# Why Pd?

Ángel Faraldo

# Computer Music Systems

# Non-realtime systems: Music-N & Csound

- Max Matthews MUSIC-N languages (1957…)
- Orchestra vs. score metaphor
- UGens…
- Audio vs. control rate

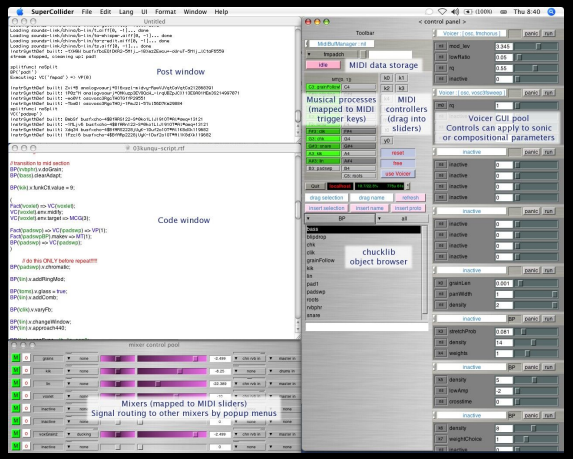
```
<CsoundSynthesizer>
<CsInstruments>

instr 1
aOut vco2 1, p4
out aOut
endin

</CsInstruments>
<CsScore>
i1 0 1 100
i1 1 1 200
i1 2 1 300
</CsScore>
</CsoundSynthesizer>
```

# Realtime systems: Supercollider

- By James McCarthy (1996)
- Server/Client interpreted model
- Real-time Synthesis for algorithmic composition and live coding
- Extremely extensible and customisable
- Supports basic GUI development via Qt
- Live-coding & algorave ‘standard’

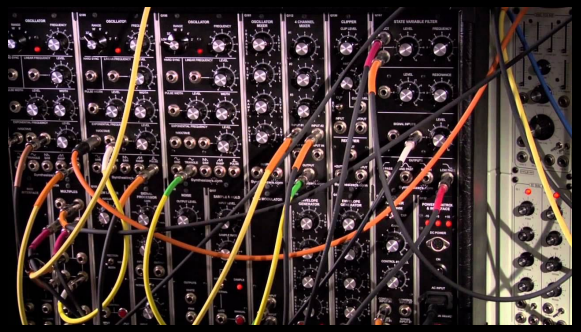


# Realtime systems: ChucK

- Ge Wang and Perry Cook (2003)
- Command-line based
- Emphasis on real-time, live coding and algorithmic composition
  - *“Strongly timed and on-the-fly language”*
  - Sample-by-sample processing
  - As realtime as it can get!
- Used by the Princeton Laptop Orchestra

# Graphical Programming Languages

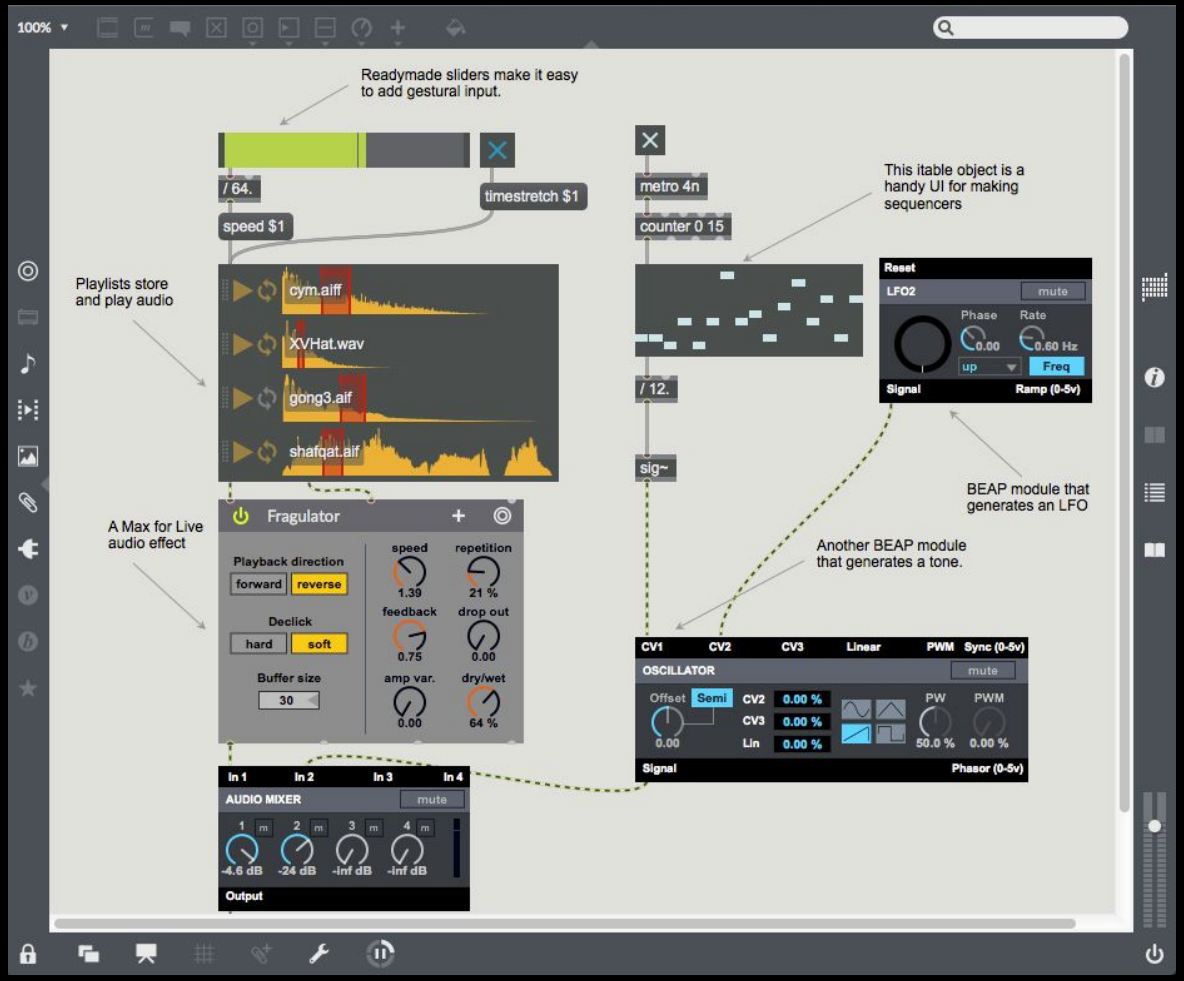
- Inspired by modular synthesisers and circuit design
- Patch cables carry data/audio to/from various modules
- Good and fast prototyping environment
- Realtime and I/O processing
- Edit vs. performance mode: the program is the instrument
- Easy to understand what’s going on for non-programmers



# Graphical Programming Languages

- Max (Puckette, IRCAM, 1980's)
  - Originally data only, controlling external synthesis equipment
- Pure Data [Pd] (Puckette, 1996)
  - Open source
  - Added audio processing, soon to be incorporated in Max
- Max/MSP (Max Signal Processing, 1997)
  - Developed and comerciallised by David Zicarelli and Cycling '74
  - Now simply called Max

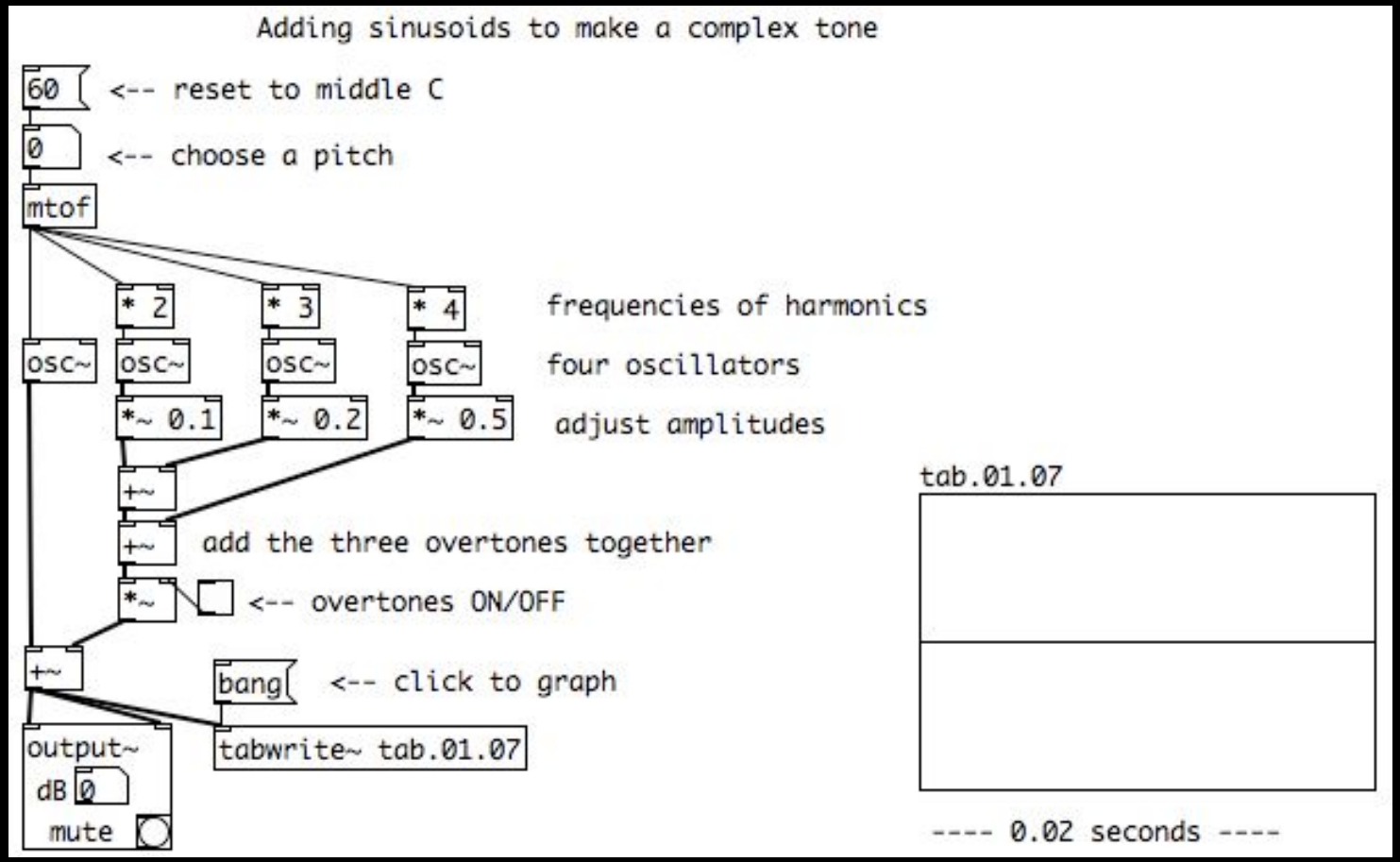
# Max/MSP


Readymade sliders make it easy to add gestural input.
/ 64.
timestretch $1
speed $1
metro 4n
counter 0 15
This itable object is a handy UI for making sequencers
Playlists store and play audio
cym.aiff
XVHat.wav
gong3.aif
shafqat.aif
/ 12.
sig~
Reset
LFO2
mute
Phase
Rate
0.00
0.60 Hz
up
Freq
Signal
Ramp (0-5v)
BEAP module that generates an LFO
A Max for Live audio effect
Fragulator
Playback direction
forward
reverse
speed
repetition
1.39
21 %
Declick
hard
soft
feedback
drop out
0.75
0.00
Buffer size
30
amp var.
dry/wet
0.00
64 %
Another BEAP module that generates a tone.
CV1
CV2
CV3
Linear
PWM
Sync (0-5v)
OSCILLATOR
mute
Offset
Semi
CV2
0.00 %
CV3
0.00 %
Lin
0.00 %
PW
PWM
50.0 %
0.00 %
0.00
Signal
Phasor (0-5v)
In 1
In 2
In 3
In 4
AUDIO MIXER
mute
-4.6 dB
-24 dB
-inf dB
-inf dB
Output

# Pros & Cons: Max/MSP

- Pros:
  - Mature, well supported with well written documentation
  - Many examples and resources (sounds, patches etc.)
  - Modern interface
  - Ableton Live integration (MaxForLive)
- Cons:
  - Expensive (although there is student licensing)
  - Not portable (only runs on Max and Windows)

# Pure Data

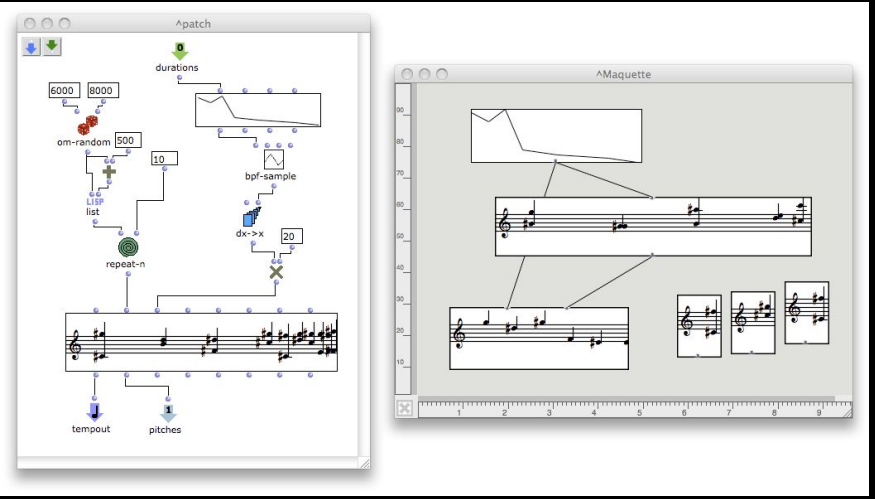

Adding sinusoids to make a complex tone
60
<-- reset to middle C
0
<-- choose a pitch
mtof
* 2
* 3
* 4
frequencies of harmonics
osc~
osc~
osc~
osc~
four oscillators
*~ 0.1
*~ 0.2
*~ 0.5
adjust amplitudes
+~
+~
add the three overtones together
*~
<-- overtones ON/OFF
+~
bang
<-- click to graph
output~
dB 0
mute
tabwrite~ tab.01.07
tab.01.07
---- 0.02 seconds ----

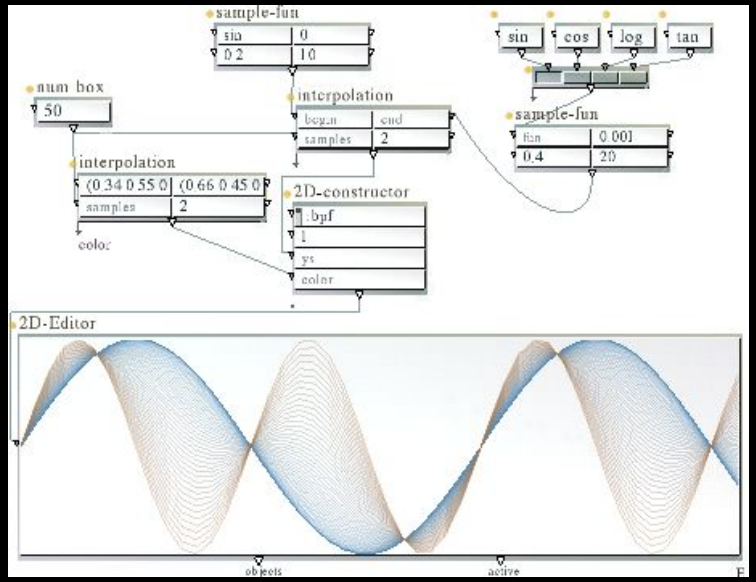
# Pros & Cons: Pd

- Pros:
  - Open-source and free
  - Portable: runs on nearly everything (Mac, Win, Linux, iOS, Android, raspberry pi, web)
  - Small language - good for learning
- Cons:
  - Documentation can be lacking
  - Comunitty-based support (not really defined)
  - Ugly interface (?)
  - Small language: Some things are hard to do in Pd

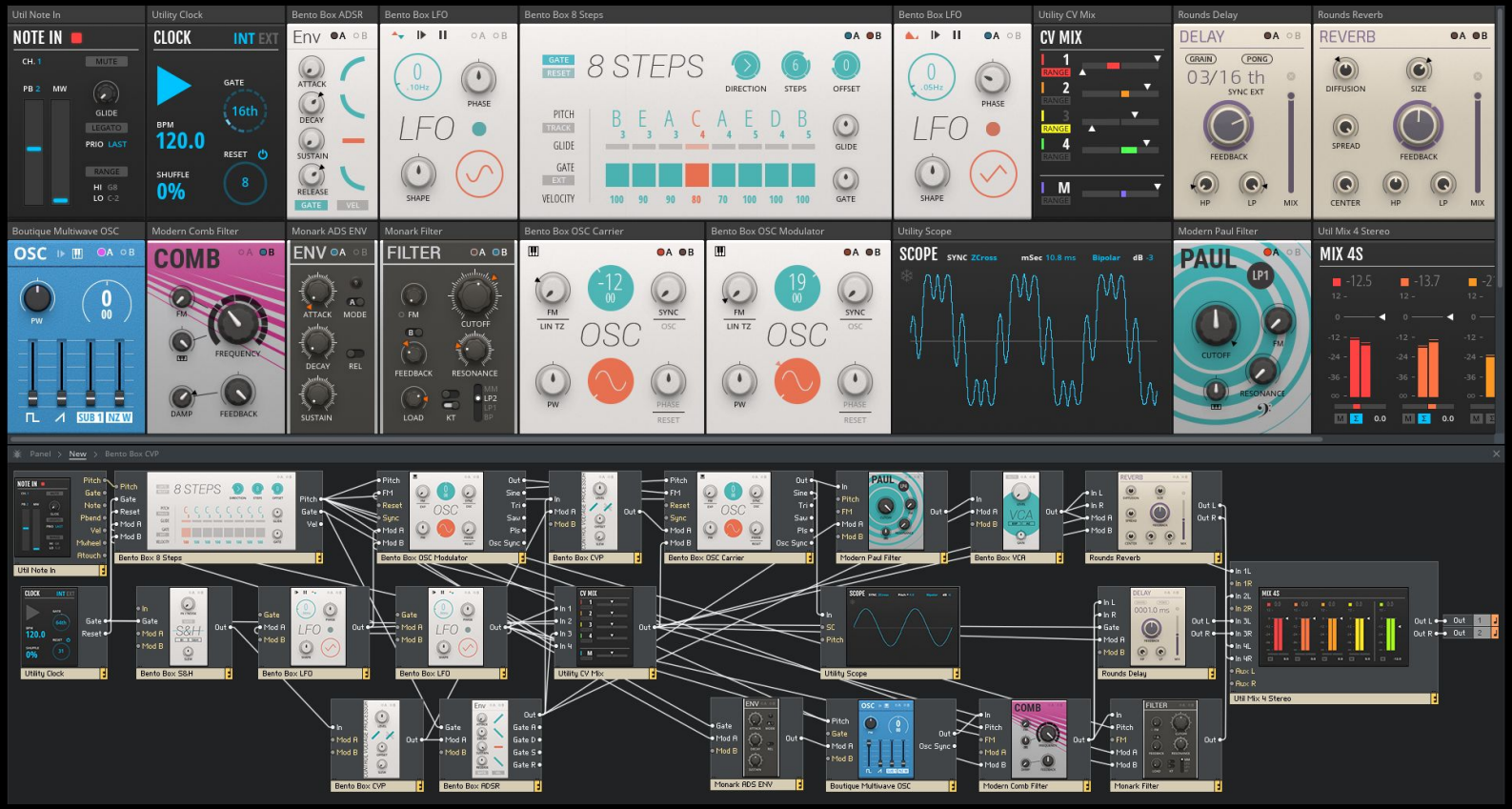
# Patchwork and Open Music

## Concerned with Computer-Assisted Composition



# Reaktor (Native Instruments)



# Pd and the World: Expanding, embedding and controlling Pd

# Interacting with Pd

- MIDI I/O
  - Native pd objects supporting all the MIDI protocol
- HID Devices
- OSC
  - udp package transmission system
  - Useful for distributed computing, mobiles, hardware
- Arduino
  - External objects to interact/write to/from arduino

# Interacting with Pd: MIDI I/O



# Interacting with Pd: Hardware controllers



# Interacting with Pd: MIDI I/O

- Native Pd objects supporting the MIDI protocol
- Protocol recognised by ALL major manufacturers
- Useful for more “musical” applications
- “Conservative” metaphors:
  - pitch, keys, velocity, panning, volume, instruments…
- Loads of hardware controllers available
  - keyboards, drumpads, faderboxes, wind-instruments…
- Useful inter-application data-exchange method

# Interacting with Pd: Hardware controllers

- MIDI I/O (example)
- Video Cameras (GEM) (example)
- Smartphones / tablets
  - As ‘controllers’ via Open Sound Control (e.g. TouchOSC)
    - UDP package transmission system
    - Useful for distributed computing, mobiles, hardware
- Motion tracking (Wii, Leap, Kinnect) (ex1) (ex2) (ex3)
  - There used to be pd interface objects to these
- Gaming devices (Joysticks, Gamepads, etc.) (ex1) (hid)
- Custom sensor systems (Arduino) (tutorial)

# Using Pd in/from pro-audio and DAW’s

- External plugins in Pd
  - [vstplugin~] can load VST effects inside Pd
  - [plugin~] can load LADSPA effects inside Pd
- Pd as VST's
  - Camomille
  - Pd Pulp

# Interacting with Pd: Embedding

- libpd
  - Pd “Vanilla” only
  - Pd and Mobile devices (android, iOS)
  - Pd as audio engine within other languages or the web
- MobMuPlat (libpd)
  - Simple front end to run Pd in mobile devices
  - With programmable GUI and access to device’s sensors
- Camomille (libpd)
  - Wrapper to use Pd as a VST plugin
- Webpd (very limited)
  - Run Pd Patches on the Web with JavaScript and Web Audio API.
- Raspberry Pi distributions

# Pd & other programming languages (1)

- Pd and MAX
  - Almost identical languages
  - Simple Pd patches can run in Max (at least on earlier versions)
  - Max has more and better documentation, expanded functionality
  - But Max is commercial and non-embeddable!
  - Pd's "cyclone" and "maxlib" externals try to bridge the gap between both
- Pd and Processing
  - Communication via OSC
  - Split sound and image processing
- Pd and Open Frameworks
  - With OFelia external

# Pd & other programming languages (2)

- Pd and Python
  - Pd inside Python (through libpd) pyata (demo)
  - Python inside Pd (py/pyext - scripting objects for Pure Data and Max)
  - OSC intercommunication (pyOSC is compatible with 2.x and 3.x)
- Pd and low-level
  - Programming externals in C/C++
  - FAUST
    - Functional language for sound synthesis and audio processing
    - Easy to wrap low-level signal processing as pd objects
    - [faustgen~] = the FAUST compiler embedded in a Pd external

# Choosing and installing your Pd flavour

# Pd Flavours

- Pd vanilla (up-to-date)
  - Core language
  - Embeddable
- Pd-lork / Purr data (up-to-date)
  - With a lot of external libraries built-in
  - Refurbished GUI in javascript
- Pd-extended (outdated)
  - Useful for working with GEM (image processing)

# Pd extensions

- External libraries
  - Extended or specialised functionality
  - Installable via deken in new Pd versions
  - Spatialisation (binaural, ambisonics, VBAP, HOA), spectral Analysis (timbreID), visuals and image processing (GEM, OFelia), algorithmic composition...
  - Not part of the core language == **not embeddable!**
- GUI plugins (definition) (github)
  - Modify and add GUI functionality (cable deletion, contextual help, etc.)

# Download and setup Pd

# Download and Setup Pd

- Download and install (https://puredata.info/downloads/pure-data)
- Configure audio (Media > Audio Settings)
- Test configuration (Media > Test Audio and Midi)
- Compute audio (in the terminal window, with cmd/ctl-/,  or in Media > DSP On)

# Help

- Help menu
- Help on objects (right-click on an object)
- List of objects (right-click on empty canvas)

# Externals

- Download the following externals via deken (Help > Find externals)
  - Zexy (advanced audio capabilities)
  - Ceammc (improved GUI objects)
  - Cyclone (useful objects to work with lists)
  - timbreID (timbre identification and classification)
- Declare external libraries via
  - Startup menu (Preferences > Startup) (zexy, timbreID)
  - Declare object [declare -stdlib zexy]
  - Object namespace [cyclone/col]

# GUI Plugins

Download GUI plugins from deken:

- Completion plugin
- Dnd-plugin (drag and drop)
- Doublechord plugin