ELSE Library for Pure Data

Porres, Alexandre Torres

EL Locus Solus (independent center)
São Paulo, SP

el.locus.solus@gmail.com

Abstract

The main computer music languages for Live Electronics nowadays are: Max, Pure Data (or just "Pd") & SuperCollider. In comparison to the other two, Pd offers a very limited set of functions in its main distribution (a.k.a "Pd vanilla"), relying heavily on external libraries as add-ons.

This paper describes the ELSE external library for Pd, which brings elements that were missing in Pd and its current external libraries when compared to other computer music environments. It also includes functionalities not available elsewhere and revises elementary building blocks of computer music already found in Pd or other systems, such as in the design of oscillators.

The ELSE library provides a large collection of objects that were carefully and meticulously designed to improve the patching experience for Pd, allowing some techniques of computer music to be implemented more conveniently.

It is also part of a didactic project for computer music that uses Pd to cover a wide range of computer music techniques and DSP topics in a tutorial with over 350 examples. The final goal is to strongly depend on ELSE to patch the examples from the tutorial.

1. Paper Structure.

Before describing and discussing the ELSE library, the paper first contextualizes the current state of Pd in section 2, so it is made clear where and how the ELSE library fits in.

Section 3 describes the motivations and goals of the ELSE library as a counterpoint to the previous section. Section 4 describes details from the and section 5 presents the final discussion and further work.

2. Contextualization: The current state of Pure Data and its external libraries.

Pure Data[1] is a visual programming language, quite similar to Cycling 74's Max[2], which is a commercial software. One of the main differences is that Pd is free an open source. Other open source environments for computer music (such as Csound[3], Chuck[4], SuperCollider[5]) are not visual, but textual instead. This makes Pd the only of its kind (an open source visual programming language), not to mention one of the most widely used.

An indicative of the user base population of these systems may be the size of its community on Facebook. The Pure Data community[6] has over 11.000 members, while SuperCollider's[7] has less than 5.000. Max[8], despite being a commercial software, still has the biggest community, with over 14.000 users.

The visual programing paradigm may be more intuitive and comfortable for non programmers (such as artists and musicians), making it more popular and known outside the computer music niche. This may explain the great number of the user base of both Pd and Max over SuperCollider. But a big user base does not imply in a proportionally large community of developers.

Most of the people involved in the SuperCollider community are experienced programmers, so they can collaborate much more to its development. Not surprisingly, the SuperCollider community is much more active than Pd's. Not only that, but its development is

1 Link: https://puredata.info/
2 Link: https://cycling74.com/products/max
3 Link: http://csound.github.io/
4 Link: http://chuck.cs.princeton.edu/
5 Link: http://supercollider.github.io/
6 Link: www.facebook.com/groups/4729684494/
7 Link: www.facebook.com/groups/supercollider/
8 Link: www.facebook.com/groups/maxmspjitter/

not centralized, while the development of the main distribution of Pure Data (Pd Vanilla) has always been centralized around Miller Puckette, its main author and distributor.

Pd Vanilla comes with a minimal set of objects. Miller Puckette has been developing Pd for over 20 years, in a notoriously slow pace, but as part of a conscious decision. Miller will only include a fix or a new functionality to Pd when he is certain that it is the best choice and won't need changes in the future. He aims for a canonical implementation of functions and is concerned that Pd patches will all run 30 years from now[9].

This orientation ends up promoting not only the development of external libraries for Pure Data, but also other distributions – or "flavors" – of Pd. One important parallel distribution named "Pd Extended" appeared in the early 2000's. It provided, besides the basic Pd Vanilla set of objects and GUI, a few other functionalities and dozens of external libraries already included.

Pd Extended had a major role in making Pd popular, but it has unfortunately been abandoned by its main author: Hans-Christoph Steiner. Its latest version, released in 2013, is very outdated when compared to recent Pd Vanilla developments. That notwithstanding, many still use Pd Extended as it is still distributed and considered convenient.

The main feature of Pd Extended is surely its provided external libraries. Therefore, with the abandonment of Pd Extended, the Pd community decided it was best to, instead of further developing on it, make it easier to install the external libraries from Pd Extended into Pd Vanilla. As a result, Pd now has an external installer since version 0.47-0. Although this did partially compensate the abandonment of the Pd Extended distribution, there is a bigger underlining issue, which is the fact that the vast majority of the external libraries available in Pd Extended hasn't had a maintainer or developer for a long time.

Thus, even before Pd Extended's last release, most of its main features and differentials had been long abandoned. Furthermore, there was not a big criteria for including libraries into Pd Extended. This means that some external libraries were included still in an experimental or early stage of development and were never further developed. The outcome is that you can easily find bugs and badly documented externals. So fixing, maintaining and developing for such libraries would be more important than bringing Pd Extended back to life.

However, so to speak, Pd-Extended has recently reincarnated as "Pd-L2ork 2.0 / Purr Data"[1], which started as a fork of Pd Extended in 2009 and became the Pd-L2ork distribution (but only for Linux). The Pd-L2ork 2.0 version is cross platform and was released in early 2017 by the name "Purr Data". It does have all the libraries and features from Pd Extended plus more of its own. But the issue of including mostly abandoned libraries it inherited from Pd Extended still remains.

3. ELSE's motivation and goal

Pd Vanilla offers a minimal package for computer music but has many external libraries that can be easily installed via its external manager, not to mention the Purr Data distribution, which provides several external objects already built-in. As follows, one could reason that Pd operates in a modular structure, relying on declaring extra packages that may be imported when needed. In this way, external libraries should compensate the lack of essential features in Pure Data Vanilla, not leaving it behind other environments.

However, that is not what we have in practice. Besides the fact that most libraries are simply abandoned nowadays, the collection of external libraries in Pd Extended presents a big kludge instead of an organized set of dedicated external packages. Most are just relatively small random sets of functions, which often carry virtually the same functionality of other externals. Hence, Pd's externals libraries needed a major overhaul and revision in order for it to claim an actual modular structure.

From the few libraries in Pd Extended that are still in active development, the main and largest one is Cyclone. Though it did spend about a decade with no significant development,

9 As stated in this video from the Pd Conference 2016, about the future of Pd. Specially from 53:52 up to 59:30 Link: https://www.youtube.com/watch?v=cAWFk1PPTtk&feature=em-lss

it went through new development phases recently, and it's under the maintenance of Alexandre Torres Porres, Derek Kwan and Matt Barber since february 2016 [2].

Cyclone clones MAX/MSP to Pure Data, being popular for providing some level of compatibility between the two environments. Its latest version (cyclone 0.3, still in the beta phase) has about 200 objects and should be merged to Purr Data soon.

In this context, the ELSE library aspires to provide a substitute for many abandoned Pd libraries, in a quite ambitious goal to centralize many essential elements into a single library, getting rid of the need to search amongst several libraries for a simple and essential object.

The initial motivation of the ELSE library was actually to include important functions that were not available in Pd Extended/Purr Data libraries. This need emerged from an educational project, which is the development of a tutorial for computer music based on Pure Data by the author[10], which now has over 350 examples and covers a wide range of computer music techniques and DSP topics [3-4].

The author has been developing this original tutorial to teach computer music for 9 years, up to a point where all the relevant existing objects in Pd Extended were used up, so there was a need for something "else".

A parallel idea was to clone the UGENs from SuperCollider, creating a dedicated library of clones such as Cyclone. But this was shortly abandoned as it would constrain to the same architecture as the original UGENs, and soon it was decided that these functions should have different design choices. One reason is the consideration that some of the original design choices were not good, and the other one is that there would have to be adaptations to the context of the Pd environment anyway. therefore, this idea was merged into the ELSE project, although a good name for a SuperCollider clone library was already thought of (SuperClonider).

A further expansion of the original scope – as already mentioned – includes the revision of many externals from Pd Extended. This decision was motivated by the fact that such externals have been abandoned and could benefit from revisions into a new library, so the didactic material by the author would rely mostly on externals maintained by himself.

Now, since the author is also a main collaborator of Cyclone[11], the idea is that the didactic material will rely heavily on both of these libraries. But for the great majority of other externals from Pd Extended used in the examples, the ultimate goal is to replace them all with a newly written external from ELSE.

It needs to be noted that even functions from objects already available in Cyclone or Pd Vanilla are also being revised by objects in ELSE, such as the case with oscillators, described in the next section.

4. The ELSE library

This section describes some of the features and objects from the ELSE library. At the time of its writing, the project is still in an alpha stage and counts over 110 objects, but a first beta release is planned to be made available by the time of the SBCM conference[12]. The following subsections describe important objects, groups of objects, and features of the ELSE library.

The name of the library stands for *EL Locus Solus' Externals*, where "EL Locus Solus" is the independent center run by the author. It functions as a production agency and a school that offers computer music courses[13].

4.1 Oscillators

The oscillators available in ELSE bring together functionalities not available elsewhere. Not only in other Pd externals, Max's and SuperCollider's internal functions, but also in other computer music languages such as Csound and Chuck and even in a modular environment like Reaktor[14]. Namely, they offer

10 Available for download at: <https://github.com/porres/LiveElectronicsTutorial>

11 The latest releases of Cyclone are available at: <https://github.com/porres/pd-cyclone>

12 The latest releases of ELSE are available at: <https://github.com/porres/pd-else>

13 Link: <http://alexandre-torres.wixsite.com/el-locus-solus>

14 Find Reaktor at <https://www.native-instruments.com/en/products/komplete/synths/reaktor-6/>

hard sync, phase modulation and accept negative frequencies.

The basic oscillators in Pure Data (such as [osc~]) have a secondary control inlet for reseting the phase. If it were a signal inlet, a signal input could trigger the oscillator to perfectly reset in sync to another frequency (i.e. hard sync technique).

The oscillators in ELSE follow the usual design from vanilla objects, in which the first inlet controls the frequency and the first secondary inlet resets the phase. The phase can be reseted in the same way as with [osc~], with a control message, but it also works with a signal impulse with a different logic.

Having a separate action for both signal input (sync at impulses) and control messages poses a challenge. It is not only unusual in Pd but it's also very tricky to implement it in the Pd API. For that, the author applied a technique developed by Matt Barber for Cyclone, nicknamed "magic" [2].

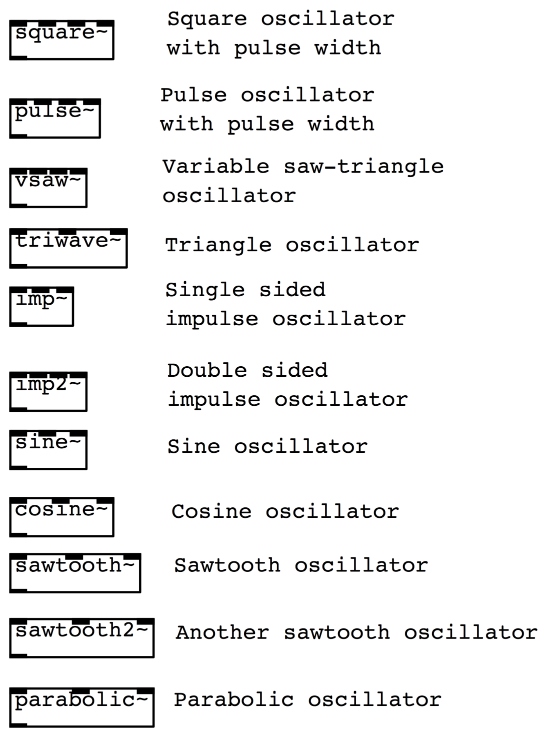

square~
Square oscillator with pulse width
pulse~
Pulse oscillator with pulse width
vsaw~
Variable saw-triangle oscillator
triwave~
Triangle oscillator
imp~
Single sided impulse oscillator
imp2~
Double sided impulse oscillator
sine~
Sine oscillator
cosine~
Cosine oscillator
sawtooth~
Sawtooth oscillator
sawtooth2~
Another sawtooth oscillator
parabolic~
Parabolic oscillator

Figure 1: Oscillators in ELSE

4.2 Sample accuracy and impulses

Many Pure Data audio objects are unable to be triggered with sample precision/accuracy. The same can be said about Max, though to a lesser extent. Dealing with sample accurate processes in both Max and Pd is tricky, being more common to use control messages (either general control messages or bangs) instead. This is surely more efficient computationally, but some computer techniques depend on controlling parameters at audio rate – such as "hard sync".

Implementing "hard sync" in Pd is not impossible, but it's just not elegant, as it is a workaround that could be avoided if only the oscillators were designed to accept a sample accurate phase reset control.

Objects from the ELSE library are strongly aimed to sample accurate processes, for which an impulse oscillator is very important. Objects can then be programmed to respond when a trigger signal is different than 0, or when they transition from 0 to a positive value, which is exactly what an impulse oscillator provides at a fixed rate, so it can be thought of as an audio rate metronome.

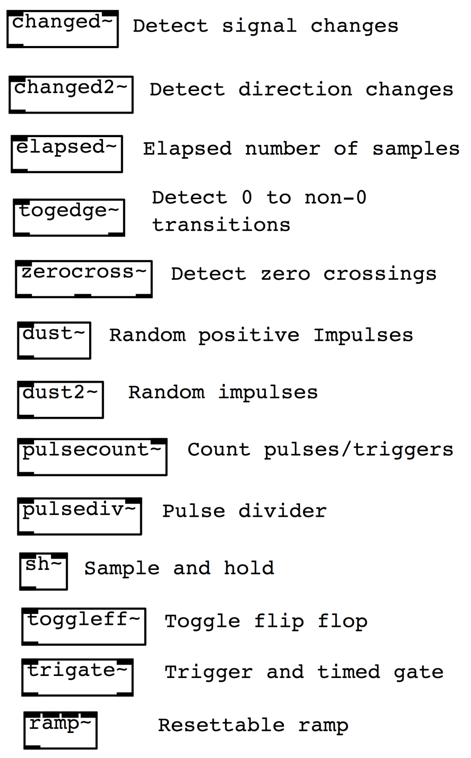
Since Pd Vanilla was not well planned for sample accurate processes, it does not have an impulse oscillator. Max's [train~] (also available in Cyclone) is the closest thing to it, but not an actual oscillator.

SuperCollider is more oriented towards sample accurate processes, and thus has an Impulse oscillator UGEN, but it does not accept negative frequencies. Pd Extended/Purr Data has [impulse~], an impulse oscillator from the sigpack library, but it has bugs and also does not accept negative frequencies.

Therefore, [impulse~] is one of the objects already available in the Pd ecosystem that needed a revision inside ELSE. It is one of the few that has the same name as a pre existing external. But in such cases, the provided externals in ELSE are fully backwards compatible to the existing ones.

Thus, [impulse~] from ELSE solves the existing bugs in [impulse~] from sigpack and provides more features (negative frequencies, phase modulation and hard sync) – note that the [impulse~] object in ELSE can also be instantiated as [imp~], for short, as in figure 1.

Following this sample accurate structure, many objects in ELSE are triggered by impulses, such as the oscillators' hard sync inlet, or negative to positive transitions. Not only that, but other objects in ELSE also generate impulses as result of signal analysis, such as [zerocross~] (when detecting zero crossings), [changed~] (when detecting signal change) and more, see figure 2 below.



changed~	Detect signal changes
changed2~	Detect direction changes
elapsed~	Elapsed number of samples
togedge~	Detect 0 to non-0 transitions
zerocross~	Detect zero crossings
dust~	Random positive Impulses
dust2~	Random impulses
pulsecount~	Count pulses/triggers
pulsediv~	Pulse divider
sh~	Sample and hold
toggleff~	Toggle flip flop
trigate~	Trigger and timed gate
ramp~	Resettable ramp

Figure 2: Objects that either respond to or generate impulses.

4.3 [sh~] and [downsample~]

The [sh~] object from ELSE is a *sample and hold* module that is triggered by impulses. Any sample and hold object needs a trigger signal input to make it “sample an input value and then hold it”. Therefore, it is inherently a sample accurate process.

Pd Vanilla has its own sample and hold object, named [samphold~]. But since Pd Vanilla isn’t aimed towards sample accurate processes and lacks an impulse generator, [samphold~] was designed to be triggered with [phasor~].

This is not a conventional design, but a workaround in order to adapt a sample and hold unit in an environment that lacks impulse trigger signals. The way [samphold~] works is that it is triggered whenever its right inlet input decreases in value. This happens with [phasor~] at period transitions, allowing it to trigger [samphold~] at that moment, but only when it has a positive frequency smaller than the sample rate.

The most critic issue is that [phasor~] is useless to trigger [samphold~] when it is running with negative frequencies, since what happens in negative frequencies is that [phasor~] inverts its direction and outputs a downwards ramp instead. Therefore, apart from the period transitions, it always outputs a value smaller than the previous one – exactly what triggers [samphold~], so it can’t work that way.

Max’s sample and hold object is [sah~], also available in Pd via Cyclone. It has a more common design, allowing it to be triggered with an impulse oscillator – since it is triggered when the signal raises above a given threshold. The [sah~] object also requires that the signal falls below the threshold so it can be triggered again. Therefore, the highest rate [sah~] can be triggered is the nyquist frequency.

The [sh~] object from ELSE is similar to [sah~], but it may also not require the signal to fall below the threshold value (this is actually done by default). This allows it to be triggered with an impulse oscillator up to the sample rate frequency. As such, it may also act as a “gate or hold”, where instead of a trigger signal, you have a gate signal that allows the input signal continuously through when it is higher than the specified threshold. This is, in fact, similar to SuperCollider’s Gate UGEN.

The [sh~] object combines features from SuperCollider’s Gate, Max’s [sah~] and Pd’s [samphold~], making it the most complete and versatile sample and hold object design. The conjunction of ELSE’s [imp~] and [sh~] objects also has more potential than previously available.

One possible application is downsampling, where it can go up to the sample rate frequency (positive or negative). But the [downsample~] object from ELSE already does this for you. One can think of it as a sample and hold object that only needs to receive a trigger rate in hertz (negative frequencies also accepted). So you’d only need to implement downsampling with [imp~] and [sh~] if you wish to use other features from these two objects. One difference, though, is that [downsample~] can also interpolate between values.

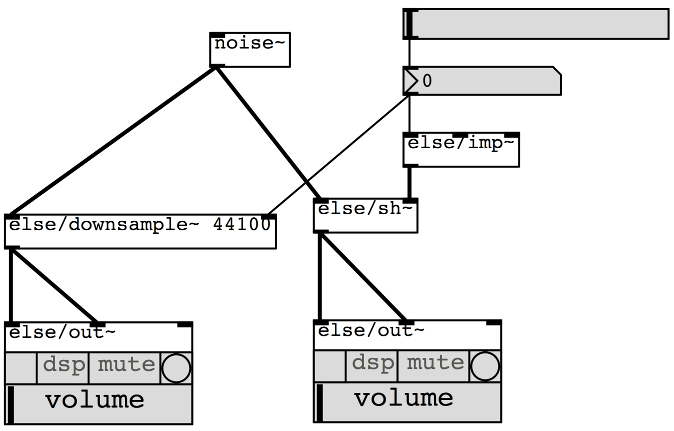


Figure 3: [downsample~] and an equivalent patch with [sh~] and [imp~]

4.4 [pimp~]

The [pimp~] object from ELSE is a combination of Pd Vanilla's [phasor~] and ELSE's [imp~] (or [impulse~]). It's basically [imp~] with an extra phase output in the left outlet. Conversely, it can be thought as a [phasor~] with an extra impulse output in the right outlet.

The impulse happens at every phase period transition (in both negative or positive frequencies/directions). Like [imp~], [pimp~] also has inlets for phase modulation and sync.

[pimp~] is very convenient for both driving a process with its phase output (such as reading a sample or envelope wavetable) and also triggering other objects at period transitions.

The need for such an object was found when dealing with granulation patches. For instance, the *B13.sampler.overlap* example, from the audio examples from Miller Puckette's book [5] (which are available in the Pd Vanilla distribution) uses [phasor~] to generate overlapping envelopes and also trigger [samphold~] objects, which samples new parameter values at the end of the envelope.

This is the basis of other granular based patches such as the time stretch/compress with independent pitch shifting from the next example in the series (*B14.sampler.rockafella*).

But this is only possible if you're reading the table/sample in the original direction. The patch does not work if you're reading the table backwards, as it would require the [phasor~] to run at a negative frequency, and [samphold~] simply does not work that way, as already explained.

The workaround here would be to analyze the signal from [phasor~] and generate an impulse signal when transitioning from one period to the next, in both positive or negative frequencies. But even so, that would require a Pd external such as [sah~] for the sample and hold function, since [samphold~] cannot be triggered like that.

The way you can generate impulses from [phasor~]'s period transitions, wether its frequency is positive or not, is by realizing that whenever it reaches the end of its period, there is a big leap in value. So you can measure the absolute difference between the current and the last value, and we can capture every period transitions up to the nyquist frequency (positive or negative) whenever it is higher than or equal to 0.5.

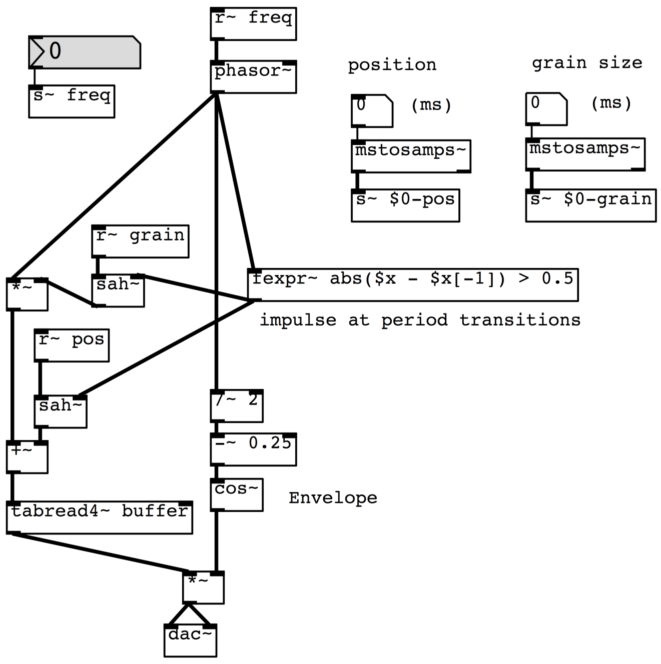


Figure 4: Example of granulation by getting an impulse from period transitions of a [phasor~] and relying on the [sah~] external.

This is not unreasonably hard to implement in Pd and can be done just with one [fexpr~] object (see figure 4 above), but it’s just not elegant. Again, all these workarounds could be avoided if simply the design choices of the objects were better.

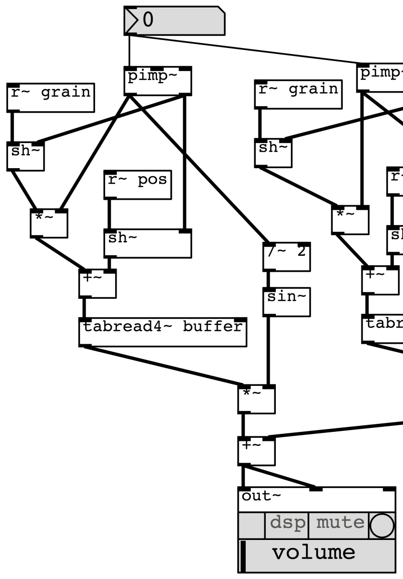


Figure 5: a simplification of the granular patch by adopting objects from ELSE such as [pimp~]

Objects from the ELSE library previously mentioned can simplify this, such as [imp~] in conjunction with [sh~] or just [downsample~]. But [pimp~] is also provided as a single and more elegant solution for this particular purpose. Check the adaptation in figure 5, and how the patch is now much cleaner and also able to include two overlapping grains into a more concise space.

4.5 Chaotic and Stochastic generators

All of the chaotic/stochastic generators in SuperCollider are being cloned. For now, these are: [brown~], [clipnoise~], [crackle~], [cusp~], [gbman~], [henon~], [lincong~], [logistic~], [latoocarfian~], [quad~], [standard~]. Apart from those objects available in SuperCollider, the Ikeda attractor has also been implemented as an extra generator (named [ikeda~]).

The [lfnoise~] object is a low frequency chaotic generator. There are similar objects to [lfnoise~] for Pd, but with less functionalities. In Max’s (and Cyclone) you have [rand~]. The zexy library for Pd has [noish~] and [noisi~]. SuperCollider has the LFNoise generators. But [lfnoise~] differs for accepting negative frequencies and allowing sync with impulses, following the same structure of ELSE’s oscillators. Else also provides [random~], which generates random signal values when receiving a trigger signal.

4.6 Filters

Pd Vanilla lack many basic filters, relying mostly on [biquad~] to generate general second order filters. But even so, it lacks a biquad coefficient generator and only accepts control input for the coefficients.

The ggee library for Pd offers several generators for [biquad~] – covering elementary filters such as lowpass, lowshelf, bandpass, and so on – but the parameters are still constrained to control messages, not allowing signal rates for simple filter sweep patches with a LFO.

Max also relies on a [biquad~] object, but at least it has signal rate input and offers a biquad coefficient generator with signal rate inlets: [filtercoeff~]. Both have been recently cloned into Cyclone, but it was considered necessary to create simpler and friendlier solutions for such elementary filters in ELSE, which would be single objects for each filter type.

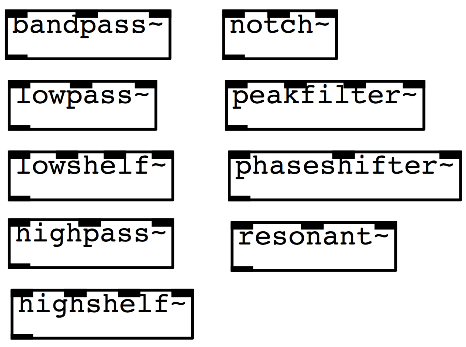


Figure 6: second order filters from ELSE

Most of such filters are made available in SuperCollider’s BEQSuite group of filters as single classes. The difference in ELSE’s objects is that, instead of a reciprocal Q, it has a more intuitive and friendly design to allow the resonance parameter to be set in either Q or bandwidth in octaves. The filter formulas are from Robert Bristow-Johnson's work[15].

5. Final Discussion and further work

This paper briefly presents and describes the ELSE library for Pure Data. The development of this library follows an experience of over a decade in patching with Pd Extended, and about 9 years of developing didactic material based on Pd Extended patches.

More recently, the author has also studied and used SuperCollider and Max to teach computer music. Throughout the course of these years of patching and teaching a wide range of computer music techniques with the existing tools in these three environments, the author accumulated many issues and frustrations with the design of the available building blocks.

A discussion and details behind the creation of some of the externals is given in this paper. The discussion could be furtherly detailed in order to describe every decision behind the design of all the objects in the library, but the given examples make the general purpose clear, which is the need found by experience to improve some elementary building blocks of computer music and include new functionalities that were missing.

Given the current state of Pure Data, with many of the existing libraries abandoned, the development of this library also aims to replace the need of many of the existing externals.

The author has also discussed the inclusion of ELSE in Purr Data as a newly available library as soon as a more stable release of ELSE is made available, and after Cyclone is first included there.

The ELSE library has a didactic motivation. Its ultimate goal is to be the applied in the development of computer music examples from the didactic material developed by the author, alongside the Cyclone library, also under the maintenance of the author and already being included in Purr Data soon in its latest state.

Currently, over 350 examples have been developed making extensive usage of available objects in Pd Extended. As soon as a first version of the ELSE library is made available and included in Purr Data, the next step is to adapt this didactic work to Purr Data and start relying on existing externals from ELSE more and more.

Short term plans for ELSE include revising the documentation and adding band limited functionalities for the oscillators. As soon as this is finished, a final 1.0 version may be released.

15 Cookbook formulae for audio equalizer biquad filter coefficients, by Robert Bristow-Johnson, available in <http://shepazu.github.io/Audio-EQ-Cookbook/audio-eq-cookbook.html>.

Acknowledgements

Flávio Luiz Schiavoni, for being the first to help me get into the art of coding externals for Pd and for coding the first version of [median~], the first object from ELSE to be coded. Derek Kwan and Matt Barber, my fellow collaborators in Cyclone, that also teach me so much and help me a lot. Ivica Bukvic and Jonathan Wilkes for being open to accept my ELSE library in Purr Data as soon as it is stable and ready for it. Miller Puckette, for being awesome and being the giant who created Pure Data and Max, plus all the others in the Pd Community, who are always very helpful.

References

[1] I. Bukvic, J. Wilkes and A. Graef, *Latest developments with Pd-L2Ork and its Development Branch Purr-Data*, PdCon16, NYU, 2016.

[2] A. Porres, D. Kwan and M. Barber, *Cloning Max/MSP Objects: a Proposal for the Upgrade of Cyclone*, PdCon16, NYU, 2016.

[3] A. Porres, *Teaching Pd and using it to teach: yet another didactic material*, PdCon09, São Paulo, 2009.

[4] A. Porres, *Patches de Pd para Computação Musical*, Revista Vórtex, Curitiba, v.2, n.2, 2014, p.198-200.

[5] M. Puckette *The Theory and Technique of Electronic Music*. World Scientific Press, 2006.