TABLE OF CONTENTS

Camomile

Camomile is a set of plugins with Pure Data (http://msp.ucsd.edu/software.html) embedded. The plugins offer to load and to control patches inside a digital audio workstation. The plugins are available as VST, VST3 and Audio Unit for Windows, Linux and Mac.

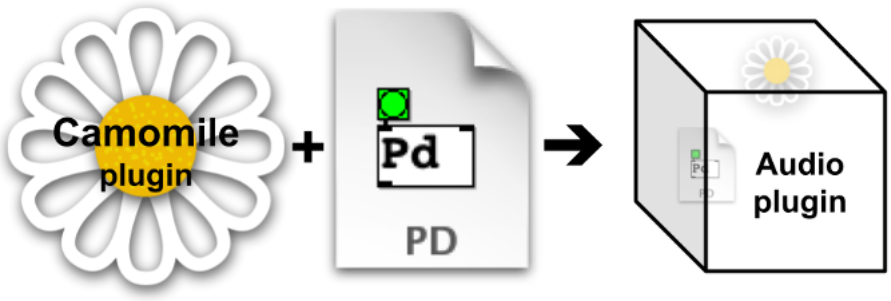
Download the plugins

The current development version of the plugin is 0.1.0 (https://github.com/pierreguillot/Camomile/releases). The current documentation is destined to this version. Even if some minor changes can happen, the general behavior od the version will be preserved.

The last stable version of the plugin is 0.0.7 (https://github.com/pierreguillot/Camomile/releases/tag/v0.0.7-beta). This version will soon become obsolete. Its documentation (https://github.com/pierreguillot/Camomile/wiki/v0.0.7---Instructions) will be never preserved.

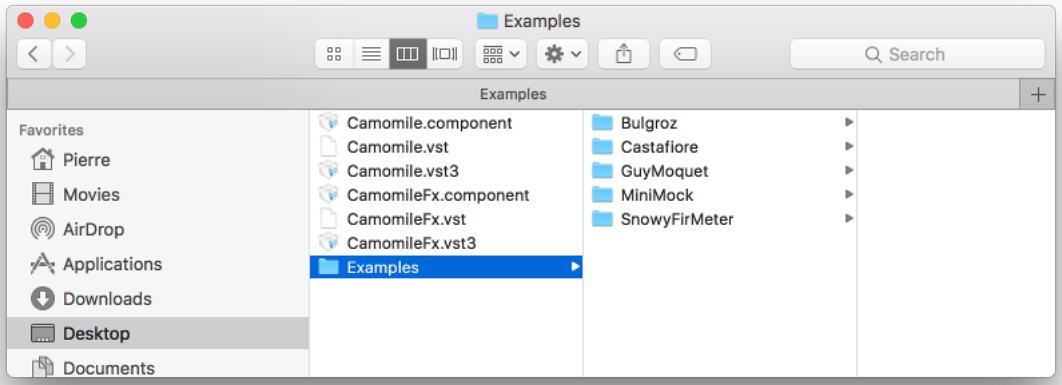
How to generate plugins

The Camomile plugins are a set of meta plugins. It means that the plugins of the distribution can't be directly loaded in a digital audio workstation but must be used to generate new plugins associated with Pure Data patches that will be loadable in the digital audio workstations.

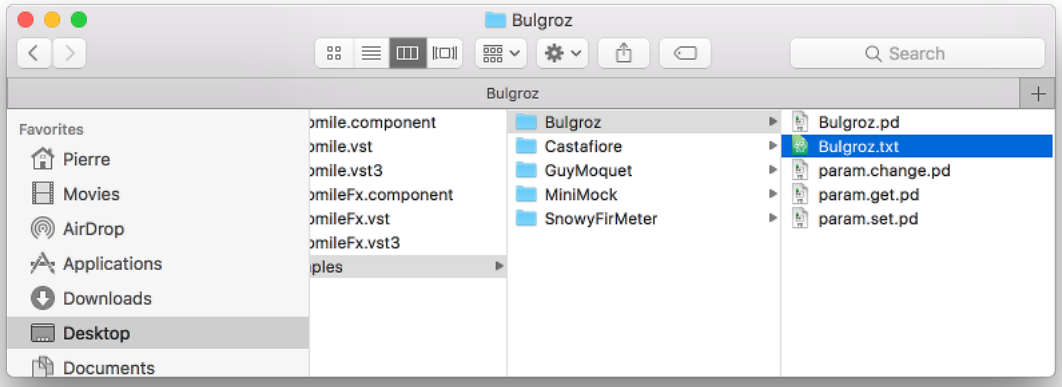


The distribution contains a set of folders that can be used to generate plugins. This tutorial presents how to generate these examples but the approach will be the same for the plugins that you'll create.

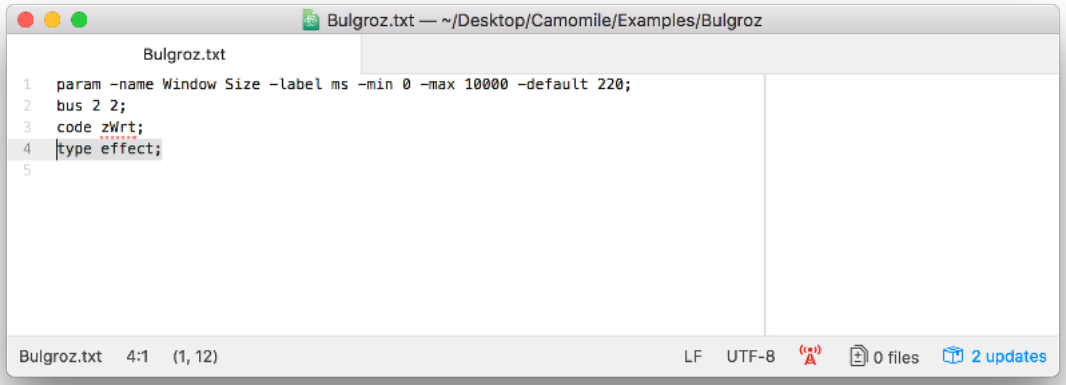
1. Open the *Examples* folder in the distribution. Each folder owns the patches and the informations of a plugin.



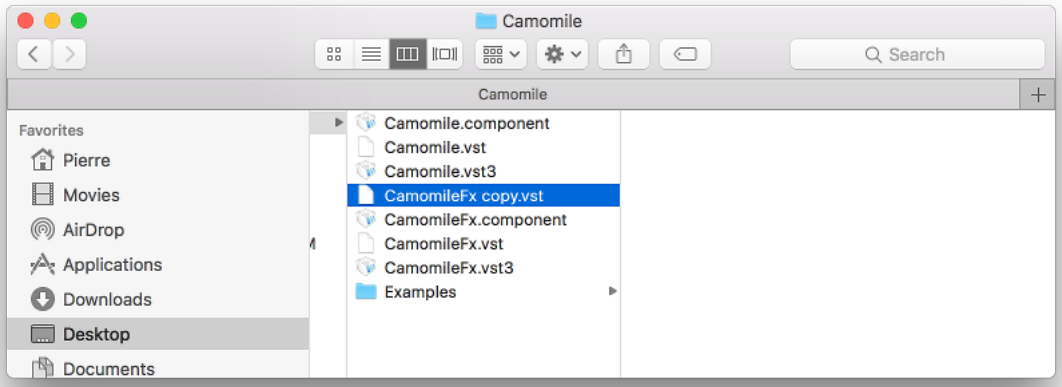
2. Choose the folder of the plugin that you want to generate.



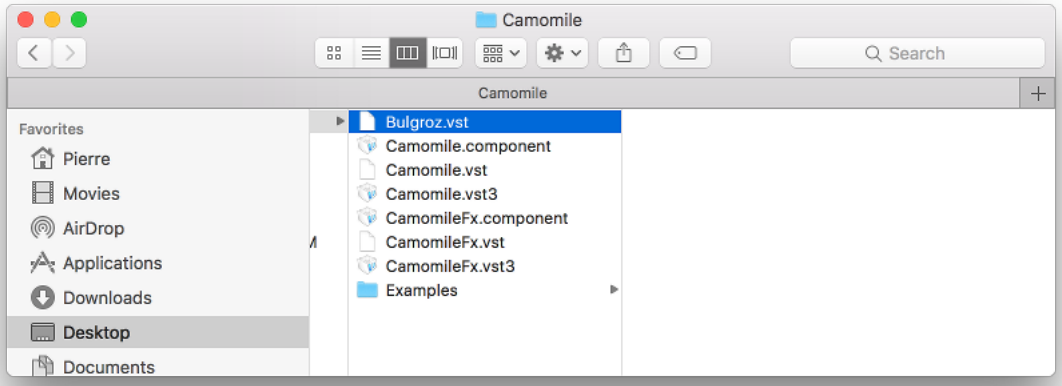
3. Open the *txt* file that has the same name as the plugin and find the line that starts with *type*.



4. If the type is *effect*, copy the *CamomileFx* plugin that you want to create (*.dll*, *.lib*, *.vst*, *.vst3*, or *.component*), otherwise if the type is *instrument* copy the *Camomile* plugin of your choice.



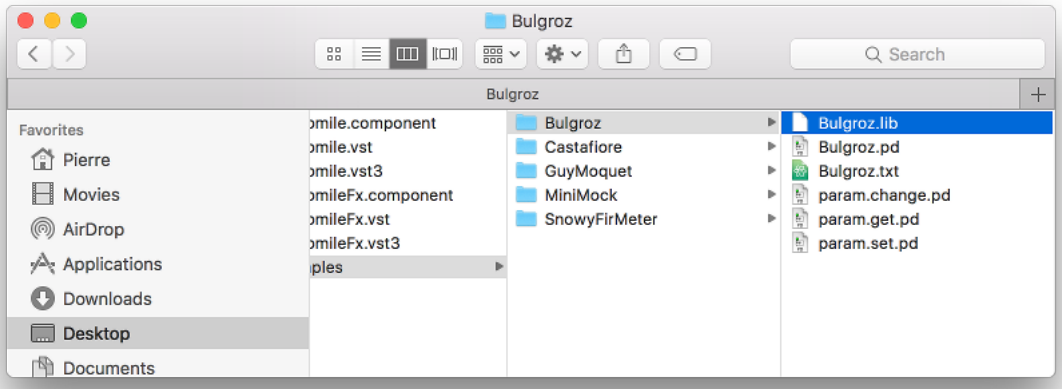
5. Rename the Camomile plugin with the name of the plugin folder.



6. Integrate the patches and the dependencies to the new plugin. The approach is slightly different depending on the operating system (Windows, MacOS and Linux) and the type (Audio Unit or VST).

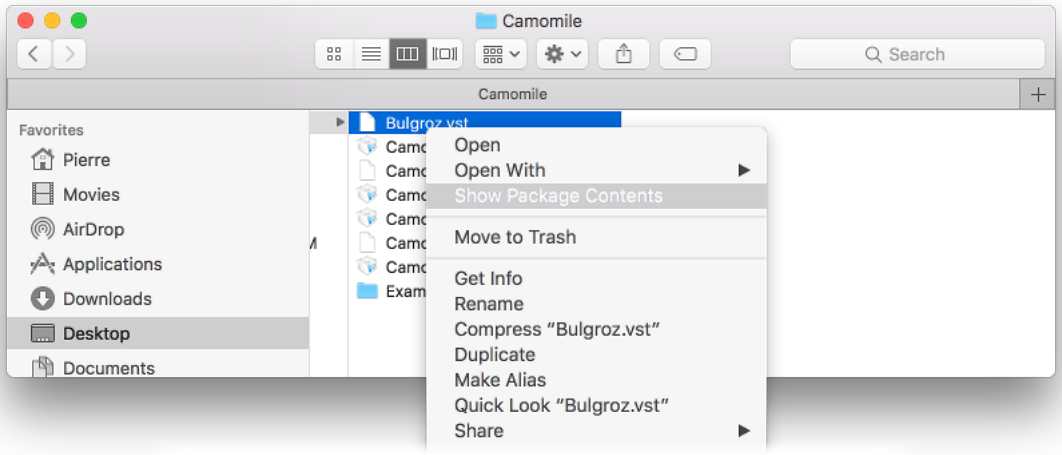
   - Windows & Linux

     Copy the plugin (*.dll*, *.vst3* or *.lib*) in the plugin folder.

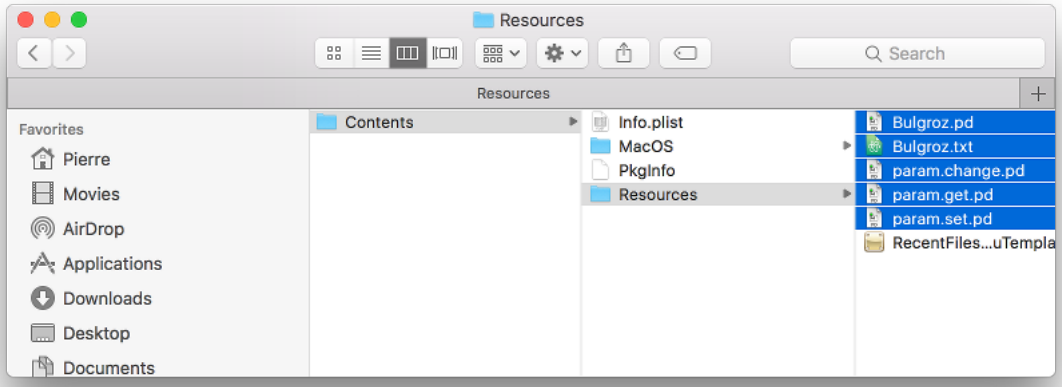


   - MacOS

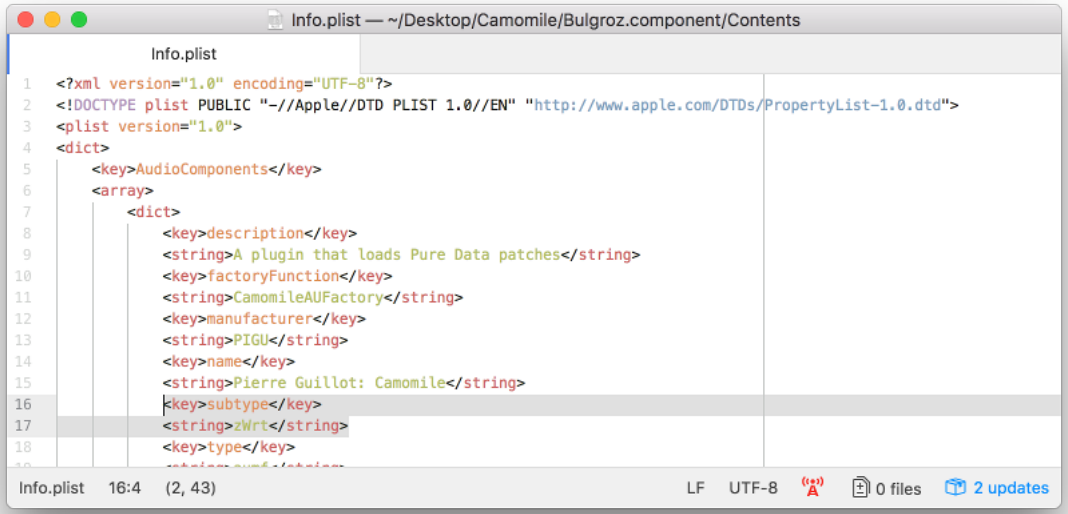
Right click on the plugin (*.vst*, *.vst3* or *.component*) and select *Show Package Contents*.



Copy the content of the plugin folder in the *Resources* folder of the plugin package.



Specific to Audio Unit plugins (*.component*), the *subtype* value of the *Info.plist* must be changed to the *code* value defined defined in the text file of the plugin folder (see 3rd step). This value is a four characters string with at least one upper case character.



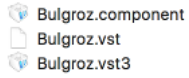
The plugin is ready to be used within a digital audio workstation. Please read the following section if you need help to install the plugin: How to install plugins.

How to install plugins

Like for any plugin, before using plugins the last step is to install them in the audio plugins' location specific to the digital audio workstation (please refer to the digital audio workstation documentation) or to the operating system's default audio plugins' location.

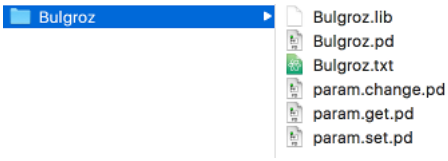
MacOS

Copy the VST (*.vst*), VST3 (*.vst3*) or Audio Unit (*.component*) packages to their respectives locations (Default VST & VST3 location (https://helpcenter.steinberg.de/hc/fr/articles/115000171310-VST-plug-in-locations-on-Mac-OS-X-and-macOS) and Default Audio Unit location (https://support.apple.com/en-us/HT201532)).



Linux

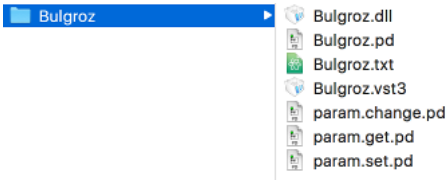
Copy the plugin folder that now contains the VST plugin itself (*.lib*) and its dependencies (the patches, the text file, etc.) to its specific location (Default VST location (http://www.manual.ardour.org/working-with-plugins/getting-plugins)).



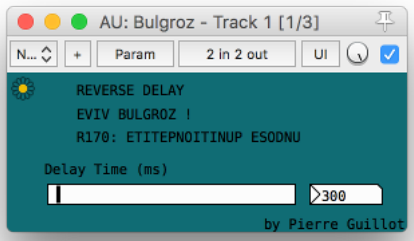
Windows

Copy the plugin folder that now contains the VST (*.dll*) and/or VST3 (*.vst3*) plugin and its dependencies (the patches, the text file, etc.) to its specific location (Default VST & VST3 location (https://helpcenter.steinberg.de/hc/fr/articles/115000177084-VST-plug-in-locations-on-Windows)).

Important: On Windows, the file libpd.dll offered by the distribution must be installed. If you use the 32bits plugins on Windows 32bits or the 64bits plugins on Windows 64bits, the file libpd.dll must be installed in the folder C:\Windows\System32. If you use the 32bits plugins on Windows 64bits (if your digital audio workstation is 32bits), the file libpd.dll must be installed in the folder C:\Windows\SysWOW64.



The plugin is now installed and ready to use. The plugin should be visible in the plugins' list of the digital audio workstation. If not, please ensure that you followed the previous steps to generate and to install the plugin. If the problem persist, look for it in pre-existing issues (https://github.com/pierreguillot/Camomile/issues) perhaps a solution has already been found. Otherwise, feel free to open a new issue (https://github.com/pierreguillot/Camomile/issues/new).



Once loaded, the graphical user interface of the plugin should be visible with a flower button at the top-left corner. If the rest of the graphical user interface only displays the message *Plugin Not Valid*, it means configuration text file of the plugin has been not found. Most likely, an error has been made while generating the plugin. Click on the flower and select the console to display further information (the path where the configuration text file is expected). Here again, ensure that you followed the previous steps to generate and to install the plugin, check for the pre-existing issues (https://github.com/pierreguillot/Camomile/issues) and feel to create a new one (https://github.com/pierreguillot/Camomile/issues/new) if you still need help.

Even if the plugin is well loaded and functional, it is wise to check errors in the console. The plugin notifies if the type defined in the configuration text file is different from the type of the plugin. Other errors can also have been done by the patch creator, in this case you should contact him. At last, if the rest of the graphical user interface only displays the message *No Graph On Parent Available*, it means that the user graphical interface has not been defined in the patch but plugin should still

be functional, the audio engine should work properly and you should have access to the available parameters and presets.

Now that you know how to create and install plugins, you should want to create your own one. Please read the following section: How to create new plugins.

How to create new plugins

Work in progress

How to compile the plugins

The cross-platform dependencies (libPd, Pure Data, JUCE and the VST SDK), are integrated as submodules to the repository. You need to pull the repository and its submodules:

```
git clone --recursive https://github.com/pierreguillot/Camomile.git
cd Camomile
```

MacOS

To compile the AU, VST & VST3 plugins on MacOS, you first need to compile the static version of libPd for multi instances and multi threads. The static library is expected to be in the folder *libpd/libs*. Thereafter, you can compile the Camomile plugins. There two options:

- Manual
  For the first step, you can compile the *libpd-osx-multi* from the project *libpd.xcodeproj* located in the *libpd* folder (don't forget to change the destination folder). Then you can compile all the targets of the the project *Camomile.xcodeproj* located in *Builds/MacOSX*. If you want the Fx version of the plugin, you must change the configuration to *ReleaseFx* or *DebugFx*.

- Command Line
  At the root of this directory, you can do the two following commands

  ```
  xcodebuild -project libpd/libpd.xcodeproj -target libpd-osx-multi -configuration Release ONLY_ACTIVE_ARCH=NO CONFIGURATION_BU
  xcodebuild -project Builds/MacOSX/Camomile.xcodeproj -configuration Release
  ```

  And/or for the Fx version of the plugin:

  ```
  xcodebuild -project Builds/MacOSX/Camomile.xcodeproj -configuration ReleaseFx
  ```

If you want to modify the Camomile project, you should use *Camomile.jucer* with the Juce's projucer application. If you want to compile the Audio Unit, after generating the XCode project, you must change the type of the *include_juce_audio_plugin_client_AU.r* located in the folder *JuceLibraryCode* to *Objective-C++ preprocessor*.

Linux

To compile VST plugins on Linux, JUCE requires a large set of dependencies, to install everything you should do this command (for further information you should refer to the JUCE documentation):

```
sudo apt-get -qq update
sudo apt-get install -y libx11-dev libxrandr-dev libxinerama-dev libxcursor-dev python-dev libfreetype6-dev libgtk-3-dev libcurl4
sudo add-apt-repository -y ppa:webkit-team/ppa
sudo apt-get -qq update
sudo apt-get install -y libwebkit2gtk-4.0-37 libwebkit2gtk-4.0-dev
```

First, you should compile libPd with multi instance and multi threads support. The static library is expected to be in the folder *libpd/libs*. For this, you can follow the libPd documentation or do:

```
make -C libpd/ UTIL=true EXTRA=true ADDITIONAL_CFLAGS="-DPDINSTANCE=1 -DPDTHREADS=1"
```

Then you can compile the Camomile plugin. The makefile are located in the folder *Builds/LinuxMakefile*. You can specify the target with with *TARGET_ARCH* (-m32 or -m64) and the configuration with *CONFIG* (Debug, DebugFx, Releaseo r ReleaseFx). For example:

```
make -C Builds/LinuxMakefile/ TARGET_ARCH=-m64 CONFIG=Release
make -C Builds/LinuxMakefile/ TARGET_ARCH=-m64 CONFIG=ReleaseFx
```

Windows

First, you should compile libPd with multi instance and multi threads support. The static library is expected to be in the folder *libpd/libs*. For this, you can follow the libPd documentation. You have to use MinGW-w64 (http://mingw-w64.org/doku.php) and to install msys2 (http://msys2.github.io/). The batch scripts offered by the libPd distribution are made for the csharp version without multi instance and multi thread supports. To compile the appropriate c version of libPd, you have change the script (or to use *libpd_mingw64_build.bat*):

```
make -C libpd libpd MULTI=true ADDITIONAL_LDFLAGS="-static -static-libgcc" ADDITIONAL_CFLAGS='-DPD_LONGINTTYPE="long long"'
```

Then you can compile the Camomile plugin. The Visual Studio 2015 solution and projects are located in the folder *Builds\VisualStudio2015*. With command, you can do

```
msbuild Builds\VisualStudio2015\Camomile.sln /property:Configuration=Release /property:Platform=x64
```

Demos

- Pierre Guillot's demos and tutorials on Vimeo (https://vimeo.com/album/4639971)

Credits

Camomile by Pierre Guillot - CICM (http://cicm.mshparisnord.org/) | Université Paris 8 (https://www.univ-paris8.fr/) | Labex Arts H2H (http://www.labex-arts-h2h.fr/)

Pure Data (http://msp.ucsd.edu/software.html) by Miller Puckette and others
JUCE (http://www.juce.com) by ROLI Ltd.
libPd (https://github.com/libpd/libpd) by the Pure Data community
VST PlugIn Technology (https://www.steinberg.net/en/company/developers.html) by Steinberg Media Technologies

License

Older versions

v0.0.7 (https://github.com/pierreguillot/Camomile/wiki/v0.0.7---Instructions)
v0.0.5 (https://github.com/pierreguillot/Camomile/releases/tag/v0.0.5-beta)
v0.0.4 (https://github.com/pierreguillot/Camomile/releases/tag/v0.0.4-beta)