pp●oll

a DSP environment for Max

project created by klaus filip

authored by c. hausch, noid & klaus filip

with development from several contributors:

klaus, hausch, noid, joe steccato, lewis kennedy, elin, paulo raposo,
leo dupleix, gus74v, oliver stotz, boris hauf, gilles aubry,
david michael, bill d., luc gross, antonio della marina,
taku unami, bill orcutt
& more

written in Cycling '74's Max

http://ppooll.klingt.org/

introduction

ppooll is an audio & video DSP (digital signal processing) environment for Max/MSP.

it's a modular network of Max patches with an internal mode of communication and graphical user interface.

it's a versatile, fully customizable toolkit that can facilitate audio manipulation, granular synthesis, live performance & improvisation, modular networking, ambisonics and more.

it can help you process signals & data in creative, dynamic, intuitive ways.

it's an open-format DSP playground employed by many experimental artists, including christian fennesz, tim hecker and taku unami.

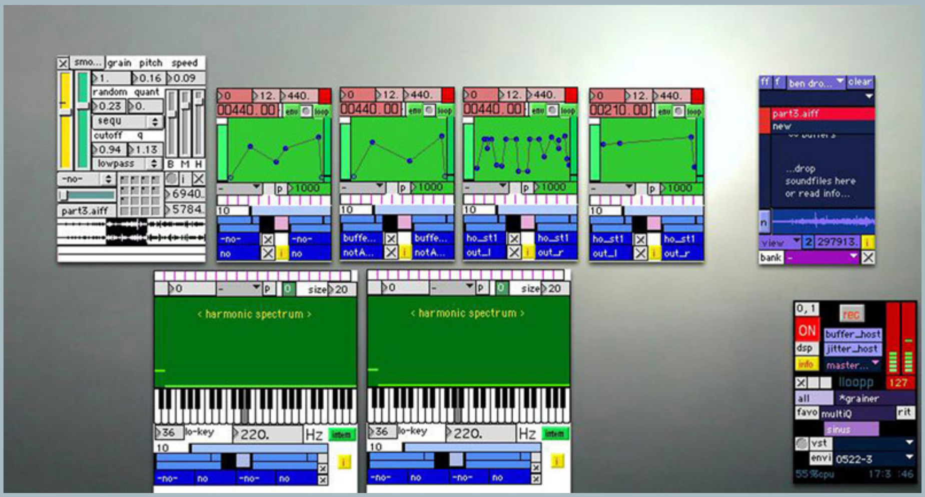
we welcome contributions and suggestions from all users and remain in a continuous state of development.

while this quickstart guide is aimed at new users, it's good to have some prior knowledge of DSP fundamentals and Max programming - but it's not essential, and won't impede you from having fun with ppooll.

ppooll is freeware, open-source, and copyleft.

a brief history

in its earliest incarnation, lloopp was authored by klaus as a sample-based looper patch with various plug-ins in 1998 for Mac OS 9.



the open architecture of the patch allowed other programmers to contribute with ease, resulting in a worldwide community of developers, helping lloopp to grow into a fully-fledged modular workstation with diverse functionality.

lloopp was subsequently ported to Mac OS X in 2003. since 2005, a revised hybrid version for Mac and Windows has been available.

from 2005, the project has been redirected as ppooll. our focus is now upon ease-of-use, working in the multichannel domain, extended development, and compatibility with modern software protocols.

installation & prerequisites

ppooll is written in Cycling 74’s Max and requires it to run.

you can download the latest version of Max here:

> https://cycling74.com/products/max

ppooll also has dependencies upon the following Max packages.
you must install them via the package manager.

> CNMAT externals
>
> cv.jit
>
> ICST ambisonics
>
> jasch objects
>
> karma
>
> link
>
> lowkeyNW
>
> PeRColate

v e r s i o n s

following Max installation, you can download the ppooll distribution packages directly from our website.

to allow our developers and users to network and contribute efficiently, we’ve recently begun hosting a development version on GitHub.

older versions of can also be found on our website, but please note that these are no longer supported. however, they may be useful for legacy systems.

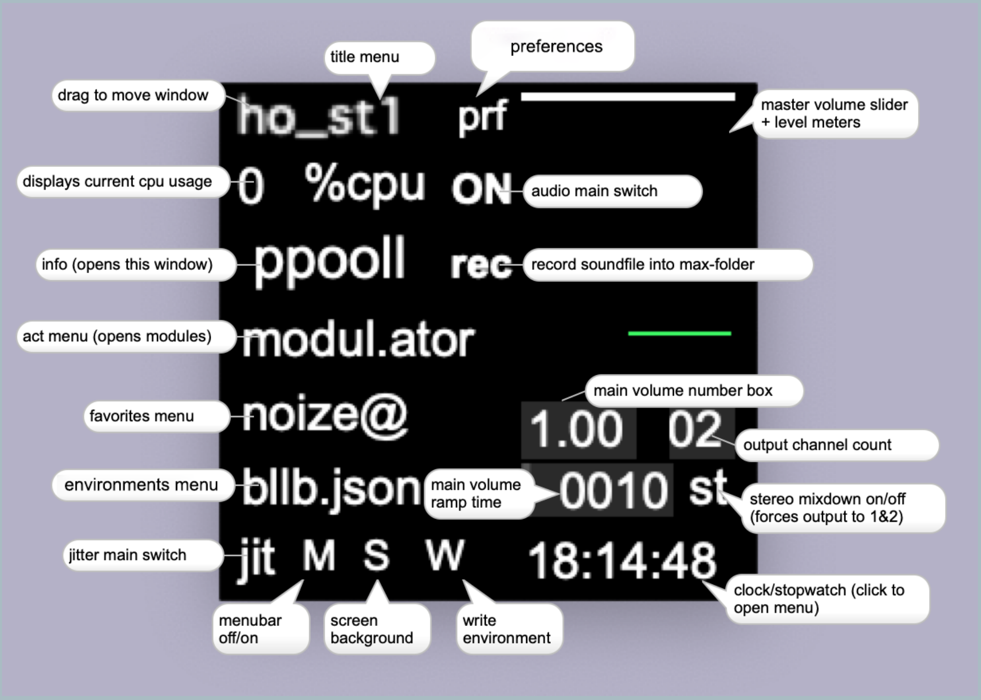
development (v.8.5, macOS / Windows)
extract & place in ~/Users/Documents/Max 8/Packages

stable (v.8.0, macOS)
extract & place in ~/Users/Documents/Max 8/Library

DSP settings

after launching Max, it’s best to configure your DSP settings to work with your soundcard of choice and route I/O correctly. find this under file > audio.

ho_st



first, load ppooll_host from the extras menu in Max.

ho_st is the beating heart of ppooll.

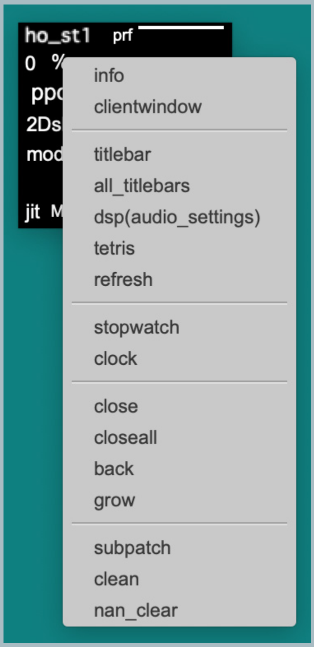
in older versions, ho_st may only be found within ppooll’s installation folder, and may look a little different.

navigation & submenus

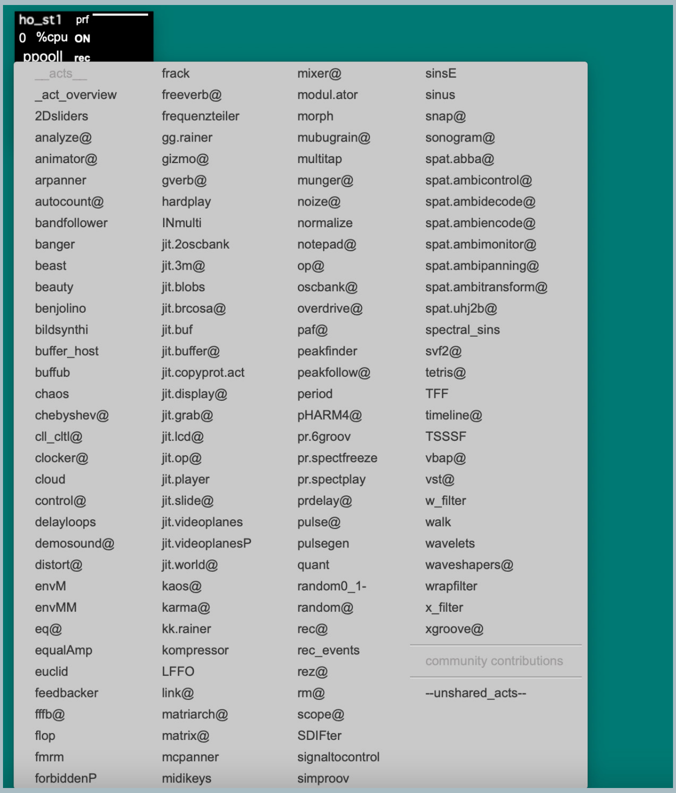
clicking and dragging to the left of an act's name will move it.

clicking to the right of its name will open a submenu of options.

some of these options will make more sense as you continue to learn ppooll.



act loading



modules within ppooll are called acts (@).

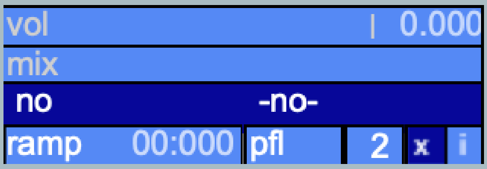
they can be loaded from the first drop-down menu of ho_st.

detailed information on acts can be found by the 'info' section within each act's submenu. shortcuts to favourite acts can be saved in the second drop-down menu of ho_st.

routing / ll.blues

audio routing within each patch is performed by ll.blues, the section of volume faders.

here, you'll find options to select a number of channels, control individual amplitudes, adjust PFL levels, and send & recieve source/ destination audio.

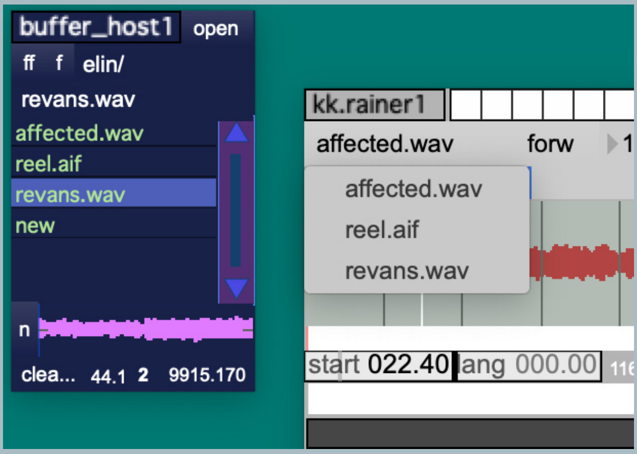


buffers / samples

the buffer_host window will automatically load whenever you open an act which deals with buffers (i.e. samples).

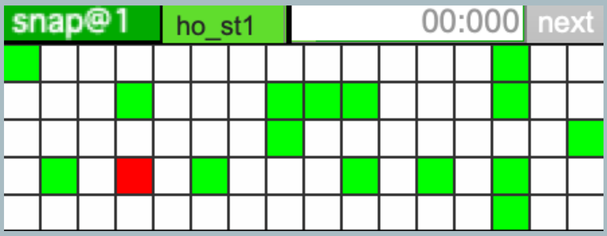
here, you can drag & drop audio samples or route the act's submenus to your sample directories.

once you've loaded a buffer into an act, you must select it from the given act's drop-down menu too.



presets & ramps

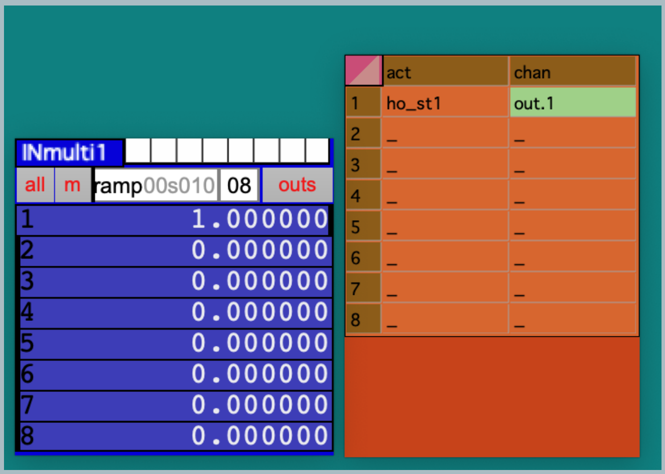
you can save and recall presets with the preset box in each act.



left-click to save a preset. cmd+click to erase a preset. you can also hold, click and drag to interpolate between presets.

the number box in each act near the preset box will determine ramp (i.e. transition) time between presets.

live input



1	1.000000
2	0.000000
3	0.000000
4	0.000000
5	0.000000
6	0.000000
7	0.000000
8	0.000000

	act	chan
1	ho_st1	out.1
2	_	_
3	_	_
4	_	_
5	_	_
6	_	_
7	_	_
8	_	_

live audio can be captured and routed by the INa / INmulti act.

you must define the routing in the ‘outs’ section of this act.

VST / AU

VST and AU plugins are supported by way of the VST act (VST@).

you can route VST@ to your plugin directories in the drop-down menu within the act.

save / recall

you can write your environment with the W button in ho_st. in older versions, select ‘write’ in the third-drop down menu in ho_st.

when working with presets, you must save individual act states by option-clicking in the actmenu and selecting ‘write’. this will allow ppooll to fully recall your environment’s settings.

it’s essential to do this with buffer_host, too.

you can recall saved environments from the third drop-down menu in ho_st.

recording

the rec function within ho_st will capture live audio from ppooll's master audio (i.e. all sources routed to ho_st), and place it within the ppooll directory in your Max folder.

you can define an alternative destination within the preferences menu in ho_st.

you can also select ST in ho_st to ensure a stereo mixdown. otherwise, your file will be rendered in multichannel audio, and may be unreadable by many common applications.

alternatively, you can use Max's quickrecord function in the extras menu, or the rec@ act to define further parameters.

recording multitrack audio out of ppooll is possible, too. you'll need to install a virtual audio router like BlackHole or Soundflower, and further configure ll.blues' light blue section to route to your DAW from individual acts.

act directory

2Dsliders	control parameters by dragging points
INmulti	(multiple) audio input
LFFO	dynamic ring modulator
SDIFter	SDIF soundfile player
TFF	4 band resonant filter
TSSSF	substractive synthesis
analyze@	loudness, brightness of audio
animator@	multiband modulation
arpanner	audiorate panner
autocount@	number generator
bandfollower	generate loudness data from filters
banger	send synchronized bangs
beast	non-trivial buffer machine
beauty	delay-feedback machine
benjolino	shift-register oscillator (rungler)
bildsynthi	video driven bandpass filter
buffer_host	sample control
buffub	records into buffers
chaos	lorenz-roessler generator
chebyshev@	distortion unit
cll_cltl@	ondomusic noise
clocker@	event sequencer
cloud	oscillator bank with pitch distribution system
control@	external device input (midi-osc-etc)
delayloops	3 delay lines
demosound@	cycling 74's demosound to ppooll
distort@	degrade sample-rate & bit-resolution
envM	mc version of multiple envelopes
envMM	multiple envelopes
eq@	crossover-filter based graphic equalizer
euclid	LFO with Euclidean sequencer
feedbacker	feedback generator for audio-inputs
fffb@	filter bank
flop	sample looper
fmrm	fm synthesis & ring modulator
forbiddenP	spectral filter or vocoder
frack	record parameter movements
freeverb@	reverb
frequenzteiler	trautonium synthesizer
gg.rainer	granular sample player
gizmo@	pitch shifter
gverb@	reverb plugin

hardplay	plays soundfiles from hd or cd
jit.2oscbank	video to oscilator bank
jit.3m©	cheap image analyser
jit.blobs	outputs a list with blobs tracked in an image
jit.brcosa@	video brcosa settings
jit.buf	store images and play (textures)
jit.buffer@	store images and play later
jit.copyprot.act	grab video from screen
jit.display@	video screening and recording
jit.grab@	camera input
jit.lcd©	draft drawing
jit.op@	image operater
jit.player	qt movie player
jit.slide@	slide and reposition incoming texture
jit.videoplanes	mix and position video
jit.videoplanesP	mix and position video (list-version)
jit.world©	texture host to movie rec
kaos@	random mouse clicks
karma@	varispeed audio looper
kk.rainer	granular sample player
kompressor	audio compressor
link@	ableton link sync interface
matriarch@	audio matrix on steroids
matrix@	audio matrix
mcpanner	simple random panner for mc signals
midikeys	midi keyboard parser
mixer@	audio mixer
modul.ator	modulates anything
morph	convolution act
mubugrain@	granular player
multitap	delay bank
munger@	live granulator
noize@	noise generator
normalize	get maximum level of audio
notepad@	write something to yourself
op@	signal/number operator
oscbank@	multiple sinus generator
overdrive@	audio overdrive
pHARM4@	4 band harmonizer
paf@	phase aligned formant synthesizer
peakfinder	dynamic gate
peakfollow@	envelope follower

period	signal-based step sequencer
pr.6groov	multiple sample player
pr.spectfreeze	spectrum freezer
pr.spectplay	spectrum player
prdelay@	simple delay with feedback
pulse@	lfo pulse generator
pulsegen	pulse wave generator
quant	signal based frequency/rhythm quantizer
random0_1-	simple randomizer for 0,1 output
random@	randomize parameters
rec@	record to harddisc
rec_events	records parameter events
rez@	spectral resonators
rm@	ring modulator
scope@	view audio signal
signaltocontrol	signals to control@
simproov	simple 4 fold sample player
sinsE	sinus bank with envelopes
sinus	sinus tone generator
snap@	snapshot all parameters as preset
sonogram@	audio signal viewer
spat.abba@	ambisonics a-to-b/b-to-a format converter
spat.ambicontrol@	ambimonitor controller
spat.ambidecode@	ambisonics b-format decoder
spat.ambiencode©	ambisonics b-format encoder
spat.ambimonitor©	monitor for ambisonics encoder
spat.ambipanning©	ambisonics panner
spat.ambitransform@	ambisonics soundfield transform
spat.uhj2b@	ambisonics uhj-to-b format converter
spectral_sins	sinewaves following incoming audio
svf2@	cutoff filter
tetris@	customize your act layout (and act-building)
timeline@	graphical timeline sequencer for parameters
vbap@	multi-speaker-spat or plugin-router
vst@	host for vst plugins
wrapfilter	n-band filter/eq
walk	random walk a parameter
wavelets	time based oscilator
waveshapers@	waveshaping functions and demos
wrapfilter	1 - 4 band stereo filter/eq
x_filter	cheyshev & butterworth filter
xgroove@	sample player

actmaking

ppooll's open architecture allows you to create your own acts - i.e. Max patches that include ppooll UI, routing & other elements.

you can find a brief guide to actmaking by clicking 'ppooll' in ho_st.

if the patch you're porting is not your own work, try to clear permissions with the original author before you share your act and upload to our GitHub.

you can also watch Klaus in a livestream offering extensive advice on the process.

here is a demonstration patch (courtesy of KNFLD) to get you started.

styleguide

- use ll_number objects as ui where possible
- use ppooll preset field instead of max presets
- put all processing into subpatch where possible
- use ll.r, ll.p or ll.mc.r~ receives
- make patches as cable-less as possible
- make patches mc-ready if possible
- choose a unique name for act to avoid conflict with existing externals / patchers
- try to save screen real state by making patcher window reasonably small
- (tip: tetris helps with organizing ui)
- keep the ui clean (look out for orphaned elements in the back)
- group elements by color
- create tetris default layout
- incorporate ll.syncs to provide tempo syncing across ppooll and external apps

FAQ

why does ppooll crash Max sometimes?

the complex nature of ppooll - i.e. several Max patches running simultaneously - means that you'll occasionally run into system errors. system status can be monitored via the Max console. it's good to keep an eye on this every once in a while, and report bugs back to us.

why do i get a quarantine error on macOS upon load?

third party max externals can trigger system security notifications. here's a workaround.

how can i route audio to multiple acts simultaneously?

both matrix@ and matriarch@ facilitate complex routing for a virtually unlimited number of source/destination channels.

how do i install custom user acts?

you can now find these on the GitHub repository.
manually, you'll need to ensure any abstractions are placed in
~/ppooll/abstractions
while the act itself must be placed in
~/ppooll/patchers/ppooll.acts

can i use ppooll without Max?

unfortunately not, but we're now beta testing live.ppooll within Ableton Live.
you can find the .amxd M4L device within the latest version's .zip folder.

community

join our mailing list:

lloopp-subscribe@klingt.org
write ‘subscribe lloopp’ in message body

since 2022, we’ve hosted a Discord server, where you can chat with developers,
seek technical assistance, report bugs, share acts, contribute, discuss
and explore ppooll together.

we also occasionally compile recordings from the community over on Bandcamp.
you can make submissions to these compilations via the server.

contact

ppooll.dsp@icloud.com

hausch@moozak.org

klaus@klingt.org