Cyclone: A set of Pure Data objects cloned from Max/MSP

> Cyclone expands Pure Data with objects cloned from cycling74's Max/MSP and provides some good level of compatibility between the two environments.

Pure Data (or just "Pd") project is found at: https://sourceforge.net/p/pure-data/pure-data/ci/master/tree/ or in its github mirror https://github.com/pure-data/pure-data. The official download page is here: http://msp.ucsd.edu/software.html

Max is found at: https://cycling74.com/

Copyright © 2003-2025 - Krzysztof Czaja, Hans-Christoph Steiner, Fred Jan Kraan, Alexandre Porres, Derek Kwan, Matt Barber and others.

This work is free. You can redistribute it and/or modify it under the terms of the BSD-3-Clause (aka Revised BSD License). See License.txt https://github.com/porres/pd-cyclone/blob/cyclone0.4/LICENSE.txt and https://opensource.org/licenses/BSD-3-Clause for more details.

Current Release: Cyclone 0.9-2 (this release needs at least Pd Vanilla 0.55-0)

Released March 14th 2025

Find Cyclone's latest releases at: https://github.com/porres/pd-cyclone/releases or directly via Pd's external manager (Help => Find Externals). Please report bugs at https://github.com/porres/pd-cyclone/issues.

About Cyclone:

Outdated versions of cyclone (0.1) are available in the long abandoned Pd-extended distribution (which no one should be using now in the 2020's) as well as in Pd-l2ork and Purr Data - both originally based on Pd-Extended but ported to nw.js (0.1 versions of cyclone here were not fully ported to nw.js at the time of this writing). If you want an up to date version of Cyclone, use Pd Vanilla or PlugData.

Do you know about PlugData? Cyclone is also part of PlugData by Timothy Schoen, which is a fork of Pd that loads as a standalone or VST with a revamped GUI. See: https://github.com/timothyschoen/PlugData

The original author of Cyclone (Krzysztof Czaja) abandoned it in 2005 at version 0.1-alpha55, when Cyclone was compatible to MAX 4.0. Cyclone was then incorporated and available in Pd-extended, where it only a had a minor update under the maintenance of Hans-Christoph Steiner in 2013 (0.1-alpha56), right before Pd-extended and Cyclone (by consequence) were abandoned altogether (this 0.1-alpha56 version was also inherited by Pd-l2ork/Purr Data). Under a new maintenance phase by Fred Jan Kraan, 0.1-alpha57 and Cyclone 0.2 beta versions were released, still closely related to the previous '0.1-alpha' releases and mostly compliant to Max 4.0!

Cyclone 0.3-0 was the major overhaul in Cyclone, where almost all of its objects got updated to the latest Max 7 version (Max 7.3.5). Many bugs were also fixed, the documentation was rewritten from scratch and new objects were included. Check the provided **CHANGELOG.txt** file for the details in all version changes.

Currently, Cyclone still hasn't reached full compatibility to "Max 7.3.5". Some functionalities that depend on "transport" or "dictionaries" haven't been implemented and actually will never be. Cyclone is not in much active development these days and the main goal is to maintain the library, and fix bugs (hence avoid including newer things).

The main current maintainer of Cyclone (Porres) is much busier with developing the ELSE library. Note that this library has alternatives for almost all cyclone objects and many objects in ELSE are actually inspired by MAX/MSP objects that were not cloned in Cyclone. By the way, ELSE is also part of PlugData, therefore, the documentation of Cyclone points to alternatives in ELSE.

Some objects in Cyclone are now borrowed from ELSE, like [pink~], [tanh~] and comment. These objects are backwards compatible and offer more stuff than the orginal MAX object, so not really fully compatible.

The only object that hasn't been updated yet to MAX 7.3.5 is [mtr] and this is on the To Do list. Cyclone may still incorporate new functionalities in existing objects from Max 8 (current release) and newer 9+ versions in the future, but we can't promise it.

A 'mc' compatiblity would be possible now, as of Pd version 0.54-0, which supports multichannel connections. Notwithstandingly, there's no plan to create such objects for Cyclone and pursue this compatibility. On the other hand, users can currently build their own "mc" like abstractions based on cyclone objetcs with [clone]. Please note that many ELSE objects have multichannel support! The [tanh~] object in Cyclone is now the same on from ELSE and has MC support, as an example.

Installing Cyclone:

You can compile Cyclone from the source provided in this repository for the current bleeding edge last state or download one of the more stable compiled releases from https://github.com/porres/pd-cyclone/releases. A good alternative is simply to use Pd's own external download manager (a.k.a deken plugin): just click on the "find externals" option under the Help menu and search for "cyclone".

When installing cyclone, make sure the Cyclone folder is included in a folder that Pd searches for, such as `~/Documents/Pd/externals` - which is what Pd suggests you to do for severl versions now.

Now you can install Cyclone by loading it in the startup: go to "Preferences => Startup", then click "New", type "cyclone" and hit OK. Next time you restart Pd, the Cyclone library binary will be loaded.

This library binary loads the non alphanumeric operators objects (which are: `!-`, `!-~`, `!/`, `!/~`, `!=~`, `%~`, `+=~`, `<=~`, `<~`, `==~`, `>=~` and `>~`) but it also adds Cyclone's path to Pd's preferences, so you can load the other objects from Cyclone (which are separate binaries and abstractions).

But note that in order to actually force a path search priority in your patch, you need to use [declare -path cyclone].

You can also use the [declare -lib cyclone] in a patch to load the library if you don't want to always have Cyclone loaded when Pd starts. Loading the Cyclone binary as an object ([cyclone]) also loads the library, see its help file for more details.

Building Cyclone for Pd Vanilla:

Since "Cyclone 0.1-alpha57", the Cyclone package has relied on the new build system called "pd-lib-builder" by Katja Vetter (check the project in: https://github.com/pure-data/pd-lib-builder).

- Compiling with pdlibbuilder

PdLibBuilder tries to find the Pd source directory at several common locations, but when this fails, you have to specify the path yourself using the pdincludepath variable. Example:

```
make pdincludepath=~/pd-0.54-0/src/  (for Windows/MinGW add 'pdbinpath=~
```

- Make Install

Use "objectsdir" to set a relative path for your build, something like:

```
make install objectsdir=../cyclone-build
```

Then move it to your preferred install folder for Pd.

Building with CMake

It is now possible to build Cyclone for Pd Vanilla or libpd using CMake. CMake is a cross-platform, open-source build system. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. This allows native compilation via Windows (Microsoft Visual Studio), Linux (GCC) and macOS (XCode).

- Dependencies:
    - CMake: You can download for your platform here.
    - Only on Windows: pthreads library
    - Pure-data or libpd: sources and binaries.

If you are using MinGW, you can use the pthreadGC-3.dll included in the `maintenance/windows_dll` directory in this repository. Alternatively, you can also download it or compile it yourself from the sources here. This will typically result in pthreadGC2.(dll/lib).

If you are using Visual Studio, you need to provide a `pthreads` library compiled for Visual Studio either by downloading it or compiling it yourself. See here. Be careful to download / compile the right version for your setup. This would typically be `pthreadVC2.(dll/lib)`.

- Configuring the build

One way to configure CMake is to use the CMake GUI. The GUI will list the variables that can be provided to configure the build. The variables can also be specified in the command-line interface (See below for an example).

In this step you can select if you want to build shared libraries with `BUILD_SHARED_LIBS` and if you want to build all Cyclone objects into one single library with `BUILD_SINGLE_LIBRARY` (more on this below).

When using Microsoft Visual Studio (MSVC), you will be requested to provide a path to the `pthreads` library and its headers using variables `CMAKE_THREAD_LIBS_INIT` and `PTHREADS_INCLUDE_DIR`.

You will be requested to provide a path to the pure-data sources and to the pure-data library.

If building Cylone for libpd, these can also be satisfied by providing the path to the `pure-data` folder inside the libpd sources and providing the path to the libpd library. The variables are: `PD_INCLUDE_DIR` and `PD_LIBRARY`.

On macOS, you can define different deployment target and architectures from your current system using the variables `CMAKE_OSX_DEPLOYMENT_TARGET` and `CMAKE_OSX_ARCHITECTURES`.

You can specify additional compilation flags using the variable `CMAKE_C_FLAGS`.

CMake can now generate Makefiles, a MSVC solution, or an XCode project.

- Building

After generation, depending on your platform you can navigate to the directory where CMake generated the build files and then:

- On Linux: run `make`
- On Windows: open the MSVC solution and build it
- On macOS: open the XCode project and build it

Of course you can also use CMake itself to build cyclone by running this on the command line:

```
    cd <path/to/build/files/generated/by/CMake>
    cmake --build .
```

- Building a single library

Per default Cyclone will build most of its objects as a single binary file (`.so` / `.dll` / `.dylib` / `.pd_darwin`). The exception is the "cyclone" object/binary that loads the non alphanumeric operators objects (which are: `!-`, `!-~`, `!/`, `!/~`, `!=~`, `%~`, `+=~`, `<=~`, `<~`, `==~`, `>=~` and `>~`).

If you want you can also build all of the Cyclone objects into one `cyclone.so/dll/dylib/pd_darwin` by activating the `BUILD_SINGLE_LIBRARY` option.

Each one of the individual libraries contain a `<name>_setup()` method that will be invoked by pure-data on library load. If you select the `BUILD_SINGLE_LIBRARY`, CMake will generate the appropriate code so that all `*_setup()` methods will be invoked in the main `cyclone_setup()`.

- Command-line examples

Here are a few examples of how to download, configure and build the latest Cyclone on the

command line using CMake and pure-data or libpd.

Linux:

```
git clone https://github.com/pure-data/pure-data
<download pure-data binaries or build it yourself>

git clone https://github.com/porres/pd-cyclone
cd pd-cyclone
mkdir build && cd build
cmake .. -DPD_INCLUDE_DIR:PATH=pure-data/src -DPD_LIBRARY:PATH=<path/to/
cmake --build .
```

Windows / MSVC:

```
git clone https://github.com/pure-data/pure-data
<download pure-data binaries or build it yourself>

#Clone the Cyclone repository from GitHub:
git clone https://github.com/porres/pd-cyclone
cd pd-cyclone
mkdir build && cd build
cmake .. -DCMAKE_THREAD_LIBS_INIT:PATH=</path/to/pthreadsVC2.lib> -DPTHR
cmake --build .
```

Using libpd in Linux:

```
# Here we compile libpd ourselves, you can skip the building steps if yo
git clone https://github.com/libpd/libpd
cd libpd
git submodule init
git submodule update
# libpd build steps:
mkdir build && cd build
cmake ..
cmake --build .
cd ../..

# Now clone the Cyclone repository
git clone https://github.com/porres/pd-cyclone
cd pd-cyclone
mkdir build && cd build
cmake .. -DPD_INCLUDE_DIR:PATH=../libpd/pure-data/src -DPD_LIBRARY:PATH=
cmake --build .
```

A Brief History of Cyclone's Development:

Excerpt from Cyclone's original Readme (by its original author Krzysztof Czaja):

- "Cyclone is a library of Pure Data classes, bringing some level of compatibility between Max/MSP and Pd environments. Although being itself in the early stage of development, it is meant to eventually become part of a much larger project, aiming at unification and standardization of computer musician's tools. In its current form, cyclone is mainly for people using both Max and Pd, and thus wanting to develop cross-platform patches. (...)." The full original readme is provided in this repository at: https://github.com/porres/pd-cyclone/blob/master/maintenance/README_original.txt

Cyclone's original author Krzysztof Czaja worked on it as part of his miXed library from 2002 to 2005 and later abandoned it all together. In parallel, miXed had been incorporated into Pd Extended and eventually ended up under the maintenance of Hans-Christoph Steiner - the main developer and maintainer of Pd-Extended. When Pd Extended was abandoned after its last release (from Jan 2013), Cyclone and miXed were left unmaintained as a result. In Dec-2014, Fred Jan Kraan took over maintenance and development for cyclone (but not the rest of the miXed library) and released 0.1-alpha57 and Cyclone 0.2 beta versions, but decided to abandon development for it in Feb-2016.

Since February 21st 2016, further development for Cyclone started on this repository by Alexandre Porres, Derek Kwan, Matt Barber and other collaborators. The first stable release was Cyclone 0.3-0 from february 2019!

About Cyclone's Repositories and its Fork History:

=> Original Repository (up to version 0.1-Alpha-56): The original repository of MiXed as part of Pd Extended - containing Cyclone and more (such as 'toxy') - resides at https://svn.code.sf.net/p/pure-data/svn/trunk/externals/miXed/cyclone and the migrated repository: https://git.puredata.info/cgit/svn2git/libraries/miXed.git/. This repository embraces work from three different maintenance phases:

- Czaja's era (until 2005 and up to 0.1-Alpha55): Czaja (the original author) worked on Cyclone from version 01-alpha-01 (2002) to 0.1-alpha-55 (2005).
- Hans era (until 2013 and 0.1-Alpha-56): Hans maintained Cyclone from 2005 to 2013. The 0.1-Alpha55 version of Cyclone is found in most of Pd-Extended versions up to Pd-Extended 0.42-5. The last release of Pd-Extended is 0.43-4 from Jan-2013 and it carries the 0.1-Alpha56 version of Cyclone, which can also be found as "cyclone-v0-0extended" when searching for externals in Pd Vanilla.
- Kraan era (up to 2015): The later work in this repository was not made available into a new release from this repository.

=> Fred Jan Kraan's Repository (0.1-Alpha57 and 0.2-beta):

Fred Jan Kraan forked the original repository to https://github.com/electrickery/pd-miXedSon, but containing only the Cyclone library. This repository has a few releases - see https://github.com/electrickery/pd-miXedSon/releases - it starts with Cyclone version 0.1-alpha-57, from October 2015, which is basically the last developments made on the original repository in its last phase. Then it moves on to a new Cyclone 0.2 version which stopped at a beta stage in february 2016.

=> This Repository (0.3-0 and onwards):

In February 2016, Porres forked from https://github.com/electrickery/pd-miXedSon to this repository that resides at: https://github.com/porres/pd-cyclone. The fork happened while cyclone was at 0.2-beta stage. Since then, Alexandre Porres, Derek Kwan, Matt Barber and other collaborators have worked on further developments of cyclone. The first stable release from this repository was cyclone 0.3-0 from february 2019. In late 2021, after the release of version 0.6-0, this repository was detached from Kraan's (electrickery) here on GitHub, after being thousands of commits ahead and with a completely restructure of the code base.

=> The 'nilwind' fork:

The 'nilwind' library is a fork of Cyclone and it starts as a fork of the last stage https://github.com/electrickery/pd-miXedSon was left at, meaning it is a is a development over cyclone 0.2-beta. The nilwind's repository is at https://github.com/electrickery/pd-nilwind. Its first release is 'nilwind 0.2.1', from November 2019. This fork of cyclone does not aim to pursue updates according to newer versions of Max and its main concern is to keep compatibility to old/legacy patches made in the Pd-Extended era (which carried cyclone 0.1). Nonetheless, versions of cyclone 0.3 onward are also compatible to Pd-Extended era, as the current development phase does not introduce breaking changes and has only offered stable releases since 0.3!

About This Repository's Goals:

This repository resides at https://github.com/porres/pd-cyclone and is faithful to the original goal of Cyclone in creating an external Pd package with a collection of objects cloned and compatible to Max/MSP objects. Bugs and issues should be reported to https://github.com/porres/pd-cyclone/issues. Releases from this repository are stable and offer many fixes and improves stability from earlier versions.

Compatibility to newer versions of Max is a concern, but Max compatibility was always the main goal of cyclone and nothing really changed, since Max itself keeps backwards compatibilities. No incompatibilities should arise between cyclone 0.3-0 onwards with the legacy stage of the library (the cyclone 0.1 phase that was available in Pd Extended). Since

this development stage of Cyclone is concerned to provide compatibility for patches made in the Pd-Extended era, if such issues arise, they should be treated as bugs and reported/fixed.

Collaborating to Cyclone:

This repository/project is open to collaboration to anyone who wishes to work (keeping in mind the key and central goal of Max/MSP compatibility). Feel free to collaborate.

Acknowledgements:

Thanks to previous maintainers, Lucas Cordiviola for working on compilation and cross compilation issues and generating binaries for many releases. Diego Barrios Romero worked on the possibility of compiling all of objects in cyclone as a single binary instead of separate binaries. Tim Schoen for helping with some issues, including a couple of objects and including Cyclone in his PlugData project (see: https://github.com/timothyschoen/PlugData)