Camomile-ELSE: creating plugins with Pure Data patches and the ELSE library

Astorga, E.M.V., Porres, Alexandre Torres

Núcleo de Pesquisas em Sonologia – Escola de Comunicação e Artes da Universidade de São Paulo
Av. Prof. Lúcio Martins Rodrigues, 443 – 055088-020, São Paulo, SP

EL Locus Solus (independent center) São Paulo, SP

esteban.astorga@usp.br, el.locus.solus@gmail.com

Abstract

Camomile is an open source project that turns patches made in the Pure Data programming language into plugins (VST, VST3, LV2 and Audio Unit) so you can load them in any plugin host on Windows, Linux and MacOS. We have a fork of this project that embeds the ELSE library of Pure Data externals into camomile so you don't have to only rely on the limited set of native Pure Data objects.

This paper describes the potential of the Camomile project, how it makes the creation of plugins very accessible to people without a background in computer science and also how this process is improved by including the extensive ELSE library. We also offer an overview of similar projects and the technical details of including the ELSE library into the camomile project.

1. Introduction

Pure data (a.k.a Pd) is an Open Source cross platform visual programming environment for interactive computer music and multimedia created by Miller Puckette in the mid 1990s. Pd is distributed for Linux, macOS and Windows. Pure Data has also been ported to iOS (Pd Party) [1] and Android (PdDroidParty) [2] operating systems thanks to libpd[1] [3].

Pure Data offers a very limited set of native objects, it misses lots of functionalities if compared to other similar environments such as Csound, SuperCollider and MAX. The Pd community has always relied on a large collection of external libraries and this also promoted forks of Pure Data that include pre-installed libraries. Nonetheless, you can currently easily install externals into the original Pure Data distribution (also known as 'Pd Vanilla').

On the other hand, Pure Data for iOS and android are only 'Vanilla' (meaning, no externals). It's not that you can't use externals with libpd, but most of the projects based on libpd are offered compiled only to include the native Pd objects. The Camomile project is yet another project that relies on libpd to embed Pure Data into an application. By default, it only allows you to use Vanilla objects. Actually, not even that as it misses some objects like [pd~] and the graphical Data Structures objects.

We have forked Camomile to compile it with externals from the ELSE library [4], which is a library that's still in an early and experimental phase, but with a huge set of externals. ELSE has been in active development since the begging of 2017 and a first beta version was presented in SBCM that year [6]. At the time of this writing, ELSE is at beta version 42.

2. The breakthrough of Camomile (History)

Bridging Pd with other softwares is not hard via softwares like jack and MIDI bridges, but this can be clumsy and limiting when trying to bring Pure Data into the workflow other softwares that process audio. Hence, a proper integration like MAX has with MAX4LIVE has always been and attractive goal. One big advantage is to render or bounce the project inside your desired software with your Pd patches as part of the process.

Attempts in that direction start with Radium, which is a DAW[2] with a tracker like interface and has Pd embedded in its Linux version, allowing us to run pd patches since 2013. Later, in 2016, we first had the possibility of running pd patches in any DAW via VST support with PD Pulp [7], which is basically a VST plugin (built using JUCE [8] and libpd) that runs Pd patches. Its interface has 10 knobs that can be assigned to any parameter in the patch and buttons to manage the patch loading. PD Pulp's limitations are the lack of MIDI support, the impossibility to use more than one instance of the plugin and the static interface which doesn't allow any customization by the user. PD Pulp was abandoned after Camomile was released by Pierre Guillot in 2016 [9].

Camomile solves most of the problems PD Pulp had and offers more features. In the first public release[3],

[1] LibPd turns Pd into an embeddable audio synthesis library that can use to create other applications

[2] Digital Audio Workstations, a kind of software that allows multi music production with track recording

[3] See: https://web.archive.org/web/20210711174627/https://forum.pdpatchrepo.info/topic/9884/camomile-an-audio-plugin-that-loads-pure-data-patches

Camomile could already deal with multiple instances of the plugin with a wrapper written by Pierre Guillot, because at that time LibPd had this limitation which was solved after this release[4]. Camomile also has MIDI support and provides the ability to create the plugin's interface. This opened the possibility for developers to design their own plugins and distribute them without the need of a Pd patch, and users can be totally unaware it was implemented with Pure Data.

Camomile is also built with LibPd and JUCE and can be used to compile and create VST3, AU and LV2 plugins in Linux, Windows and macOS operating systems. This covers virtually all the DAWs in use these days, plus other hosts like sound editors and other softwares.

Camomile allows the regular Pd user to create and distribute its own plugin without a background in computer science. However, the process of compiling isn't an obvious thing to all Pd users. The way Camomile compiles plugins is simple enough because knowledge about the JUCE environments isn't required and can be done with a minimum set of instructions, provided by tutorials available in the Wiki session of the official Camomile's repository. Compiling scripts are provided for Linux and MacOS, but Windows compilation, at the moment, needs this set of instructions (which is also valid for all operating systems).

In short, Camomile provides support for multiple instances, basic Pd GUI objects (toggle, slider, radio, comment, numbox and array graph), up to 64 automatable parameters with automatic recognition (name, label, range, minimum, maximum, etc.), MIDI in and out, play head position, BPM and up to 16 audio channels.

One current limitation is the lack of support for externals, forcing developers to work with the limited set of Vanilla objects. So we have developed a version of Camomile that includes objects from the ELSE library, which makes the process of creating plugins much easier and friendlier.

3. The ELSE library

The ELSE library [4] is a huge collection of externals and provides many ready made building blocks of computer and electronic music, such as filters, oscillators, noise generators, envelopes, sequencers, random generators, tools for algorithmic composition, etcetera…

While most of these could be implemented with Vanilla objects, the amount of work would be quite huge and you'd need a deep knowledge of Digital Signal Processing and Pure Data. Not only that, but ELSE also provides things you cannot have with only Pd Vanilla.

ELSE makes the patching process much easier and accessible to the general public. Since Camomile's main selling point is that it allows people without a background in computer science to develop plugins, the ELSE library takes this premise to a whole new level, where you don't have to be a Pd expert and you can more intuitively program and create instruments, effects, MIDI processors and controllers.

ELSE has a structure that resembles the one of modular synthesizers and also has many ready made effects (chorus~, flanger~ and phaser~ for instance), plus advanced algorithmic composition tools like markov chain.

Originally, ELSE also comes with a vast computer music tutorial, with over 430 examples that covers a wide range of synthesis, algorithmic composition and DSP techniques. All of it just has a single dependency of ELSE externals, so you can easily create plugins with all that is provided in this tutorial.

4. Technical Details

In order to provide a version of Camomile-ELSE [5], Camomile with support for ELSE externals we had to recompile Camomile with the externals' code embedded into Camomile's source code. Camomile's main dependencies are JUCE and LibPd. JUCE provides classes to manage audio I/O, audio processing, media reading and writing and the hooks to compile plugins VST, LV2 (for Linux, Windows and MacOS) and AU plugins (MacOS only). All our compiling tests are done using a MacOS machine, a Virtual Machine with Ubuntu 12.04 LTS and another old notebook running Windows 10.

The process of setting Camomile to be compiled with ELSE externals is documented in Camomile repository[5], it consists in adding the externals from ELSE into LibPd and then set JUCE to compile Camomile with these dependencies. Some modifications to the original code of ELSE were necessary in order to conform with limitations and conflicts with the Windows compiler.

The ELSE library consists in a collection of externals that are binaries (compiled from C code) and abstractions (Pd patches, where many use objects from the ELSE library). The binaries are compiled with LibPd and this procedure makes Pd see the ELSE externals from ELSE as if they were Pd native objects. This means you cannot load the externals with namespaces (as in else/sine~), but originally all the abstractions from ELSE that has externals from ELSE use namespaces in order to

[4] See: https://web.archive.org/web/20210711174255/https://github.com/libpd/libpd/issues/13

[5] See: https://github.com/pierreguillot/Camomile/issues/214#issuecomment-704670696

force loading the correct object. So we have a specially modified abstraction folder without namespaces and use the [declare] object (as in: [declare -path else]). This allows Camomile to find the abstractions inside the provided modified *else* folder, that you also need to include in your compilation project.

This allows support for all objects in the ELSE library (both binaries and abstractions) but most GUI objects aren't functional because of a limitation from Camomile. Objects from the ELSE library that are binaries needed to be ported like the native Pd GUIs were, and many GUI abstractions use data structures, which are also not implemented in Camomile. The discussion related can be found in Camomile repository[6] and the difficulty consists in reimplementing Pd's Data Structure drawing features in the JUCE interface. So we are looking for other workarounds to provide better GUI support in Camomile.

Camomile-ELSE is compiled on the last stable version of Camomile, 1.0.7. There is already available a beta version 1.0.8 that supports the *pd~* object. This object makes it possible something like open Pd inside Pd, it was created to allow open and control different DSP chains in different threads controlled by operational system. On the last beta version of Camomile, the pd~ object can used to open the Pd application and the patch once you call the plugin. This can be great for Pd users but is not an ideal situation for the distribution of a plugin for non Pd users as it basically just opens Pd inside your plugin host and exposes the inner workings. On the other hand, this process allows you to run Pd with any externals that it can load, because now we're not compiling the externals in camomile anymore, just opening a Pd binary with its included external binaries. Nonetheless, this new camomile feature does a good job by not requiring the user to install Pd, the externals and the patch, because all of this is embedded in the generated plugin.

Besides the non optimal plugin design that just cannot hide Pd as the regular way of compiling plugins with externals, we still need to test and measure if this process has also some technical disadvantages, such as quantifying the increase in latency that this way of loading externals into camomile causes[7]. But one way or another, the possibility to have any externals available like that is very promising and exciting.

One disadvantage with this workaround is the increase of the generated plugins' memory size, once each plugin needs to have a copy of Pure Data and the externals needed embedded in the plugin. To mitigate this issue, it is possible to install Pd and the externals manually on the machine in order to use the plugins. But then, this includes yet more steps and it is defnitely not something you want for a regular plugin user that never heard of Pd. Though it is an option for the Pd enthusiasts.

But since the amount memory we're talking about is not that significant these days, the first workaround can be the usual way to easily include externals into camomile without doing what we're providing here with camomile-else.

5. Conclusion

Pure Data is a cross platform visual programming environment for interactive computer music and multimedia with a friendly visual paradigm that is well suited for artists and non programmers. For this reason, it made the world of computer music more accessible and increased the range of users capable of buildings sounds and compose music with computers.

Pd also provides support to create smartphone apps, create strange devices and installations with microcomputers like Raspberry Pi, new instruments with devices like Bela and so on. Its portability and accessibility is unparalleled. The integration of Pd and any plugin host is now also a reality, which brings the Pd world into the workflow of DAWs and sound editors.

This is true for either the Pd enthusiast that can now easily also work with Pd inside plugin hosts but it also a way for non programmers to design and distribute plugins with Pd patches (without a background in programming or computer science). This opens the doors for the commercialization of open source plugins and a fertile terrain to explore and research computer music and experimental techniques in for yet a broader range of users.

This is where our work with Camomile-ELSE has a big contribution as it allows users to design plugins with an extensive set of externals provided by the ELSE library, as well as the extensive examples available in its included tutorial.

This development effort also opened us to new aims like provide better accessibility in the creation of mobile apps by also compiling ELSE into other libpd projects such as PdDroidParty and PdParty.

Another development from this work would be a basis for other projects to also include other external libraries.

[6] See: https://web.archive.org/web/20210711175310/https://github.com/pierreguillot/Camomile/discussions/256

[7] See: https://web.archive.org/web/20210711175710/https://github.com/pierreguillot/Camomile/discussions/233

References

[1] PdParty see:
https://web.archive.org/web/20210711154116/http://danomatika.com/code/pdparty

[2] PdDroidParty see:
https://web.archive.org/web/20210410181502/https://droidparty.net/

[3] LibPd see:
https://web.archive.org/web/20201125151522/https://github.com/libpd/libpd

[4] ELSE Library see:
https://web.archive.org/web/20210711155857/https://github.com/porres/pd-else

[5] Camomile-ELSE see:
https://web.archive.org/web/20210711180229/https://github.com/emviveros/Camomile-ELSE

[6] Porres, Alexandre Torres. “ELSE Library for Pure Data”, 2017, 8. See:
https://web.archive.org/web/20210711160308/http://compmus.ime.usp.br/sbcm/2017/papers/sbcm-2017-6.pdf

[7] PD Pulp see:
https://web.archive.org/web/20201207051840/https://github.com/logsol/Pd-Pulp

[8] JUCE see:
https://web.archive.org/web/20210708055657/https://juce.com/

[9] Camomile see:
https://web.archive.org/web/20210711163045/https://github.com/pierreguillot/Camomile