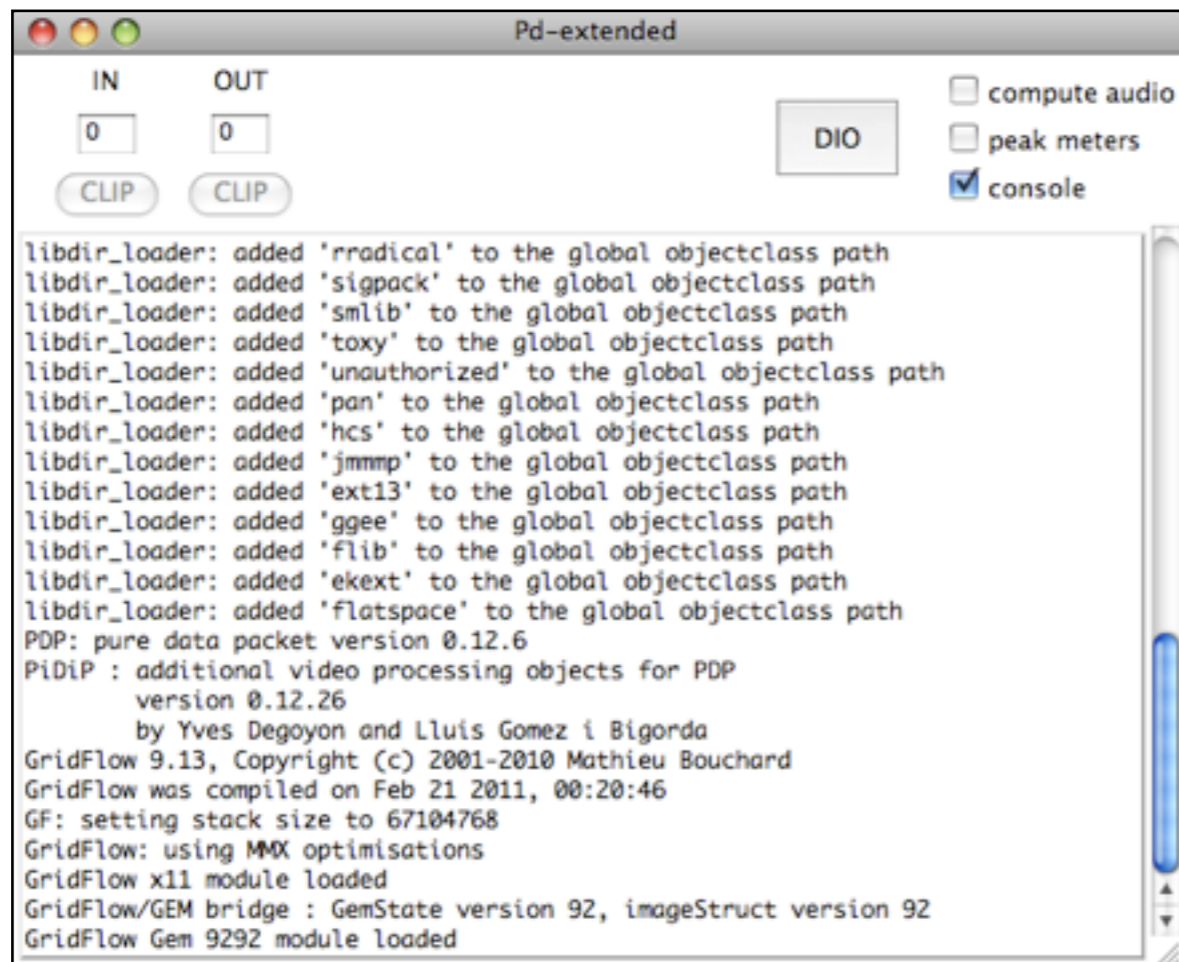
## Accessing Data

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

# Getting started

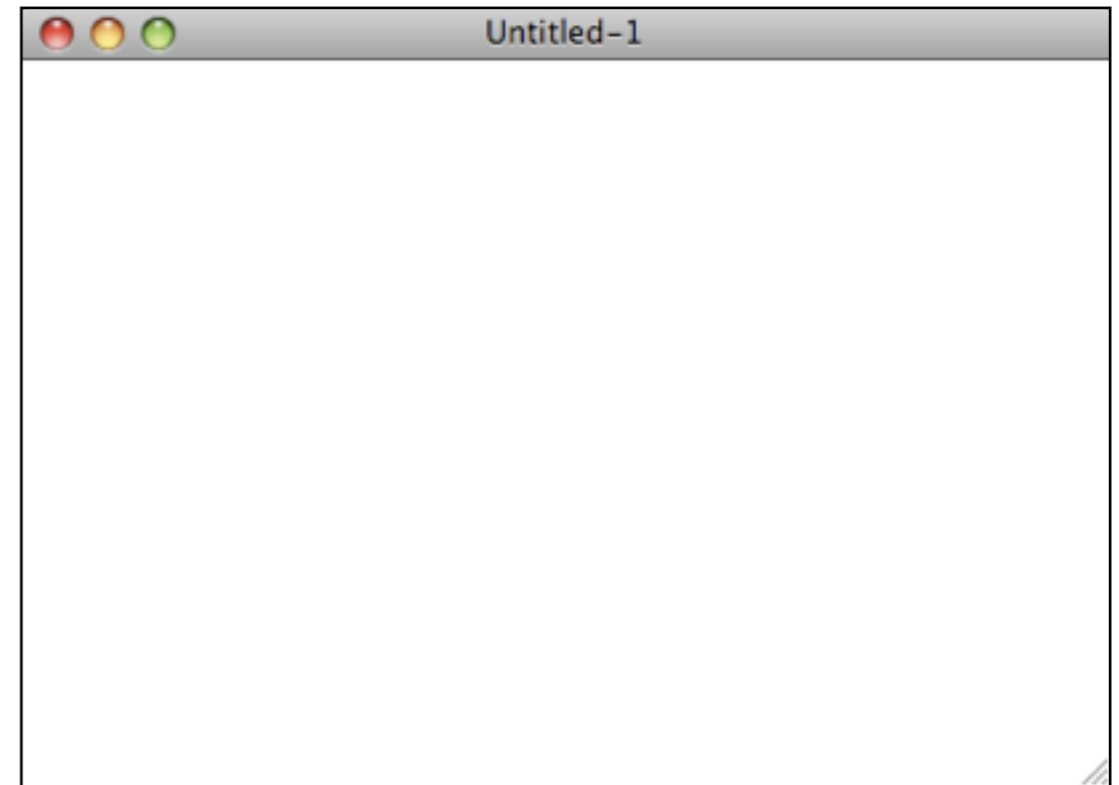
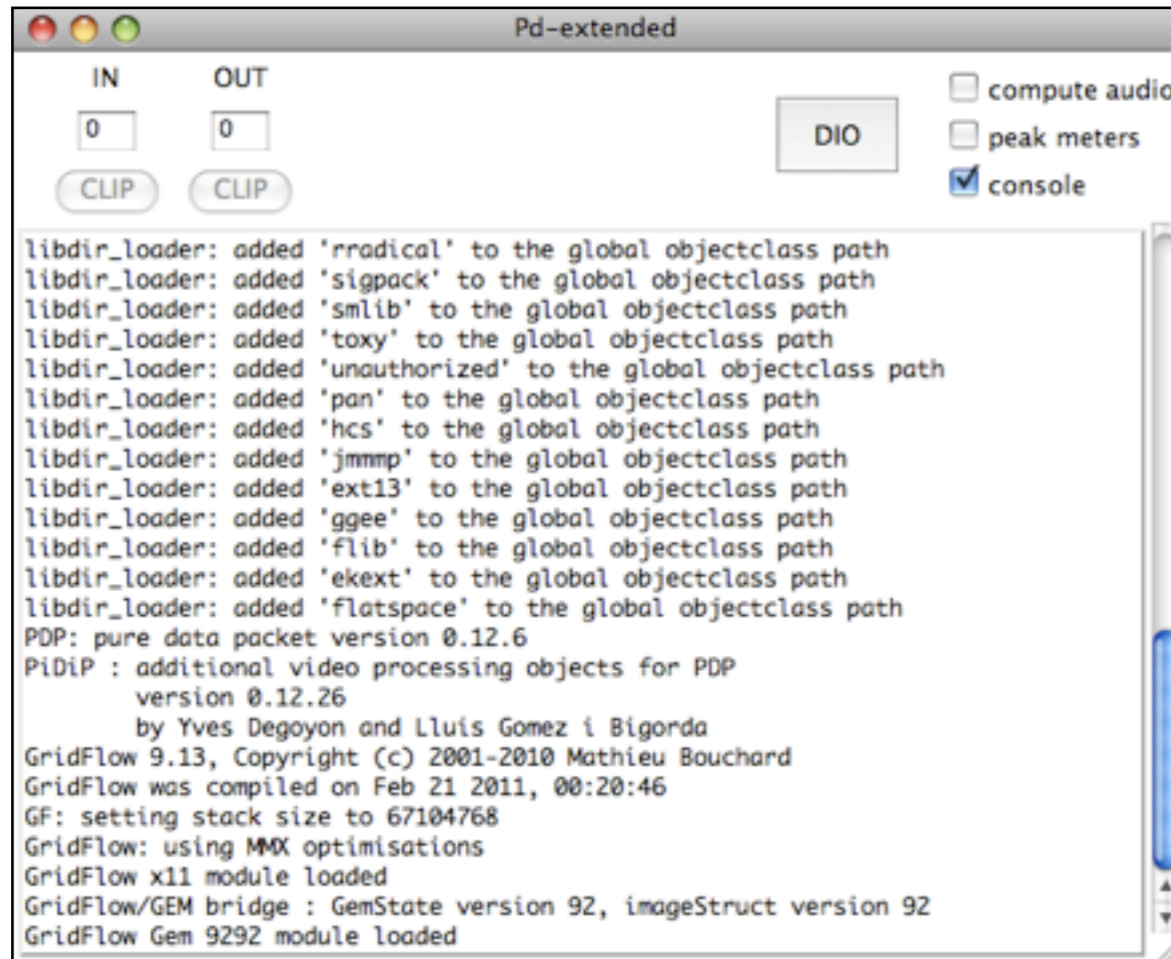
- Start PD
- File -> New

# Getting started

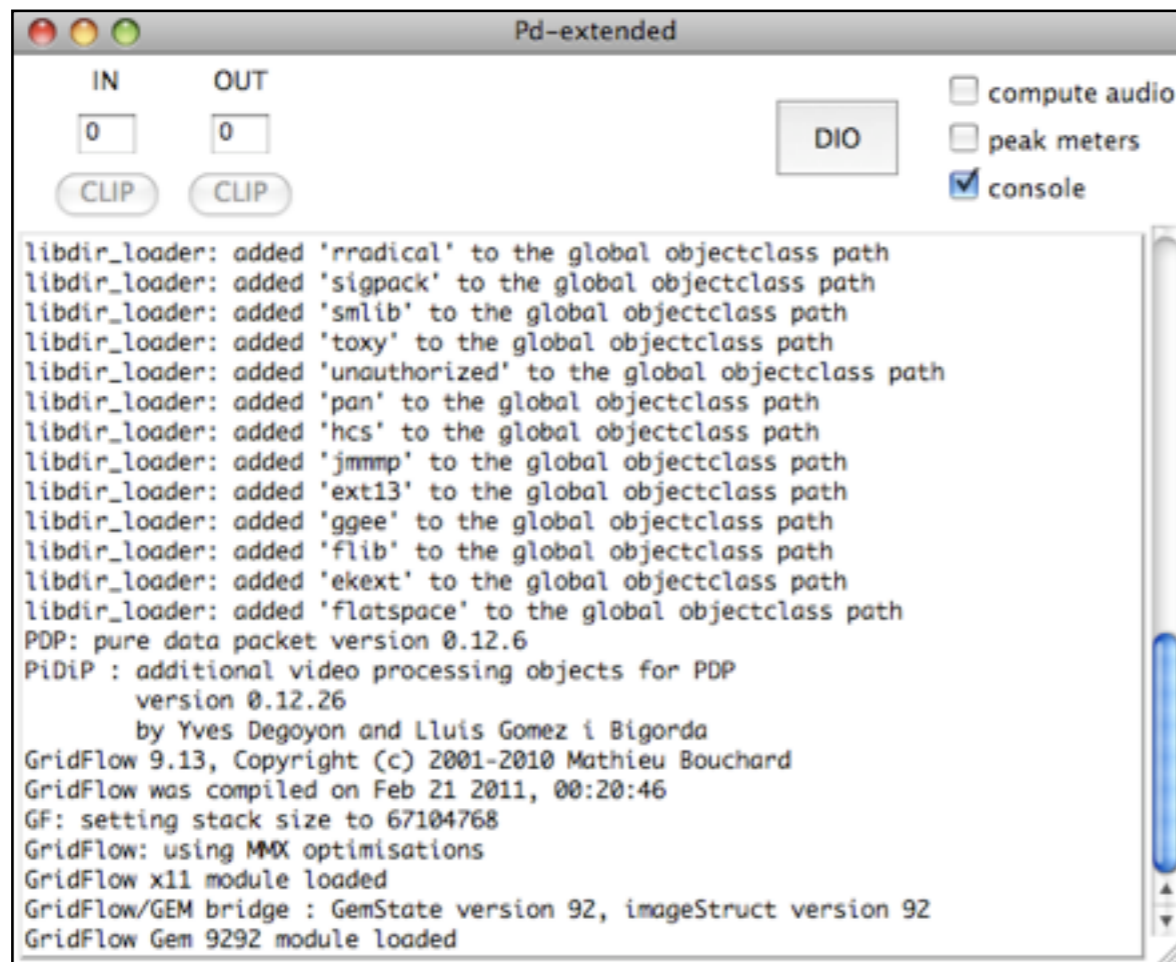




# Getting started

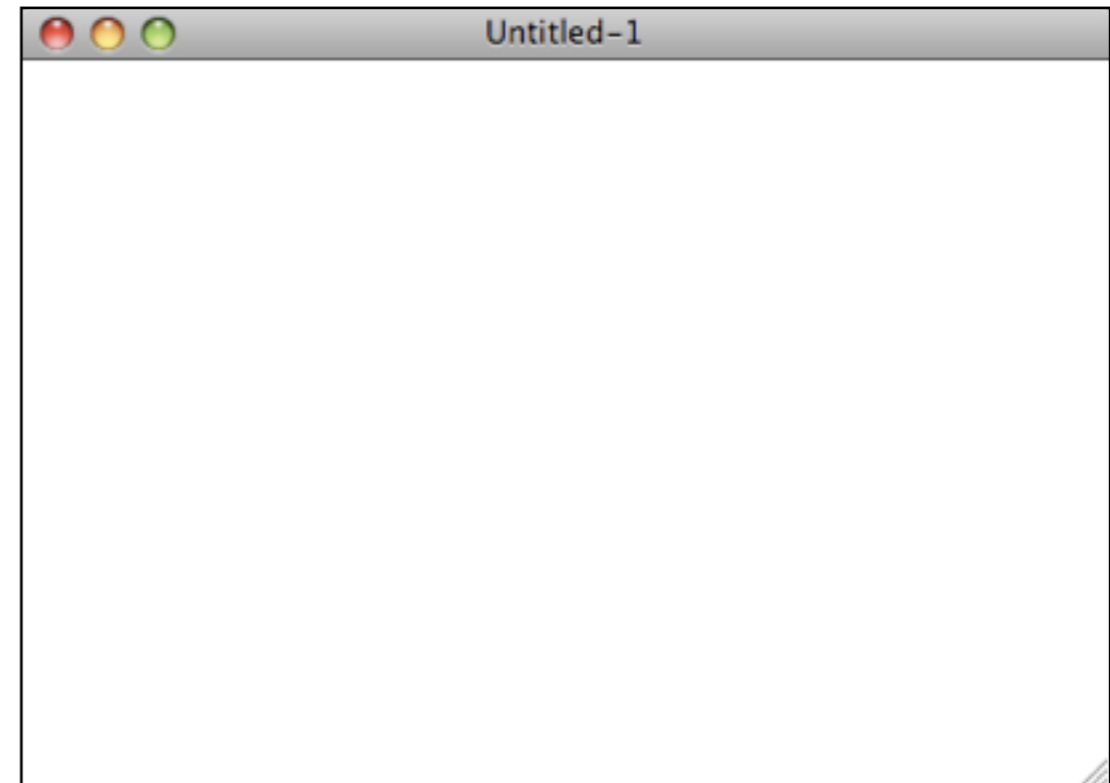


# Getting started



The screenshot shows the Pd-extended application window. At the top, there are two input fields labeled 'IN' and 'OUT', both containing the number '0'. Below them are two 'CLIP' buttons. To the right, there is a 'DIO' button and three checkboxes: 'compute audio' (unchecked), 'peak meters' (unchecked), and 'console' (checked). The main area of the window is a text area containing the following output:

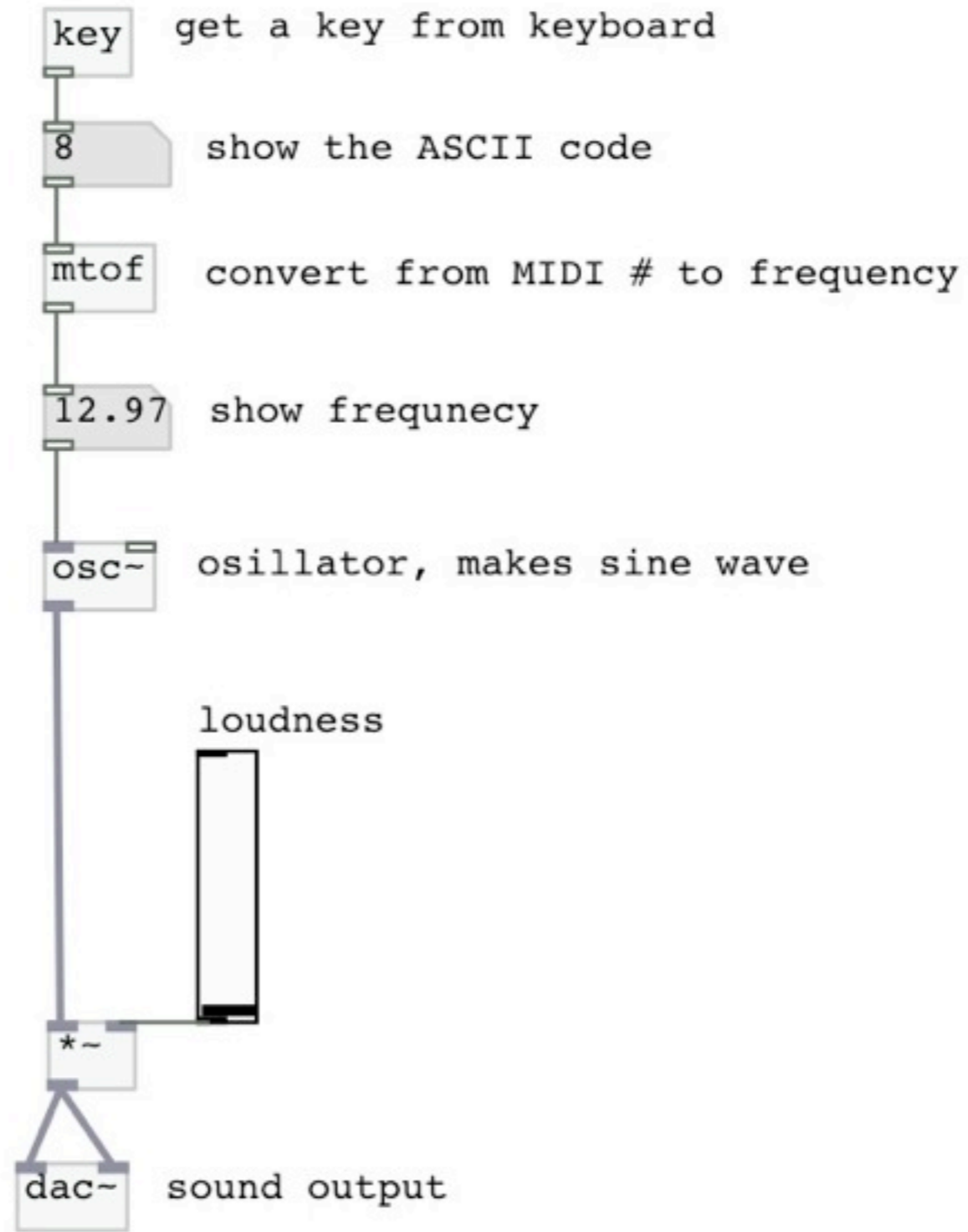
```
libdir_loader: added 'rradical' to the global objectclass path
libdir_loader: added 'sigpack' to the global objectclass path
libdir_loader: added 'smlib' to the global objectclass path
libdir_loader: added 'toxy' to the global objectclass path
libdir_loader: added 'unauthorized' to the global objectclass path
libdir_loader: added 'pan' to the global objectclass path
libdir_loader: added 'hcs' to the global objectclass path
libdir_loader: added 'jmmp' to the global objectclass path
libdir_loader: added 'ext13' to the global objectclass path
libdir_loader: added 'ggee' to the global objectclass path
libdir_loader: added 'flib' to the global objectclass path
libdir_loader: added 'ekext' to the global objectclass path
libdir_loader: added 'flatspace' to the global objectclass path
PDP: pure data packet version 0.12.6
PiDiP : additional video processing objects for PDP
        version 0.12.26
        by Yves Degoyon and Lluís Gomez i Bigorda
GridFlow 9.13, Copyright (c) 2001-2010 Mathieu Bouchard
GridFlow was compiled on Feb 21 2011, 00:20:46
GF: setting stack size to 67104768
GridFlow: using MMX optimisations
GridFlow x11 module loaded
GridFlow/GEM bridge : GemState version 92, imageStruct version 92
GridFlow Gem 9292 module loaded
```



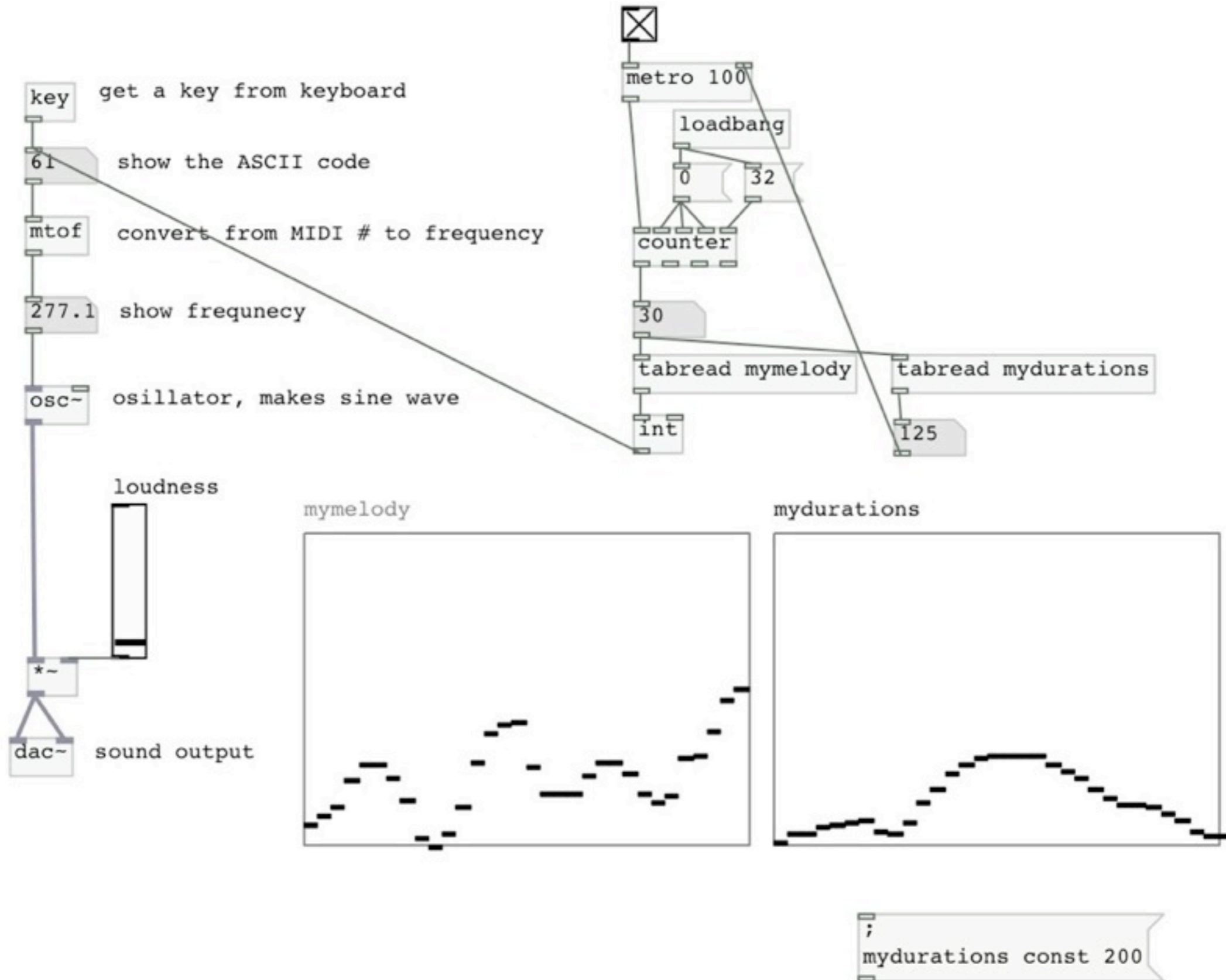
switch edit mode / run mode: **cmd + e**



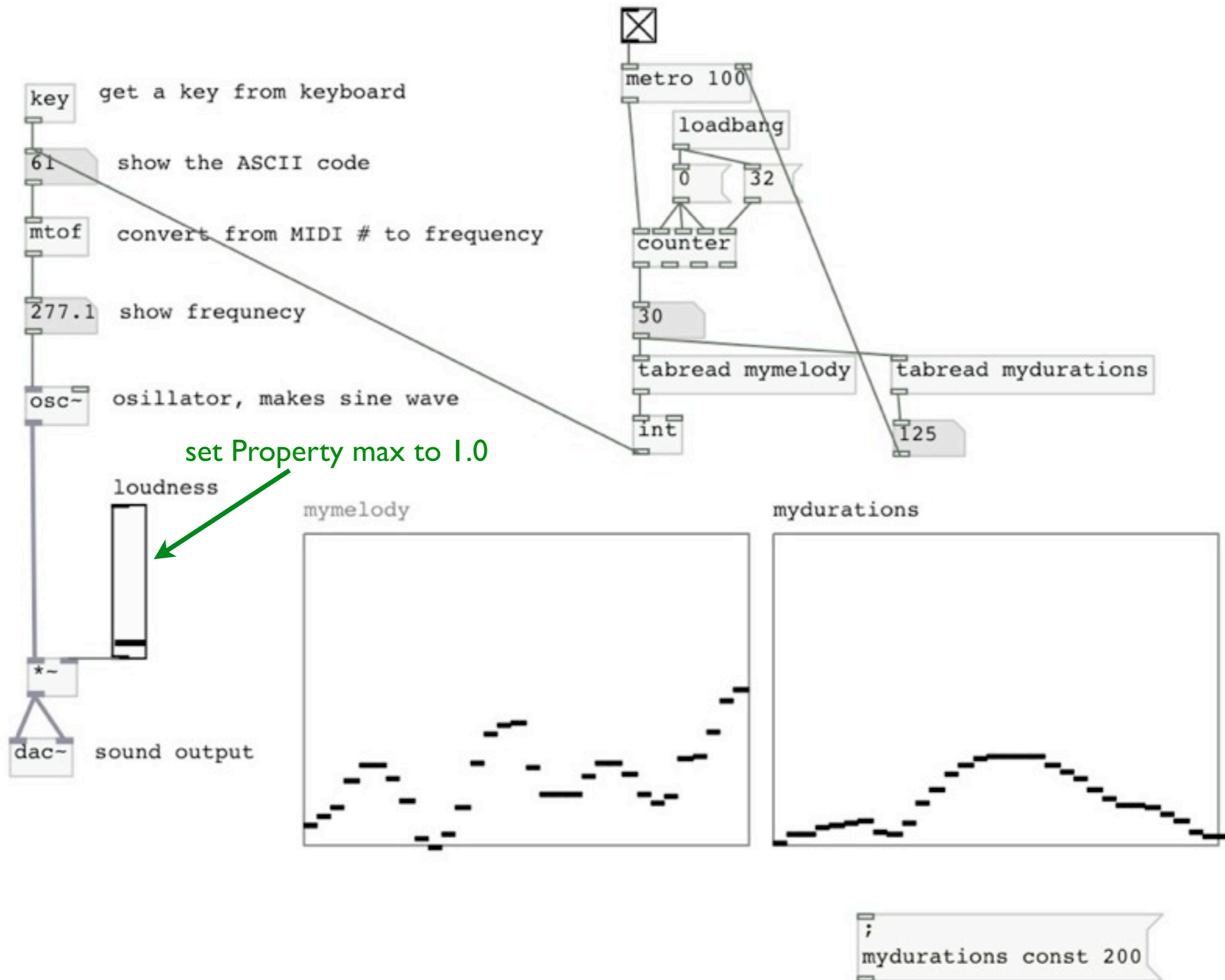
# Your first pd patch



# 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](http://www.idc.ul.ie/mikael/sounds/CAO-2012Examples.zip)

Questions?