CRT – Composition in Real Time

Patch by Tom Mays
(contact@tommays.net)
2015-2019

Originally made available to composition students at the Strasbourg Conservatory, and to students of the 2018 Academy of Composition – Philippe Manoury / Musica, Strasbourg.

Runs in Max8 on Mac OSX and on Windows. See Cycling74 website (https://cycling74.com) for minimal hardware and OS requirements.

Includes some external objects from IRCAM, Paris – including elements of Spat (ircamverb, spat.pan, spat.viewer) and Max Sound Box (psych~, psychoirtrist~, yin~).

Also, makes use of software developed at the Center for New Music and Audio Technologies (CNMAT) at the University of California, Berkeley (list-interpolate); and from the PeRColate collection by Dan Trueman, modified by Ivica Ico Bukvic et Ji-Sun Kim (munger~).

Special thanks to composers Philippe Manoury and Daniel D'Adamo for suggestions and feedback.

Current CRT version 0.21, 14 August 2019

Patch philosophy

CRT is a Max patch, running best on the most recent version of Max 8. The patch is saved in the form of a *maxproject*, so all necessary files are included.

CRT provides a basic architecture and a set of simple modules for audio input/output, soundfile playback, delay, amplitude LFO, filters, frequency shifters, harmonizers, reverbs, some basic FFT synthesis and a granular motor – all passing through a powerful centralized audio matrix allowing interconnections among all modules (including a module back to itself). All modules contain a switch parameter (sw) which mutes the module when off so that only active modules utilize CPU.

All module parameters are a part of a message-based system for creating *events*, storing the parameters of all active (sw = 1) modules in a "traditional" but very effective *qlist* text file. Once an *event* is created and filled with an initial parameter state, it can be modified to include time-variant values (ramp functions with *line* or *line~*) or time-delayed messages forming a sequence within the same *event*.

CRT is not intended to include EVERYTHING you might need for a real-time computer environment, but to provide a solid framework to build on. The user is expected to be familiar with Max which she/he will need in order to make more interesting interactive relationships between the instruments and the electronics. For example, the user might program an algorithmic player in order to modify existing CRT parameters, or synthesized or sampled sounds. A user may want to incorporate a custom audio processing patch, which they can easily tie into CRT by using the auxiliary send/receive in order to benefit from the central matrix and spatialization.

Every module has access to spatialization including reverb, configurable from 4 to 8 loudspeakers. These are provided by the use of Ircam objects spat.pan and ircamverb~. All positions are determined by azimuth and distance independent of speaker configuration. This means that one can easily work in a 4-speaker setup and then move to 7 or 8 speaker configurations for the concert.

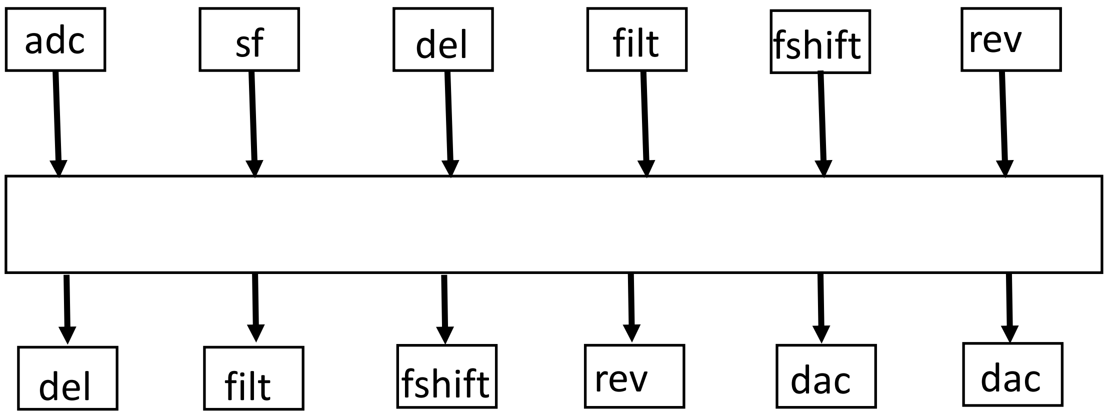
The patch contains a fixed number of each module, but it is relatively simple to add more of a certain module, or else to delete unused ones to lighten the patch.

Since working with CRT involves modifying the contents of the maxproject folder, always keep an unmodified copy of the CRT project folder at hand in order to start a new project.

Architecture

The *nerve center* of the patch is the audio matrix. In this simplified diagram, we can see that *sources* such as audio input (adc) and soundfile player output into the matrix, as well as audio processing modules such as delays, filters, frequency shifters and reverbs. The outputs of the matrix in return go to the inputs of all of the audio processing modules, but also to the dac outputs which correspond to the audio interface and the speakers. In this way, any interconnection of modules can be achieved, even sending a module's output to its own input! These connections can be made in a graduated dB scale, and even ramped over time.

Independent spatialization of each module is achieved via the matrix as well, with each module sending the proper commands to the matrix so that its pan position and reverb level are set using a simple panning interface for the user.



Current module list

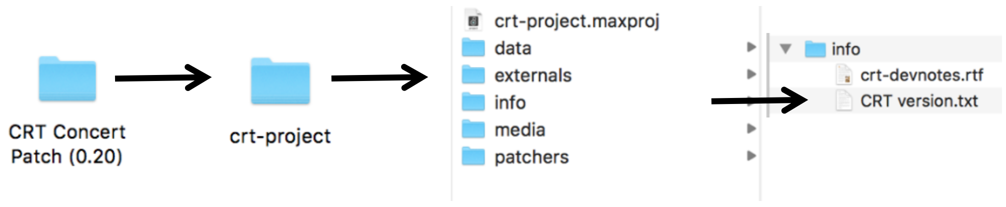
This is the list of all modules pre-installed in this version of CRT, with the number of each :

1 - mtrx – central audio matrix for all modules
2 - adc – audio interface input (adc~)
2 - src – noise and tone source (noise~ and cycle~)
4 - sf – mono soundfile player (sfplay~)
4 - sfst – stereo soundfile player (sfplay~ 2)
4 - sfq – quad soundfile player (sfplay~ 4)
4 - del – simple delay with feedback (tapin~/tapout~)
4 - amod – amplitude modulation (LFO or audio rate, *~ and cycle~)
4 - filt – basic multi-type filter module (biquad~)
4 - fshift – dual side band frequency shifter (freqshift~)
1 - fftfilt – frequency domain (FFT) filter/synthsizer (pfft~)
4 - harm – harmonizer with delay (gizmo~)
4 - choir – vocal choral pitch shifter with independent delays (psychoirtrist~)
2 - rev – reverb (Ircamverb~)
2 - mng – granular processing or synthesis with live input (munger~)
1 - zerox – zero crossing detector and noise/voiced audio splitter (zerox~)
2 - aux – mono send/return to/from custom audio patch or to additional dac~
2 - auxst – stereo send/return to/from custom audio patch or to additional dac~
4 to 8 -dac – audio interface output (dac~)

Depending on your Max skills and your needs for a specific project, other functions can be added such as pitch and timbre detection, random or algorithmic processes, sample playback, other forms of synthesis, and in fact any other patch that you would like to add. These patches can "talk" with any parameters of the existing modules via messages and sends, and can be incorporated into the audio matrix via aux(mono) or auxst(stereo).

Getting CRT Running

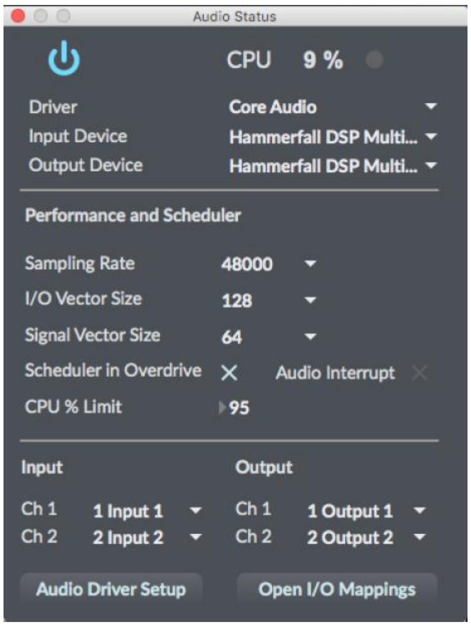
Everything you need to run CRT with Max is in the folder *CRT Concert Patch (0.20)*. Duplicate this folder or make a .zip file of it and keep it in a safe place so that you always have a non-modified version to go back to if ever you have a problem, or just to start a new project. This folder contains the maxproject folder *crt-project* with contains in turn the file *crt-project.maxproj* and several folders (*data*, *externals*, *info*, *media* and *patchers*. Do **NOT** change the name of the maxproject folder *crt-project* or the name of any of its contents, otherwise the patch may either not open or just not work properly. You CAN however change the name of the enclosing folder *CRT Concert Patch (0.20)* to anything you like. It may be useful to something like *project title CRT*. Even if you take the version number out of the folder name, it is always possible to find the CRT version of your project by looking in the folder *info* and opening the file *CRT version.txt*. This file contains a history of versions and release dates with a summary of fixes or new features for that version.



Preparing Max

CRT was designed in Max 7, and this version was completed with the latest version as of this date, 7.3.4. We are running in 32 bit mode and NOT in 64 bit mode, so we recommend 32 bit mode if possible.

First, launch Max by itself, **without** opening the *crt-project*. Go into the *Audio Status* window in the *Options* menu and choose your audio interface for input and output, sampling rate, i/o Vector Size and Signal Vector Size. We recommend 44100 or 48000 sampling rate, with 128 I/O vector and 64 Signal Vector – keeping latency down. Also, we would recommend checking *Scheduler in Overdrive* on, but *Audio Interupt* off. Double check your Input and Output mappings, then open the *Audiotester* in the *Extras* menu, and test your audio inputs and outputs.


Audio Status

CPU 9 %

Driver: Core Audio
Input Device: Hammerfall DSP Multi...
Output Device: Hammerfall DSP Multi...

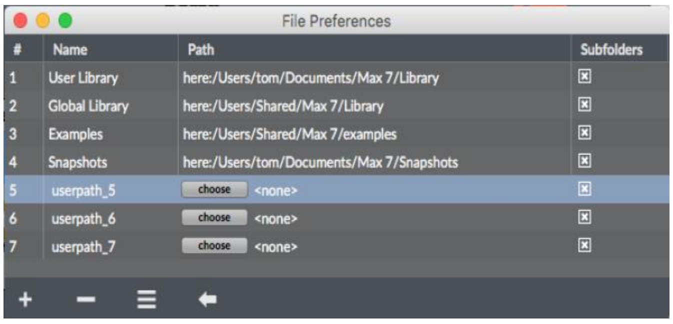
Performance and Scheduler

Sampling Rate: 48000
I/O Vector Size: 128
Signal Vector Size: 64
Scheduler in Overdrive: ✕
Audio Interrupt
CPU % Limit: 95

Input		Output	
Ch 1	1 Input 1	Ch 1	1 Output 1
Ch 2	2 Input 2	Ch 2	2 Output 2

You might also want to remove the user file paths in the File Preferences window in the Options menu. Click on each “userpath” and then click the “-“ at the bottom of the window. You can not and should not remove the 1st four “factory” paths.

If everything is set, QUIT MAX ! Do NOT just open the CRT project directly after modifying the audio and path preferences. If Max ever crashes (not that it will, but it can happen), you will lose all of your preferences if you didn’t quit “normally”. Max’s preferences are made “permanent” only when you quit max correctly. If max crashes, it automatically goes back to the last settings it had the last time it quit correctly. So always quit after modifying and testing your settings.


File Preferences

#	Name	Path	Subfolders
1	User Library	here:/Users/tom/Documents/Max 7/Library	☒
2	Global Library	here:/Users/Shared/Max 7/Library	☒
3	Examples	here:/Users/Shared/Max 7/examples	☒
4	Snapshots	here:/Users/tom/Documents/Max 7/Snapshots	☒
5	userpath_5	choose <none>	☒
6	userpath_6	choose <none>	☒
7	userpath_7	choose <none>	☒

Launching CRT

After modifying, verifying and testing the audio and file path settings in Max, and then quitting Max… RE-Launch Max, and preferably open the *Max Console* using Shift-Command-M so that you will be able to read messages that will be printed from within the CRT patch, or any errors that might occur.

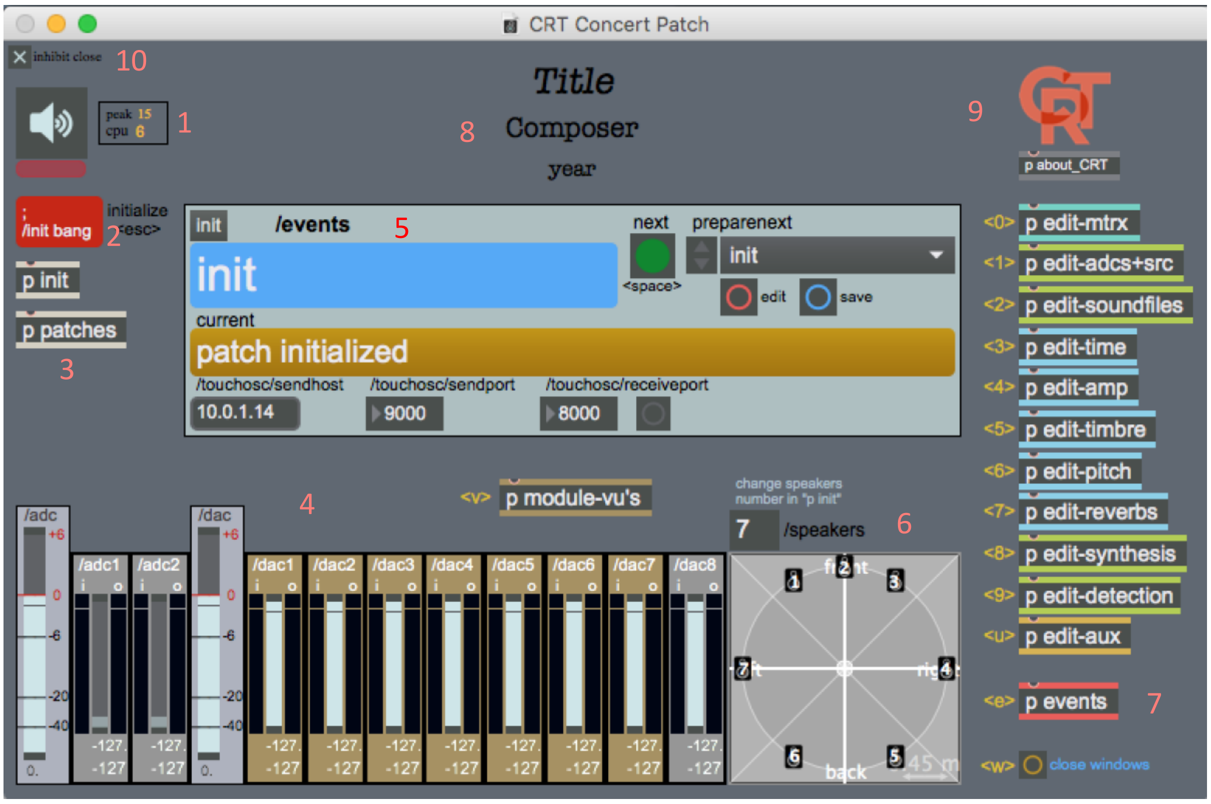
Open the project file *crt-project.maxproj*, either by double-clicking on it once Max is running, or (safer) by going to the *File/Open…* menu from within Max. You can also drag the *crt-project.maxproj* file from the finder window to the Max Console. Or, in the finder, you can drag the *crt-project.maxproj* file to the Max icon in the Dock.

CRT is a fairly large patch and takes a while to load, so expect it to take about 30 seconds to load and initialize. The top-level patcher will appear, lines will be printed to the Max Console, and the window will turn red while the patch initializes before turning back to its normal dark blue-grey color. "Init : ------------------------ done" should be printed at the bottom of the Max Console, and the Max Console should contain no errors.

CRT descriptions

Top-level patcher

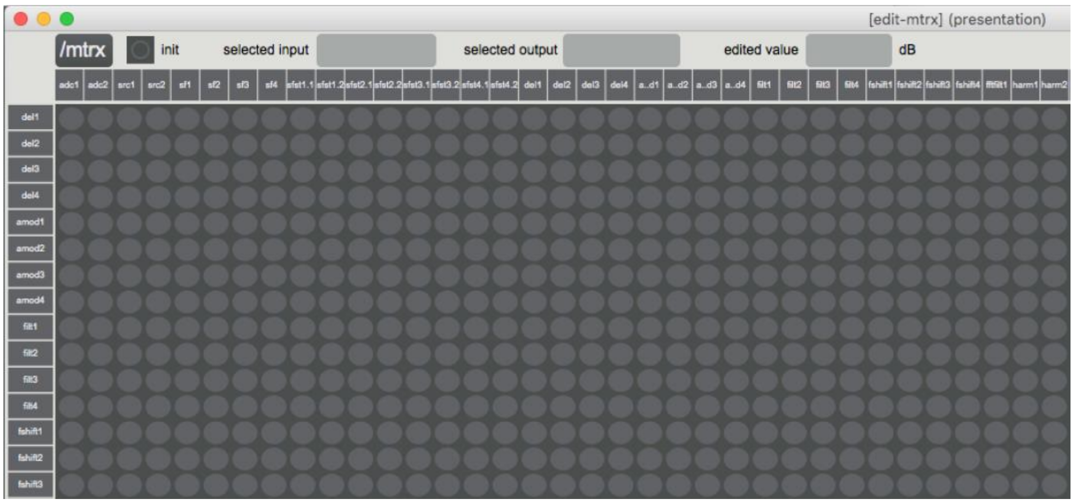
The top-level patch window of CRT is called *CRT Concert Patch*.



Its parts are:
1 – *Audio on/off*, cpu usage (current and peak)
2 – Button and message to initialize the entire patch (*; /init bang*)
3 – *init* subpatch (where you can add personalized init params) and *patches* (containing all module patches and other functions)
4 – ADC and DAC i/o level meters including volume control per adc and dac, as well as a global control for each. Golden brown color indicates active module. There is also the sub patch for level meters of all modules (*module-vu's*).
5 – The *events* section where you trigger events, display current event and descriptive text, edit and execute event sequence, select starting event in sequence, etc.
6 – Spat speaker configuration (display only). Speakers number can be changed in *init* sub patch
7 – Sub patches for all editor windows, organized by category. Also, the *events* sub patch, where events are created
8 – Editable text for *Title*, *Composer* and *Year*
9 – CRT logo and *about* sub patch for credits and version. Click on CRT logo to open about patch
10 – *Inhibit close* toggle. When this is checked, the window can not be closed by accident.

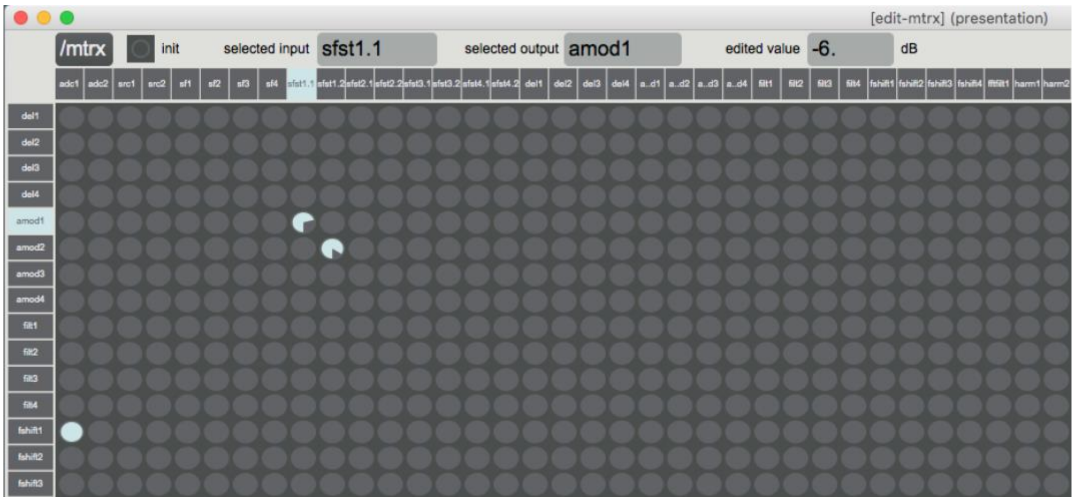
The Matrix

Double-click on the *p edit-mtrx* sub patch on the upper right to open the audio matrix editor, or just press the “0” (zero) key on the computer keyboard to open the matrix window. The audio matrix is quite large, and we are only showing a part of it here. The columns represent *inputs* to the matrix, and the rows represent matrix *outputs*. All of the matrix *inputs* come from the *outputs* of all of the modules, and the *outputs* of the matrix go to the *inputs* of the modules. Modules which have both *inputs* and *outputs* are found on both sides of the matrix. This includes all audio processing modules. Other modules such as *adc* and *sf* or *sfst* only have outputs, and so only appear on the input side of the matrix. The *dacs* are modules that are connected to the output of the matrix, but they do not appear on the matrix editor to avoid confusion, since all connections to the *dacs* are made via the *spat* interface of each module. This will be explained later.



[edit-mtrx] (presentation)
/mtrx
init
selected input
selected output
edited value
dB
adc1 adc2 src1 src2 sf1 sf2 sf3 sf4 sfst1.1 sfst1.2 sfst2.1 sfst2.2 sfst3.1 sfst3.2 sfst4.1 sfst4.2 del1 del2 del3 del4 a..d1 a..d2 a..d3 a..d4 filt1 filt2 filt3 filt4 fshift1 fshift2 fshift3 fshift4 fftfilt1 harm1 harm2
del1 del2 del3 del4 amod1 amod2 amod3 amod4 filt1 filt2 filt3 filt4 fshift1 fshift2 fshift3

The numbers of rows and columns and the labels of each are generated automatically, so if we add or subtract modules later, we will not have to do anything to the matrix editor in order for it to show the proper connections. However, on the input side the labels are sometimes hard to read, and are missing characters for lack of space. If you click on a name, it will show in full in the selected input and selected output section at the top, so you can verify the right location before making a connection. To connect, you click on a circle corresponding to the intersection of an input you want to connect to an output, and you glide the mouse to raise the *dial* level to the desired amount. This will probably be 0 dB, or “full”. The edited value will show at the top right in the space labeled *edited value dB*. Here is an example of a connected matrix.



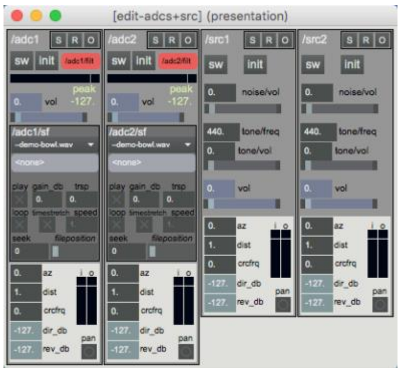
[edit-mtrx] (presentation)
/mtrx
init
selected input sfst1.1
selected output amod1
edited value -6.
dB
adc1 adc2 src1 src2 sf1 sf2 sf3 sf4 sfst1.1 sfst1.2 sfst2.1 sfst2.2 sfst3.1 sfst3.2 sfst4.1 sfst4.2 del1 del2 del3 del4 a..d1 a..d2 a..d3 a..d4 filt1 filt2 filt3 filt4 fshift1 fshift2 fshift3 fshift4 fftfilt1 harm1 harm2
del1 del2 del3 del4 amod1 amod2 amod3 amod4 filt1 filt2 filt3 filt4 fshift1 fshift2 fshift3

Stereo soundfile 1 (sfst1) has two outputs, sfst1.1 and sfst1.2. 1.1 is connected to amp modulation 1 (amod1) at a value of -6 dB and 1.2 is connected to amod2 at a value of -3 dB. At the same time, adc1 is connected to frequency shift 1 (fshift1) at full value, 0 dB.

If you double click on the sub patch *events*, or just press the keyboard shortcut “e”, you will open the events window. Click on the red ; /events/capture bang to open a text window with all active module params, including all matrix connection messges. Here’s what the above connections look like in message format:

```
;
// ---------- mtrx ----------;
/mtrx adc1 fshift1 0.;
/mtrx sfst1.1 amod1 -6.;
/mtrx sfst1.2 amod2 -3.;
```

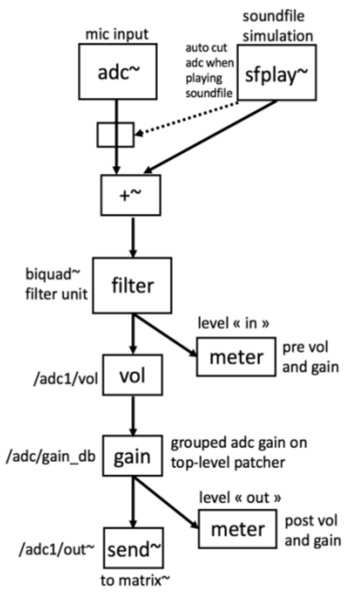
The *adc* and *src* modules

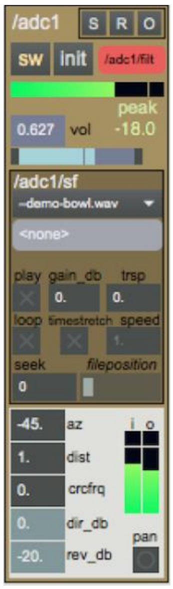


Double-click on the edit-adcs+src subpatch or press keyboard shortcut “1” to open the adc+src module editor window. This window contains 2 adc modules and two src (source) modules. The adc modules receive signal from the mic inputs of the audio interface, and also contain a soundfile player so that soundfiles can be played to simulate adc input, and then simply replaced by the actual mic input when available without changing any signal routing. The source module (src) generates white noise and a sine tone of variable frequency, with separate volumes.

The adc module

Block diagram of adc module:





The *adc* modules (/adc1 and /adc2) contain an adc~ input and an sfplay~ summed to one single mono signal which then passer through a simple biquad multi-type filter

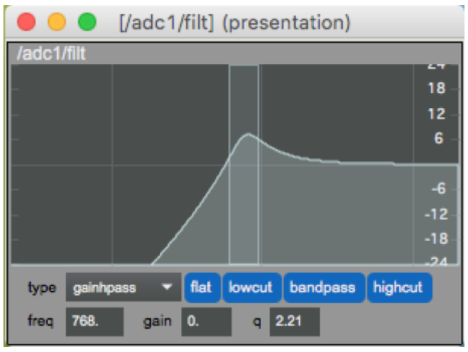
The adc editor. Certain controls are found in ALL editors.

S R O for temp preset Store, Recall, Open allowing you to store a temp preset and open the text editor to copy the messages. Or you can initialize the module and Recall the earlier preset.

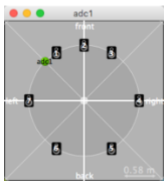
“sw” on/off switch which activates and deactivates the module, muting cpu use when off.

Init – click to initialize module. Also send “bang” to /adc1/init

Specific adc elements are: Input level meter, Output volume, Soundfile play section (similar to sf module) with soundfile menu selection, play, loop, gain_db and transpose – also activate time stretch and adjust trsp and speed independently. Soundfiles are located in the project folder in /media/sounds.



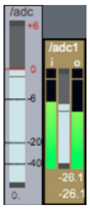
Clicking on the red /adc1/filt button opens the adc filter window if you want to filter the adc before . Here choose filter type, cutoff or center frequency, gain and Q (resonance). The blue buttons are shortcuts to get quickly to *flat*, *lowcut*, *bandpass* and *highcut*.



The bottom beige section is for spatializaton (azimuth, distance, a circular rotation frequency (crcfrq), direct out gain in dB (dir_db) and reverb send (rev_db).

Click the *pan* button to open pan editor.

I and *o* meters show pre-volume and post volume levels, as per the above schematic. This is very useful when debugging an output problem with a module. If the *i* meter is moving than the module is producing or receiving sound, but the *o* level meter will not move unless the /vol parameter AND the grouped /adc/gain_db control (main top-level patcher) are both on. The adc meters on the top-level contain the *I* and *o*, meters as well as a /vol slider, equivalent to the one in the adc editor.



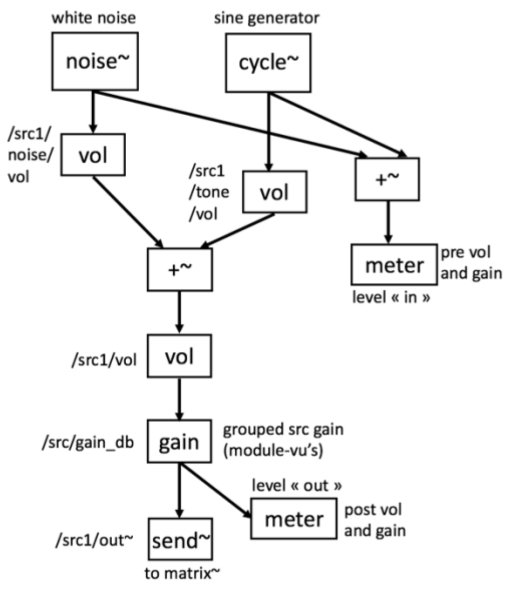
Clicking on Store and then on Open allows us to see and copy the paramters representing the current state of the module. These are the messages that represent the state of the above examples.

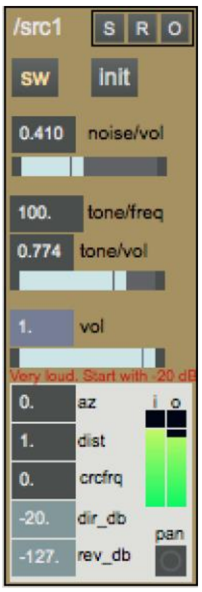
```
// ---------- adc1 ----------;
/adc1/sw 1;
/adc1/vol 0.627455;
/adc1/filt/type gainhpass;
/adc1/filt/q 2.21;
/adc1/filt/gain 0.;
/adc1/filt/freq 768.;
/adc1/sf/file <none>;
/adc1/sf/play 0;
/adc1/sf/seek 0;
/adc1/sf/loop 0;
/adc1/sf/trsp 0;
/adc1/sf/speed 1;
/adc1/sf/timestretch 0;
/adc1/sf/gain_db 0;
/adc1/az -45.;
/adc1/dist 1;
/adc1/crcfrq 0;
/adc1/dir_db 0;
/adc1/rev_db -20;
```

The src module

A simple source of noise and or sine tone with variable frequency.

Block diagram of src module:



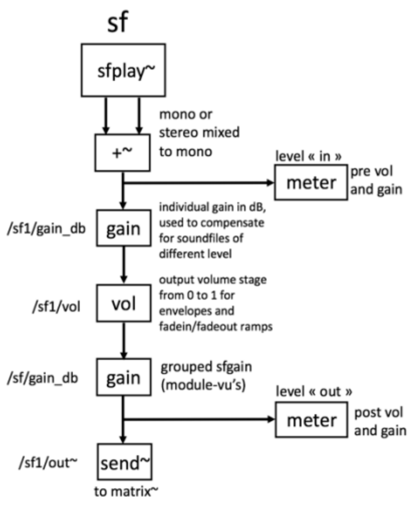


```
// ---------- src1 ----------;
/src1/sw 1;
/src1/noise/vol 0.410268;
/src1/tone/freq 100.;
/src1/tone/vol 0.773661;
/src1/vol 1.;
/src1/az 0;
/src1/dist 1;
/src1/crcfrq 0;
/src1/dir_db -20.;
/src1/rev_db -127.;
```

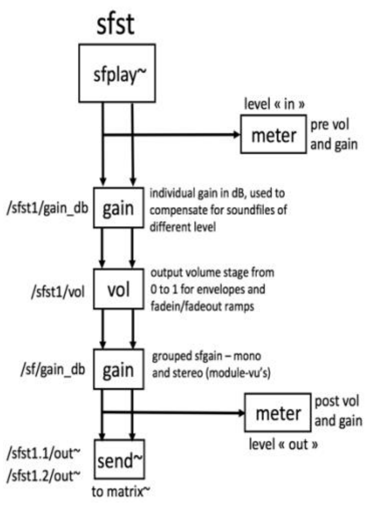
Soundfiles

Double-click on the *edit-soundfiles* subpatch or press keyboard shortcut "2" to open the module editor window. This window contains 4 mono soundfile players and 4 stereo soundfile players. They are identical except that the first outputs in mono whether playing a mono soundfile or a stereo soundfile which will be mixed to mono. The sfst outputs 2 channels either for separate channels of stereo soundfile, or in the case of mono, the 1 channel output duplicated to the 2nd channel. They both include a "file" menu that reads the soundfiles that are found in the project folder /media/sounds. If you put sounds into this folder and initialize the CRT patch, the menu will automatically be updated. Both modules also contain a gain_db parameter and a trsp (transpose) parameter and loop on/off. A timestretch function allows separation of speed and transposition. This function can sometimes add noise to the end of the file if playing a soundfile that was created at a different sampling rate than the current sampling rate.

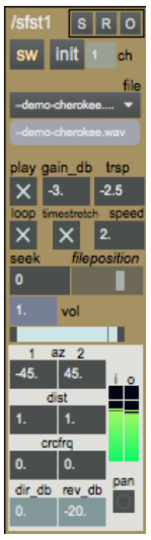
The sf and sfst modules



sf
sfplay~
mono or stereo mixed to mono
+~
level « in »
meter
pre vol and gain
/sf1/gain_db
gain
individual gain in dB, used to compensate for soundfiles of different level
/sf1/vol
vol
output volume stage from 0 to 1 for envelopes and fadein/fadeout ramps
/sf/gain_db
gain
grouped sfgain (module-vu's)
level « out »
meter
post vol and gain
/sf1/out~
send~
to matrix~



sfst
sfplay~
level « in »
meter
pre vol and gain
/sfst1/gain_db
gain
individual gain in dB, used to compensate for soundfiles of different level
/sfst1/vol
vol
output volume stage from 0 to 1 for envelopes and fadein/fadeout ramps
/sf/gain_db
gain
grouped sfgain – mono and stereo (module-vu's)
meter
post vol and gain
level « out »
/sfst1.1/out~
/sfst1.2/out~
send~
to matrix~



```
// ---------- sfst1 ----------;
/sfst1/file –demo-cherokee.wav;
/sfst1/sw 1;
/sfst1/play 1;
/sfst1/seek 0;
/sfst1/loop 1;
/sfst1/trsp -2.500667;
/sfst1/speed 2.006667;
/sfst1/timestretch 1;
/sfst1/gain_db -3.;
/sfst1/vol 1.;
/sfst1/az1 -45;
/sfst1/dist1 1;
/sfst1/crcfrq1 0;
/sfst1/az2 45;
/sfst1/dist2 1;
/sfst1/crcfrq2 0;
/sfst1/dir_db 0;
/sfst1/rev_db -20;
```

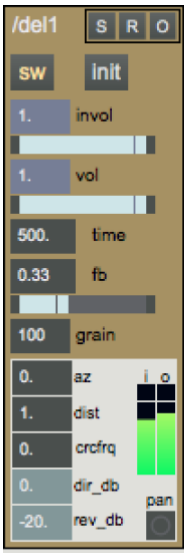
Time processing

Double-click on the *edit-time* subpatch or press keyboard shortcut "3" to open the module editor window.

The del module

The del module is a continuously variable delay up to 30 seconds, with feedback. Here is an example of a complete event "state" with adc input, matrix routing and the del1 module. It was created by setting the parameters in the editors, and then going to the *events* subpatch and clicking on *; /events/capture bang*.

```
// ---------- mtrx ----------;
/mtrx adc1 del1 0.;
// ---------- adc1 ----------;
/adc1/sw 1;
/adc1/vol 1.;
/adc1/filt/type lowpass;
/adc1/filt/q 0.707;
/adc1/filt/gain 0.;
/adc1/filt/freq 23950.;
/adc1/sf/file <none>;
/adc1/sf/play 0;
/adc1/sf/seek 0;
/adc1/sf/loop 0;
/adc1/sf/trsp 0;
/adc1/sf/speed 1;
/adc1/sf/timestretch 0;
/adc1/sf/gain_db 0;
/adc1/az 0;
/adc1/dist 1;
/adc1/crcfrq 0;
/adc1/dir_db 0;
/adc1/rev_db -127;
// ---------- del1 ----------;
/del1/sw 1;
/del1/invol 1;
/del1/time 500.;
/del1/fb 0.33;
/del1/vol 1;
/del1/grain 100;
/del1/az 0;
/del1/dist 1;
/del1/crcfrq 0;
/del1/dir_db 0;
/del1/rev_db -20;
```



Amplitude processing

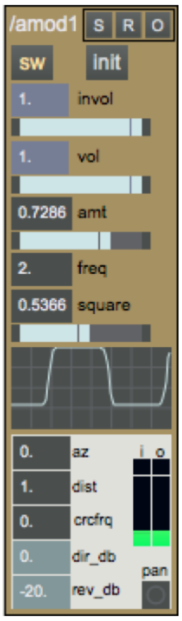
Double-click on the *edit-amp* subpatch or press keyboard shortcut "4" to open the module editor window.

The amod module

A simple amplitude modulation LFO, with parameters for frequency, amount of modulation, and a waveform distortion parameter, square, which deforms the otherwise sine LFO progressively into a square wave.

In the event example below, note that the adc1 has dir_db and rev_db set to -127 so that there is no spat output of the adc directly. In this case all of the audio comes from the amod module which has dir_db and rev_db. In this way we hear the transformation better because it is not masqued by amplification of the direct sound.

```
// ---------- mtrx ----------;
/mtrx adc1 amod1 0.;
// ---------- adc1 ----------;
/adc1/sw 1;
/adc1/vol 1.;
/adc1/filt/type lowpass;
/adc1/filt/q 0.707;
/adc1/filt/gain 0.;
/adc1/filt/freq 23950.;
/adc1/sf/file <none>;
/adc1/sf/play 0;
/adc1/sf/seek 0;
/adc1/sf/loop 0;
/adc1/sf/trsp 0;
/adc1/sf/speed 1;
/adc1/sf/timestretch 0;
/adc1/sf/gain_db 0;
/adc1/az 0;
/adc1/dist 1;
/adc1/crcfrq 0;
/adc1/dir_db -127.;
/adc1/rev_db -127;
// ---------- amod1 ----------;
/amod1/sw 1;
/amod1/invol 1.;
/amod1/amt 0.727619;
/amod1/freq 2;
/amod1/square 0.53756;
/amod1/vol 1.;
/amod1/az 0;
/amod1/dist 1;
/amod1/crcfrq 0;
/amod1/dir_db 0;
/amod1/rev_db -20;
```



Timbre processing

Double-click on the *edit-timbre* subpatch or press keyboard shortcut “5” to open the module editor window.

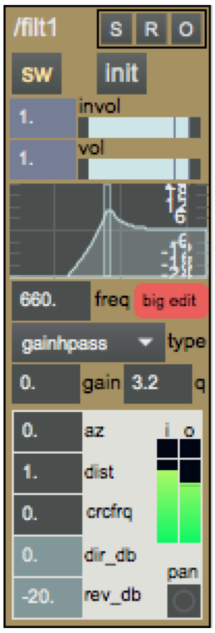
This Window contains editors for a basic filter module (filt), a frequency shift module (fshift) and a spectral domain filter (fftfilt) which can process live audio, but which can also be used as a form of synthesis.

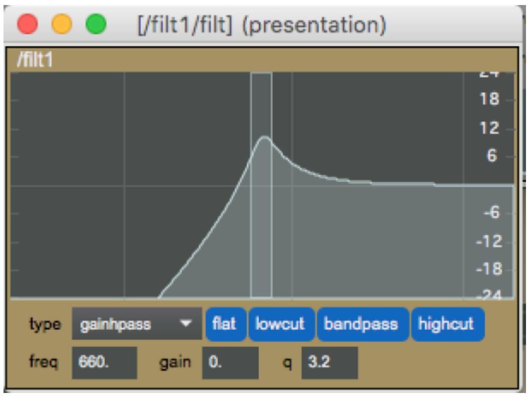
The filt module

Basic filter module based on biquad~ and filtercoef~. Editor includes a small version of the graphical filter interface, but also a *big edit* button for a larger, easier to edit version.

Example event state. Note than in this example, there is no spat amplification of the adc so that the sound only gets sent out by the filt module to avoid masquing the filtering by the amplification of the direct signal.

```
// ---------- mtrx ----------;
/mtrx adc1 filt1 0.;
// ---------- adc1 ----------;
/adc1/sw 1;
/adc1/vol 1.;
/adc1/filt/type lowpass;
/adc1/filt/q 0.707;
/adc1/filt/gain 0;
/adc1/filt/freq 23950.;
/adc1/sf/file <none>;
/adc1/sf/play 0;
/adc1/sf/seek 0;
/adc1/sf/loop 0;
/adc1/sf/trsp 0;
/adc1/sf/speed 1;
/adc1/sf/timestretch 0;
/adc1/sf/gain_db 0;
/adc1/az 0;
/adc1/dist 1;
/adc1/crcfrq 0;
/adc1/dir_db -127;
/adc1/rev_db -127;
// ---------- filt1 ----------;
/filt1/sw 1;
/filt1/invol 1;
/filt1/type gainhpass;
/filt1/q 3.2;
/filt1/gain 0.;
/filt1/freq 660.;
/filt1/vol 1;
/filt1/az 0;
/filt1/dist 1;
/filt1/crcfrq 0;
/filt1/dir_db 0;
/filt1/rev_db -20;
```

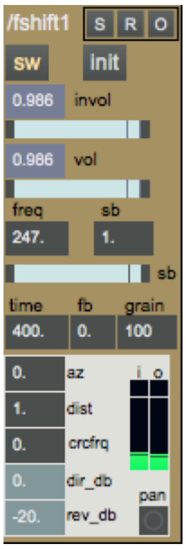




The fshift module

Simple frequency shift module based on freqshift~, including delay with feedback. Since freqshift~ provides bith positive and negative sideband outputs, this module includes a *sb* parameter in order to select or mix these sidebands. The parameter can go progressively from -1 to 1, -1 being negative side band, 1 being positive side band, and 0 being a mix of the two. A side band (*sb*) param value of 0 is the equivalent of Ring Modulation.

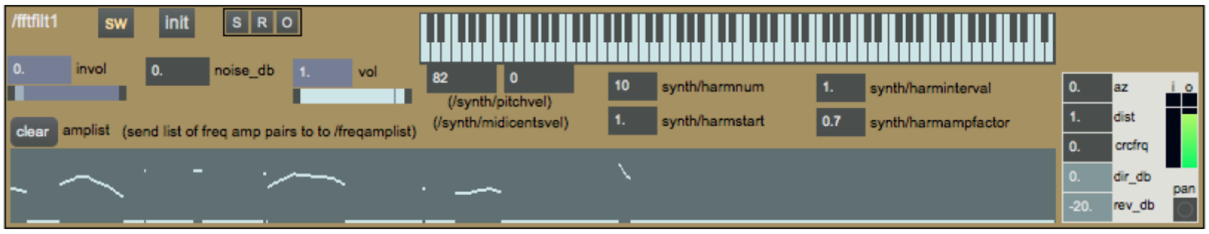
```
// ---------- fshift1 ----------;
/fshift1/sw 1;
/fshift1/invol 0.985714;
/fshift1/freq 247.;
/fshift1/sb 1.;
/fshift1/time 400.;
/fshift1/fb 0;
/fshift1/vol 0.985714;
/fshift1/grain 100;
/fshift1/az 0;
/fshift1/dist 1;
/fshift1/crcfrq 0;
/fshift1/dir_db 0;
/fshift1/rev_db -20;
```



The fftfilt module

A frequency domain fft filter which filters a live input or just white noise using a spectral envelop that you can draw. You can also play it like a synthesizer, most easily perceived by using the noise_db parameter to use noise as the source. The synth/harmnum param determines number of harmonics to calculate, the synth/harmstart determines harmonic number to start on, synth/harminterval determines harmonic multiplier (decimal values = inharmonic), and synth/harmampfactor determines relative weight of harmonics – closer to 1. = brighter sound.

If you use live input, you must set the invol param to 1, but not if you use noise input. The vol param must always be set to 1 in order to have output. The spectral envelope is stored as a long list.



```
// ---------- fftfilt1 ----------;
/fftfilt1/sw 1;
/fftfilt1/invol 0.;
/fftfilt1/noise_db 0.;
/fftfilt1/amplist 0.500 0.475 0.470 0.465 0.460 0.455 0.450 0.445 0.440 0.430 0.420 0.420 0.420 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.520 0.520 0.540 0.550 0.560
0.560 0.570 0.580 0.580 0.587 0.593 0.600 0.620 0.620 0.620 0.620 0.620 0.620 0.620 0.620 0.620 0.610 0.600 0.592 0.583 0.575 0.567
0.558 0.550 0.542 0.533 0.525 0.517 0.508 0.500 0.500 0.480 0.460 0.460 0.440 0.420 0.410 0.400 0.387 0.373 0.360 0.360 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.700 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.700 0.700 0.700 0.700 0.700 0.700 0.700 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.660 0.660 0.000 0.480 0.500 0.513 0.527 0.540 0.548
0.556 0.564 0.572 0.580 0.588 0.596 0.604 0.612 0.620 0.630 0.640 0.650 0.660 0.660 0.660 0.658 0.657 0.655 0.653 0.651 0.650 0.648
0.646 0.644 0.643 0.641 0.639 0.637 0.636 0.634 0.632 0.630 0.629 0.627 0.625 0.623 0.622 0.620 0.617 0.613 0.610 0.607 0.603 0.580
0.573 0.567 0.560 0.550 0.540 0.530 0.520 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.460 0.460
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.400 0.400 0.400 0.400 0.400 0.400 0.400 0.400 0.404 0.408 0.412 0.416 0.420 0.425 0.430 0.435 0.440 0.440 0.440 0.444 0.448 0.452
0.456 0.460 0.460 0.460 0.460 0.453 0.447 0.440 0.435 0.430 0.425 0.420 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.780 0.760 0.730 0.680 0.665 0.650 0.635 0.620 0.600 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.;
/fftfilt1/synth/harmnum 10;
/fftfilt1/synth/harmstart 1;
/fftfilt1/synth/harminterval 1;
/fftfilt1/synth/harmampfactor 0.7;
/fftfilt1/vol 1.;
/fftfilt1/az 0;
/fftfilt1/dist 1;
/fftfilt1/crcfrq 0;
/fftfilt1/dir_db 0;
/fftfilt1/rev_db -20;
```

Pitch processing

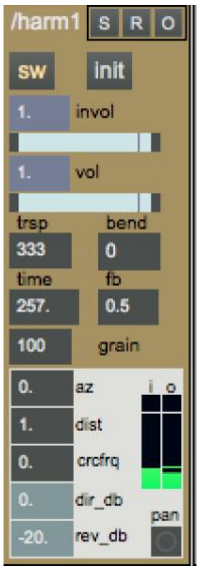
Double-click on the *edit-pitch* subpatch or press keyboard shortcut "6" to open the module editor window.

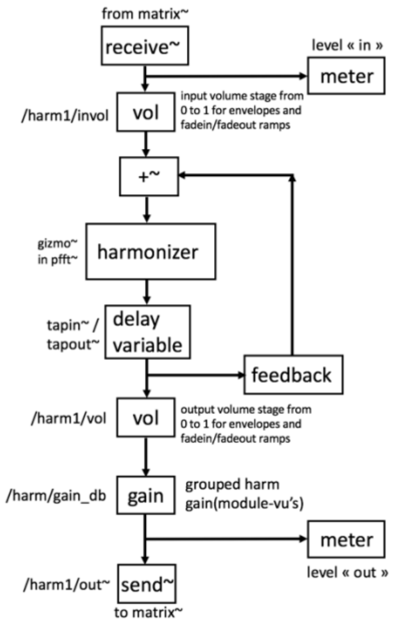
This Window contains editors for a basic harmonizer module (harm) based on gizmo~, and a vocal transposition module which maintains vocal formants and makes for very realistic "choir" harmonisation – based on Ircam's psychoirtrist~ object. The choir module has up to four transpositions, each with separate delay, gain, volume and pan. The choir module has 2 outputs, /choir1.1/out~ and /choir1.2/out~, using 2 inputs in the matrix. Each channel can be panned freely in the multichannel spat panning space.

The harm module

Sample event parameters:

```
// ---------- harm1 ----------;
/harm1/sw 1;
/harm1/invol 1;
/harm1/trsp 333;
/harm1/bend 0;
/harm1/time 257.;
/harm1/fb 0.5;
/harm1/vol 1;
/harm1/grain 100;
/harm1/az 0;
/harm1/dist 1;
/harm1/crcfrq 0;
/harm1/dir_db 0;
/harm1/rev_db -20;
```



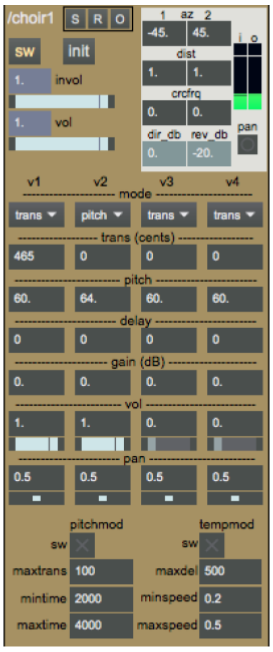


The choir module

Built with Ircam's psychoirtrist~ object, choir has up to 4 simultaneous transpositions with separate delay, gain, volume and pan. Each voice has two modes: *trans* for relative transposition in cents, and *pitch* for absolute pitch as a float midi referenced note (i.e. C4 = 60.0) no matter what the input.

Here is sample event state for choir, with voice 1 in relative trasposition mode and voice 2 in fixed pitch mode. In the editor, the pitch value is ignored in trans mode, and the trans value is ignored in pitch mode. This module is meant to work best with voice because it tries to respect the spectral envelope of formants throughout the change in transposition. It may however work well with certain instruments.

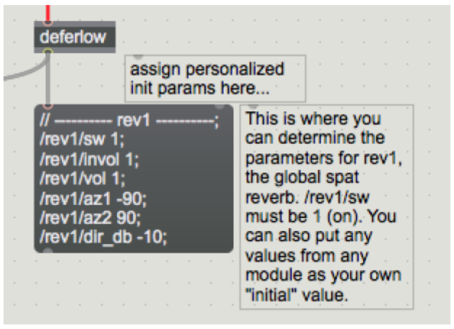
```
// ---------- choir1 ----------;
/choir1/sw 1;
/choir1/invol 1;
/choir1/vol 1;
/choir1/v1/mode trans;
/choir1/v1/trans 465;
/choir1/v1/pitch 60;
/choir1/v1/delay 0;
/choir1/v1/gain 0;
/choir1/v1/vol 1.;
/choir1/v1/pan 0.5;
/choir1/v2/mode pitch;
/choir1/v2/trans 0;
/choir1/v2/pitch 64.;
/choir1/v2/delay 0;
/choir1/v2/gain 0;
/choir1/v2/vol 1.;
/choir1/v2/pan 0.5;
/choir1/v3/mode trans;
/choir1/v3/trans 0;
/choir1/v3/pitch 60;
/choir1/v3/delay 0;
/choir1/v3/gain 0;
/choir1/v3/vol 0;
/choir1/v3/pan 0.5;
/choir1/v4/mode trans;
/choir1/v4/trans 0;
/choir1/v4/pitch 60;
/choir1/v4/delay 0;
/choir1/v4/gain 0;
/choir1/v4/vol 0;
/choir1/v4/pan 0.5;
/choir1/pitchmod/sw 0;
/choir1/pitchmod/maxtrans 100;
/choir1/pitchmod/mintime 2000;
/choir1/pitchmod/maxtime 4000;
/choir1/tempmod/sw 0;
/choir1/tempmod/maxdel 500;
/choir1/tempmod/minspeed 0.2;
/choir1/tempmod/maxspeed 0.5;
/choir1/az1 -45;
/choir1/dist1 1;
/choir1/crcfrq1 0;
/choir1/az2 45;
/choir1/dist2 1;
/choir1/crcfrq2 0;
/choir1/dir_db 0;
/choir1/rev_db -20
```



Spatial processing

Double-click on the *edit-reverbs* subpatch or press keyboard shortcut "7" to open the module editor window.

There are two reverb modules in the edit-reverbs patcher, /rev1 and /rev2. Rev1 is a "reserved" module in CRT because it is used exclusively as the spatialization reverb for all of the other modules. The *rev_db* parameter in all module editors sends signal to rev1. Rev1 is always on when CRT initializes and is panned to be in all speakers. Rev1 appears in the matrix, but is not meant to be used there, but instead only via the *rev_db* parameters in the module editors. You must therefore adjust the parameters of rev1 to modify the spatial characteristics of the global spatializaton in the patch. If you do modify the defaut values, *Store*, then *Open* in the temp preset section of the rev1 editor, and copy the parameter messages. Then open the *init* sub patcher in the main top-level window and paste the messages into the message box marked "assign personalized init params here..." which already contains some rev1 parameters.



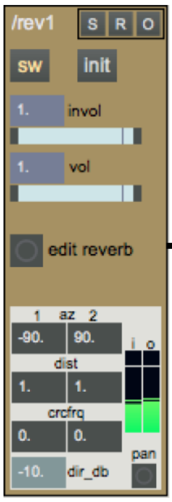
```
// ---------- rev1 ----------;
/rev1/sw 1;
/rev1/invol 1;
/rev1/vol 1;
/rev1/az1 -90;
/rev1/az2 90;
/rev1/dir_db -10;
```

This is where you can determine the parameters for rev1, the global spat reverb. /rev1/sw must be 1 (on). You can also put any values from any module as your own "initial" value.

Rev2 is available for rever "effects" such as long or infinite reverb.

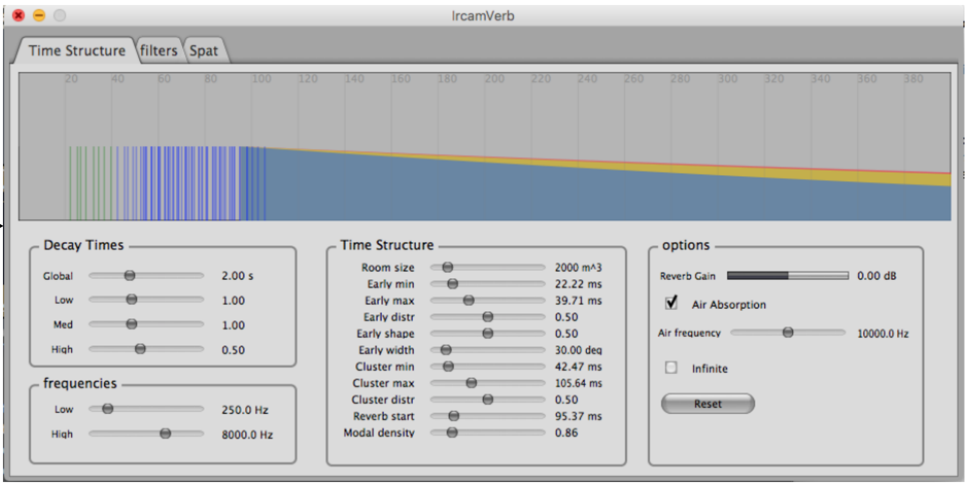
The rev module

The reverb object in use is Ircam's Ircamverb~, which is extracted from the Ircam Spat. Note the very few parameters found on the editor. This is because the reverb parameters are all in another window that we open with the "edit reverb" button. Note also that the panning section of the editor does NOT include rev_db to send to the reverb, for obvious reasons…



invol	1.
vol	1.
az 1	-90.
az 2	90.
dist	1. / 1.
crcfrq	0. / 0.
dir_db	-10.

IrcamVerb editor, *Time Structure* tab



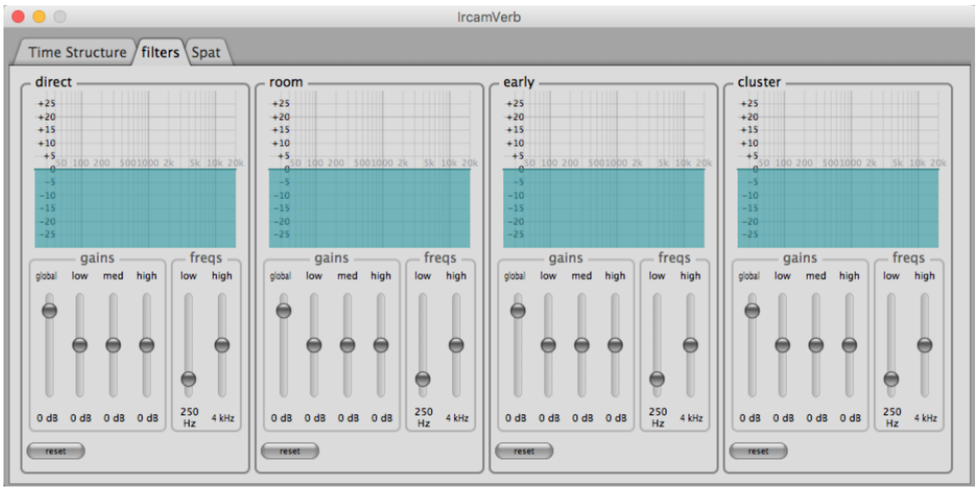
Decay Times: Global 2.00 s; Low 1.00; Med 1.00; High 0.50

frequencies: Low 250.0 Hz; High 8000.0 Hz

Time Structure	
Room size	2000 m^3
Early min	22.22 ms
Early max	39.71 ms
Early distr	0.50
Early shape	0.50
Early width	30.00 deg
Cluster min	42.47 ms
Cluster max	105.64 ms
Cluster distr	0.50
Reverb start	95.37 ms
Modal density	0.86

options: Reverb Gain 0.00 dB; [✓] Air Absorption; Air frequency 10000.0 Hz; ☐ Infinite

IrcamVerb editor, *filters* tab



direct, room, early, cluster — gains: global 0 dB, low 0 dB, med 0 dB, high 0 dB; freqs: low 250 Hz, high 4 kHz

All internal Ircamverb parameters are integrated in the CRT environment, so the temp store and the /events/capture functions in CRT will collect all of these parameters. For more information, see documentation for Ircam Spat. If you click in the temp store section at the top of the CRT rev editor, you can see in text format all of the reverb parameters. They are all compatible with remote messages via semi-colon message box or send objects, and they can also be ramped, as can any CRT parameter that it makes sense to ramp. See below.

```
// ---------- rev1 ----------;
/rev1/sw 1;
/rev1/invol 1;
/rev1/vol 1;
/rev1/cluster/filters 0.000 0.000 0.000 0.000 250.000 4000.;
/rev1/cluster/distr 0.5;
/rev1/cluster/end 105.639999;
/rev1/cluster/start 42.470001;
/rev1/direct/filters 0.000 0.000 0.000 0.000 250.000 4000.;
/rev1/early/filters 0.000 0.000 0.000 0.000 250.000 4000.;
/rev1/early/distr 0.5;
/rev1/early/end 39.709999;
/rev1/early/shape 0.5;
/rev1/early/start 22.219999;
/rev1/early/width 30.;
/rev1/reverb/fh 8000.;
/rev1/reverb/fl 250.;
/rev1/reverb/gain 0.;
/rev1/reverb/infinite 0;
/rev1/reverb/modaldensity 0.86;
/rev1/reverb/start 95.370003;
/rev1/reverb/air 1;
/rev1/reverb/airfreq 10000.;
/rev1/reverb/tr0 2.;
/rev1/reverb/trh 0.5;
/rev1/reverb/trl 1.;
/rev1/reverb/trm 1.;
/rev1/room/filters 0.000 0.000 0.000 0.000 250.000 4000.;
/rev1/az1 -90;
/rev1/dist1 1;
/rev1/crcfrq1 0;
/rev1/az2 90;
/rev1/dist2 1;
/rev1/crcfrq2 0;
/rev1/dir_db -10;
/rev1/rev_db -127;
```

Synthesis

Double-click on the *edit-synthesis* subpatch or press keyboard shortcut “8” to open the module editor window.

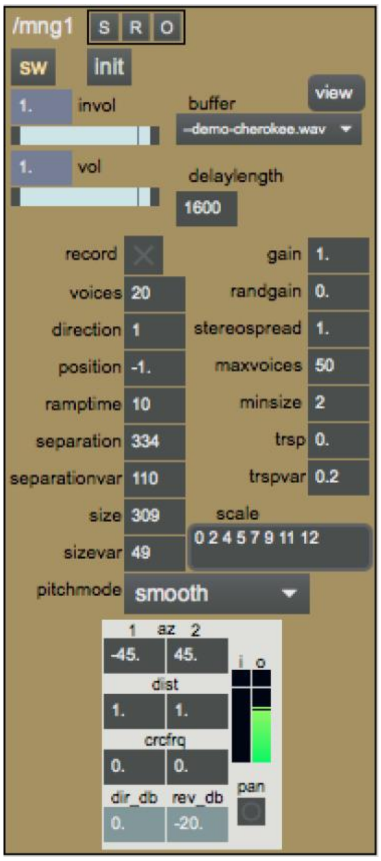
The synthesis section contains a granular synthesis module, the munger~ object by Dan Trueman. It can work as a real time audio processing module for live input, or else you can load a soundfile to work from.

Another module that you can use for synthesis of pitches or timbres is the *fftfilt* module, discussed in the *Timbre Processing* section.

The mng module

Sample parameter list, and parameter explanations.

```
// ---------- mng1 ----------;
/mng1/sw 1;
/mng1/invol 1;
/mng1/vol 1;
/mng1/buffer –demo-cherokee.wav;
/mng1/delaylength 1600;
/mng1/record 0;
/mng1/voices 20;
/mng1/direction 1;
/mng1/position -1;
/mng1/ramptime 10;
/mng1/separation 334;
/mng1/separationvar 110;
/mng1/size 309;
/mng1/sizevar 49;
/mng1/trsp 0;
/mng1/trspvar 0.2;
/mng1/scale 0 2 4 5 7 9 11 12;
/mng1/pitchmode smooth;
/mng1/gain 1;
/mng1/randgain 0;
/mng1/stereospread 1.;
/mng1/maxvoices 50;
/mng1/minsize 2;
/mng1/az1 -45;
/mng1/dist1 1;
/mng1/crcfrq1 0;
/mng1/az2 45;
/mng1/dist2 1;
/mng1/crcfrq2 0;
/mng1/dir_db 0;
/mng1/rev_db -20;
```



/buffer : choose soundfile from /media/sounds folder to use instead of live input

/delaylength: length of sound in ms to read with grains (maxiumum 5000)

/record : 1 = process live input, 0 = freeze buffer

/voices : current voice limit

/direction : 1 = fwd, -1 = bkwd, 0 = random

/position: -1 = random, 0 to 1 = relative fixed position

/ramptime : grain ramp in ms

/separation : time between grains in ms

/separationvar : separation time variation ms

/size : grain size in ms

/sizevar : grain size variation in ms

/pitchmode : tempered for quantize to semitones, smooth for non-quantized, scale for use scale.

/scale : list of ints for scalar transposition randomly as a function of trspvar

/gain : grain gain as multiplication factor

/randgain : random gain variation

/stereospread : 0.0 = mono, 1.0 = full stereo

/maxvoices : limit on maximum number of voices for voices param

/minsize : minimum grain size

/trsp : transposition value in semitones

/trspvar : variable transposition amount, 0. - 1.

Detection

Double-click on the *edit-detection* subpatch or press keyboard shortcut “9” to open the module editor window.

Detections of audio and musical parameters is one of the areas where you will need to do some Max programming. A tutorial on this subject will be released shortly, but you can apply objects such as Miller Puckette’s group of objects sigmund~ for pitch and peak detection, bonk~ for attack detection, or centroid~ for spectral centroid. There is also the excellent yin~ from Ircam for voiced-only pitch detection. These are not standard Max objects, but have been included in the CRT release in the externals folder

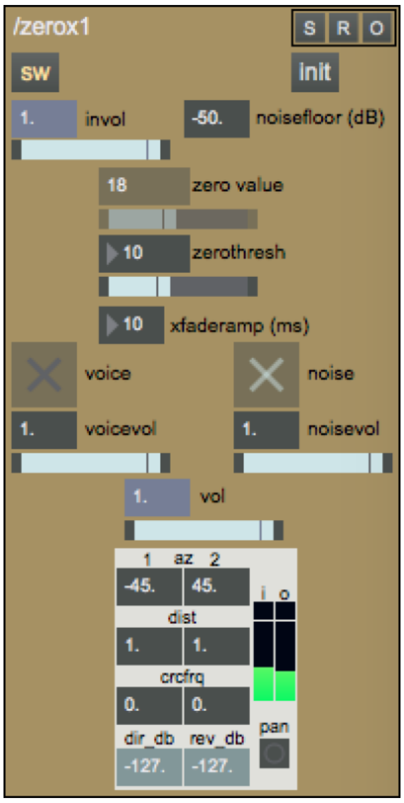
For now there is just zerox which is the max built-in zero crossing detector which gives a basic idea of the presence of noise.

The zero crossing detection will output voiced sounds out the first audio output of zerox, and noise elements out the second channel.

Below is an example event state whereas adc1 goes into zerox1, and the voiced output of zerox 1 goes into fshift1 while the noise element goes into del1.

The zerox module

```
// ---------- mtrx ----------;
/mtrx adc1 zerox1 0.;
/mtrx zerox1.1 fshift1 0.;
/mtrx zerox1.2 del1 0.;
// ---------- adc1 ----------;
/adc1/sw 1;
/adc1/vol 1;
/adc1/filt/type lowpass;
/adc1/filt/q 0.707;
/adc1/filt/gain 0.;
/adc1/filt/freq 22000.;
/adc1/sf/file <none>;
/adc1/sf/play 0;
/adc1/sf/seek 0;
/adc1/sf/loop 0;
/adc1/sf/trsp 0;
/adc1/sf/speed 1;
/adc1/sf/timestretch 0;
/adc1/sf/gain_db 0;
/adc1/az 0;
/adc1/dist 1;
/adc1/crcfrq 0;
/adc1/dir_db -127;
/adc1/rev_db -127;
// ---------- del1 ----------;
/del1/sw 1;
/del1/invol 1.;
/del1/time 423.;
/del1/fb 0.496;
/del1/vol 1.;
/del1/grain 100;
/del1/az 0;
/del1/dist 1;
/del1/crcfrq 0;
/del1/dir_db 0;
/del1/rev_db -20;
// ---------- fshift1 ----------;
/fshift1/sw 1;
/fshift1/invol 1;
/fshift1/freq 313.;
/fshift1/sb 0;
/fshift1/time 0;
/fshift1/fb 0;
/fshift1/vol 1;
/fshift1/grain 100;
/fshift1/az 0;
/fshift1/dist 1;
/fshift1/crcfrq 0;
/fshift1/dir_db 0;
/fshift1/rev_db -20;
// ---------- zerox1 ----------;
/zerox1/sw 1;
/zerox1/invol 1;
/zerox1/noisefloor -50;
/zerox1/zerothresh 10;
/zerox1/xfaderamp 10;
/zerox1/voicevol 1.;
/zerox1/noisevol 1.;
/zerox1/vol 1.;
/zerox1/az1 -45;
/zerox1/dist1 1;
/zerox1/crcfrq1 0;
/zerox1/az2 45;
/zerox1/dist2 1;
/zerox1/crcfrq2 0;
/zerox1/dir_db -127;
/zerox1/rev_db -127;
```



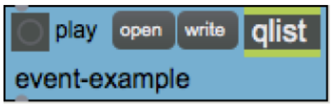
Events

Text-based qlist events

Double-click on the *events* subpatch or press keyboard shortcut “e” to open the events window.

EVENTS are in the form of text files containing lines of parameter messages and possibly delay times (int number by itself on a line) or comments (lines starting with "//"). Each line must end with a semicolon. Each event is accessed via bpatcher "crt.event". Event text files are in the format "eventname.txt" and are automatically created and saved in /crt-project/data/events.

To make a new event, duplicate event-example or any other event bpatcher, "command-i" to open inspector on the bpatcher, and scroll down to "args" or "Argument(s)" and change the existing name to the name of your new event. Do NOT use spaces in the names.



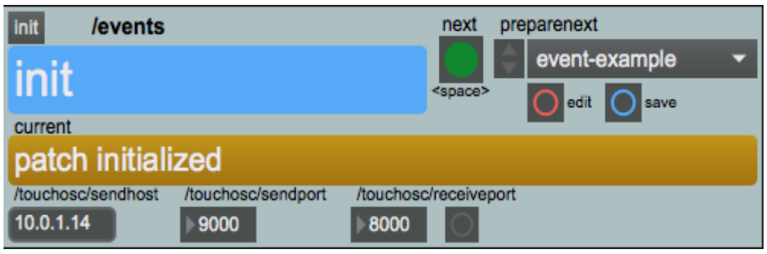
To edit event, double-click on "qlist" or click "open" button to open text window. Paste lines of parameter messages, edit existing lines or type new ones. Cmd-s in open qlist window (the simplist method) OR close window, click "save" in dialog, then click "write" in event bpatcher window.

The simplest way to initially build an event is to adjust module editors for the desired result, then click on red-colored messaage
"; /events/capture bang"

to capture parameters of all active modules (modules whos switch parameter, /sw, is 1) to text window. The contents of the text window can be copied and then pasted into the text window of an event.

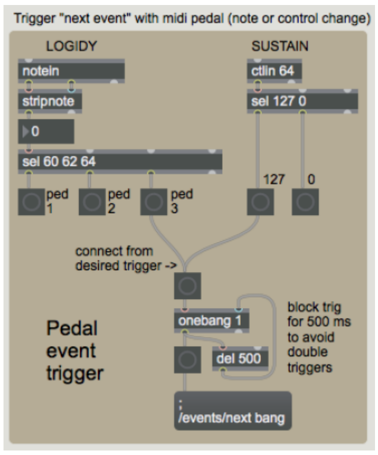
Play event by clicking the play button on the event, or esle send the message
"; eventname play". One event can trigger another if "eventname play;" is included as a line of an event.

Event sequencer



On the main top-level patcher of CRT is the event sequencer section called /events. It can be used to sequence a series of event qlists to be triggered either by spacebar or via a midi pedal that outputs sustain or midi notes.

Click on the red *edit* button of the event sequencer and type the list of events for the sequence in order. After each name you can type a text that can include spaces and that will be displayed when the event triggers. When you’re done, close the window and click on the blue *save* button to save this text file to /data/events with the text events themselves.


Trigger "next event" with midi pedal (note or control change)
LOGIDY
SUSTAIN
notein
ctlin 64
stripnote
sel 127 0
0
sel 60 62 64
127
0
ped 1
ped 2
ped 3
connect from desired trigger ->
onebang 1
block trig for 500 ms to avoid double triggers
del 500
Pedal event trigger
;
/events/next bang

Then initialize the patch and you can hit the spacebar or midi input to trigger. See the midi input patch in events. You can easily modify the midi inputs patch to accommodate different midi hardware.