pdlua

Claude Heiland-Allen

2007–2018

Contents

1 AUTHORS

Claude Heiland-Allen
Frank Barknecht
Martin Peach

2 CHANGES

```
0.8     2013-11-01
        Building:   Makefile.static takes settings on command line
        Building:   Makefile.static patches lua with -fPIC for amd64 compat
        Building:   Makefile.static uses lua 5.1.5
        Feature:    menu open uses editor, merged from SVN (Martin Peach)
        Bug fix:    stack leakage fix, merged from SVN (Martin Peach)
        Internal:   use private Pd headers instead of redefining things

0.6~svn 2009-09-14
        Building:   autoconf system uses -fPIC for amd64 compatibility

0.5     2008-06-18
        Building:   Makefile.static (recommended)
        Building:   autoconfiscated (for advanced users)
        Feature:    interaction with [value], see [revalue] and [lexpr]
        Feature:    interaction with [table], see [ltabdump] and [ltabfill]
        Feature:    require() looks relative to .pd_lua and .pd_luax files
        Internal:   support pd >= 0.41 and pd < 0.41 (pd table API change)

0.4     2008-03-26
        API change: "load $1"--[lua] users must add the file name extension
        Bug fix:    "load $1"--[lua] works as advertised
        API change: users must name files "*.pd_lua" instead of "*.lua"
        API change: users must name files "*.pd_luax" instead of "*.luax"
        Bug fix:    help patches should be found correctly
        Examples++: [lurn], [luametro], [lpipe]
        Internal:   Doxygenated source comments
        Internal:   fewer leading underscores
        Internal:   renamed all examples for API change and tested them

0.3     2007-12-02
        clock support
        receive support
        send support
        support static builds

0.2     2007-09-25
        renumbered inlets/outlets: now follow Lua 1,2,... conventions
        fixed all examples for new numbering
        more and updated documentation
        opening via Pd's canvas path with obj:dofile(filename)
        better error reporting
        lexpr example
        bugfix in dispatch (in_3_foo doesn't pass "foo" as an arg any more)

0.1     2007-09-24
        initial release
```

3 configure.ac

```
# Process this file with autoconf to produce a configure script.
AC_PREREQ(2.61)
# package information
AC_INIT([pdlua],[0.5],[claude@mathr.co.uk])
AM_INIT_AUTOMAKE([pdlua],[0.5])
AC_CONFIG_SRCDIR([src/lua.c])
AC_CONFIG_HEADER([src/config.h])
```

```
# require C
AC_PROG_CC
AC_HEADER_STDC
AC_C_CONST
AC_TYPE_SIZE_T
AC_TYPE_SSIZE_T
AC_FUNC_ERROR_AT_LINE
AC_FUNC_MALLOC
# require Lua
if test -z "$PKG_CONFIG"; then
  AC_PATH_PROG(PKG_CONFIG, pkg-config)
fi
AC_ARG_WITH(lua, AS_HELP_STRING([--with-lua=lua5.1], [Lua name varies on ↙
    ↳ different systems]), [WITH_LUA=$withval], [WITH_LUA=lua5.1])
PKG_CHECK_MODULES(lua, $WITH_LUA >= 5.1)
CPPFLAGS="$CPPFLAGS $lua_CFLAGS"
LIBS="$LIBS $lua_LIBS"
AC_CHECK_HEADER([lua.h],,[AC_MSG_ERROR([cannot find <lua.h> header file])])
AC_CHECK_HEADER([lauxlib.h],,[AC_MSG_ERROR([cannot find <lauxlib.h> header file↙
    ↳ ])])
AC_CHECK_HEADER([lualib.h],,[AC_MSG_ERROR([cannot find <lualib.h> header file])↙
    ↳ ])
AC_CHECK_LIB([$WITH_LUA],[lua_newstate],,[AC_MSG_ERROR([cannot find $WITH_LUA ↙
    ↳ library])])
# require Pd
AC_CHECK_HEADER([m_pd.h],,[AC_MSG_ERROR([cannot find <m_pd.h> header file])])
pdextradir="\${prefix}/lib/pd/extra/lua"
AC_SUBST(pdextradir)
pddocdir="\${prefix}/lib/pd/doc/lua"
AC_SUBST(pddocdir)
# check system type
if test `uname -s` = Linux;
then
  LFLAGS="$LFLAGS -shared"
  PDEXT="pd_linux"
elif test `uname -s` = Darwin;
then
  LFLAGS="$LFLAGS -bundle -undefined suppress -flat_namespace"
  PDEXT="pd_darwin"
fi
AC_SUBST(PDEXT)
AC_SUBST(LFLAGS)
# output
AC_CONFIG_FILES([Makefile src/Makefile])
AC_OUTPUT
```

4 COPYING

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

```
 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                    51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law:

that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not

signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions

of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

```
295
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.
300
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
305
    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

310
Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:
315
    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
320
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.
325
You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

330   Yoyodyne, Inc., hereby disclaims all copyright interest in the program
    `Gnomovision' (which makes passes at compilers) written by James Hacker.

    <signature of Ty Coon>, 1 April 1989
    Ty Coon, President of Vice
335
This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
340 Public License instead of this License.
```

5 doc/internal.txt

```
Internal Stuff
==============

This is a developer document, not a user document.  You probably don't
5 need to read this.


Private/Internal Variables
```

```
    ---------------------------

    Everything that the user shouldn't dabble with starts with '_'.
    For example 'pd._constructor', 'self._object', etc.  This includes
    everything that messes with pointers, for many purposes have a
    stub: function f(self, ...) return _f(self._object, ...) end

    Things that the user can use shouldn't start with '_'.

    Everything in pdlua should be in the 'pd' global table.


    Table Indices
    -------------

    As per Lua conventions, Lua tables start at index 1.  See:

    lua.c/pdlua_pushatomtable()  (C->Lua)
    lua.c/pdlua_popatomtable()   (Lua->C) (makes additional assumptions)


    Inlet/Outlet Numbers
    --------------------

    C   code uses 0,1,...
    Lua code uses 1,2,...

    Translations are performed in:

    lua.c/pdlua_dispatch() (C->Lua)
    lua.c/pdlua_outlet()   (Lua->C)


    Pointers
    --------

    Pointers are all Light User Data values.  This means there is no type
    safety, make sure you're using the right pointers!

    pd._classes     :: string   => Lua object (class prototypes)
    object._class   :: t_class*

    pd._objects     :: t_pdlua* => Lua object (object instances)
    object._object  :: t_pdlua*

    pd._clocks      :: t_clock* => Lua object (clock instances)
    clock._clock    :: t_clock*

    pdtable._array  :: PDLUA_ARRAY_ELEM*

    Pointer atoms are also stored as Light User Data.  It's possible for
    things to crash if they are used/stored/etc, as far as I understand the
    way Pd uses them.


    Architecture Issues
    -------------------
```

```
:initialize() is called before the object is created.
:postinitialize() is called after the object is created.

Other Issues
------------

"#t does not invoke the __len metamethod when t is a table."
See end of: http://lua-users.org/wiki/GeneralizedPairsAndIpairs

This means pd.Table will remain ugly, with :length() :set() :get()
```

6 doc/lua.txt

```
lua
===

The Lua loader included in -lib lua allows externals for Pd to be
written in the Lua programming language.

If you try to create an object [foo] in Pd, Pd checks if the class
"foo" exists.  If it doesn't, it tries to load an external file that
"probably" will contain code for "foo".  The Lua loader adds support
for loading "foo.pd_lua" when you try to create [foo].


Class Creation
--------------

The first expression/statement in the file should be of the form:

    local foo = pd.Class:new():register("foo")

This creates a new Pd class called "foo".  The 'local' declaration
is optional, but recommended -- without it, 'foo' is global, which
means any Lua code can modify it (possibly by accident).


Object Initialization
---------------------

Then you can add methods to the Pd class.  The most important one
is 'initialize', which is executed when a new object is created.

    function foo:initialize(sel, atoms)
      -- code
    end

or equivalently:

   foo.initialize = function (self, sel, atoms)
     -- code
   end

'sel' is usually (always?) the class name, 'atoms' are the creation
arguments in a Lua table.  An example:
```

```
    Pd     ::  [foo a b 1 2 3 c]
    sel    ==  "foo"
    atoms  ==  { "a", "b", 1, 2, 3, "c" }
```

Being a method, 'initialize' has a 'self' variable (which is the object to be created), and if you want your objects to have inlets or outlets you need need to set those fields in this method (Pd doesn't support changing the number of inlets or outlets after an object is created):

```
    self.inlets = 1
    self.outlets = atoms[1]
```

The default inlet/outlet counts are 0.

The return value of 'initialize' is used to allow objects to fail to create (for example, if the creation arguments are bad). Most of the time you will 'return true', but if you really can't create then you can 'return false'.

If you need to do things after the Pd object is created, but before control is returned to Pd, you can use the 'postinitialize' method:

```
  function foo:postinitialize()
    -- code
  end
```

Object Finalization

The 'finalize' method is called when the object is deleted by Pd. You can clean up stuff here if needed. The default implementation does nothing.

Inlet Methods

FIXME: write about inlet methods/dispatching
FIXME: for now, see examples/*.pd_lua and src/pd.lua

Sending To Outlets

FIXME: write about self:outlet(outletNumber, selector, atoms)
FIXME: for now, see examples/*.pd_lua and src/pd.lua

Sending To Receivers

You can send messages to receivers like this:

```
    pd.send("receiver", "selector", { "a", "message", 1, 2, 3 }
```

```
100
    See examples/lsend.pd_lua for details.


    Receivers
105 ---------

    You can bind methods to receivers, to get messages from
    [send receiver] and "; receiver message".

110 See examples/lreceive.pd_lua for details.

    Remember to clean up your receivers in object:finalize(), or weird
    things will happen.

115
    Clocks
    ------

    You can bind methods to clocks, for timing based on Pd's logical clock.
120
    See examples/ldelay.pd_lua for details.

    Remember to clean up your clocks in object:finalize(), or weird things
    will happen.
125

    Miscellaneous Object Methods
    ----------------------------

130 Execute a Lua file using Pd's path to find it:

        self:dofile("filename")

    Report an error to Pd's console:
135
        self:error("message")


    Miscellaneous Functions
140 -----------------------

    Print a string to Pd's console:

        pd.post("a string")
145
    Note that pd.post() should not really be used for errors.

    FIXME: add pd.error() for error messages
```

7 doc/luax.txt

```
    luax
    ====

    The luax class allows "volatile" loading of Lua source code files
 5  that define Pd object behaviour.
```

The [luax foo] object loads "foo.pd_luax" at object creation time.

Advantages

+ You can edit "foo.pd_luax" and new [luax foo] objects will reflect
  the changes in the file.

+ Good for rapid development/testing cycles.

+ Good for live coding.

+ No need to restart Pd if you made a little mistake.

Disadvantages

- Reloading the file each time is slower.

- Syntax is different to the syntax expected by the Lua loader
  (see below for discussion).

- There is no "reload" functionality, so you can have multiple
  objects called [luax foo] but that have different behaviours.

- Data shared between objects must be accessible globally.

- The above two points mean some mistakes/changes mean you have
  to restart Pd anyway.

How To Write Code For luax

The last expression/statement in the file should be of the form:

```
return function (self, sel, atoms)
  -- code here
end
```

This function is executed in the context of the 'initialize'
method of the luax class, and has the same arguments:

'self' is the object to be created.
'sel' is the name of the class.
'atoms' are the creation arguments.

To add methods to the new object you need to add code inside
the returned function. There are two syntaxes for this:

```
function self:in_1_float(f) ... end
```

or

```
    self.in_1_float = function(self, f) ... end

If using the second form, remember the self argument.

If you need a shared state between objects, you need to use a
global name.  Try to pick something unique to avoid conflicts
with other scripts.  You also need to ensure that you don't
clobber this state - remember the script can be executed more
than once.
```

8 doc/Makefile

```
all: pdlua.pdf

%.pdf: %.tex
	pdflatex $<
	pdflatex $<
```

9 doc/pdlua.tex

```
\documentclass{article}

\title{pdlua}
\author{Claude Heiland-Allen
\\
\tt{claude@mathr.co.uk}}

\begin{document}

\maketitle

\begin{abstract}
Pd (aka Pure-data) is a real-time visual programming environment
primarily used for multimedia processing. Lua is a powerful, fast,
light-weight, embeddable scripting language. pdlua is a Lua embedding
for Pd.
\end{abstract}

\section{Classes and objects}

\begin{verbatim}
-- example1.pd_lua
local example1 = pd.Class:new():register("example1")
function example1:initialize(sel, atoms)
  return true
end
\end{verbatim}

{\tt pd} is a package automatically available to scripts loaded by pdlua.
pdlua uses a prototype-based object system and {\tt pd.Class} is the
prototype for classes that define patchable objects in Pd.  To create a
new class, use the {\tt :new()} method.  This creates an anonymous class,
which needs to be registered with Pd using the {\tt :register()} method.

The new class {\tt example1} cannot be instantiated yet, as the default
{\tt :initialize()} method returns {\tt false}, indicating to pdlua that
```

```
the Pd object should not be created.

With the code above, {\tt example1} can be created in Pd, but it will
have neither inlets nor outlets.

\section{Inlets and methods}

\begin{verbatim}
-- example2.pd_lua
local example2 = pd.Class:new():register("example2")
function example2:initialize(sel, atoms)
  self.inlets = 3
  return true
end
\end{verbatim}

Setting {\tt self.inlets} in the {\tt :initialize()} method will give
the created objects some inlets, in this case three of them.  Not very
interesting yet, as sending messages to these inlets will result in
errors as there are no methods to respond to messages at these inlets.

Messages arriving at the Pd object's inlets are dispatched to the Lua
object's {\tt :in\_*()} methods.  There are five predefined selectors:

\begin{itemize}
\item {\tt bang}
\item {\tt float}
\item {\tt symbol}
\item {\tt pointer}
\item {\tt list}
\end{itemize}

They can be used like this:

\begin{verbatim}
function example2:in_1_bang()
  pd.post("inlet 1 got a bang")
end
function example2:in_1_float(f)
  pd.post("inlet 1 got a float: " .. f)
end
function example2:in_1_symbol(s)
  pd.post("inlet 1 got a symbol: " .. s)
end
function example2:in_1_pointer(p)
  pd.post("inlet 1 got a pointer")
end
function example2:in_1_list(atoms)
  pd.post("inlet 1 got a list: " .. #atoms .. " elements")
end
\end{verbatim}

In the above, the methods are defined for the leftmost inlet.  To add
methods for the other inlets, replace {\tt :in\_1\_*()} with
{\tt :in\_2\_*()} for the second inlet, or {\tt :in\_3\_*()} for the third,
and so on.
```

```
It is possible to add methods for other selectors:

\begin{verbatim}
function example2:in_2_baabaa(atoms)
  pd.post("inlet 2 got a baabaa: " .. #atoms .. " elements")
end
function example2:in_2_moomoo(atoms)
  pd.post("inlet 2 got a moomoo: " .. #atoms .. " elements")
end
\end{verbatim}

It is also possible to add methods that catch any selector:

\begin{verbatim}
function example2:in_3(sel, atoms)
  pd.post("inlet 3 got a " .. sel .. ": .. #atoms .. " elements")
end
\end{verbatim}

Or methods that catch messages at any inlet:

\begin{verbatim}
function example2:in_n_float(i, f)
  pd.post("inlet " .. i .. " got a float: " .. f)
end
function example2:in_n_quack(i, atoms)
  pd.post("inlet " .. i .. " got a quack: " .. #atoms .. " elements")
end
\end{verbatim}

Or even catch any message at any inlet:

\begin{verbatim}
function example2:in_n(i, sel, atoms)
  pd.post("inlet " .. i .. " got a " .. sel .. ": " .. #atoms .. " elements")
end
\end{verbatim}

The more specific methods are called before the more general methods:

\begin{itemize}
\item {\tt :in\_1\_selector()}
\item {\tt :in\_n\_selector()}
\item {\tt :in\_1()}
\item {\tt :in\_n()}
\item {\tt :error("no method found")}
\end{itemize}

\section{Outlets}

\section{Sends}

\section{Receives}

\section{Values}

\section{Tables}
```

```
\section{Clocks}

\section{Paths}

\end{document}
```

10 examples/complex-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 144 114 luax complex;
#X obj 143 142 complex;
```

11 examples/complex.lua

```
local P = {}
if _REQUIREDNAME == nil then
  complex = P
else
  _G[_REQUIREDNAME] = P
end

-- imports
-- local sqrt = math.sqrt

-- no more external access after this point
setfenv(1, P)

function new (r, i) return {r=r, i=i} end

i = new(0, 1)

function add(c1, c2)
  return new(c1.r + c2.r, c1.i + c2.i)
end

function sub(c1, c2)
  return new(c1.r - c2.r, c1.i - c2.i)
end

function mul(c1, c2)
  return new(c1.r * c2.r - c1.i * c2.i, c1.r * c2.i + c1.i * c2.r)
end

return P
```

12 examples/complex.pd_lua

```
require("complex")

local C = pd.Class:new():register("complex")

function C:initialize(sel, atoms)
  for k,v in pairs(complex) do
    pd.post("complex." .. tostring(k) .. " = " .. tostring(v))
  end
```

```
  self.inlets = 0
  self.outlets = 0
  return true
end
```

13 examples/complex.pd_luax

```
require("complex")
return function (self, sel, atoms)
  for k,v in pairs(complex) do
    pd.post("complex." .. tostring(k) .. " = " .. tostring(v))
  end
  self.inlets = 0
  self.outlets = 0
  return true
end
```

14 examples/dispatchertest-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 98 174 dispatchertest;
#X floatatom 98 42 5 0 0 0 - - -;
#X symbolatom 144 75 10 0 0 0 - - -;
#X msg 191 110 a b c 1 2 3;
#X connect 1 0 0 0;
#X connect 2 0 0 1;
#X connect 3 0 0 2;
```

15 examples/dispatchertest.pd_lua

```
local DispatcherTest = pd.Class:new():register("dispatchertest")

function DispatcherTest:initialize(name, atoms)
  self.inlets = 3
  return true
end

function DispatcherTest:in_1_float(f)
  pd.post("float: " .. f)
end

function DispatcherTest:in_2_symbol(s)
  pd.post("symbol: " .. s)
end

function DispatcherTest:in_3(sel, atoms)
  pd.post(sel .. ":")
  for i,v in ipairs(atoms) do
    pd.post(i .. " = " .. v)
  end
end
```

16 examples/dispatchertest.pd_luax

```
return function (self, sel, atoms)
  self.inlets = 3
  function self:in_1_float(f)  pd.post("float: "  .. f) end
  function self:in_2_symbol(s) pd.post("symbol: " .. s) end
  function self:in_3_foo(atoms)
    pd.post("foo(")
    for i,v in ipairs(atoms) do
      pd.post("  " .. i .. " = " .. v)
    end
    pd.post(")")
  end
  return true
end
```

17 examples/dumptypes.pd_luax

```
-- dump Lua types for atoms, just to see what we get for pointers
return function(self, sel, atoms)
  self.inlets = 1
  function self:in_1(sel, atoms)
    pd.post(sel .. "[")
    for _,v in ipairs(atoms) do
      pd.post(type(v))
    end
    pd.post("]")
  end
  return true
end
```

18 examples/errors-help.pd

```
#N canvas 0 22 450 300 10;
#X obj 58 171 errors;
#X msg 31 64 moo;
#X msg 100 100 list moo baa;
#X msg 130 132 1;
#X msg 70 67 symbol baa;
#X connect 1 0 0 0;
#X connect 2 0 0 2;
#X connect 3 0 0 3;
#X connect 4 0 0 1;
```

19 examples/errors.pd_lua

```
-- no methods, deliberately: test the error reporting!
local errors = pd.Class:new():register("errors")
function errors:initialize(sel, atoms)
  self.inlets = 4
  self.outlets = 0
  return true
end
```

20 examples/ldelay2-help.pd

```
#N canvas 256 192 450 300 10;
#X msg 259 89 500;
#X msg 272 112 1000;
#X msg 197 107 1500;
#X obj 187 76 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 187 199 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X msg 139 108 stop;
#X obj 351 265 delay 1000;
#X text 278 266 see also:;
#X obj 187 150 ldelay2 1000;
#X obj 266 199 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X text 17 17 Testing multiple clocks in one object...;
#X connect 0 0 8 1;
#X connect 1 0 8 1;
#X connect 2 0 8 0;
#X connect 3 0 8 0;
#X connect 5 0 8 0;
#X connect 8 0 4 0;
#X connect 8 1 9 0;
```

21 examples/ldelay2.pd_lua

```
-- test multiple clocks

local LDelay = pd.Class:new():register("ldelay2")

function LDelay:initialize(name, atoms)
  if type(atoms[1]) ~= "number" or atoms[1] < 0 then
    self.delay = 1000
  else
    self.delay = atoms[1]
  end
  self.inlets = 2
  self.outlets = 2
  return true
end

function LDelay:postinitialize()
  self.clock1 = pd.Clock:new():register(self, "trigger1")
  self.clock2 = pd.Clock:new():register(self, "trigger2")
end

function LDelay:finalize()
  self.clock1:destruct()
  self.clock2:destruct()
end

function LDelay:in_2_float(f)
  self.delay = math.max(0, f)
end

function LDelay:in_1_float(f)
  self:in_2_float(f)
  self:in_1_bang()
```

```
end

function LDelay:in_1_bang()
  self.clock1:delay(self.delay)
  self.clock2:delay(self.delay * 2)
end

function LDelay:in_1_stop()
  self.clock1:unset()
  self.clock2:unset()
end

function LDelay:trigger1()
  self:outlet(1, "bang", {})
end

function LDelay:trigger2()
  self:outlet(2, "bang", {})
end
```

22 examples/ldelay-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 187 150 ldelay 1000;
#X msg 259 89 500;
#X msg 272 112 1000;
#X msg 197 107 1500;
#X obj 187 76 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 187 199 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X msg 139 108 stop;
#X text 17 17 Reimplementation of [delay] in Lua to prove clocks work...
;
#X obj 351 265 delay 1000;
#X text 278 266 see also:;
#X connect 0 0 5 0;
#X connect 1 0 0 1;
#X connect 2 0 0 1;
#X connect 3 0 0 0;
#X connect 4 0 0 0;
#X connect 6 0 0 0;
```

23 examples/ldelay.pd_lua

```
-- reimplementation of [delay] in Lua to test clock support

local LDelay = pd.Class:new():register("ldelay")

function LDelay:initialize(name, atoms)
  if type(atoms[1]) ~= "number" or atoms[1] < 0 then
    self.delay = 1000
  else
    self.delay = atoms[1]
  end
  self.inlets = 2
```

```
  self.outlets = 1
  return true
end

function LDelay:postinitialize()
  self.clock = pd.Clock:new():register(self, "trigger")
end

function LDelay:finalize()
  self.clock:destruct()
end

function LDelay:in_2_float(f)
  self.delay = math.max(0, f)
end

function LDelay:in_1_float(f)
  self:in_2_float(f)
  self:in_1_bang()
end

function LDelay:in_1_bang()
  self.clock:delay(self.delay)
end

function LDelay:in_1_stop()
  self.clock:unset()
end

function LDelay:trigger()
  self:outlet(1, "bang", {})
end
```

24 examples/ldemux-help.pd

```
#N canvas 31 53 510 441 10;
#X floatatom 371 118 5 0 0 0 - - -;
#X floatatom 250 103 5 0 0 0 - - -;
#X msg 252 73 symbol x;
#X msg 252 49 list a b c;
#X msg 251 27 a b c;
#X floatatom 108 212 5 0 0 0 - - -;
#X floatatom 64 258 5 0 0 0 - - -;
#X floatatom 64 214 5 0 0 0 - - -;
#X floatatom 50 92 5 0 0 0 - - -;
#X floatatom 57 137 5 0 0 0 - - -;
#X floatatom 102 141 5 0 0 0 - - -;
#X floatatom 148 139 5 0 0 0 - - -;
#X obj 64 235 ldemux zwei;
#X floatatom 131 262 5 0 0 0 - - -;
#X floatatom 109 309 5 0 0 0 - - -;
#X floatatom 65 355 5 0 0 0 - - -;
#X floatatom 65 308 5 0 0 0 - - -;
#X floatatom 106 356 5 0 0 0 - - -;
#X obj 65 332 ldemux;
#X obj 250 141 ldemux 6 ---------;
#X text 20 72 2nd arg gives default outlet;
```

```
#X obj 51 112 ldemux 3 2;
#X obj 219 286 print lu1;
#X obj 286 282 print lu2;
#X obj 307 252 print lu3;
#X obj 324 225 print lu4;
#X obj 329 194 print lu5;
#X obj 366 175 print lu6;
#X text 169 346 outlet number starts from 1 like Lua!;
#X connect 0 0 19 1;
#X connect 1 0 19 0;
#X connect 2 0 19 0;
#X connect 3 0 19 0;
#X connect 4 0 19 0;
#X connect 5 0 12 1;
#X connect 7 0 12 0;
#X connect 8 0 21 0;
#X connect 12 0 6 0;
#X connect 12 1 13 0;
#X connect 14 0 18 1;
#X connect 16 0 18 0;
#X connect 18 0 15 0;
#X connect 18 1 17 0;
#X connect 19 0 22 0;
#X connect 19 1 23 0;
#X connect 19 2 24 0;
#X connect 19 3 25 0;
#X connect 19 4 26 0;
#X connect 19 5 27 0;
#X connect 21 0 9 0;
#X connect 21 1 10 0;
#X connect 21 2 11 0;
```

25 examples/ldemux.pd_lua

```
-- contributed by Frank Barknecht

local LDemux = pd.Class:new():register("ldemux")

function LDemux:initialize(name, atoms)
    local n = atoms[1] or 2 -- default to 2 outlets.
    if type(n) ~= "number" or n < 2 then
        pd.post("ldemux: wrong outlet-count argument, using 2 outlets instead")
        n = 2
    end
    self.outlets = n
    self.inlets = 2
    self.to = 1
    -- second arg, if a number, selects default outlet
    if type(atoms[2]) == "number" then
        self:in_2_float(atoms[2])
    end
    return true
end

function LDemux:in_2_float(f)
    -- clip selection between left- and rightmost outlet
    self.to = math.max(1, math.min(self.outlets, f))
```

```
end

function LDemux:in_1(s, m)
  self:outlet(self.to, s, m)
end
```

26 examples/lexpr-help.pd

```
#N canvas 0 22 936 656 10;
#X floatatom 110 102 5 0 0 0 - - -;
#X floatatom 110 56 5 0 0 0 - - -;
#X floatatom 202 56 5 0 0 0 - - -;
#X floatatom 294 57 5 0 0 0 - - -;
#X obj 72 53 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 26 232 5 0 0 0 - - -;
#X floatatom 205 236 5 0 0 0 - - -;
#X floatatom 385 235 5 0 0 0 - - -;
#X floatatom 26 278 5 0 0 0 - - -;
#X obj 26 255 lexpr phi rho gamma -> 100*cos(phi) + gamma*sin(rho)
;
#X msg 115 205 lexpr foo bar baz -> foo * bar / baz;
#X msg 115 181 lexpr x y z -> x+y+z;
#X text 12 11 First come variable names \, then -> surrounded by spaces
\, then an expression (in Lua syntax).;
#X obj 110 77 lexpr a b c -> min(a \, b \, c);
#X text 14 122 Messages can change the expression (provided inlet count
stays the same). This resets the state of all variables. Note: weird
tricks are needed for commas in messages :(;
#X obj 86 441 lexpr a b c -> a + b + c;
#X msg 190 412 hot \$1;
#X obj 190 392 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X msg 270 412 hot \$1;
#X obj 270 392 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X msg 110 412 hot \$1;
#X obj 110 392 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X floatatom 86 367 5 0 0 0 - - -;
#X floatatom 167 367 5 0 0 0 - - -;
#X floatatom 249 367 5 0 0 0 - - -;
#X obj 68 387 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 86 469 5 0 0 0 - - -;
#X text 13 305 Eaxh inlet supports a "hot <float>" method \, that makes
the inlet hot or not. By default only the first inlet is hot. Sending
a bang to the first inlet always performs the calculation \, no matter
if that inlet is set to be cold.;
#X text 22 491 Multiple outlets are supported \, with multiple expressions
separated by commas.;
#X obj 82 546 lexpr a b -> a + b \, a - b;
#X floatatom 82 528 5 0 0 0 - - -;
#X floatatom 252 528 5 0 0 0 - - -;
#X floatatom 82 568 5 0 0 0 - - -;
#X floatatom 252 568 5 0 0 0 - - -;
#N canvas 2 22 450 300 \$0-weird-tricks-to-get-commas-in-messages 0
```

```
;
#X obj 58 231 lexpr a b -> max(a \, b) \, min(a \, b);
#X floatatom 32 194 5 0 0 0 - - -;
#X floatatom 33 273 5 0 0 0 - - -;
#X floatatom 284 198 5 0 0 0 - - -;
#X floatatom 284 275 5 0 0 0 - - -;
#X obj 123 148 makefilename %c;
#X msg 123 128 44;
#X obj 123 68 makefilename %c;
#X msg 123 48 44;
#X msg 123 88 lexpr a b -> min(a \$1 b) \$1 max(a \$1 b);
#X msg 123 168 lexpr a b -> max(a \$1 b) \$1 min(a \$1 b);
#X text 28 18 This is really ugly \, but it seems to work...;
#X connect 0 0 2 0;
#X connect 0 1 4 0;
#X connect 1 0 0 0;
#X connect 3 0 0 1;
#X connect 5 0 10 0;
#X connect 6 0 5 0;
#X connect 7 0 9 0;
#X connect 8 0 7 0;
#X connect 9 0 0 0;
#X connect 10 0 0 0;
#X restore 27 606 pd \$0-weird-tricks-to-get-commas-in-messages;
#X text 502 5 New in pdlua-0.5: interaction with Pd [value] objects.
;
#X obj 608 70 value \$0-foo;
#X floatatom 608 45 5 0 0 0 - - -;
#X obj 528 97 lexpr x y -> val("\$0-foo") * x / y;
#X floatatom 528 72 5 0 0 0 - - -;
#X floatatom 763 72 5 0 0 0 - - -;
#X floatatom 528 129 5 0 0 0 - - -;
#X text 504 161 Trying to access a [value] that doesn't exist returns
0:;
#X obj 539 224 lexpr z -> z + val("\$0-bar");
#X floatatom 539 202 5 0 0 0 - - -;
#X floatatom 539 253 5 0 0 0 - - -;
#X obj 508 384 lexpr scale name -> scale * val(name);
#X obj 782 319 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 672 343 symbol \$0-foo;
#X obj 672 319 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 782 343 symbol \$0-baz;
#X floatatom 792 377 5 0 0 0 - - -;
#X obj 792 399 value \$0-baz;
#X floatatom 508 354 5 0 0 0 - - -;
#X floatatom 508 415 5 0 0 0 - - -;
#X text 505 292 New in lexpr for pdlua-0.5: inputs can be symbols.
;
#X obj 511 531 lexpr left right -> "" .. left .. right;
#X symbolatom 511 497 10 0 0 0 - - -;
#X symbolatom 781 497 10 0 0 0 - - -;
#X symbolatom 511 570 0 0 0 0 - - -;
#X text 501 471;
#X text 494 446 New in lexpr for pdlua-0.5: outputs can be symbols
too. The inital "" is to make sure that we have a string.;
```

```
#X floatatom 737 496 5 0 0 0 - - -;
#X floatatom 594 497 5 0 0 0 - - -;
#X connect 1 0 13 0;
#X connect 2 0 13 1;
#X connect 3 0 13 2;
#X connect 4 0 13 0;
#X connect 5 0 9 0;
#X connect 6 0 9 1;
#X connect 7 0 9 2;
#X connect 9 0 8 0;
#X connect 10 0 9 0;
#X connect 11 0 9 0;
#X connect 13 0 0 0;
#X connect 15 0 26 0;
#X connect 16 0 15 1;
#X connect 17 0 16 0;
#X connect 18 0 15 2;
#X connect 19 0 18 0;
#X connect 20 0 15 0;
#X connect 21 0 20 0;
#X connect 22 0 15 0;
#X connect 23 0 15 1;
#X connect 24 0 15 2;
#X connect 25 0 15 0;
#X connect 29 0 32 0;
#X connect 29 1 33 0;
#X connect 30 0 29 0;
#X connect 31 0 29 1;
#X connect 37 0 36 0;
#X connect 38 0 41 0;
#X connect 39 0 38 0;
#X connect 40 0 38 1;
#X connect 43 0 45 0;
#X connect 44 0 43 0;
#X connect 46 0 54 0;
#X connect 47 0 50 0;
#X connect 48 0 46 1;
#X connect 49 0 48 0;
#X connect 50 0 46 1;
#X connect 51 0 52 0;
#X connect 53 0 46 0;
#X connect 56 0 59 0;
#X connect 57 0 56 0;
#X connect 58 0 56 1;
#X connect 62 0 56 1;
#X connect 63 0 56 0;
```

27 examples/lexpr.pd_lua

```
local lexpr = pd.Class:new():register("lexpr")

local function sandbox(e, f) -- only supports nullary f() with one return
  local g = getfenv(f)
  setfenv(f, e)
  local r = f()
  setfenv(f, g)
  return r
```

```
end

local lexpr_globals = {
  abs    = math.abs,
  acos   = math.acos,
  asin   = math.asin,
  atan   = math.atan,
  atan2  = math.atan2,
  ceil   = math.ceil,
  cos    = math.cos,
  cosh   = math.cosh,
  deg    = math.deg,
  exp    = math.exp,
  floor  = math.floor,
  fmod   = math.fmod,
  log    = math.log,
  log10  = math.log10,
  max    = math.max,
  min    = math.min,
  int    = function (x) i,f = math.modf(x) ; return i end,
  wrap   = function (x) i,f = math.modf(x) ; return f end,
  pi     = math.pi,
  pow    = math.pow,
  rad    = math.rad,
  sin    = math.sin,
  sinh   = math.sinh,
  sqrt   = math.sqrt,
  tan    = math.tan,
  tanh   = math.tanh,
  val    = function (s) return (pd.getvalue(s) or 0) end,
  choose = function (b,t,f) if b then return t else return f end end
}

function lexpr:readexpr(atoms)
  local vname = { }
  local context = { }
  local expr
  local i
  local k
  local v
  local j = 1
  local inlets
  local f
  local phase = "vars"
  for k,v in pairs(lexpr_globals) do
    context[k] = v
  end
  for i,v in ipairs(atoms) do
    if phase == "vars" then        -- create variables
      if v == "->" then
        inlets = i - 1
        phase = "expr"
        expr = ""
      else
        if type(v) == "string" then
          vname[j] = v
          context[v] = 0
```

```
          j = j + 1
        else
          self:error("lexpr: variable names must be symbols")
          return -1
        end
      end
    else if phase == "expr" then  -- build string
      expr = expr .. " " .. v
    else
      self:error("lexpr: internal error parsing expression")
      return -1
    end end
  end
  f = assert(loadstring("return {" .. expr .. " }"))
  local outlets = #(sandbox(context, f))
  return inlets, vname, context, f, outlets
end

function lexpr:initialize(sel, atoms)
  self.vname = { }
  self.context = { }
  self.hot = { }
  self.f = function () return 0 end
  function self:in_1_bang()
    local r = sandbox(self.context, self.f)
    local i
    for i = self.outlets,1,-1 do
      if type(r[i]) == "number" then
        self:outlet(i, "float", { r[i] })
      else if type(r[i]) == "string" then
        self:outlet(i, "symbol", { r[i] })
      else
        self:error("calculated a " .. type(r[i]) .. " but expected a number or a
            ↪ string")
      end end
    end
  end
  function self:in_n_float(i, f)
    self.context[self.vname[i]] = f
    if self.hot[i] then self:in_1_bang() end
  end
  function self:in_n_symbol(i, s)
    self.context[self.vname[i]] = s
    if self.hot[i] then self:in_1_bang() end
  end
  function self:in_n_hot(i, atoms)
    if type(atoms[1]) == "number" then
      self.hot[i] = atoms[1] ~= 0
    else
      self:error("hot method expects a float")
    end
  end
  function self:in_1_lexpr(atoms)
    local inlets
    local vname
    local context
    local f
```

```
    local outlets
    inlets, vname, context, f, outlets = self:readexpr(atoms)
    if (inlets == self.inlets) and (outlets == self.outlets) then
      self.vname = vname
      self.context = context
      self.f = f
    else
      self:error("new expression has different inlet/outlet count")
    end
  end
  self.inlets, self.vname, self.context, self.f, self.outlets = self:readexpr(atoms)
  if self.inlets < 1 then
    pd.post("lexpr: error: would have no inlets")
    return false
  end
  if self.outlets < 1 then
    pd.post("lexpr: error: would have no outlets")
    return false
  end
  for i = 1,self.inlets,1 do
    self.hot[i] = i == 1
  end
  return true
end
```

28 examples/list-pak-help.pd

```
#N canvas 188 72 556 526 10;
#X obj 106 244 list-pak;
#X obj 106 277 print list-pak;
#X floatatom 106 202 5 0 0 0 - - -;
#X msg 147 197 bang;
#X msg 149 219 symbol x;
#X obj 306 301 print list-pak;
#X floatatom 306 176 5 0 0 0 - - -;
#X msg 306 129 bang;
#X msg 306 152 symbol x;
#X floatatom 338 239 5 0 0 0 - - -;
#X floatatom 383 243 5 0 0 0 - - -;
#X obj 306 268 list-pak 3;
#X msg 380 221 any;
#X msg 338 217 bang;
#X obj 103 385 list-pak -3;
#X text 101 367 Don't do this:;
#X text 76 32 list-pak;
#X text 84 58 Like [pack] for any kind of type. Argument specifies
number of inlets. All inlets are hot as in Max' [pak].;
#X text 83 94 Stored elements default to 0;
#X text 101 467 Frank Barknecht \, 2007;
#X connect 0 0 1 0;
#X connect 2 0 0 0;
#X connect 3 0 0 0;
#X connect 4 0 0 0;
#X connect 6 0 11 0;
#X connect 7 0 11 0;
#X connect 8 0 11 0;
```

```
#X connect 9 0 11 1;
#X connect 10 0 11 2;
#X connect 11 0 5 0;
#X connect 12 0 11 2;
#X connect 13 0 11 1;
```

29 examples/list-pak.pd_lua

```
--[[

list-pak

    Like [pack] for any kind of type. Argument specifies number of inlets. All
    inlets are hot (as in Max' [pak])

    --Written by Frank Barknecht

--]]

-- Some footils --

-- selectors
local selectors = {"float", "list", "symbol", "bang", "pointer"}

-- check if item x is in table t:
local function contains(t, x)
  for _,v in ipairs(t) do
    if v == x then return true end
  end
  return false
end

-- Pd class

local ListPak = pd.Class:new():register("list-pak")

function ListPak:initialize(name, atoms)
    self.outlets = 1
    self.stored = {}
    if atoms[1] == nil then
        self.inlets = 1
    elseif type(atoms[1]) == "number" and atoms[1] >= 0 then
        self.inlets = math.max(atoms[1], 1)
    else
        pd.post("list-pak: First arg must be a positive float or empty")
        return false
    end
    for i=1,self.inlets do
        table.insert(self.stored, 0)
    end
    return true
end

function ListPak:in_n(i, sel, atoms)
    if not contains(selectors, sel) then
        -- insert selector
        self.stored[i] =  sel
```

```
50      else
            if table.getn(atoms) > 0 then
                self.stored[i] =  atoms[1]
            end
        end
55      self:outlet(1, "list", self.stored)
    end
```

30 examples/list-unpack-help.pd

```
    #N canvas 168 175 727 437 10;
    #X msg 246 129 1 2 3;
    #X msg 149 129 1;
    #X obj 149 221 list-unpack 3;
 5  #X obj 149 302 print lu0;
    #X obj 192 282 print lu1;
    #X obj 235 262 print lu2;
    #X msg 389 128 a b c;
    #X msg 461 128 1 2 3 4;
10  #X msg 179 129 symbol x;
    #X msg 295 128 list a b c;
    #X text 272 103 lists;
    #X text 153 106 float;
    #X text 192 105 symbol;
15  #X text 385 105 anything;
    #X text 461 105 too long;
    #X obj 340 221 list-unpack;
    #X obj 429 221 list-unpack 0;
    #X obj 339 330 list-unpack -1;
20  #X text 58 53 Like [unpack] for any kind of type. Argument specifies
    number of outlets. Pointers untested rsp. not supported;
    #X text 42 25 [list-unpack];
    #X text 64 389 2007 Frank Barknecht;
    #X obj 341 267 list-unpack 28 ----------------------------------;
25  #X text 337 244 Second arg is ignored and can be used for stretching
    ;
    #X text 338 196 Default number of outlets is 1 \, even if you want
    0:;
    #X text 338 302 But these won't create:;
30  #X obj 339 353 list-unpack three;
    #X connect 0 0 2 0;
    #X connect 1 0 2 0;
    #X connect 2 0 3 0;
    #X connect 2 1 4 0;
35  #X connect 2 2 5 0;
    #X connect 6 0 2 0;
    #X connect 7 0 2 0;
    #X connect 8 0 2 0;
    #X connect 9 0 2 0;
```

31 examples/list-unpack.pd_lua

```
    --[[

    list-unpack
        Like [unpack] for any kind of type. Argument specifies number of outlets.
```

```
    pointers untested rsp. not supported.
    --Written by Frank Barknecht
    --Fixed for pdlua-0.2svn by Clahde Heiland-Allen
--]]

-- Some footils --

-- outlet selectors
local selectors = {"float", "list", "symbol", "bang", "pointer"}

-- check if item x is in table t:
local function contains(t, x)
  for _,v in ipairs(t) do
    if v == x then return true end
  end
  return false
end

-- Pd class

local ListUnpack = pd.Class:new():register("list-unpack")

function ListUnpack:initialize(name, atoms)
    self.inlets = 1
    if atoms[1] == nil then
        self.outlets = 1
        return true
    elseif type(atoms[1]) == "number" and atoms[1] >= 0 then
        self.outlets = math.max(atoms[1], 1)
        return true
    else
        pd.post("list-unpack: First arg must be a positive float or empty")
        return false
    end
end

function ListUnpack:Outlet(num, atom)
    -- a better outlet: automatically selects selector
    -- map lua types to pd selectors
    local selectormap = {string = "symbol", number="float", userdata="pointer"}
    self:outlet(num, selectormap[type(atom)], {atom})
end

function ListUnpack:in_1(sel, atoms)
    if not contains(selectors, sel) then
        -- also unpack selector of anythings
        table.insert(atoms, 1, sel)
    end
    local size = math.min(self.outlets, table.getn(atoms))
    for i=size, 1, -1 do
        self:Outlet(i, atoms[i])
    end
end
```

32 examples/list-unpack-test-gem.pd

```
#N canvas 0 0 450 300 10;
```

```
#X obj 22 42 gemhead;
#X obj 222 124 gemwin;
#X msg 266 46 create;
#X msg 274 76 destroy;
#X obj 222 46 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X obj 22 70 list-unpack 3;
#X obj 108 102 print gem3;
#X obj 65 123 print gem2;
#X obj 22 143 print gem1;
#X connect 0 0 5 0;
#X connect 2 0 1 0;
#X connect 3 0 1 0;
#X connect 4 0 1 0;
#X connect 5 0 8 0;
#X connect 5 1 7 0;
#X connect 5 2 6 0;
```

33 examples/lpipe-help.pd

```
#N canvas 343 338 450 300 10;
#X obj 114 179 lpipe;
#X obj 114 214 print;
#X obj 114 127 unpack 0 0;
#X msg 135 95 0 0 \, 1000 1000 \, 2000 2000 \, 500 500;
#X msg 114 65 4000 4000;
#X msg 56 65 250 250;
#X msg 50 128 a b c d;
#X floatatom 179 183 5 0 0 0 - - -;
#X connect 0 0 1 0;
#X connect 2 0 0 0;
#X connect 2 1 0 1;
#X connect 2 1 7 0;
#X connect 3 0 2 0;
#X connect 4 0 2 0;
#X connect 5 0 2 0;
#X connect 6 0 0 0;
```

34 examples/lpipe.pd_lua

```
-- reimplementation of [pipe] for any message
-- claude(+fbar) 2008

local M = pd.Class:new():register("lpipe")

function M:initialize(name, atoms)
    self.inlets = 2
    self.outlets = 1
    self.nextID = 0
    return true
end

function M:in_2_float(f)
   self.deltatime = math.max(0, f)
end
```

```
function M:in_1(sel, atoms)
    -- we need unique method names for our clock callbacks
    self.nextID = self.nextID + 1
    local id = "trigger" .. self.nextID
    local clock = pd.Clock:new():register(self, id)
    -- the clock callback outlets the data and cleans up
    self[id] = function(self)
        self:outlet(1, sel, atoms)
        clock:destruct()
        self[id] = nil
    end
    -- now start the clock
    clock:delay(self.deltatime)
end
```

35 examples/lreceive-help.pd

```
#N canvas 0 0 540 307 10;
#X obj 27 41 lreceive lreceive- 25 10;
#X obj 27 76 print lreceive-LEFT;
#X obj 190 76 print lreceive-RIGHT;
#X obj 26 252 send lreceive-25;
#X msg 26 196 foo bar 1 2 3;
#X msg 146 196 123 bar 1 2 3;
#X obj 146 252 send lreceive-27;
#X obj 272 252 send;
#X floatatom 295 204 5 24 35 2 n - -;
#X obj 295 226 makefilename lreceive-%d;
#X symbolatom 272 171 10 0 0 0 - - -;
#X obj 272 143 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X connect 0 0 1 0;
#X connect 0 1 2 0;
#X connect 4 0 3 0;
#X connect 5 0 6 0;
#X connect 8 0 9 0;
#X connect 9 0 7 1;
#X connect 10 0 7 0;
#X connect 11 0 10 0;
```

36 examples/lreceive.pd_lua

```
-- test receive support

local LReceive = pd.Class:new():register("lreceive")

function LReceive:initialize(name, atoms)
  if type(atoms[1]) ~= "string" then
    pd.post("lreceive needs a prefix for the receive names!")
    return false
  else
    self.prefix = atoms[1]
  end
  if type(atoms[2]) ~= "number" or atoms[2] < 1 then
    self.start = 1
  else
```

```
    self.start = math.floor(atoms[2])
  end
  if type(atoms[3]) ~= "number" or atoms[3] < 1 then
    self.count = 1
  else
    self.count = math.floor(atoms[3])
  end
  self.receives = { }
  self.inlets = 0
  self.outlets = 2
  return true
end

function LReceive:postinitialize()
  local i = 0
  while (i < self.count) do
    local n = self.start + i
    self.receives[i+1] = pd.Receive:new():register(self, self.prefix .. n, "↙
        ↳ receive_" .. n)
    self["receive_" .. n] = function (self, sel, atoms)
      self:outlet(2, "float", { n })
      self:outlet(1, sel, atoms)
    end
    i = i + 1
  end
end

function LReceive:finalize()
  for _,r in ipairs(self.receives) do r:destruct() end
end
```

37 examples/lsend-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 55 142 lsend lsend-x;
#X obj 49 188 r lsend-x;
#X obj 158 189 r lsend-y;
#X msg 141 87 symbol lsend-x;
#X msg 149 107 symbol lsend-y;
#X floatatom 69 112 5 0 0 0 - - -;
#X msg 32 57 foo bar 1 2 4 8;
#X obj 48 224 print lsend-x;
#X obj 158 224 print lsend-y;
#X symbolatom 55 87 10 0 0 0 - - -;
#X text 21 23 Test send support...;
#X connect 1 0 7 0;
#X connect 2 0 8 0;
#X connect 3 0 0 1;
#X connect 4 0 0 1;
#X connect 5 0 0 0;
#X connect 6 0 0 0;
#X connect 9 0 0 0;
```

38 examples/lsend.pd_lua

```
-- test send support
```

```
local LSend = pd.Class:new():register("lsend")

function LSend:initialize(name, atoms)
  if type(atoms[1]) ~= "string" then
    pd.post("lsend needs a symbol")
    return false
  else
    self.sendto = atoms[1]
  end
  self.inlets = 2
  self.outlets = 0
  return true
end

function LSend:in_2_symbol(s)
  self.sendto = s
end

function LSend:in_1(sel, atoms)
  pd.send(self.sendto, sel, atoms)
end
```

39 examples/ltabdump-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 102 201 ltabdump \$0-tab-A;
#X obj 102 241 print ltabdump;
#X floatatom 218 242 5 0 0 0 - - -;
#X obj 11 34 table \$0-tab-B 8;
#X obj 11 14 table \$0-tab-A 4;
#N canvas 18 116 450 300 \$0-fill-tables 0;
#X obj 27 37 inlet;
#X obj 27 61 t b b;
#X obj 220 134 until;
#X obj 220 177 random 16;
#X obj 220 155 t b b;
#X obj 219 205 tabwrite \$0-tab-B;
#X obj 311 166 f 0;
#X obj 350 151 + 1;
#X obj 350 174 mod 8;
#X msg 223 112 8;
#X obj 30 134 until;
#X obj 30 177 random 16;
#X obj 30 155 t b b;
#X obj 121 166 f 0;
#X obj 160 151 + 1;
#X msg 33 112 4;
#X obj 160 174 mod 4;
#X obj 29 205 tabwrite \$0-tab-A;
#X connect 0 0 1 0;
#X connect 1 0 15 0;
#X connect 1 1 9 0;
#X connect 2 0 4 0;
#X connect 3 0 5 0;
#X connect 4 0 3 0;
#X connect 4 1 6 0;
```

```
#X connect 6 0 7 0;
#X connect 6 0 5 1;
#X connect 7 0 8 0;
#X connect 8 0 6 1;
#X connect 9 0 2 0;
#X connect 10 0 12 0;
#X connect 11 0 17 0;
#X connect 12 0 11 0;
#X connect 12 1 13 0;
#X connect 13 0 14 0;
#X connect 13 0 17 1;
#X connect 14 0 16 0;
#X connect 15 0 10 0;
#X connect 16 0 13 1;
#X restore 171 39 pd \$0-fill-tables;
#X obj 171 12 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 193 14 loadbang;
#X obj 102 82 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 132 99 symbol \$0-tab-A;
#X obj 132 179 symbol \$0-tab-C;
#X obj 132 161 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 132 121 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 132 81 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 132 139 symbol \$0-tab-B;
#X connect 0 0 1 0;
#X connect 0 1 2 0;
#X connect 6 0 5 0;
#X connect 7 0 5 0;
#X connect 8 0 0 0;
#X connect 9 0 0 0;
#X connect 10 0 0 0;
#X connect 11 0 10 0;
#X connect 12 0 14 0;
#X connect 13 0 9 0;
#X connect 14 0 0 0;
```

40 examples/ltabdump.pd_lua

```
local LTabDump = pd.Class:new():register("ltabdump")

function LTabDump:initialize(sel, atoms)
  self.inlets = 1
  self.outlets = 2
  if type(atoms[1]) == "string" then
    self.name = atoms[1]
  else
    self.name = nil
  end
  return true
end

-- output table length and data
```

```
function LTabDump:in_1_bang()
  -- need to sync each time before accessing if ...
  -- ... control-flow has gone outside of our scope
  local t = pd.Table:new():sync(self.name)
  if t ~= nil then
    local l = t:length()
    local a = { }
    local i
    for i = 1,l do
      a[i] = t:get(i-1)
    end
    -- copied above before outlet() to avoid race condition
    self:outlet(2, "float", { l })
    self:outlet(1, "list", a)
  end
end

-- choose a table name, then output
function LTabDump:in_1_symbol(s)
  self.name = s
  self:in_1_bang()
end
```

41 examples/ltabfill-help.pd

```
#N canvas 0 0 453 328 10;
#X obj 82 239 ltabfill \$0-table w -> sin(2*pi*w*x);
#X obj 33 192 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 331 196 5 0 0 2 w - -;
#X obj 82 159 symbol \$0-table-1;
#X obj 94 205 symbol \$0-table-2;
#X obj 94 184 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 82 136 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 254 134 table \$0-table-1 333;
#X obj 254 157 table \$0-table-2 555;
#X text 15 17 [ltabfill] is a clone of [lexpr] that writes a function
to a table. The first argument and inlet is the name of the table \,
then the usual [lexpr] stuff follows. There is an additional implicit
parameter to the expression \, named 'x' \, which ranges from [0..1)
no matter the size of the array. Note: you might have to close/reopen
the table displays for them to refresh correctly.;
#X text 14 280 See also:;
#X obj 53 299 lexpr x -> x;
#X connect 1 0 0 0;
#X connect 2 0 0 1;
#X connect 3 0 0 0;
#X connect 4 0 0 0;
#X connect 5 0 4 0;
#X connect 6 0 3 0;
```

42 examples/ltabfill.pd_lua

```
local ltabfill = pd.Class:new():register("ltabfill")
```

```
local function sandbox(e, f, x) -- only supports unary f() with one return
  local g = getfenv(f)
  setfenv(f, e)
  local r = f(x)
  setfenv(f, g)
  return r
end

local ltabfill_globals = {
  abs   = math.abs,
  acos  = math.acos,
  asin  = math.asin,
  atan  = math.atan,
  atan2 = math.atan2,
  ceil  = math.ceil,
  cos   = math.cos,
  cosh  = math.cosh,
  deg   = math.deg,
  exp   = math.exp,
  floor = math.floor,
  fmod  = math.fmod,
  log   = math.log,
  log10 = math.log10,
  max   = math.max,
  min   = math.min,
  int   = function (x) i,f = math.modf(x) ; return i end,
  wrap  = function (x) i,f = math.modf(x) ; return f end,
  pi    = math.pi,
  pow   = math.pow,
  rad   = math.rad,
  sin   = math.sin,
  sinh  = math.sinh,
  sqrt  = math.sqrt,
  tan   = math.tan,
  tanh  = math.tanh,
  val   = function (s) return (pd.getvalue(s) or 0) end
}

function ltabfill:readexpr(atoms)
  local vname = { }
  local context = { }
  local expr
  local i
  local k
  local v
  local j = 2
  local inlets
  local f
  local phase = "table"
  for k,v in pairs(ltabfill_globals) do
    context[k] = v
  end
  for i,v in ipairs(atoms) do
    if phase == "table" then
      if type(v) == "string" then
        self.tabname = v
```

```
        phase = "vars"
      else
        self:error("ltabfill: table name must be a symbol")
        return -1
      end
    else if phase == "vars" then        -- create variables
      if v == "->" then
        inlets = i - 1
        phase = "expr"
        expr = ""
      else
        if type(v) == "string" then
          vname[j] = v
          context[v] = 0
          j = j + 1
        else
          self:error("ltabfill: variable names must be symbols")
          return -1
        end
      end
    else if phase == "expr" then  -- build string
      expr = expr .. " " .. v
    else
      self:error("ltabfill: internal error parsing expression")
      return -1
    end end end
  end
  f = assert(loadstring("return function(x) return " .. expr .. " end"))()
  return inlets, vname, context, f, 0
end

function ltabfill:initialize(sel, atoms)
  self.tabname = nil
  self.vname = { }
  self.context = { }
  self.hot = { }
  self.f = function (x) return 0 end
  function self:in_1_bang()
    if self.tabname ~= nil then
      local t = pd.Table:new():sync(self.tabname)
      if t ~= nil then
        local i
        local l = t:length()
        for i = 1,l do
          local y = sandbox(self.context, self.f, (i-1)/l)
          t:set(i-1, y)
        end
        t:redraw()
      end
    end
  end
  function self:in_1_symbol(s)
    self.tabname = s
    if self.hot[1] then self:in_1_bang() end
  end
  function self:in_1_float(f)
    self:error("table name expected, got a float")
```

```
  end
  function self:in_n_float(i, f)
    self.context[self.vname[i]] = f
    if self.hot[i] then self:in_1_bang() end
  end
  function self:in_n_symbol(i, s)
    self.context[self.vname[i]] = s
    if self.hot[i] then self:in_1_bang() end
  end
  function self:in_n_hot(i, atoms)
    if type(atoms[1]) == "number" then
      self.hot[i] = atoms[1] ~= 0
    else
      self:error("hot method expects a float")
    end
  end
  function self:in_1_ltabfill(atoms)
    local inlets
    local vname
    local context
    local f
    local outlets
    inlets, vname, context, f, outlets = self:readexpr(atoms)
    if (inlets == self.inlets) and (outlets == self.outlets) then
      self.vname = vname
      self.context = context
      self.f = f
    else
      self:error("new expression has different inlet/outlet count")
    end
  end
  self.inlets, self.vname, self.context, self.f, self.outlets = self:readexpr(
    ↪ atoms)
  if self.inlets < 1 then
    pd.post("ltabfill: error: would have no inlets")
    return false
  end
  for i = 1,self.inlets,1 do
    self.hot[i] = i == 1
  end
  return true
end
```

43 examples/luametro-help.pd

```
#N canvas 0 0 636 570 10;
#X obj 144 423 t b b;
#X obj 144 449 timer;
#X floatatom 144 474 5 0 0 0 - - -;
#X obj 144 319 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X msg 95 311 stop;
#X msg 96 288 bang;
#X obj 169 395 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X floatatom 293 335 5 0 0 0 - - -;
#X msg 293 312 1000;
```

```
#X msg 332 311 500;
#X msg 367 420 1000 a 2000;
#X msg 365 399 list error bug;
#X msg 366 334 750 250 500 250;
#X obj 144 498 print;
#X text 369 377 Only floats allowed in period list;
#X text 43 14 lmetro;
#X msg 96 333 start;
#X text 294 290 lists or floats set new periods;
#X obj 307 457 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X msg 174 320 mode alea;
#X msg 173 293 mode junk;
#X obj 144 360 luametro 500 125 500 250;
#X msg 176 339 mode 2;
#X text 89 34 Like [metro] in Pd \, but allows more than one period.
The periods will be evaluated in sequence from left to right as default.
Min. period length is 0.01 msec.;
#X text 86 90 With "mode" you can set the mode the metro uses to select
delay periods. Four modes are available:;
#X text 115 128 1: alea - select randomly;
#X text 114 148 2: series - select randomly without repetitions;
#X text 113 167 3: sequence - select in sequence from left to right
(default mode);
#X text 113 197 4: rota - go left to right \, then turn around and
select right to left.;
#X text 94 244 "mode 2" or "mode series" work both.;
#X connect 0 0 1 0;
#X connect 0 1 1 1;
#X connect 1 0 2 0;
#X connect 2 0 13 0;
#X connect 3 0 21 0;
#X connect 4 0 21 0;
#X connect 5 0 21 0;
#X connect 7 0 21 1;
#X connect 8 0 7 0;
#X connect 9 0 7 0;
#X connect 10 0 21 1;
#X connect 11 0 21 1;
#X connect 12 0 21 1;
#X connect 16 0 21 0;
#X connect 19 0 21 0;
#X connect 20 0 21 0;
#X connect 21 0 6 0;
#X connect 21 0 0 0;
#X connect 21 1 18 0;
#X connect 22 0 21 0;
```

44 examples/luametro.pd_lua

```
-- reimplementation of [metro] with a twist
-- fbar 2007

-- we need an urn:
local urn = {}

function urn.new(size)
```

```
    assert(size > 0, "Error: size of urn must be greater than 0")
    local t = {size=size, last=size}
    for i=1,size do
        t[i] = i
    end
    return t
end

function urn.get(u)
    if u.last > 0 then
        local i = math.random(u.last)
        local res = u[i]
        u[i] = u[u.last]
        u.last = u.last - 1
        return res
    else
        return nil
    end
end

function urn.reset(u)
    u.last = u.size
    for i=1,u.size do
        u[i] = i
    end
end

local modes = {"alea", "series", "sequence", "rota"}

local function contains(t,x)
    for k,v in pairs(t) do
        if v == x then return true end
    end
    return false
end

-- pd:

local M = pd.Class:new():register("luametro")

function M:initialize(name, atoms)
    self.periods = {}
    self.i = 1
    self.mode = "sequence"
    self.direction = 1
    self.u = urn.new(math.max(1, #atoms))
    if not self:setperiods(atoms) then return false end
    self.inlets = 2
    self.outlets = 2
    return true
end

function M:postinitialize()
    self.clock = pd.Clock:new():register(self, "tick")
end

function M:finalize()
```

```
    self.clock:destruct()
end

function M:setperiods(periods)
    self.periods = {}
    for i,j in ipairs(periods) do
        if type(j) == "number" then
            self.periods[i] = math.max(0.01, j)
        else
            self:error("non-number in period list!")
            return false
        end
    end
    -- start over at front:
    self.i = 1
    return true
end

function M:in_2(sel, atoms)
    if sel == "list" or sel == "float" then
        self:setperiods(atoms)
    end
end

function M:in_1_float(f)
    if f ~= 0 then
        self:tick()
    else
        self.clock:unset()
    end
end

function M:in_1_bang()
    self.i = 1
    self:tick()
end

function M:in_1_start()
    self.i = 1
    self:tick()
end

function M:in_1_stop()
    self.clock:unset()
end

function M:in_1_mode(atoms)
    if type(atoms[1]) == "number" then
        self.mode = modes[atoms[1]]
        if self.mode == "series" then
            self.u = urn.new(#self.periods)
        end
        self.i = 1
    elseif type(atoms[1]) == "string" then
        if contains(modes, atoms[1]) then
            self.mode = atoms[1]
        else
```

```
            self:error("no such mode: " .. atoms[1])
        end
    end
    pd.post("Mode set to: " .. self.mode)
end


function M:tick()
    self:outlet(1, "bang", {})
    -- pd.post("selection: " .. tostring(self.i))
    if type(self.periods[self.i]) == "number" then
        self.clock:delay(self.periods[self.i])
    end
    if self.mode == "sequence" then
        self.i = self.i + 1
        if self.i > #self.periods then
            self.i = 1
            self:outlet(2, "bang", {})
        end
    elseif self.mode == "alea" then
        self.i = math.random(#self.periods)
    elseif self.mode == "series" then
        local f = urn.get(self.u)
        if not f then
            urn.reset(self.u)
            f = urn.get(self.u)
            self:outlet(2, "bang", {})
        end
        self.i = f
    elseif self.mode == "rota" then
        self.i = self.i + self.direction
        if self.i > #self.periods or self.i < 1 then
            self.direction = -self.direction
            self.i = self.i + self.direction
            self:outlet(2, "bang", {})
        end
    else
        self:error("Unknown mode")
    end
end
```

45 examples/lurn-help.pd

```
#N canvas 457 204 823 609 10;
#X msg 81 156 bang;
#X obj 104 275 print urne;
#X obj 125 251 bng 15 250 50 0 empty empty empty 0 -6 0 8 -262144 -1
-1;
#X msg 155 158 seed 10;
#X msg 157 181 clear;
#X floatatom 139 231 5 0 0 0 - - -;
#X text 78 41 generates random numbers without repetition;
#X text 37 23 urne: Unique Random Generator;
#X text 352 177 bang: output next random number without repetitions.
;
#X text 351 200 clear: put all numbers back into urn.;
#X text 351 224 seed NUM: seed random number generator;
```

```
#X text 331 157 Inlet 0:;
#X text 332 258 Inlet 1:;
#X text 335 319 Outlet 0:;
#X text 361 338 random number;
#X text 337 371 Outlet 1:;
#X text 373 279 int - set range of random numbers and reset the urn.
;
#X text 333 125 Arguments: int - range of random numbers in urn (optional)
;
#X text 83 68 Clone of the Max object [urn]. Stops when all values
have been chosen and bangs right outlet. Send "clear" message to make
it start over.;
#X msg 83 371 bang;
#N canvas 0 0 450 300 display 0;
#X obj 46 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 63 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 80 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 97 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 114 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 156 135 list append 0;
#X obj 46 134 list append 1;
#N canvas 0 0 450 300 count 0;
#X obj 158 193 + 1;
#X obj 122 158 until;
#X obj 122 194 f;
#X obj 122 101 inlet;
#X msg 167 157 0;
#X obj 122 120 t a b;
#X obj 122 224 outlet;
#X connect 0 0 2 1;
#X connect 1 0 2 0;
#X connect 2 0 0 0;
#X connect 2 0 6 0;
#X connect 3 0 5 0;
#X connect 4 0 2 1;
#X connect 5 0 1 0;
#X connect 5 1 4 0;
#X restore 156 90 pd count;
#X obj 46 160 route 0 1 2 3 4 5 6 7 8 9;
#X obj 131 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 148 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 165 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 182 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 199 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 46 36 inlet;
#X obj 156 37 inlet;
#X obj 156 64 max 1;
```

```
#X connect 5 0 8 0;
#X connect 6 0 8 0;
#X connect 7 0 5 0;
#X connect 8 0 0 0;
#X connect 8 1 1 0;
#X connect 8 2 2 0;
#X connect 8 3 3 0;
#X connect 8 4 4 0;
#X connect 8 5 9 0;
#X connect 8 6 10 0;
#X connect 8 7 11 0;
#X connect 8 8 12 0;
#X connect 8 9 13 0;
#X connect 14 0 6 0;
#X connect 15 0 16 0;
#X connect 16 0 7 0;
#X coords 0 -1 1 1 180 40 1 40 170;
#X restore 83 519 pd display;
#X text 363 389 bang \, when an empty urn receives a bang into first
inlet.;
#X msg 101 395 clear \, bang;
#X obj 127 448 t b b;
#X obj 127 468 bng 15 250 50 0 empty empty empty 0 -6 0 8 -262144 -1
-1;
#X obj 256 493 f 10;
#X floatatom 150 425 5 0 0 0 - - -;
#N canvas 0 0 450 300 display 0;
#X obj 46 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 63 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 80 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 97 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0 1
;
#X obj 114 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 156 135 list append 0;
#X obj 46 134 list append 1;
#N canvas 0 0 450 300 count 0;
#X obj 158 193 + 1;
#X obj 122 158 until;
#X obj 122 194 f;
#X obj 122 101 inlet;
#X msg 167 157 0;
#X obj 122 120 t a b;
#X obj 122 224 outlet;
#X connect 0 0 2 1;
#X connect 1 0 2 0;
#X connect 2 0 0 0;
#X connect 2 0 6 0;
#X connect 3 0 5 0;
#X connect 4 0 2 1;
#X connect 5 0 1 0;
#X connect 5 1 4 0;
#X restore 156 90 pd count;
#X obj 46 160 route 0 1 2 3 4 5 6 7 8 9;
```

```
#X obj 131 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 148 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 165 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 182 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 199 191 tgl 15 0 empty empty empty 0 -6 0 8 -262144 -1 -1 0
1;
#X obj 46 36 inlet;
#X obj 156 37 inlet;
#X obj 156 66 max 1;
#X connect 5 0 8 0;
#X connect 6 0 8 0;
#X connect 7 0 5 0;
#X connect 8 0 0 0;
#X connect 8 1 1 0;
#X connect 8 2 2 0;
#X connect 8 3 3 0;
#X connect 8 4 4 0;
#X connect 8 5 9 0;
#X connect 8 6 10 0;
#X connect 8 7 11 0;
#X connect 8 8 12 0;
#X connect 8 9 13 0;
#X connect 14 0 6 0;
#X connect 15 0 16 0;
#X connect 16 0 7 0;
#X coords 0 -1 1 1 180 40 1 40 170;
#X restore 81 300 pd display;
#X obj 254 273 f 10;
#X obj 254 247 b;
#X text 365 425 Use the second outlet to make the urn refill automatically
if it gets empty as show in the second example with a combined "clear
\, bang" message.;
#X obj 100 492 print urne_auto;
#X obj 81 230 lurn 10;
#X obj 83 425 lurn 10;
#X connect 0 0 32 0;
#X connect 3 0 32 0;
#X connect 4 0 29 0;
#X connect 4 0 32 0;
#X connect 5 0 32 1;
#X connect 19 0 33 0;
#X connect 22 0 33 0;
#X connect 23 0 22 0;
#X connect 23 0 24 0;
#X connect 23 1 25 0;
#X connect 25 0 20 1;
#X connect 26 0 25 0;
#X connect 26 0 33 1;
#X connect 28 0 27 1;
#X connect 29 0 28 0;
#X connect 32 0 1 0;
#X connect 32 0 27 0;
#X connect 32 1 2 0;
```

```
#X connect 33 0 20 0;
#X connect 33 0 31 0;
#X connect 33 1 23 0;
```

46 examples/lurn.pd_lua

```
-- urn class: random selection without repetitions.
-- fbar 2007

-- urn interface:

local urn = {}

function urn.new(size)
    assert(size > 0, "Error: size of urn must be greater than 0")
    local t = {size=size, last=size}
    for i=1,size do
        t[i] = i
    end
    return t
end

function urn.get(u)
    if u.last > 0 then
        local i = math.random(u.last)
        local res = u[i]
        u[i] = u[u.last]
        u.last = u.last - 1
        return res
    else
        return nil
    end
end

function urn.reset(u)
    u.last = u.size
    for i=1,u.size do
        u[i] = i
    end
end

-- Pd class:

local M = pd.Class:new():register("lurn")

function M:initialize(name, atoms)
    if type(atoms[1]) == "number" and atoms[1] >= 1 then
        self.u = urn.new(math.floor(math.max(atoms[1]), 1))
    else
        self.u = urn.new(1)
    end
    self.inlets = 2
    self.outlets = 2
    return true
end

function M:finalize()
```

```
    self.u = nil
end

function M:in_2_float(f)
    if f >= 1 then
        self.u = urn.new(math.floor(f))
    else
        self:error("size of urn too small. needs to be 1 at least")
    end
end

function M:in_1_clear(atoms)
    urn.reset(self.u)
end

function M:in_1_seed(atoms)
    if type(atoms[1]) == "number" then
        math.randomseed(atoms[1])
    else
        self:error("seed needs a number")
    end
end

function M:in_1_bang()
    local f = urn.get(self.u)
    if type(f) == "number" then
        self:outlet(1, "float", {f - 1})
    else
        self:outlet(2, "bang", {})
    end
end
```

47 examples/mutatee.pd_lua

```
Mutatee = pd.Class:new():register("mutatee")

function Mutatee:initialize()
  self.inlets = 1
  return true
end

function Mutatee:in_1_bang()
  pd.post("I'm happy and carefree!")
end
```

48 examples/mutator-help.pd

```
#N canvas 0 22 450 300 10;
#X obj 91 98 mutator;
#X obj 82 150 mutatee;
#X obj 71 13 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
 -1;
#X obj 71 39 t b b b b b;
#X obj 171 100 print checkpoint-charlie;
#X connect 2 0 3 0;
#X connect 3 0 1 0;
```

```
#X connect 3 1 4 0;
#X connect 3 2 0 0;
#X connect 3 3 4 0;
#X connect 3 4 1 0;
```

49 examples/mutator.pd_lua

```
Mutator = pd.Class:new():register("mutator")

function Mutator:initialize()
  self.inlets = 1
  return true
end

function Mutator:in_1_bang()
  pd.post("Radiation Alert!")
  Mutatee.in_1_bang = function(self)
    pd.post("Oh noes! I've been mutated!")
  end
end
```

50 examples/nop-help.pd

```
#N canvas 0 22 450 300 10;
#X obj 131 138 nop;
#X obj 131 179 print nop;
#X floatatom 94 47 5 0 0 0 - - -;
#X symbolatom 131 80 10 0 0 0 - - -;
#X msg 160 103 foo foo foo foo foo;
#X text 165 139 <---- (should) do nothing!;
#X connect 0 0 1 0;
#X connect 2 0 0 0;
#X connect 3 0 0 0;
#X connect 4 0 0 0;
```

51 examples/nop.pd_lua

```
local Nop = pd.Class:new():register("nop")

function Nop:initialize()
  self.inlets = 1
  self.outlets = 1
  return true
end

function Nop:in_1(s, m)
  self:outlet(1, s, m)
end
```

52 examples/nop-test-gem.pd

```
#N canvas 0 0 450 300 10;
#X obj 73 129 nop;
#X obj 73 76 gemhead;
#X obj 73 194 cube;
```

```
#X obj 202 194 gemwin;
#X obj 202 109 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X msg 247 111 create;
#X msg 249 150 destroy;
#X connect 0 0 2 0;
#X connect 1 0 0 0;
#X connect 4 0 3 0;
#X connect 5 0 3 0;
#X connect 6 0 3 0;
```

53 examples/peekbag-help.pd

```
#N canvas 229 223 555 414 10;
#X msg 96 243 60 64;
#X msg 147 243 60 0;
#X msg 191 243 62 64;
#X msg 238 243 62 0;
#X obj 96 370 print;
#X text 141 371 Output is in the printout window.;
#X msg 238 289 clear;
#X text 148 26 - COLLECTION OF NUMBERS;
#X text 32 94 The bag object takes (value \, flag) pairs. If the flag
is true (nonzero) \, the value is added to the collection \; if false
\, it's removed. The collection may have many copies of the same value.
You can output the collection (and empty it) with a "flush" message
\, or just empty it with "clear." You can use this to mimic a sustain
pedal \, for example.;
#X msg 237 266 flush;
#X text 287 243 <-- add or delete elements;
#X text 291 266 <-- output them;
#X text 293 290 <-- start over;
#X obj 96 340 peekbag;
#X obj 65 309 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 86 27 peekbag;
#X text 81 64 like;
#X obj 124 64 bag;
#X text 159 64 but with a bang for taking a peek.;
#X msg 237 309 aslist;
#X text 295 311 <-- get elements in a single list.;
#X text 33 177 Use a "bang" to take a peek at the bag's content without
clearing it \, use "aslist" to get the bag's elements in a single list.
;
#X connect 0 0 13 0;
#X connect 1 0 13 0;
#X connect 2 0 13 0;
#X connect 3 0 13 0;
#X connect 6 0 13 0;
#X connect 9 0 13 0;
#X connect 13 0 4 0;
#X connect 14 0 13 0;
#X connect 19 0 13 0;
```

54 examples/peekbag.pd_lua

```
-- contributed by Frank Barknecht

local PeekBag = pd.Class:new():register("peekbag")

function PeekBag:initialize(name, atoms)
  self.inlets = 2
  self.outlets = 1
  self.bag = {}
  self.add = false
  return true
end

function PeekBag:in_1_float(f)
    if self.add then
        table.insert(self.bag, f)
    else
        for i=table.getn(self.bag),1,-1 do
            if self.bag[i]==f then
                table.remove(self.bag, i)
                break
            end
        end
    end
end

function PeekBag:in_1_list(l)
    if type(l[2]) == "number" then
        self:in_2_float(l[2])
    end
    if type(l[1]) == "number" then
        self:in_1_float(l[1])
    end
end

function PeekBag:in_1_clear(l)
  self.bag = {}
  self.add = false
end

function PeekBag:in_1_flush(l)
    self:in_1_bang()
    self:in_1_clear()
end

function PeekBag:in_2_float(f)
    if f == 0 then
        self.add = false
    else
        self.add = true
    end
end

function PeekBag:in_1_bang()
    -- print all values of array
    for i,v in ipairs(self.bag) do
        self:outlet(1, "float", {v})
    end
```

```
end

function PeekBag:in_1_aslist()
    -- print all values of array as list
    if table.getn(self.bag) == 1 then
        self:outlet(1, "float", {self.bag[1]})
    elseif table.getn(self.bag) > 1 then
        self:outlet(1, "list", self.bag)
    end
end
```

55 examples/requirer-help.pd

```
#N canvas 0 0 450 300 10;
#X obj 144 88 requirer complex;
#X obj 146 173 requirer sqlite3;
#X text 57 27 This should find the package next to the .pd_lua?;
#X text 55 54 Or should it look next to the containing .pd patch?;
#X text 56 142 This only "works" because of a dirty dirty hack...;
#X text 52 222 TODO: some kind of "my location" necessary for .pd_lua
scripts to access?;
```

56 examples/requirer.pd_lua

```
complex = require("complex")
sqlite3 = require("luasql.sqlite3") -- evil hack, below is lame

local R = pd.Class:new():register("requirer")

function R:initialize(sel, atoms)
  if type(atoms[1]) ~= "string" then return false end
  -- require(atoms[1]) -- will this ever work?
  for k,v in pairs(_G[atoms[1]]) do
    pd.post(atoms[1].. "." .. tostring(k) .. " = " .. tostring(v))
  end
  self.inlets = 0
  self.outlets = 0
  return true
end
```

57 examples/revalue-help.pd

```
#N canvas 0 0 558 379 10;
#X obj 109 166 revalue;
#X obj 155 203 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X floatatom 109 205 5 0 0 0 - - -;
#X obj 83 121 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 109 121 5 0 0 0 - - -;
#X floatatom 110 268 5 0 0 0 - - -;
#X obj 83 267 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 110 297 value \$0-foo;
#X floatatom 110 331 5 0 0 0 - - -;
#X floatatom 240 268 5 0 0 0 - - -;
```

```
#X obj 213 267 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X floatatom 240 331 5 0 0 0 - - -;
#X obj 240 297 value \$0-bar;
#X obj 155 142 symbol \$0-foo;
#X obj 185 122 symbol \$0-bar;
#X obj 155 120 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 185 100 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 215 80 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 215 102 symbol \$0-oof;
#X obj 294 167 revalue \$0-foo;
#X obj 294 143 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X floatatom 294 199 5 0 0 0 - - -;
#X text 23 23 Interface to Pd's native [value]. Behaves like [value]
but with a settable name \, and another outlet to report if there is
no [value] with the current name.;
#X connect 0 0 2 0;
#X connect 0 1 1 0;
#X connect 3 0 0 0;
#X connect 4 0 0 0;
#X connect 5 0 7 0;
#X connect 6 0 7 0;
#X connect 7 0 8 0;
#X connect 9 0 12 0;
#X connect 10 0 12 0;
#X connect 12 0 11 0;
#X connect 13 0 0 1;
#X connect 14 0 0 1;
#X connect 15 0 13 0;
#X connect 16 0 14 0;
#X connect 17 0 18 0;
#X connect 18 0 0 1;
#X connect 19 0 21 0;
#X connect 20 0 19 0;
```

58 examples/revalue.pd_lua

```
local Revalue = pd.Class:new():register("revalue")

function Revalue:initialize(sel, atoms)
  self.inlets = 2
  self.outlets = 2
  if type(atoms[1]) == "string" then
    self.name = atoms[1]
  else
    self.name = nil
  end
  return true
end

-- get a value or report if it doesn't exist
function Revalue:in_1_bang()
  if self.name ~= nil then
```

```
    local r = pd.getvalue(self.name)
    if r ~= nil then
      self:outlet(1, "float", { r })
      return
    end
  end
  self:outlet(2, "bang", { })
end

-- set a value or report if it doesn't exist
function Revalue:in_1_float(f)
  if self.name ~= nil then
    if pd.setvalue(self.name, f) then
      return
    end
  end
  self:outlet(2, "bang", { })
end

-- set the [value] name
function Revalue:in_2_symbol(s)
  self.name = s
end
```

59 examples/reverb-calculator-help.pd

```
#N canvas 0 0 513 360 10;
#X obj 117 203 reverb-calculator 8;
#X obj 13 229 reverb;
#X floatatom 89 169 5 0 0 0 - - -;
#X obj 15 146 noise~;
#X obj 14 185 *~;
#X obj 62 115 vline~;
#X obj 18 265 dac~;
#X obj 70 263 print reverb;
#X obj 11 7 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0 1
;
#X obj 8 60 t b b;
#X obj 158 24 spigot;
#X obj 197 -4 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 0
1;
#X obj 25 166 noise~;
#X obj 25 205 *~;
#X obj 10 28 metro 1000;
#X obj 158 47 t b b b;
#X floatatom 158 146 5 0 0 0 - - -;
#X obj 158 172 pack f f f;
#X obj 158 71 random 1000;
#X obj 251 71 random 1000;
#X obj 340 70 random 1000;
#X floatatom 251 146 5 0 0 0 - - -;
#X floatatom 341 148 5 0 0 0 - - -;
#X obj 157 97 expr $f1*$f1/100000+1;
#X obj 251 121 expr $f1*$f1/100000+1;
#X obj 341 101 expr $f1*$f1/100000+1;
#X msg 10 89 0.5 1 \, 0 1 1;
#X obj 84 4 f 0;
```

```
#X obj 115 4 + 1;
#X obj 84 62 sel 0;
#X obj 115 24 mod 5;
#X text 255 174 room dimensions in meters;
#X text 79 148 decay (%);
#X text 98 245 delay times in ms;
#X obj 212 22 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X connect 0 0 1 2;
#X connect 0 0 7 0;
#X connect 1 0 6 0;
#X connect 1 1 6 1;
#X connect 2 0 1 3;
#X connect 3 0 4 0;
#X connect 4 0 1 0;
#X connect 5 0 4 1;
#X connect 5 0 13 1;
#X connect 8 0 14 0;
#X connect 9 0 26 0;
#X connect 9 1 27 0;
#X connect 10 0 15 0;
#X connect 11 0 10 1;
#X connect 12 0 13 0;
#X connect 13 0 1 1;
#X connect 14 0 9 0;
#X connect 15 0 18 0;
#X connect 15 1 19 0;
#X connect 15 2 20 0;
#X connect 16 0 17 0;
#X connect 17 0 0 0;
#X connect 18 0 23 0;
#X connect 19 0 24 0;
#X connect 20 0 25 0;
#X connect 21 0 17 1;
#X connect 22 0 17 2;
#X connect 23 0 16 0;
#X connect 24 0 21 0;
#X connect 25 0 22 0;
#X connect 26 0 5 0;
#X connect 27 0 28 0;
#X connect 27 0 29 0;
#X connect 28 0 30 0;
#X connect 29 0 10 0;
#X connect 30 0 27 1;
#X connect 34 0 15 0;
```

60 examples/reverb-calculator.pd_lua

```
local R = pd.Class:new():register("reverb-calculator")

function R:initialize(sel, atoms)
  if type(atoms[1]) ~= "number" then
    return false
  end
  self.count = math.max(atoms[1], 1)
  self.inlets = 1
  self.outlets = 1
```

```
  return true
end

local gcd = function(m, n)
  while m ~= 0 do m, n = math.mod(n, m), m end
  return n
end

function R:fundamental(nx, ny, nz)
  if (nx == 0 and ny == 0 and nz == 0) then return false end
  if (nx == 0 and ny == 0 and nz == 1) then return true  end
  if (nx == 0 and ny == 1 and nz == 0) then return true  end
  if (nx == 1 and ny == 0 and nz == 0) then return true  end
  return gcd(gcd(nx, ny), nz) == 1
end

function R:delay(lx, ly, lz, nx, ny, nz)
  return 1000 / ((340/2) * math.sqrt((nx*nx)/(lx*lx)+(ny*ny)/(ly*ly)+(nz*nz)/(lz
    ↘ *lz)))
end

function R:in_1_list(atoms)
  local lx = atoms[1]
  local ly = atoms[2]
  local lz = atoms[3]
  local delays = { }
  for nx = 0, 16 do
  for ny = 0, 16 do
  for nz = 0, 16 do
    if self:fundamental(nx, ny, nz) then
      table.insert(delays, self:delay(lx, ly, lz, nx, ny, nz))
    end
  end end end
  table.sort(delays, function (a, b) return a > b end)
  local out = { }
  local j = 1
  for i = 1, self.count do
    out[i] = delays[j]
    repeat
      j = j + 1
    until delays[j] ~= delays[j-1]
  end
  self:outlet(1, "list", out)
end
```

61 examples/reverb.pd

```
#N canvas 0 0 690 613 10;
#X obj 388 33 inlet~;
#X obj 448 33 inlet~;
#X obj 399 215 outlet~;
#X obj 459 215 outlet~;
#X obj 13 53 delread~ \$0-A 29.4118;
#X obj 32 73 delread~ \$0-B 23.5294;
#X obj 121 94 delread~ \$0-C 18.3734;
#X obj 138 113 delread~ \$0-D 17.6471;
#X obj 205 51 delread~ \$0-E 15.1322;
```

```
#X obj 223 73 delread~ \$0-F 14.1176;
#X obj 311 96 delread~ \$0-G 12.7274;
#X obj 327 118 delread~ \$0-H 12.4706;
#X obj 204 221 +~;
#X obj 310 222 +~;
#X obj 246 221 -~;
#X obj 350 221 -~;
#X obj 203 277 +~;
#X obj 247 278 +~;
#X obj 291 279 -~;
#X obj 330 279 -~;
#X obj 14 221 +~;
#X obj 120 222 +~;
#X obj 56 221 -~;
#X obj 160 221 -~;
#X obj 13 277 +~;
#X obj 57 278 +~;
#X obj 101 279 -~;
#X obj 141 278 -~;
#X obj 272 349 -~;
#X obj 311 349 -~;
#X obj 13 347 +~;
#X obj 57 348 +~;
#X obj 101 349 +~;
#X obj 141 348 +~;
#X obj 184 347 -~;
#X obj 228 348 -~;
#X obj 587 53 clip 0 100;
#X obj 512 339 / 282.843;
#X obj 13 397 *~ 0;
#X obj 57 398 *~ 0;
#X obj 101 399 *~ 0;
#X obj 141 398 *~ 0;
#X obj 184 397 *~ 0;
#X obj 228 398 *~ 0;
#X obj 272 399 *~ 0;
#X obj 311 399 *~ 0;
#X obj 587 31 inlet;
#X obj 310 174 +~;
#X obj 341 175 +~;
#X obj 540 28 inlet;
#X obj 125 20 unpack f f f f f f f f;
#X obj 13 435 delwrite~ \$0-A 1000;
#X obj 23 455 delwrite~ \$0-B 1000;
#X obj 33 475 delwrite~ \$0-C 1000;
#X obj 43 495 delwrite~ \$0-D 1000;
#X obj 53 515 delwrite~ \$0-E 1000;
#X obj 63 535 delwrite~ \$0-F 1000;
#X obj 73 555 delwrite~ \$0-G 1000;
#X obj 83 575 delwrite~ \$0-H 1000;
#X connect 0 0 47 1;
#X connect 1 0 48 1;
#X connect 4 0 20 0;
#X connect 4 0 22 0;
#X connect 5 0 20 1;
#X connect 5 0 22 1;
#X connect 6 0 21 0;
```

```
#X connect 6 0 23 0;
#X connect 7 0 21 1;
#X connect 7 0 23 1;
#X connect 8 0 12 0;
#X connect 8 0 14 0;
#X connect 9 0 12 1;
#X connect 9 0 14 1;
#X connect 10 0 47 0;
#X connect 11 0 48 0;
#X connect 12 0 16 0;
#X connect 12 0 18 0;
#X connect 13 0 18 1;
#X connect 13 0 16 1;
#X connect 14 0 17 0;
#X connect 14 0 19 0;
#X connect 15 0 17 1;
#X connect 15 0 19 1;
#X connect 16 0 30 1;
#X connect 16 0 34 1;
#X connect 17 0 31 1;
#X connect 17 0 35 1;
#X connect 18 0 32 1;
#X connect 18 0 28 1;
#X connect 19 0 33 1;
#X connect 19 0 29 1;
#X connect 20 0 24 0;
#X connect 20 0 26 0;
#X connect 21 0 26 1;
#X connect 21 0 24 1;
#X connect 22 0 25 0;
#X connect 22 0 27 0;
#X connect 23 0 25 1;
#X connect 23 0 27 1;
#X connect 24 0 30 0;
#X connect 24 0 34 0;
#X connect 25 0 31 0;
#X connect 25 0 35 0;
#X connect 26 0 32 0;
#X connect 26 0 28 0;
#X connect 27 0 33 0;
#X connect 27 0 29 0;
#X connect 28 0 44 0;
#X connect 29 0 45 0;
#X connect 30 0 38 0;
#X connect 31 0 39 0;
#X connect 32 0 40 0;
#X connect 33 0 41 0;
#X connect 34 0 42 0;
#X connect 35 0 43 0;
#X connect 36 0 37 0;
#X connect 37 0 45 1;
#X connect 37 0 44 1;
#X connect 37 0 43 1;
#X connect 37 0 42 1;
#X connect 37 0 41 1;
#X connect 37 0 40 1;
#X connect 37 0 39 1;
```

```
#X connect 37 0 38 1;
#X connect 38 0 51 0;
#X connect 39 0 52 0;
#X connect 40 0 53 0;
#X connect 41 0 54 0;
#X connect 42 0 55 0;
#X connect 43 0 56 0;
#X connect 44 0 57 0;
#X connect 45 0 58 0;
#X connect 46 0 36 0;
#X connect 47 0 2 0;
#X connect 47 0 13 0;
#X connect 47 0 15 0;
#X connect 48 0 3 0;
#X connect 48 0 15 1;
#X connect 48 0 13 1;
#X connect 49 0 50 0;
#X connect 50 0 4 0;
#X connect 50 1 5 0;
#X connect 50 2 6 0;
#X connect 50 3 7 0;
#X connect 50 4 8 0;
#X connect 50 5 9 0;
#X connect 50 6 10 0;
#X connect 50 7 11 0;
```

62 examples/shared-help.pd

```
#N canvas 0 22 450 300 10;
#X obj 20 71 shared foo;
#X obj 139 72 shared foo;
#X obj 22 191 shared bar;
#X obj 141 194 shared bar;
#X obj 141 241 print bar;
#X obj 139 115 print foo;
#X floatatom 85 28 5 0 0 0 - - -;
#X symbolatom 204 29 10 0 0 0 - - -;
#X obj 139 28 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 20 28 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 87 158 5 0 0 0 - - -;
#X symbolatom 206 159 10 0 0 0 - - -;
#X obj 141 158 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 22 158 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 289 72 shared foo;
#X obj 289 28 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X msg 354 27 a b c;
#X obj 291 194 shared bar;
#X obj 291 158 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X msg 356 158 d e f;
#X connect 0 0 5 0;
#X connect 1 0 5 0;
```

```
#X connect 2 0 4 0;
#X connect 3 0 4 0;
#X connect 6 0 0 1;
#X connect 7 0 1 1;
#X connect 8 0 1 0;
#X connect 9 0 0 0;
#X connect 10 0 2 1;
#X connect 11 0 3 1;
#X connect 12 0 3 0;
#X connect 13 0 2 0;
#X connect 14 0 5 0;
#X connect 15 0 14 0;
#X connect 16 0 14 1;
#X connect 17 0 4 0;
#X connect 18 0 17 0;
#X connect 19 0 17 1;
```

63 examples/shared.pd_lua

```
local Shared = pd.Class:new():register("shared")
local sharedStore = { }

function Shared:initialize(sel, atoms)
  if type(atoms[1]) == "string" then
    if sharedStore[atoms[1]] == nil then
      sharedStore[atoms[1]] = { sel = "bang", msg = { }, count = 1 }
    else
      sharedStore[atoms[1]].count = sharedStore[atoms[1]].count + 1
    end
    self.name = atoms[1]
    self.inlets = 2
    self.outlets = 1
    return true
  else
    return false
  end
end

function Shared:in_1_bang()
  self:outlet(1, sharedStore[self.name].sel, sharedStore[self.name].msg)
end

function Shared:in_2(s, m)
  local c = sharedStore[self.name].count
  sharedStore[self.name] = { sel = s, msg = m, count = c }
end

function Shared:finalize()
  sharedStore[self.name].count = sharedStore[self.name].count - 1
  if sharedStore[self.name].count == 0 then
    sharedStore[self.name] = nil
  end
end
```

64 examples/shared.pd_luax

```
if sharedluaxstore == nil then
  sharedluaxstore = { }  -- initialize store only once
end

return function (self, sel, atoms)
  if type(atoms[1]) == "string" then
    if sharedluaxstore[atoms[1]] == nil then
      sharedluaxstore[atoms[1]] = { sel = "bang", msg = { }, count = 1 }
    else
      sharedluaxstore[atoms[1]].count = sharedluaxstore[atoms[1]].count + 1
    end
    self.name = atoms[1]
    self.inlets = 2
    self.outlets = 1
    self.in_1_bang = function(self)
      local s = sharedluaxstore[self.name]
      self:outlet(1, s.sel, s.msg)
    end
    self.in_2 = function(self, s, m)
      local c = sharedluaxstore[self.name].count
      sharedluaxstore[self.name] = { sel = s, msg = m, count = c }
    end
    self.finalize = function (self)
      sharedluaxstore[self.name].count = sharedluaxstore[self.name].count - 1
      if sharedluaxstore[self.name].count == 0 then
        sharedluaxstore[self.name] = nil
      end
    end
    return true
  else
    return false
  end
end
```

65 examples/simplecounter-help.pd

```
#N canvas 0 0 450 300 10;
#X text 21 17 these three are equivalent \, default start count is
0;
#X obj 22 78 simplecounter;
#X floatatom 22 113 5 0 0 0 - - -;
#X obj 22 50 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 152 116 5 0 0 0 - - -;
#X obj 152 53 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 152 81 simplecounter 0;
#X floatatom 287 113 5 0 0 0 - - -;
#X obj 287 50 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X obj 287 78 simplecounter foo;
#X text 19 141 the first creation argument (when a float) is start
count;
#X floatatom 17 252 5 0 0 0 - - -;
#X obj 17 189 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X floatatom 140 252 5 0 0 0 - - -;
```

```
#X obj 140 189 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 17 217 simplecounter 10;
#X obj 140 217 simplecounter -20;
#X floatatom 284 250 5 0 0 0 - - -;
#X obj 284 187 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144
-1 -1;
#X obj 284 215 simplecounter 10 foo;
#X text 18 157 further creation arguments are ignored;
#X connect 1 0 2 0;
#X connect 3 0 1 0;
#X connect 5 0 6 0;
#X connect 6 0 4 0;
#X connect 8 0 9 0;
#X connect 9 0 7 0;
#X connect 12 0 15 0;
#X connect 14 0 16 0;
#X connect 15 0 11 0;
#X connect 16 0 13 0;
#X connect 18 0 19 0;
#X connect 19 0 17 0;
```

66 examples/simplecounter.pd_lua

```
local SimpleCounter = pd.Class:new():register("simplecounter")

function SimpleCounter:initialize(sel, atoms)
  self.inlets = 1
  self.outlets = 1
  self.count = 0
  if type(atoms[1]) == "number" then self.count = atoms[1] end
  return true
end

function SimpleCounter:in_1_bang()
  self:outlet(1, "float", {self.count})
  self.count = self.count + 1
end
```

67 examples/simplecounter.pd_luax

```
return function (self, sel, atoms)
  self.inlets = 1
  self.outlets = 1
  self.count = 0
  if type(atoms[1]) == "number" then self.count = atoms[1] end
  function self:in_1_bang()
    self:outlet(1, "float", {self.count})
    self.count = self.count + 1
  end
  return true
end
```

68 examples/swarm-help.pd

```
#N canvas 0 0 453 370 10;
#N canvas 0 0 450 300 \$0-swarm-visualisation 0;
#X obj 155 111 bng 8 250 50 0 \$0-bang-s \$0-1-r empty 17 7 0 10 -258113
-1 -1;
#X obj 164 85 bng 8 250 50 0 \$0-bang-s \$0-2-r empty 17 7 0 10 -258113
-1 -1;
#X obj 150 139 bng 8 250 50 0 \$0-bang-s \$0-3-r empty 17 7 0 10 -258113
-1 -1;
#X obj 121 102 bng 8 250 50 0 \$0-bang-s \$0-4-r empty 17 7 0 10 -258113
-1 -1;
#X obj 118 102 bng 8 250 50 0 \$0-bang-s \$0-5-r empty 17 7 0 10 -258113
-1 -1;
#X obj 129 145 bng 8 250 50 0 \$0-bang-s \$0-6-r empty 17 7 0 10 -258113
-1 -1;
#X obj 129 147 bng 8 250 50 0 \$0-bang-s \$0-7-r empty 17 7 0 10 -258113
-1 -1;
#X obj 128 101 bng 8 250 50 0 \$0-bang-s \$0-8-r empty 17 7 0 10 -258113
-1 -1;
#X obj 128 94 bng 8 250 50 0 \$0-bang-s \$0-9-r empty 17 7 0 10 -258113
-1 -1;
#X obj 97 124 bng 8 250 50 0 \$0-bang-s \$0-10-r empty 17 7 0 10 -258113
-1 -1;
#X obj 172 134 bng 8 250 50 0 \$0-bang-s \$0-11-r empty 17 7 0 10 -258113
-1 -1;
#X obj 125 146 bng 8 250 50 0 \$0-bang-s \$0-12-r empty 17 7 0 10 -258113
-1 -1;
#X obj 125 96 bng 8 250 50 0 \$0-bang-s \$0-13-r empty 17 7 0 10 -258113
-1 -1;
#X obj 94 126 bng 8 250 50 0 \$0-bang-s \$0-14-r empty 17 7 0 10 -258113
-1 -1;
#X obj 116 101 bng 8 250 50 0 \$0-bang-s \$0-15-r empty 17 7 0 10 -258113
-1 -1;
#X obj 107 107 bng 8 250 50 0 \$0-bang-s \$0-16-r empty 17 7 0 10 -258113
-1 -1;
#X obj 126 147 bng 8 250 50 0 \$0-bang-s \$0-17-r empty 17 7 0 10 -4034
-1 -1;
#X obj 125 148 bng 8 250 50 0 \$0-bang-s \$0-18-r empty 17 7 0 10 -258113
-1 -1;
#X obj 156 119 bng 8 250 50 0 \$0-bang-s \$0-19-r empty 17 7 0 10 -258113
-1 -1;
#X obj 103 92 bng 8 250 50 0 \$0-bang-s \$0-20-r empty 17 7 0 10 -258113
-1 -1;
#X obj 156 122 bng 8 250 50 0 \$0-bang-s \$0-21-r empty 17 7 0 10 -258113
-1 -1;
#X obj 117 146 bng 8 250 50 0 \$0-bang-s \$0-22-r empty 17 7 0 10 -258113
-1 -1;
#X obj 116 104 bng 8 250 50 0 \$0-bang-s \$0-23-r empty 17 7 0 10 -258113
-1 -1;
#X obj 130 155 bng 8 250 50 0 \$0-bang-s \$0-24-r empty 17 7 0 10 -258113
-1 -1;
#X obj 118 100 bng 8 250 50 0 \$0-bang-s \$0-25-r empty 17 7 0 10 -258113
-1 -1;
#X obj 159 82 bng 8 250 50 0 \$0-bang-s \$0-26-r empty 17 7 0 10 -258113
-1 -1;
#X obj 126 157 bng 8 250 50 0 \$0-bang-s \$0-27-r empty 17 7 0 10 -258113
-1 -1;
#X obj 125 157 bng 8 250 50 0 \$0-bang-s \$0-28-r empty 17 7 0 10 -258113
```

```
-1 -1;
#X obj 119 147 bng 8 250 50 0 \$0-bang-s \$0-29-r empty 17 7 0 10 -258113
-1 -1;
#X obj 150 125 bng 8 250 50 0 \$0-bang-s \$0-30-r empty 17 7 0 10 -258113
-1 -1;
#X obj 150 125 bng 8 250 50 0 \$0-bang-s \$0-31-r empty 17 7 0 10 -258113
-1 -1;
#X obj 155 103 bng 8 250 50 0 \$0-bang-s \$0-32-r empty 17 7 0 10 -258113
-1 -1;
#X coords 0 -1 1 1 200 200 2 0 0;
#X restore 220 31 pd \$0-swarm-visualisation;
#X obj 15 202 + 100;
#X obj 84 201 + 100;
#X obj 25 255 pack f f s;
#X obj 31 231 makefilename \$0-%d-r;
#X msg 25 279 \; \$3 pos \$1 \$2;
#X obj 13 4 tgl 15 0 empty empty empty 17 7 0 10 -262144 -1 -1 1 1
;
#X obj 29 57 bng 15 250 50 0 empty empty empty 17 7 0 10 -262144 -1
-1;
#X msg 75 72 randomize;
#X obj 85 178 * 100;
#X obj 16 179 * 100;
#X obj 17 132 swarm 2 32;
#X obj 17 157 unpack f f;
#X obj 133 256 list prepend;
#X obj 120 336 dac~;
#N canvas 0 0 1009 708 \$0-audio 0;
#X obj 9 74 unpack f f;
#X obj 7 97 * 200;
#X obj 6 121 + 300;
#X obj 5 143 osc~;
#X obj 71 102 * 0.5;
#X obj 69 129 + 0.5;
#X obj 137 28 route 1 2 3 4;
#X obj 5 174 *~ 0;
#X obj 127 77 unpack f f;
#X obj 125 100 * 200;
#X obj 124 124 + 300;
#X obj 123 146 osc~;
#X obj 189 105 * 0.5;
#X obj 187 132 + 0.5;
#X obj 123 177 *~ 0;
#X obj 198 220 *~ 0.25;
#X obj 236 78 unpack f f;
#X obj 234 101 * 200;
#X obj 233 125 + 300;
#X obj 232 147 osc~;
#X obj 298 106 * 0.5;
#X obj 296 133 + 0.5;
#X obj 232 178 *~ 0;
#X obj 354 81 unpack f f;
#X obj 352 104 * 200;
#X obj 351 128 + 300;
#X obj 350 150 osc~;
#X obj 416 109 * 0.5;
#X obj 414 136 + 0.5;
```

```
#X obj 350 181 *~ 0;
#X obj 490 76 unpack f f;
#X obj 488 99 * 200;
#X obj 487 123 + 300;
#X obj 486 145 osc~;
#X obj 552 104 * 0.5;
#X obj 550 131 + 0.5;
#X obj 486 176 *~ 0;
#X obj 608 79 unpack f f;
#X obj 606 102 * 200;
#X obj 605 126 + 300;
#X obj 604 148 osc~;
#X obj 670 107 * 0.5;
#X obj 668 134 + 0.5;
#X obj 604 179 *~ 0;
#X obj 679 222 *~ 0.25;
#X obj 717 80 unpack f f;
#X obj 715 103 * 200;
#X obj 714 127 + 300;
#X obj 713 149 osc~;
#X obj 779 108 * 0.5;
#X obj 777 135 + 0.5;
#X obj 713 180 *~ 0;
#X obj 835 83 unpack f f;
#X obj 833 106 * 200;
#X obj 832 130 + 300;
#X obj 831 152 osc~;
#X obj 897 111 * 0.5;
#X obj 895 138 + 0.5;
#X obj 831 183 *~ 0;
#X obj 618 30 route 5 6 7 8;
#X obj 137 7 inlet;
#X obj 9 294 unpack f f;
#X obj 7 317 * 200;
#X obj 6 341 + 300;
#X obj 5 363 osc~;
#X obj 71 322 * 0.5;
#X obj 69 349 + 0.5;
#X obj 5 394 *~ 0;
#X obj 127 297 unpack f f;
#X obj 125 320 * 200;
#X obj 124 344 + 300;
#X obj 123 366 osc~;
#X obj 189 325 * 0.5;
#X obj 187 352 + 0.5;
#X obj 123 397 *~ 0;
#X obj 198 440 *~ 0.25;
#X obj 236 298 unpack f f;
#X obj 234 321 * 200;
#X obj 233 345 + 300;
#X obj 232 367 osc~;
#X obj 298 326 * 0.5;
#X obj 296 353 + 0.5;
#X obj 232 398 *~ 0;
#X obj 354 301 unpack f f;
#X obj 352 324 * 200;
#X obj 351 348 + 300;
```

```
#X obj 350 370 osc~;
#X obj 416 329 * 0.5;
#X obj 414 356 + 0.5;
#X obj 350 401 *~ 0;
#X obj 490 296 unpack f f;
#X obj 488 319 * 200;
#X obj 487 343 + 300;
#X obj 486 365 osc~;
#X obj 552 324 * 0.5;
#X obj 550 351 + 0.5;
#X obj 486 396 *~ 0;
#X obj 608 299 unpack f f;
#X obj 606 322 * 200;
#X obj 605 346 + 300;
#X obj 604 368 osc~;
#X obj 670 327 * 0.5;
#X obj 668 354 + 0.5;
#X obj 604 399 *~ 0;
#X obj 679 442 *~ 0.25;
#X obj 717 300 unpack f f;
#X obj 715 323 * 200;
#X obj 714 347 + 300;
#X obj 713 369 osc~;
#X obj 779 328 * 0.5;
#X obj 777 355 + 0.5;
#X obj 713 400 *~ 0;
#X obj 835 303 unpack f f;
#X obj 833 326 * 200;
#X obj 832 350 + 300;
#X obj 831 372 osc~;
#X obj 897 331 * 0.5;
#X obj 895 358 + 0.5;
#X obj 831 403 *~ 0;
#X obj 137 248 route 9 10 11 12;
#X obj 618 250 route 13 14 15 16;
#X obj 4 514 unpack f f;
#X obj 2 537 * 200;
#X obj 1 561 + 300;
#X obj 0 583 osc~;
#X obj 66 542 * 0.5;
#X obj 64 569 + 0.5;
#X obj 0 614 *~ 0;
#X obj 122 517 unpack f f;
#X obj 120 540 * 200;
#X obj 119 564 + 300;
#X obj 118 586 osc~;
#X obj 184 545 * 0.5;
#X obj 182 572 + 0.5;
#X obj 118 617 *~ 0;
#X obj 193 660 *~ 0.25;
#X obj 231 518 unpack f f;
#X obj 229 541 * 200;
#X obj 228 565 + 300;
#X obj 227 587 osc~;
#X obj 293 546 * 0.5;
#X obj 291 573 + 0.5;
#X obj 227 618 *~ 0;
```

```
#X obj 349 521 unpack f f;
#X obj 347 544 * 200;
#X obj 346 568 + 300;
#X obj 345 590 osc~;
#X obj 411 549 * 0.5;
#X obj 409 576 + 0.5;
#X obj 345 621 *~ 0;
#X obj 485 516 unpack f f;
#X obj 483 539 * 200;
#X obj 482 563 + 300;
#X obj 481 585 osc~;
#X obj 547 544 * 0.5;
#X obj 545 571 + 0.5;
#X obj 481 616 *~ 0;
#X obj 603 519 unpack f f;
#X obj 601 542 * 200;
#X obj 600 566 + 300;
#X obj 599 588 osc~;
#X obj 665 547 * 0.5;
#X obj 663 574 + 0.5;
#X obj 599 619 *~ 0;
#X obj 674 662 *~ 0.25;
#X obj 712 520 unpack f f;
#X obj 710 543 * 200;
#X obj 709 567 + 300;
#X obj 708 589 osc~;
#X obj 774 548 * 0.5;
#X obj 772 575 + 0.5;
#X obj 708 620 *~ 0;
#X obj 830 523 unpack f f;
#X obj 828 546 * 200;
#X obj 827 570 + 300;
#X obj 826 592 osc~;
#X obj 892 551 * 0.5;
#X obj 890 578 + 0.5;
#X obj 826 623 *~ 0;
#X obj 4 734 unpack f f;
#X obj 2 757 * 200;
#X obj 1 781 + 300;
#X obj 0 803 osc~;
#X obj 66 762 * 0.5;
#X obj 64 789 + 0.5;
#X obj 0 834 *~ 0;
#X obj 122 737 unpack f f;
#X obj 120 760 * 200;
#X obj 119 784 + 300;
#X obj 118 806 osc~;
#X obj 184 765 * 0.5;
#X obj 182 792 + 0.5;
#X obj 118 837 *~ 0;
#X obj 193 880 *~ 0.25;
#X obj 231 738 unpack f f;
#X obj 229 761 * 200;
#X obj 228 785 + 300;
#X obj 227 807 osc~;
#X obj 293 766 * 0.5;
#X obj 291 793 + 0.5;
```

```
#X obj 227 838 *~ 0;
#X obj 349 741 unpack f f;
#X obj 347 764 * 200;
#X obj 346 788 + 300;
#X obj 345 810 osc~;
#X obj 411 769 * 0.5;
#X obj 409 796 + 0.5;
#X obj 345 841 *~ 0;
#X obj 485 736 unpack f f;
#X obj 483 759 * 200;
#X obj 482 783 + 300;
#X obj 481 805 osc~;
#X obj 547 764 * 0.5;
#X obj 545 791 + 0.5;
#X obj 481 836 *~ 0;
#X obj 603 739 unpack f f;
#X obj 601 762 * 200;
#X obj 600 786 + 300;
#X obj 599 808 osc~;
#X obj 665 767 * 0.5;
#X obj 663 794 + 0.5;
#X obj 599 839 *~ 0;
#X obj 674 882 *~ 0.25;
#X obj 712 740 unpack f f;
#X obj 710 763 * 200;
#X obj 709 787 + 300;
#X obj 708 809 osc~;
#X obj 774 768 * 0.5;
#X obj 772 795 + 0.5;
#X obj 708 840 *~ 0;
#X obj 830 743 unpack f f;
#X obj 828 766 * 200;
#X obj 827 790 + 300;
#X obj 826 812 osc~;
#X obj 892 771 * 0.5;
#X obj 890 798 + 0.5;
#X obj 826 843 *~ 0;
#X obj 132 468 route 17 18 19 20;
#X obj 613 470 route 21 22 23 24;
#X obj 132 689 route 25 26 27 28;
#X obj 613 690 route 29 30 31 32;
#X obj 418 910 outlet~;
#X text 360 4 stupid copy and paste \, but whatever...;
#X obj 410 887 *~ 0.125;
#X connect 0 0 1 0;
#X connect 0 1 4 0;
#X connect 1 0 2 0;
#X connect 2 0 3 0;
#X connect 3 0 7 0;
#X connect 4 0 5 0;
#X connect 5 0 7 1;
#X connect 6 0 0 0;
#X connect 6 1 8 0;
#X connect 6 2 16 0;
#X connect 6 3 23 0;
#X connect 6 4 59 0;
#X connect 7 0 15 0;
```

```
#X connect 8 0 9 0;
#X connect 8 1 12 0;
#X connect 9 0 10 0;
#X connect 10 0 11 0;
#X connect 11 0 14 0;
#X connect 12 0 13 0;
#X connect 13 0 14 1;
#X connect 14 0 15 0;
#X connect 15 0 243 0;
#X connect 16 0 17 0;
#X connect 16 1 20 0;
#X connect 17 0 18 0;
#X connect 18 0 19 0;
#X connect 19 0 22 0;
#X connect 20 0 21 0;
#X connect 21 0 22 1;
#X connect 22 0 15 0;
#X connect 23 0 24 0;
#X connect 23 1 27 0;
#X connect 24 0 25 0;
#X connect 25 0 26 0;
#X connect 26 0 29 0;
#X connect 27 0 28 0;
#X connect 28 0 29 1;
#X connect 29 0 15 0;
#X connect 30 0 31 0;
#X connect 30 1 34 0;
#X connect 31 0 32 0;
#X connect 32 0 33 0;
#X connect 33 0 36 0;
#X connect 34 0 35 0;
#X connect 35 0 36 1;
#X connect 36 0 44 0;
#X connect 37 0 38 0;
#X connect 37 1 41 0;
#X connect 38 0 39 0;
#X connect 39 0 40 0;
#X connect 40 0 43 0;
#X connect 41 0 42 0;
#X connect 42 0 43 1;
#X connect 43 0 44 0;
#X connect 44 0 243 0;
#X connect 45 0 46 0;
#X connect 45 1 49 0;
#X connect 46 0 47 0;
#X connect 47 0 48 0;
#X connect 48 0 51 0;
#X connect 49 0 50 0;
#X connect 50 0 51 1;
#X connect 51 0 44 0;
#X connect 52 0 53 0;
#X connect 52 1 56 0;
#X connect 53 0 54 0;
#X connect 54 0 55 0;
#X connect 55 0 58 0;
#X connect 56 0 57 0;
#X connect 57 0 58 1;
```

```
#X connect 58 0 44 0;
#X connect 59 0 30 0;
#X connect 59 1 37 0;
#X connect 59 2 45 0;
#X connect 59 3 52 0;
#X connect 59 4 119 0;
#X connect 60 0 6 0;
#X connect 61 0 62 0;
#X connect 61 1 65 0;
#X connect 62 0 63 0;
#X connect 63 0 64 0;
#X connect 64 0 67 0;
#X connect 65 0 66 0;
#X connect 66 0 67 1;
#X connect 67 0 75 0;
#X connect 68 0 69 0;
#X connect 68 1 72 0;
#X connect 69 0 70 0;
#X connect 70 0 71 0;
#X connect 71 0 74 0;
#X connect 72 0 73 0;
#X connect 73 0 74 1;
#X connect 74 0 75 0;
#X connect 75 0 243 0;
#X connect 76 0 77 0;
#X connect 76 1 80 0;
#X connect 77 0 78 0;
#X connect 78 0 79 0;
#X connect 79 0 82 0;
#X connect 80 0 81 0;
#X connect 81 0 82 1;
#X connect 82 0 75 0;
#X connect 83 0 84 0;
#X connect 83 1 87 0;
#X connect 84 0 85 0;
#X connect 85 0 86 0;
#X connect 86 0 89 0;
#X connect 87 0 88 0;
#X connect 88 0 89 1;
#X connect 89 0 75 0;
#X connect 90 0 91 0;
#X connect 90 1 94 0;
#X connect 91 0 92 0;
#X connect 92 0 93 0;
#X connect 93 0 96 0;
#X connect 94 0 95 0;
#X connect 95 0 96 1;
#X connect 96 0 104 0;
#X connect 97 0 98 0;
#X connect 97 1 101 0;
#X connect 98 0 99 0;
#X connect 99 0 100 0;
#X connect 100 0 103 0;
#X connect 101 0 102 0;
#X connect 102 0 103 1;
#X connect 103 0 104 0;
#X connect 104 0 243 0;
```

```
#X connect 105 0 106 0;
#X connect 105 1 109 0;
#X connect 106 0 107 0;
#X connect 107 0 108 0;
#X connect 108 0 111 0;
#X connect 109 0 110 0;
#X connect 110 0 111 1;
#X connect 111 0 104 0;
#X connect 112 0 113 0;
#X connect 112 1 116 0;
#X connect 113 0 114 0;
#X connect 114 0 115 0;
#X connect 115 0 118 0;
#X connect 116 0 117 0;
#X connect 117 0 118 1;
#X connect 118 0 104 0;
#X connect 119 0 61 0;
#X connect 119 1 68 0;
#X connect 119 2 76 0;
#X connect 119 3 83 0;
#X connect 119 4 120 0;
#X connect 120 0 90 0;
#X connect 120 1 97 0;
#X connect 120 2 105 0;
#X connect 120 3 112 0;
#X connect 120 4 237 0;
#X connect 121 0 122 0;
#X connect 121 1 125 0;
#X connect 122 0 123 0;
#X connect 123 0 124 0;
#X connect 124 0 127 0;
#X connect 125 0 126 0;
#X connect 126 0 127 1;
#X connect 127 0 135 0;
#X connect 128 0 129 0;
#X connect 128 1 132 0;
#X connect 129 0 130 0;
#X connect 130 0 131 0;
#X connect 131 0 134 0;
#X connect 132 0 133 0;
#X connect 133 0 134 1;
#X connect 134 0 135 0;
#X connect 135 0 243 0;
#X connect 136 0 137 0;
#X connect 136 1 140 0;
#X connect 137 0 138 0;
#X connect 138 0 139 0;
#X connect 139 0 142 0;
#X connect 140 0 141 0;
#X connect 141 0 142 1;
#X connect 142 0 135 0;
#X connect 143 0 144 0;
#X connect 143 1 147 0;
#X connect 144 0 145 0;
#X connect 145 0 146 0;
#X connect 146 0 149 0;
#X connect 147 0 148 0;
```

```
#X connect 148 0 149 1;
#X connect 149 0 135 0;
#X connect 150 0 151 0;
#X connect 150 1 154 0;
#X connect 151 0 152 0;
#X connect 152 0 153 0;
#X connect 153 0 156 0;
#X connect 154 0 155 0;
#X connect 155 0 156 1;
#X connect 156 0 164 0;
#X connect 157 0 158 0;
#X connect 157 1 161 0;
#X connect 158 0 159 0;
#X connect 159 0 160 0;
#X connect 160 0 163 0;
#X connect 161 0 162 0;
#X connect 162 0 163 1;
#X connect 163 0 164 0;
#X connect 164 0 243 0;
#X connect 165 0 166 0;
#X connect 165 1 169 0;
#X connect 166 0 167 0;
#X connect 167 0 168 0;
#X connect 168 0 171 0;
#X connect 169 0 170 0;
#X connect 170 0 171 1;
#X connect 171 0 164 0;
#X connect 172 0 173 0;
#X connect 172 1 176 0;
#X connect 173 0 174 0;
#X connect 174 0 175 0;
#X connect 175 0 178 0;
#X connect 176 0 177 0;
#X connect 177 0 178 1;
#X connect 178 0 164 0;
#X connect 179 0 180 0;
#X connect 179 1 183 0;
#X connect 180 0 181 0;
#X connect 181 0 182 0;
#X connect 182 0 185 0;
#X connect 183 0 184 0;
#X connect 184 0 185 1;
#X connect 185 0 193 0;
#X connect 186 0 187 0;
#X connect 186 1 190 0;
#X connect 187 0 188 0;
#X connect 188 0 189 0;
#X connect 189 0 192 0;
#X connect 190 0 191 0;
#X connect 191 0 192 1;
#X connect 192 0 193 0;
#X connect 193 0 243 0;
#X connect 194 0 195 0;
#X connect 194 1 198 0;
#X connect 195 0 196 0;
#X connect 196 0 197 0;
#X connect 197 0 200 0;
```

```
#X connect 198 0 199 0;
#X connect 199 0 200 1;
#X connect 200 0 193 0;
#X connect 201 0 202 0;
#X connect 201 1 205 0;
#X connect 202 0 203 0;
#X connect 203 0 204 0;
#X connect 204 0 207 0;
#X connect 205 0 206 0;
#X connect 206 0 207 1;
#X connect 207 0 193 0;
#X connect 208 0 209 0;
#X connect 208 1 212 0;
#X connect 209 0 210 0;
#X connect 210 0 211 0;
#X connect 211 0 214 0;
#X connect 212 0 213 0;
#X connect 213 0 214 1;
#X connect 214 0 222 0;
#X connect 215 0 216 0;
#X connect 215 1 219 0;
#X connect 216 0 217 0;
#X connect 217 0 218 0;
#X connect 218 0 221 0;
#X connect 219 0 220 0;
#X connect 220 0 221 1;
#X connect 221 0 222 0;
#X connect 222 0 243 0;
#X connect 223 0 224 0;
#X connect 223 1 227 0;
#X connect 224 0 225 0;
#X connect 225 0 226 0;
#X connect 226 0 229 0;
#X connect 227 0 228 0;
#X connect 228 0 229 1;
#X connect 229 0 222 0;
#X connect 230 0 231 0;
#X connect 230 1 234 0;
#X connect 231 0 232 0;
#X connect 232 0 233 0;
#X connect 233 0 236 0;
#X connect 234 0 235 0;
#X connect 235 0 236 1;
#X connect 236 0 222 0;
#X connect 237 0 121 0;
#X connect 237 1 128 0;
#X connect 237 2 136 0;
#X connect 237 3 143 0;
#X connect 237 4 238 0;
#X connect 238 0 150 0;
#X connect 238 1 157 0;
#X connect 238 2 165 0;
#X connect 238 3 172 0;
#X connect 238 4 239 0;
#X connect 239 0 179 0;
#X connect 239 1 186 0;
#X connect 239 2 194 0;
```

```
#X connect 239 3 201 0;
#X connect 239 4 240 0;
#X connect 240 0 208 0;
#X connect 240 1 215 0;
#X connect 240 2 223 0;
#X connect 240 3 230 0;
#X connect 243 0 241 0;
#X restore 133 282 pd \$0-audio;
#X obj 13 33 metro 40;
#X text 181 309 boids-style swarm in pd with Lua;
#X connect 1 0 3 0;
#X connect 2 0 3 1;
#X connect 3 0 5 0;
#X connect 4 0 3 2;
#X connect 6 0 16 0;
#X connect 7 0 11 0;
#X connect 8 0 11 0;
#X connect 9 0 2 0;
#X connect 10 0 1 0;
#X connect 11 0 12 0;
#X connect 11 0 13 0;
#X connect 11 1 4 0;
#X connect 11 1 13 1;
#X connect 12 0 10 0;
#X connect 12 1 9 0;
#X connect 13 0 15 0;
#X connect 15 0 14 0;
#X connect 15 0 14 1;
#X connect 16 0 11 0;
```

69 examples/swarm.pd_lua

```
-- see also: http://www.vergenet.net/~conrad/boids/pseudocode.html
local swarm = pd.Class:new():register("swarm")

function swarm:initialize(sel, atoms) -- constructor
  if type(atoms[1]) ~= "number" or atoms[1] < 2 then return false end
  if type(atoms[2]) ~= "number" or atoms[2] < 3 then return false end
  self.dim = math.floor(atoms[1])
  self.count = math.floor(atoms[2])
  self.cluster = 0.05 -- magic values look ok in the help patch..
  self.distance2 = 0.2
  self.similar2 = 0.1
  self.friction = 0.96
  self.flock = { }
  self:in_1_randomize()
  self.inlets = 2
  self.outlets = 2
  return true
end

function swarm:in_1_randomize() -- randomize positions, no movement
  for i = 1, self.count do
    self.flock[i] = { x = { }, dx = { } }
    for j = 1, self.dim do
      self.flock[i].x[j] = math.random() - 0.5
      self.flock[i].dx[j] = 0
```

```
    end
    self.flock[i].w = math.random() + 0.5
  end
end

function swarm:in_1_bang()  -- update and output
  local c = self:center()
  for i = 1, self.count do
    f = self.flock[i]                       -- update
    local v1 = self:rule1(c, f)
    local v2 = self:rule2(i, f)
    local v3 = self:rule3(i, f)
    for k = 1, self.dim do f.dx[k] = f.dx[k] + v1[k] + v2[k] + v3[k] end
    for k = 1, self.dim do f.dx[k] = f.dx[k] * self.friction end
    for k = 1, self.dim do f.x[k] = f.x[k] + f.dx[k] end
    self:outlet(2, "float", { i })      -- output
    self:outlet(1, "list", f.x)
  end
end

function swarm:center() -- center of mass
  local c = { }
  local w = 0
  for k = 1, self.dim do c[k] = 0 end
  for i = 1, self.count do
    w = w + self.flock[i].w
    for k = 1, self.dim do c[k] = c[k] + self.flock[i].w * self.flock[i].x[k] ↙
        ↳ end
  end
  for k = 1, self.dim do c[k] = c[k] / w end
  return c
end

function swarm:rule1(c, f) -- clustering
  local v = { }
  for k = 1, self.dim do v[k] = self.cluster * (c[k] - (1 + f.w) * f.x[k]) end
  return v
end

function swarm:rule2(i, f) -- avoidance
  local v = { }
  for k = 1, self.dim do v[k] = 0 end
  for j = 1, self.count do
    if i ~= j then
      g = self.flock[j]
      local d = { }
      local m = 0
      for k = 1, self.dim do d[k] = g.x[k] - f.x[k] ; m = m + d[k] * d[k] end
      if m < self.distance2 then
        for k = 1, self.dim do v[k] = v[k] - d[k] end
      end
    end
  end
  for k = 1, self.dim do v[k] = 0.01 * v[k] end
  return v
end
```

```
function swarm:rule3(i, f) -- similarity
  local v = { }
  for k = 1, self.dim do v[k] = 0 end
  for j = 1, self.count do
    if i ~= j then
      g = self.flock[j]
      local d = { }
      local m = 0
      for k = 1, self.dim do d[k] = g.dx[k] - f.dx[k] ; m = m + d[k] * d[k] end
      if m < self.similar2 then
        for k = 1, self.dim do v[k] = v[k] + d[k] end
      end
    end
  end
  for k = 1, self.dim do v[k] = 0.004 * v[k] end
  return v
end

-- exercise: make the right inlet control individual elements
```

70 .gitignore

```
Makefile
Makefile.in
aclocal.m4
autom4te.cache/
compile
config.log
config.status
configure
depcomp
install-sh
missing
src/.deps/
src/Makefile
src/Makefile.in
src/config.h
src/config.h.in
src/lua-lua.o
src/lua.pd_linux
src/stamp-h1
```

71 Makefile.am

```
AUTOMAKE_OPTIONS = foreign
SUBDIRS = src
```

72 Makefile.static

```
#############################################################################
# User configuration goes here

# Uncomment the particular line according to your platform
PLATFORM ?= linux
#PLATFORM = macosx
#PLATFORM = mingw
```

```
# If Pd is in an unusual place, put -I/path/to m_pd.h here:
PDINCLUDE ?=
#PDINCLUDE = -I./
#PDINCLUDE = -I$(HOME)/include
#PDINCLUDE = -I/opt/puredata/include
#PDINCLUDE = -I/Applications/Pd-0.41-4.app/Contents/Resources/src

# End of user configuration
##############################################################################

# which Pd external to build
lua_src = src/lua.c
lua_linux = src/lua.pd_linux
lua_macosx = src/lua.pd_darwin
lua_mingw = src/lua.dll

# how to build a Pd external
# FIXME: Windows?
CFLAGS_linux = -shared
LIBS_linux =
CFLAGS_macosx = -bundle -undefined suppress -flat_namespace
LIBS_macosx =
CFLAGS_mingw = -shared -DMSW
LIBS_mingw = pd.dll

# Lua setup
LUA = lua-5.1.5
LUATGZ = $(LUA).tar.gz
LUAURL = http://www.lua.org/ftp/$(LUATGZ)
LUAI = -I$(LUA)/include/
LUAL = $(LUA)/lib/liblua.a
CFLAGS = -pedantic -Wall -Wextra -O2 -fPIC $(LUAI) $(PDINCLUDE)

# build
default: $(lua_$(PLATFORM))

# clean up
clean:
	rm -f $(lua_$(PLATFORM))

# install
install:
	echo "TODO"

# compile
$(lua_$(PLATFORM)): $(lua_src) $(LUA)/build.stamp
	gcc $(CFLAGS) $(CFLAGS_$(PLATFORM)) -o $(lua_$(PLATFORM)) $(lua_src) $(↙
	    ↳ LUAL) $(LIBS_$(PLATFORM))

# get Lua: download, unpack, compile, install locally
$(LUA)/build.stamp: $(LUA)/patch.stamp
	make -C $(LUA) $(PLATFORM) local
	touch $(LUA)/build.stamp

$(LUA)/patch.stamp: $(LUA)/unpack.stamp
	sed -i 's/^CFLAGS= -O2 -Wall $$(MYCFLAGS)$$/CFLAGS= -fPIC -O2 -Wall $$(↙
```

```
            ↳ MYCFLAGS)/g' $(LUA)/src/Makefile
        touch $(LUA)/patch.stamp

$(LUA)/unpack.stamp: $(LUATGZ)
        tar xzf $(LUATGZ)
        touch $(LUA)/unpack.stamp

$(LUATGZ):
        wget $(LUAURL)
```

73 README

```
pdlua -- a Lua embedding for Pd
Copyright (C) 2007,2008,2009,2013 Claude Heiland-Allen <claude@mathr.co.uk>
Copyright (C) 2012 Martin Peach martin.peach@sympatico.ca

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.


Lua 5.1 is highly recommended, other versions are untested and
unsupported - use at your own risk.

For a newer fork with Lua 5.3 support see:
https://github.com/agraef/pd-lua


Compilation Instructions:

make -f Makefile.static PLATFORM=linux PDINCLUDE=-I/path/to/pd/src


Usage instructions:

pd -lib /path/to/pdlua/src/lua


Alternatively, for the brave:

aclocal
autoheader
automake -a -c
autoconf
./configure
make
```

```
make install


Additional notes for mingw (thanks to David Shimamomto):

Copy the following files to the same directory as 'Makefile.static':
1. lua-5.1.5.tar.gz
2. m_pd.h
3. pd.dll

make -f Makefile.static PLATFORM=mingw PDINCLUDE=-I./

To install, copy the following files to 'extra':
1. src/lua.dll
2. src/pd.lua
```

74 src/Doxyfile

```
# Doxyfile 1.5.1

# This file describes the settings to be used by the documentation system
# doxygen (www.doxygen.org) for a project
#
# All text after a hash (#) is considered a comment and will be ignored
# The format is:
#       TAG = value [value, ...]
# For lists items can also be appended using:
#       TAG += value [value, ...]
# Values that contain spaces should be placed between quotes (" ")

#---------------------------------------------------------------------------
# Project related configuration options
#---------------------------------------------------------------------------

# The PROJECT_NAME tag is a single word (or a sequence of words surrounded
# by quotes) that should identify the project.

PROJECT_NAME           = pdlua

# The PROJECT_NUMBER tag can be used to enter a project or revision number.
# This could be handy for archiving the generated documentation or
# if some version control system is used.

PROJECT_NUMBER         = 0.8

# The OUTPUT_DIRECTORY tag is used to specify the (relative or absolute)
# base path where the generated documentation will be put.
# If a relative path is entered, it will be relative to the location
# where doxygen was started. If left blank the current directory will be used.

OUTPUT_DIRECTORY       = ../doc

# If the CREATE_SUBDIRS tag is set to YES, then doxygen will create
# 4096 sub-directories (in 2 levels) under the output directory of each output
# format and will distribute the generated files over these directories.
# Enabling this option can be useful when feeding doxygen a huge amount of
# source files, where putting all generated files in the same directory would
```

```
# otherwise cause performance problems for the file system.

CREATE_SUBDIRS         = NO

# The OUTPUT_LANGUAGE tag is used to specify the language in which all
# documentation generated by doxygen is written. Doxygen will use this
# information to generate all constant output in the proper language.
# The default language is English, other supported languages are:
# Afrikaans, Arabic, Brazilian, Catalan, Chinese, Chinese-Traditional,
# Croatian, Czech, Danish, Dutch, Finnish, French, German, Greek, Hungarian,
# Italian, Japanese, Japanese-en (Japanese with English messages), Korean,
# Korean-en, Lithuanian, Norwegian, Polish, Portuguese, Romanian, Russian,
# Serbian, Slovak, Slovene, Spanish, Swedish, and Ukrainian.

OUTPUT_LANGUAGE        = English

# This tag can be used to specify the encoding used in the generated output.
# The encoding is not always determined by the language that is chosen,
# but also whether or not the output is meant for Windows or non-Windows users.
# In case there is a difference, setting the USE_WINDOWS_ENCODING tag to YES
# forces the Windows encoding (this is the default for the Windows binary),
# whereas setting the tag to NO uses a Unix-style encoding (the default for
# all platforms other than Windows).

USE_WINDOWS_ENCODING   = NO

# If the BRIEF_MEMBER_DESC tag is set to YES (the default) Doxygen will
# include brief member descriptions after the members that are listed in
# the file and class documentation (similar to JavaDoc).
# Set to NO to disable this.

BRIEF_MEMBER_DESC      = YES

# If the REPEAT_BRIEF tag is set to YES (the default) Doxygen will prepend
# the brief description of a member or function before the detailed description.
# Note: if both HIDE_UNDOC_MEMBERS and BRIEF_MEMBER_DESC are set to NO, the
# brief descriptions will be completely suppressed.

REPEAT_BRIEF           = YES

# This tag implements a quasi-intelligent brief description abbreviator
# that is used to form the text in various listings. Each string
# in this list, if found as the leading text of the brief description, will be
# stripped from the text and the result after processing the whole list, is
# used as the annotated text. Otherwise, the brief description is used as-is.
# If left blank, the following values are used ("$name" is automatically
# replaced with the name of the entity): "The $name class" "The $name widget"
# "The $name file" "is" "provides" "specifies" "contains"
# "represents" "a" "an" "the"

ABBREVIATE_BRIEF       =

# If the ALWAYS_DETAILED_SEC and REPEAT_BRIEF tags are both set to YES then
# Doxygen will generate a detailed section even if there is only a brief
# description.

ALWAYS_DETAILED_SEC    = NO
```

```
# If the INLINE_INHERITED_MEMB tag is set to YES, doxygen will show all
# inherited members of a class in the documentation of that class as if those
# members were ordinary class members. Constructors, destructors and assignment
# operators of the base classes will not be shown.

INLINE_INHERITED_MEMB  = NO

# If the FULL_PATH_NAMES tag is set to YES then Doxygen will prepend the full
# path before files name in the file list and in the header files. If set
# to NO the shortest path that makes the file name unique will be used.

FULL_PATH_NAMES        = YES

# If the FULL_PATH_NAMES tag is set to YES then the STRIP_FROM_PATH tag
# can be used to strip a user-defined part of the path. Stripping is
# only done if one of the specified strings matches the left-hand part of
# the path. The tag can be used to show relative paths in the file list.
# If left blank the directory from which doxygen is run is used as the
# path to strip.

STRIP_FROM_PATH        =

# The STRIP_FROM_INC_PATH tag can be used to strip a user-defined part of
# the path mentioned in the documentation of a class, which tells
# the reader which header file to include in order to use a class.
# If left blank only the name of the header file containing the class
# definition is used. Otherwise one should specify the include paths that
# are normally passed to the compiler using the -I flag.

STRIP_FROM_INC_PATH    =

# If the SHORT_NAMES tag is set to YES, doxygen will generate much shorter
# (but less readable) file names. This can be useful is your file systems
# doesn't support long names like on DOS, Mac, or CD-ROM.

SHORT_NAMES            = NO

# If the JAVADOC_AUTOBRIEF tag is set to YES then Doxygen
# will interpret the first line (until the first dot) of a JavaDoc-style
# comment as the brief description. If set to NO, the JavaDoc
# comments will behave just like the Qt-style comments (thus requiring an
# explicit @brief command for a brief description.

JAVADOC_AUTOBRIEF      = NO

# The MULTILINE_CPP_IS_BRIEF tag can be set to YES to make Doxygen
# treat a multi-line C++ special comment block (i.e. a block of //! or ///
# comments) as a brief description. This used to be the default behaviour.
# The new default is to treat a multi-line C++ comment block as a detailed
# description. Set this tag to YES if you prefer the old behaviour instead.

MULTILINE_CPP_IS_BRIEF = NO

# If the DETAILS_AT_TOP tag is set to YES then Doxygen
# will output the detailed description near the top, like JavaDoc.
# If set to NO, the detailed description appears after the member
```

```
# documentation.

DETAILS_AT_TOP         = NO

# If the INHERIT_DOCS tag is set to YES (the default) then an undocumented
# member inherits the documentation from any documented member that it
# re-implements.

INHERIT_DOCS           = YES

# If the SEPARATE_MEMBER_PAGES tag is set to YES, then doxygen will produce
# a new page for each member. If set to NO, the documentation of a member will
# be part of the file/class/namespace that contains it.

SEPARATE_MEMBER_PAGES  = NO

# The TAB_SIZE tag can be used to set the number of spaces in a tab.
# Doxygen uses this value to replace tabs by spaces in code fragments.

TAB_SIZE               = 8

# This tag can be used to specify a number of aliases that acts
# as commands in the documentation. An alias has the form "name=value".
# For example adding "sideeffect=\par Side Effects:\n" will allow you to
# put the command \sideeffect (or @sideeffect) in the documentation, which
# will result in a user-defined paragraph with heading "Side Effects:".
# You can put \n's in the value part of an alias to insert newlines.

ALIASES                =

# Set the OPTIMIZE_OUTPUT_FOR_C tag to YES if your project consists of C
# sources only. Doxygen will then generate output that is more tailored for C.
# For instance, some of the names that are used will be different. The list
# of all members will be omitted, etc.

OPTIMIZE_OUTPUT_FOR_C  = YES

# Set the OPTIMIZE_OUTPUT_JAVA tag to YES if your project consists of Java
# sources only. Doxygen will then generate output that is more tailored for Java↙
    ↳ .
# For instance, namespaces will be presented as packages, qualified scopes
# will look different, etc.

OPTIMIZE_OUTPUT_JAVA   = NO

# If you use STL classes (i.e. std::string, std::vector, etc.) but do not want ↙
    ↳ to
# include (a tag file for) the STL sources as input, then you should
# set this tag to YES in order to let doxygen match functions declarations and
# definitions whose arguments contain STL classes (e.g. func(std::string); v.s.
# func(std::string) {}). This also make the inheritance and collaboration
# diagrams that involve STL classes more complete and accurate.

BUILTIN_STL_SUPPORT    = NO

# If member grouping is used in the documentation and the DISTRIBUTE_GROUP_DOC
# tag is set to YES, then doxygen will reuse the documentation of the first
```

```
# member in the group (if any) for the other members of the group. By default
# all members of a group must be documented explicitly.

DISTRIBUTE_GROUP_DOC   = NO

# Set the SUBGROUPING tag to YES (the default) to allow class member groups of
# the same type (for instance a group of public functions) to be put as a
# subgroup of that type (e.g. under the Public Functions section). Set it to
# NO to prevent subgrouping. Alternatively, this can be done per class using
# the \nosubgrouping command.

SUBGROUPING            = YES

#---------------------------------------------------------------------------
# Build related configuration options
#---------------------------------------------------------------------------

# If the EXTRACT_ALL tag is set to YES doxygen will assume all entities in
# documentation are documented, even if no documentation was available.
# Private class members and static file members will be hidden unless
# the EXTRACT_PRIVATE and EXTRACT_STATIC tags are set to YES

EXTRACT_ALL            = YES

# If the EXTRACT_PRIVATE tag is set to YES all private members of a class
# will be included in the documentation.

EXTRACT_PRIVATE        = YES

# If the EXTRACT_STATIC tag is set to YES all static members of a file
# will be included in the documentation.

EXTRACT_STATIC         = YES

# If the EXTRACT_LOCAL_CLASSES tag is set to YES classes (and structs)
# defined locally in source files will be included in the documentation.
# If set to NO only classes defined in header files are included.

EXTRACT_LOCAL_CLASSES  = YES

# This flag is only useful for Objective-C code. When set to YES local
# methods, which are defined in the implementation section but not in
# the interface are included in the documentation.
# If set to NO (the default) only methods in the interface are included.

EXTRACT_LOCAL_METHODS  = NO

# If the HIDE_UNDOC_MEMBERS tag is set to YES, Doxygen will hide all
# undocumented members of documented classes, files or namespaces.
# If set to NO (the default) these members will be included in the
# various overviews, but no documentation section is generated.
# This option has no effect if EXTRACT_ALL is enabled.

HIDE_UNDOC_MEMBERS     = NO

# If the HIDE_UNDOC_CLASSES tag is set to YES, Doxygen will hide all
# undocumented classes that are normally visible in the class hierarchy.
```

```
# If set to NO (the default) these classes will be included in the various
# overviews. This option has no effect if EXTRACT_ALL is enabled.

HIDE_UNDOC_CLASSES     = NO

# If the HIDE_FRIEND_COMPOUNDS tag is set to YES, Doxygen will hide all
# friend (class|struct|union) declarations.
# If set to NO (the default) these declarations will be included in the
# documentation.

HIDE_FRIEND_COMPOUNDS  = NO

# If the HIDE_IN_BODY_DOCS tag is set to YES, Doxygen will hide any
# documentation blocks found inside the body of a function.
# If set to NO (the default) these blocks will be appended to the
# function's detailed documentation block.

HIDE_IN_BODY_DOCS      = NO

# The INTERNAL_DOCS tag determines if documentation
# that is typed after a \internal command is included. If the tag is set
# to NO (the default) then the documentation will be excluded.
# Set it to YES to include the internal documentation.

INTERNAL_DOCS          = YES

# If the CASE_SENSE_NAMES tag is set to NO then Doxygen will only generate
# file names in lower-case letters. If set to YES upper-case letters are also
# allowed. This is useful if you have classes or files whose names only differ
# in case and if your file system supports case sensitive file names. Windows
# and Mac users are advised to set this option to NO.

CASE_SENSE_NAMES       = YES

# If the HIDE_SCOPE_NAMES tag is set to NO (the default) then Doxygen
# will show members with their full class and namespace scopes in the
# documentation. If set to YES the scope will be hidden.

HIDE_SCOPE_NAMES       = NO

# If the SHOW_INCLUDE_FILES tag is set to YES (the default) then Doxygen
# will put a list of the files that are included by a file in the documentation
# of that file.

SHOW_INCLUDE_FILES     = YES

# If the INLINE_INFO tag is set to YES (the default) then a tag [inline]
# is inserted in the documentation for inline members.

INLINE_INFO            = YES

# If the SORT_MEMBER_DOCS tag is set to YES (the default) then doxygen
# will sort the (detailed) documentation of file and class members
# alphabetically by member name. If set to NO the members will appear in
# declaration order.

SORT_MEMBER_DOCS       = YES
```

```
# If the SORT_BRIEF_DOCS tag is set to YES then doxygen will sort the
# brief documentation of file, namespace and class members alphabetically
# by member name. If set to NO (the default) the members will appear in
# declaration order.

SORT_BRIEF_DOCS        = YES

# If the SORT_BY_SCOPE_NAME tag is set to YES, the class list will be
# sorted by fully-qualified names, including namespaces. If set to
# NO (the default), the class list will be sorted only by class name,
# not including the namespace part.
# Note: This option is not very useful if HIDE_SCOPE_NAMES is set to YES.
# Note: This option applies only to the class list, not to the
# alphabetical list.

SORT_BY_SCOPE_NAME     = NO

# The GENERATE_TODOLIST tag can be used to enable (YES) or
# disable (NO) the todo list. This list is created by putting \todo
# commands in the documentation.

GENERATE_TODOLIST      = YES

# The GENERATE_TESTLIST tag can be used to enable (YES) or
# disable (NO) the test list. This list is created by putting \test
# commands in the documentation.

GENERATE_TESTLIST      = YES

# The GENERATE_BUGLIST tag can be used to enable (YES) or
# disable (NO) the bug list. This list is created by putting \bug
# commands in the documentation.

GENERATE_BUGLIST       = YES

# The GENERATE_DEPRECATEDLIST tag can be used to enable (YES) or
# disable (NO) the deprecated list. This list is created by putting
# \deprecated commands in the documentation.

GENERATE_DEPRECATEDLIST= YES

# The ENABLED_SECTIONS tag can be used to enable conditional
# documentation sections, marked by \if sectionname ... \endif.

ENABLED_SECTIONS       =

# The MAX_INITIALIZER_LINES tag determines the maximum number of lines
# the initial value of a variable or define consists of for it to appear in
# the documentation. If the initializer consists of more lines than specified
# here it will be hidden. Use a value of 0 to hide initializers completely.
# The appearance of the initializer of individual variables and defines in the
# documentation can be controlled using \showinitializer or \hideinitializer
# command in the documentation regardless of this setting.

MAX_INITIALIZER_LINES  = 30
```

```
380 # Set the SHOW_USED_FILES tag to NO to disable the list of files generated
    # at the bottom of the documentation of classes and structs. If set to YES the
    # list will mention the files that were used to generate the documentation.

    SHOW_USED_FILES        = YES
385
    # If the sources in your project are distributed over multiple directories
    # then setting the SHOW_DIRECTORIES tag to YES will show the directory hierarchy
    # in the documentation. The default is NO.

390 SHOW_DIRECTORIES        = NO

    # The FILE_VERSION_FILTER tag can be used to specify a program or script that
    # doxygen should invoke to get the current version for each file (typically from↙
        ↳ the
    # version control system). Doxygen will invoke the program by executing (via
395 # popen()) the command <command> <input-file>, where <command> is the value of
    # the FILE_VERSION_FILTER tag, and <input-file> is the name of an input file
    # provided by doxygen. Whatever the program writes to standard output
    # is used as the file version. See the manual for examples.

400 FILE_VERSION_FILTER    =

    #---------------------------------------------------------------------------
    # configuration options related to warning and progress messages
    #---------------------------------------------------------------------------
405
    # The QUIET tag can be used to turn on/off the messages that are generated
    # by doxygen. Possible values are YES and NO. If left blank NO is used.

    QUIET                  = NO
410
    # The WARNINGS tag can be used to turn on/off the warning messages that are
    # generated by doxygen. Possible values are YES and NO. If left blank
    # NO is used.

415 WARNINGS               = YES

    # If WARN_IF_UNDOCUMENTED is set to YES, then doxygen will generate warnings
    # for undocumented members. If EXTRACT_ALL is set to YES then this flag will
    # automatically be disabled.
420
    WARN_IF_UNDOCUMENTED   = YES

    # If WARN_IF_DOC_ERROR is set to YES, doxygen will generate warnings for
    # potential errors in the documentation, such as not documenting some
425 # parameters in a documented function, or documenting parameters that
    # don't exist or using markup commands wrongly.

    WARN_IF_DOC_ERROR      = YES

430 # This WARN_NO_PARAMDOC option can be abled to get warnings for
    # functions that are documented, but have no documentation for their parameters
    # or return value. If set to NO (the default) doxygen will only warn about
    # wrong or incomplete parameter documentation, but not about the absence of
    # documentation.
435
```

```
WARN_NO_PARAMDOC       = NO

# The WARN_FORMAT tag determines the format of the warning messages that
# doxygen can produce. The string should contain the $file, $line, and $text
# tags, which will be replaced by the file and line number from which the
# warning originated and the warning text. Optionally the format may contain
# $version, which will be replaced by the version of the file (if it could
# be obtained via FILE_VERSION_FILTER)

WARN_FORMAT            = "$file:$line: $text"

# The WARN_LOGFILE tag can be used to specify a file to which warning
# and error messages should be written. If left blank the output is written
# to stderr.

WARN_LOGFILE           =

#---------------------------------------------------------------------------
# configuration options related to the input files
#---------------------------------------------------------------------------

# The INPUT tag can be used to specify the files and/or directories that contain
# documented source files. You may enter file names like "myfile.cpp" or
# directories like "/usr/src/myproject". Separate the files or directories
# with spaces.

INPUT                  = "lua.c"

# If the value of the INPUT tag contains directories, you can use the
# FILE_PATTERNS tag to specify one or more wildcard pattern (like *.cpp
# and *.h) to filter out the source-files in the directories. If left
# blank the following patterns are tested:
# *.c *.cc *.cxx *.cpp *.c++ *.java *.ii *.ixx *.ipp *.i++ *.inl *.h *.hh *.hxx
# *.hpp *.h++ *.idl *.odl *.cs *.php *.php3 *.inc *.m *.mm *.py

FILE_PATTERNS          =

# The RECURSIVE tag can be used to turn specify whether or not subdirectories
# should be searched for input files as well. Possible values are YES and NO.
# If left blank NO is used.

RECURSIVE              = NO

# The EXCLUDE tag can be used to specify files and/or directories that should
# excluded from the INPUT source files. This way you can easily exclude a
# subdirectory from a directory tree whose root is specified with the INPUT tag.

EXCLUDE                =

# The EXCLUDE_SYMLINKS tag can be used select whether or not files or
# directories that are symbolic links (a Unix filesystem feature) are excluded
# from the input.

EXCLUDE_SYMLINKS       = NO

# If the value of the INPUT tag contains directories, you can use the
# EXCLUDE_PATTERNS tag to specify one or more wildcard patterns to exclude
```

```
# certain files from those directories. Note that the wildcards are matched
# against the file with absolute path, so to exclude all test directories
# for example use the pattern */test/*

EXCLUDE_PATTERNS       =

# The EXAMPLE_PATH tag can be used to specify one or more files or
# directories that contain example code fragments that are included (see
# the \include command).

EXAMPLE_PATH           =

# If the value of the EXAMPLE_PATH tag contains directories, you can use the
# EXAMPLE_PATTERNS tag to specify one or more wildcard pattern (like *.cpp
# and *.h) to filter out the source-files in the directories. If left
# blank all files are included.

EXAMPLE_PATTERNS       =

# If the EXAMPLE_RECURSIVE tag is set to YES then subdirectories will be
# searched for input files to be used with the \include or \dontinclude
# commands irrespective of the value of the RECURSIVE tag.
# Possible values are YES and NO. If left blank NO is used.

EXAMPLE_RECURSIVE      = NO

# The IMAGE_PATH tag can be used to specify one or more files or
# directories that contain image that are included in the documentation (see
# the \image command).

IMAGE_PATH             =

# The INPUT_FILTER tag can be used to specify a program that doxygen should
# invoke to filter for each input file. Doxygen will invoke the filter program
# by executing (via popen()) the command <filter> <input-file>, where <filter>
# is the value of the INPUT_FILTER tag, and <input-file> is the name of an
# input file. Doxygen will then use the output that the filter program writes
# to standard output.  If FILTER_PATTERNS is specified, this tag will be
# ignored.

INPUT_FILTER           =

# The FILTER_PATTERNS tag can be used to specify filters on a per file pattern
# basis.  Doxygen will compare the file name with each pattern and apply the
# filter if there is a match.  The filters are a list of the form:
# pattern=filter (like *.cpp=my_cpp_filter). See INPUT_FILTER for further
# info on how filters are used. If FILTER_PATTERNS is empty, INPUT_FILTER
# is applied to all files.

FILTER_PATTERNS        =

# If the FILTER_SOURCE_FILES tag is set to YES, the input filter (if set using
# INPUT_FILTER) will be used to filter the input files when producing source
# files to browse (i.e. when SOURCE_BROWSER is set to YES).

FILTER_SOURCE_FILES    = NO
```

```
#---------------------------------------------------------------------------
# configuration options related to source browsing
#---------------------------------------------------------------------------

# If the SOURCE_BROWSER tag is set to YES then a list of source files will
# be generated. Documented entities will be cross-referenced with these sources.
# Note: To get rid of all source code in the generated output, make sure also
# VERBATIM_HEADERS is set to NO.

SOURCE_BROWSER         = YES

# Setting the INLINE_SOURCES tag to YES will include the body
# of functions and classes directly in the documentation.

INLINE_SOURCES         = NO

# Setting the STRIP_CODE_COMMENTS tag to YES (the default) will instruct
# doxygen to hide any special comment blocks from generated source code
# fragments. Normal C and C++ comments will always remain visible.

STRIP_CODE_COMMENTS    = YES

# If the REFERENCED_BY_RELATION tag is set to YES (the default)
# then for each documented function all documented
# functions referencing it will be listed.

REFERENCED_BY_RELATION = YES

# If the REFERENCES_RELATION tag is set to YES (the default)
# then for each documented function all documented entities
# called/used by that function will be listed.

REFERENCES_RELATION    = YES

# If the REFERENCES_LINK_SOURCE tag is set to YES (the default)
# and SOURCE_BROWSER tag is set to YES, then the hyperlinks from
# functions in REFERENCES_RELATION and REFERENCED_BY_RELATION lists will
# link to the source code.  Otherwise they will link to the documentstion.

REFERENCES_LINK_SOURCE = YES

# If the USE_HTAGS tag is set to YES then the references to source code
# will point to the HTML generated by the htags(1) tool instead of doxygen
# built-in source browser. The htags tool is part of GNU's global source
# tagging system (see http://www.gnu.org/software/global/global.html). You
# will need version 4.8.6 or higher.

USE_HTAGS              = NO

# If the VERBATIM_HEADERS tag is set to YES (the default) then Doxygen
# will generate a verbatim copy of the header file for each class for
# which an include is specified. Set to NO to disable this.

VERBATIM_HEADERS       = YES

#---------------------------------------------------------------------------
# configuration options related to the alphabetical class index
```

```
#---------------------------------------------------------------------------

# If the ALPHABETICAL_INDEX tag is set to YES, an alphabetical index
# of all compounds will be generated. Enable this if the project
# contains a lot of classes, structs, unions or interfaces.

ALPHABETICAL_INDEX     = YES

# If the alphabetical index is enabled (see ALPHABETICAL_INDEX) then
# the COLS_IN_ALPHA_INDEX tag can be used to specify the number of columns
# in which this list will be split (can be a number in the range [1..20])

COLS_IN_ALPHA_INDEX    = 5

# In case all classes in a project start with a common prefix, all
# classes will be put under the same header in the alphabetical index.
# The IGNORE_PREFIX tag can be used to specify one or more prefixes that
# should be ignored while generating the index headers.

IGNORE_PREFIX          =

#---------------------------------------------------------------------------
# configuration options related to the HTML output
#---------------------------------------------------------------------------

# If the GENERATE_HTML tag is set to YES (the default) Doxygen will
# generate HTML output.

GENERATE_HTML          = YES

# The HTML_OUTPUT tag is used to specify where the HTML docs will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank 'html' will be used as the default path.

HTML_OUTPUT            = html

# The HTML_FILE_EXTENSION tag can be used to specify the file extension for
# each generated HTML page (for example: .htm,.php,.asp). If it is left blank
# doxygen will generate files with .html extension.

HTML_FILE_EXTENSION    = .html

# The HTML_HEADER tag can be used to specify a personal HTML header for
# each generated HTML page. If it is left blank doxygen will generate a
# standard header.

HTML_HEADER            =

# The HTML_FOOTER tag can be used to specify a personal HTML footer for
# each generated HTML page. If it is left blank doxygen will generate a
# standard footer.

HTML_FOOTER            =

# The HTML_STYLESHEET tag can be used to specify a user-defined cascading
# style sheet that is used by each HTML page. It can be used to
# fine-tune the look of the HTML output. If the tag is left blank doxygen
```

```
# will generate a default style sheet. Note that doxygen will try to copy
# the style sheet file to the HTML output directory, so don't put your own
# stylesheet in the HTML output directory as well, or it will be erased!

HTML_STYLESHEET        =

# If the HTML_ALIGN_MEMBERS tag is set to YES, the members of classes,
# files or namespaces will be aligned in HTML using tables. If set to
# NO a bullet list will be used.

HTML_ALIGN_MEMBERS     = NO

# If the GENERATE_HTMLHELP tag is set to YES, additional index files
# will be generated that can be used as input for tools like the
# Microsoft HTML help workshop to generate a compressed HTML help file (.chm)
# of the generated HTML documentation.

GENERATE_HTMLHELP      = NO

# If the GENERATE_HTMLHELP tag is set to YES, the CHM_FILE tag can
# be used to specify the file name of the resulting .chm file. You
# can add a path in front of the file if the result should not be
# written to the html output directory.

CHM_FILE               =

# If the GENERATE_HTMLHELP tag is set to YES, the HHC_LOCATION tag can
# be used to specify the location (absolute path including file name) of
# the HTML help compiler (hhc.exe). If non-empty doxygen will try to run
# the HTML help compiler on the generated index.hhp.

HHC_LOCATION           =

# If the GENERATE_HTMLHELP tag is set to YES, the GENERATE_CHI flag
# controls if a separate .chi index file is generated (YES) or that
# it should be included in the master .chm file (NO).

GENERATE_CHI           = NO

# If the GENERATE_HTMLHELP tag is set to YES, the BINARY_TOC flag
# controls whether a binary table of contents is generated (YES) or a
# normal table of contents (NO) in the .chm file.

BINARY_TOC             = NO

# The TOC_EXPAND flag can be set to YES to add extra items for group members
# to the contents of the HTML help documentation and to the tree view.

TOC_EXPAND             = NO

# The DISABLE_INDEX tag can be used to turn on/off the condensed index at
# top of each HTML page. The value NO (the default) enables the index and
# the value YES disables it.

DISABLE_INDEX          = NO

# This tag can be used to set the number of enum values (range [1..20])
```

```
# that doxygen will group on one line in the generated HTML documentation.

ENUM_VALUES_PER_LINE   = 4

# If the GENERATE_TREEVIEW tag is set to YES, a side panel will be
# generated containing a tree-like index structure (just like the one that
# is generated for HTML Help). For this to work a browser that supports
# JavaScript, DHTML, CSS and frames is required (for instance Mozilla 1.0+,
# Netscape 6.0+, Internet explorer 5.0+, or Konqueror). Windows users are
# probably better off using the HTML help feature.

GENERATE_TREEVIEW      = NO

# If the treeview is enabled (see GENERATE_TREEVIEW) then this tag can be
# used to set the initial width (in pixels) of the frame in which the tree
# is shown.

TREEVIEW_WIDTH         = 250

#---------------------------------------------------------------------------
# configuration options related to the LaTeX output
#---------------------------------------------------------------------------

# If the GENERATE_LATEX tag is set to YES (the default) Doxygen will
# generate Latex output.

GENERATE_LATEX         = NO

# The LATEX_OUTPUT tag is used to specify where the LaTeX docs will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `latex' will be used as the default path.

LATEX_OUTPUT           = latex

# The LATEX_CMD_NAME tag can be used to specify the LaTeX command name to be
# invoked. If left blank `latex' will be used as the default command name.

LATEX_CMD_NAME         = latex

# The MAKEINDEX_CMD_NAME tag can be used to specify the command name to
# generate index for LaTeX. If left blank `makeindex' will be used as the
# default command name.

MAKEINDEX_CMD_NAME     = makeindex

# If the COMPACT_LATEX tag is set to YES Doxygen generates more compact
# LaTeX documents. This may be useful for small projects and may help to
# save some trees in general.

COMPACT_LATEX          = NO

# The PAPER_TYPE tag can be used to set the paper type that is used
# by the printer. Possible values are: a4, a4wide, letter, legal and
# executive. If left blank a4wide will be used.

PAPER_TYPE             = a4wide
```

```
# The EXTRA_PACKAGES tag can be to specify one or more names of LaTeX
# packages that should be included in the LaTeX output.

EXTRA_PACKAGES         =

# The LATEX_HEADER tag can be used to specify a personal LaTeX header for
# the generated latex document. The header should contain everything until
# the first chapter. If it is left blank doxygen will generate a
# standard header. Notice: only use this tag if you know what you are doing!

LATEX_HEADER           =

# If the PDF_HYPERLINKS tag is set to YES, the LaTeX that is generated
# is prepared for conversion to pdf (using ps2pdf). The pdf file will
# contain links (just like the HTML output) instead of page references
# This makes the output suitable for online browsing using a pdf viewer.

PDF_HYPERLINKS         = NO

# If the USE_PDFLATEX tag is set to YES, pdflatex will be used instead of
# plain latex in the generated Makefile. Set this option to YES to get a
# higher quality PDF documentation.

USE_PDFLATEX           = NO

# If the LATEX_BATCHMODE tag is set to YES, doxygen will add the \\batchmode.
# command to the generated LaTeX files. This will instruct LaTeX to keep
# running if errors occur, instead of asking the user for help.
# This option is also used when generating formulas in HTML.

LATEX_BATCHMODE        = NO

# If LATEX_HIDE_INDICES is set to YES then doxygen will not
# include the index chapters (such as File Index, Compound Index, etc.)
# in the output.

LATEX_HIDE_INDICES     = NO

#---------------------------------------------------------------------------
# configuration options related to the RTF output
#---------------------------------------------------------------------------

# If the GENERATE_RTF tag is set to YES Doxygen will generate RTF output
# The RTF output is optimized for Word 97 and may not look very pretty with
# other RTF readers or editors.

GENERATE_RTF           = NO

# The RTF_OUTPUT tag is used to specify where the RTF docs will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank `rtf' will be used as the default path.

RTF_OUTPUT             = rtf

# If the COMPACT_RTF tag is set to YES Doxygen generates more compact
# RTF documents. This may be useful for small projects and may help to
# save some trees in general.
```

```
835
COMPACT_RTF            = NO

# If the RTF_HYPERLINKS tag is set to YES, the RTF that is generated
# will contain hyperlink fields. The RTF file will
840 # contain links (just like the HTML output) instead of page references.
# This makes the output suitable for online browsing using WORD or other
# programs which support those fields.
# Note: wordpad (write) and others do not support links.

845 RTF_HYPERLINKS       = NO

# Load stylesheet definitions from file. Syntax is similar to doxygen's
# config file, i.e. a series of assignments. You only have to provide
# replacements, missing definitions are set to their default value.
850
RTF_STYLESHEET_FILE    =

# Set optional variables used in the generation of an rtf document.
# Syntax is similar to doxygen's config file.
855
RTF_EXTENSIONS_FILE    =

#---------------------------------------------------------------------------
# configuration options related to the man page output
860 #---------------------------------------------------------------------------

# If the GENERATE_MAN tag is set to YES (the default) Doxygen will
# generate man pages

865 GENERATE_MAN        = NO

# The MAN_OUTPUT tag is used to specify where the man pages will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank 'man' will be used as the default path.
870
MAN_OUTPUT             = man

# The MAN_EXTENSION tag determines the extension that is added to
# the generated man pages (default is the subroutine's section .3)
875
MAN_EXTENSION          = .3

# If the MAN_LINKS tag is set to YES and Doxygen generates man output,
# then it will generate one additional man file for each entity
880 # documented in the real man page(s). These additional files
# only source the real man page, but without them the man command
# would be unable to find the correct page. The default is NO.

MAN_LINKS              = NO
885
#---------------------------------------------------------------------------
# configuration options related to the XML output
#---------------------------------------------------------------------------

890 # If the GENERATE_XML tag is set to YES Doxygen will
# generate an XML file that captures the structure of
```

```
# the code including all documentation.

GENERATE_XML           = NO

# The XML_OUTPUT tag is used to specify where the XML pages will be put.
# If a relative path is entered the value of OUTPUT_DIRECTORY will be
# put in front of it. If left blank 'xml' will be used as the default path.

XML_OUTPUT             = xml

# The XML_SCHEMA tag can be used to specify an XML schema,
# which can be used by a validating XML parser to check the
# syntax of the XML files.

XML_SCHEMA             =

# The XML_DTD tag can be used to specify an XML DTD,
# which can be used by a validating XML parser to check the
# syntax of the XML files.

XML_DTD                =

# If the XML_PROGRAMLISTING tag is set to YES Doxygen will
# dump the program listings (including syntax highlighting
# and cross-referencing information) to the XML output. Note that
# enabling this will significantly increase the size of the XML output.

XML_PROGRAMLISTING     = YES

#---------------------------------------------------------------------------
# configuration options for the AutoGen Definitions output
#---------------------------------------------------------------------------

# If the GENERATE_AUTOGEN_DEF tag is set to YES Doxygen will
# generate an AutoGen Definitions (see autogen.sf.net) file
# that captures the structure of the code including all
# documentation. Note that this feature is still experimental
# and incomplete at the moment.

GENERATE_AUTOGEN_DEF   = NO

#---------------------------------------------------------------------------
# configuration options related to the Perl module output
#---------------------------------------------------------------------------

# If the GENERATE_PERLMOD tag is set to YES Doxygen will
# generate a Perl module file that captures the structure of
# the code including all documentation. Note that this
# feature is still experimental and incomplete at the
# moment.

GENERATE_PERLMOD       = NO

# If the PERLMOD_LATEX tag is set to YES Doxygen will generate
# the necessary Makefile rules, Perl scripts and LaTeX code to be able
# to generate PDF and DVI output from the Perl module output.
```

```
PERLMOD_LATEX          = NO

# If the PERLMOD_PRETTY tag is set to YES the Perl module output will be
# nicely formatted so it can be parsed by a human reader.  This is useful
# if you want to understand what is going on.  On the other hand, if this
# tag is set to NO the size of the Perl module output will be much smaller
# and Perl will parse it just the same.

PERLMOD_PRETTY         = YES

# The names of the make variables in the generated doxyrules.make file
# are prefixed with the string contained in PERLMOD_MAKEVAR_PREFIX.
# This is useful so different doxyrules.make files included by the same
# Makefile don't overwrite each other's variables.

PERLMOD_MAKEVAR_PREFIX =

#---------------------------------------------------------------------------
# Configuration options related to the preprocessor
#---------------------------------------------------------------------------

# If the ENABLE_PREPROCESSING tag is set to YES (the default) Doxygen will
# evaluate all C-preprocessor directives found in the sources and include
# files.

ENABLE_PREPROCESSING   = YES

# If the MACRO_EXPANSION tag is set to YES Doxygen will expand all macro
# names in the source code. If set to NO (the default) only conditional
# compilation will be performed. Macro expansion can be done in a controlled
# way by setting EXPAND_ONLY_PREDEF to YES.

MACRO_EXPANSION        = NO

# If the EXPAND_ONLY_PREDEF and MACRO_EXPANSION tags are both set to YES
# then the macro expansion is limited to the macros specified with the
# PREDEFINED and EXPAND_AS_DEFINED tags.

EXPAND_ONLY_PREDEF     = NO

# If the SEARCH_INCLUDES tag is set to YES (the default) the includes files
# in the INCLUDE_PATH (see below) will be search if a #include is found.

SEARCH_INCLUDES        = YES

# The INCLUDE_PATH tag can be used to specify one or more directories that
# contain include files that are not input files but should be processed by
# the preprocessor.

INCLUDE_PATH           =

# You can use the INCLUDE_FILE_PATTERNS tag to specify one or more wildcard
# patterns (like *.h and *.hpp) to filter out the header-files in the
# directories. If left blank, the patterns specified with FILE_PATTERNS will
# be used.

INCLUDE_FILE_PATTERNS  =
```

```
# The PREDEFINED tag can be used to specify one or more macro names that
# are defined before the preprocessor is started (similar to the -D option of
# gcc). The argument of the tag is a list of macros of the form: name
# or name=definition (no spaces). If the definition and the = are
# omitted =1 is assumed. To prevent a macro definition from being
# undefined via #undef or recursively expanded use the := operator
# instead of the = operator.

PREDEFINED             =

# If the MACRO_EXPANSION and EXPAND_ONLY_PREDEF tags are set to YES then
# this tag can be used to specify a list of macro names that should be expanded.
# The macro definition that is found in the sources will be used.
# Use the PREDEFINED tag if you want to use a different macro definition.

EXPAND_AS_DEFINED      =

# If the SKIP_FUNCTION_MACROS tag is set to YES (the default) then
# doxygen's preprocessor will remove all function-like macros that are alone
# on a line, have an all uppercase name, and do not end with a semicolon. Such
# function macros are typically used for boiler-plate code, and will confuse
# the parser if not removed.

SKIP_FUNCTION_MACROS   = YES

#---------------------------------------------------------------------------
# Configuration::additions related to external references
#---------------------------------------------------------------------------

# The TAGFILES option can be used to specify one or more tagfiles.
# Optionally an initial location of the external documentation
# can be added for each tagfile. The format of a tag file without
# this location is as follows:
#   TAGFILES = file1 file2 ...
# Adding location for the tag files is done as follows:
#   TAGFILES = file1=loc1 "file2 = loc2" ...
# where "loc1" and "loc2" can be relative or absolute paths or
# URLs. If a location is present for each tag, the installdox tool
# does not have to be run to correct the links.
# Note that each tag file must have a unique name
# (where the name does NOT include the path)
# If a tag file is not located in the directory in which doxygen
# is run, you must also specify the path to the tagfile here.

TAGFILES               =

# When a file name is specified after GENERATE_TAGFILE, doxygen will create
# a tag file that is based on the input files it reads.

GENERATE_TAGFILE       =

# If the ALLEXTERNALS tag is set to YES all external classes will be listed
# in the class index. If set to NO only the inherited external classes
# will be listed.

ALLEXTERNALS           = NO
```

```
# If the EXTERNAL_GROUPS tag is set to YES all external groups will be listed
# in the modules index. If set to NO, only the current project's groups will
# be listed.

EXTERNAL_GROUPS        = YES

# The PERL_PATH should be the absolute path and name of the perl script
# interpreter (i.e. the result of 'which perl').

PERL_PATH              = /usr/bin/perl

#---------------------------------------------------------------------------
# Configuration options related to the dot tool
#---------------------------------------------------------------------------

# If the CLASS_DIAGRAMS tag is set to YES (the default) Doxygen will
# generate a inheritance diagram (in HTML, RTF and LaTeX) for classes with base
# or super classes. Setting the tag to NO turns the diagrams off. Note that
# this option is superseded by the HAVE_DOT option below. This is only a
# fallback. It is recommended to install and use dot, since it yields more
# powerful graphs.

CLASS_DIAGRAMS         = YES

# If set to YES, the inheritance and collaboration graphs will hide
# inheritance and usage relations if the target is undocumented
# or is not a class.

HIDE_UNDOC_RELATIONS   = YES

# If you set the HAVE_DOT tag to YES then doxygen will assume the dot tool is
# available from the path. This tool is part of Graphviz, a graph visualization
# toolkit from AT&T and Lucent Bell Labs. The other options in this section
# have no effect if this option is set to NO (the default)

HAVE_DOT               = YES

# If the CLASS_GRAPH and HAVE_DOT tags are set to YES then doxygen
# will generate a graph for each documented class showing the direct and
# indirect inheritance relations. Setting this tag to YES will force the
# the CLASS_DIAGRAMS tag to NO.

CLASS_GRAPH            = YES

# If the COLLABORATION_GRAPH and HAVE_DOT tags are set to YES then doxygen
# will generate a graph for each documented class showing the direct and
# indirect implementation dependencies (inheritance, containment, and
# class references variables) of the class with other documented classes.

COLLABORATION_GRAPH    = YES

# If the GROUP_GRAPHS and HAVE_DOT tags are set to YES then doxygen
# will generate a graph for groups, showing the direct groups dependencies

GROUP_GRAPHS           = YES
```

```
# If the UML_LOOK tag is set to YES doxygen will generate inheritance and
# collaboration diagrams in a style similar to the OMG's Unified Modeling
# Language.

UML_LOOK               = NO

# If set to YES, the inheritance and collaboration graphs will show the
# relations between templates and their instances.

TEMPLATE_RELATIONS     = NO

# If the ENABLE_PREPROCESSING, SEARCH_INCLUDES, INCLUDE_GRAPH, and HAVE_DOT
# tags are set to YES then doxygen will generate a graph for each documented
# file showing the direct and indirect include dependencies of the file with
# other documented files.

INCLUDE_GRAPH          = YES

# If the ENABLE_PREPROCESSING, SEARCH_INCLUDES, INCLUDED_BY_GRAPH, and
# HAVE_DOT tags are set to YES then doxygen will generate a graph for each
# documented header file showing the documented files that directly or
# indirectly include this file.

INCLUDED_BY_GRAPH      = YES

# If the CALL_GRAPH and HAVE_DOT tags are set to YES then doxygen will
# generate a call dependency graph for every global function or class method.
# Note that enabling this option will significantly increase the time of a run.
# So in most cases it will be better to enable call graphs for selected
# functions only using the \callgraph command.

CALL_GRAPH             = YES

# If the CALLER_GRAPH and HAVE_DOT tags are set to YES then doxygen will
# generate a caller dependency graph for every global function or class method.
# Note that enabling this option will significantly increase the time of a run.
# So in most cases it will be better to enable caller graphs for selected
# functions only using the \callergraph command.

CALLER_GRAPH           = YES

# If the GRAPHICAL_HIERARCHY and HAVE_DOT tags are set to YES then doxygen
# will graphical hierarchy of all classes instead of a textual one.

GRAPHICAL_HIERARCHY    = YES

# If the DIRECTORY_GRAPH, SHOW_DIRECTORIES and HAVE_DOT tags are set to YES
# then doxygen will show the dependencies a directory has on other directories
# in a graphical way. The dependency relations are determined by the #include
# relations between the files in the directories.

DIRECTORY_GRAPH        = YES

# The DOT_IMAGE_FORMAT tag can be used to set the image format of the images
# generated by dot. Possible values are png, jpg, or gif
# If left blank png will be used.
```

```
DOT_IMAGE_FORMAT       = png

# The tag DOT_PATH can be used to specify the path where the dot tool can be
# found. If left blank, it is assumed the dot tool can be found in the path.

DOT_PATH               =

# The DOTFILE_DIRS tag can be used to specify one or more directories that
# contain dot files that are included in the documentation (see the
# \dotfile command).

DOTFILE_DIRS           =

# The MAX_DOT_GRAPH_WIDTH tag can be used to set the maximum allowed width
# (in pixels) of the graphs generated by dot. If a graph becomes larger than
# this value, doxygen will try to truncate the graph, so that it fits within
# the specified constraint. Beware that most browsers cannot cope with very
# large images.

MAX_DOT_GRAPH_WIDTH    = 1024

# The MAX_DOT_GRAPH_HEIGHT tag can be used to set the maximum allows height
# (in pixels) of the graphs generated by dot. If a graph becomes larger than
# this value, doxygen will try to truncate the graph, so that it fits within
# the specified constraint. Beware that most browsers cannot cope with very
# large images.

MAX_DOT_GRAPH_HEIGHT   = 1024

# The MAX_DOT_GRAPH_DEPTH tag can be used to set the maximum depth of the
# graphs generated by dot. A depth value of 3 means that only nodes reachable
# from the root by following a path via at most 3 edges will be shown. Nodes
# that lay further from the root node will be omitted. Note that setting this
# option to 1 or 2 may greatly reduce the computation time needed for large
# code bases. Also note that a graph may be further truncated if the graph's
# image dimensions are not sufficient to fit the graph (see MAX_DOT_GRAPH_WIDTH
# and MAX_DOT_GRAPH_HEIGHT). If 0 is used for the depth value (the default),
# the graph is not depth-constrained.

MAX_DOT_GRAPH_DEPTH    = 0

# Set the DOT_TRANSPARENT tag to YES to generate images with a transparent
# background. This is disabled by default, which results in a white background.
# Warning: Depending on the platform used, enabling this option may lead to
# badly anti-aliased labels on the edges of a graph (i.e. they become hard to
# read).

DOT_TRANSPARENT        = NO

# Set the DOT_MULTI_TARGETS tag to YES allow dot to generate multiple output
# files in one run (i.e. multiple -o and -T options on the command line). This
# makes dot run faster, but since only newer versions of dot (>1.8.10)
# support this, this feature is disabled by default.

DOT_MULTI_TARGETS      = NO

# If the GENERATE_LEGEND tag is set to YES (the default) Doxygen will
```

```
# generate a legend page explaining the meaning of the various boxes and
# arrows in the dot generated graphs.

GENERATE_LEGEND        = YES

# If the DOT_CLEANUP tag is set to YES (the default) Doxygen will
# remove the intermediate dot files that are used to generate
# the various graphs.

DOT_CLEANUP            = YES

#---------------------------------------------------------------------------
# Configuration::additions related to the search engine
#---------------------------------------------------------------------------

# The SEARCHENGINE tag specifies whether or not a search engine should be
# used. If set to NO the values of all tags below this one will be ignored.

SEARCHENGINE           = NO
```

75 src/hello-help.pd

```
#N canvas 43 60 476 189 10;
#X obj 34 62 hello;
#X obj 94 62 hello;
#X obj 154 62 hello;
#X obj 214 62 hello;
#X text 33 95 See also:;
#X obj 49 125 lua;
#X obj 93 125 luax hello;
#X text 32 14 [lua] registers a loader that allows [hello] to be defined
by a source file "hello.pd_lua".;
```

76 src/hello.lua

```
pd.post("Hello, you!")
```

77 src/hello.pd_lua

```
pd.post("Hello, universe!")

local Hello = pd.Class:new():register("hello")

function Hello:initialize(name, atoms)
  pd.post("Hello, world!")
  return true
end

function Hello:finalize()
  pd.post("Bye bye, world!")
end
```

78 src/hello.pd_luax

```
return function(self, sel, atoms)
  pd.post("Hello, world!")
  return true
end
```

79 src/lua.c

```
/** @file lua.c
 *  @brief pdlua -- a Lua embedding for Pd.
 *  @author Claude Heiland-Allen <claude@mathr.co.uk>
 *  @date 2013
 *  @version 0.8
 *
 * Copyright (C) 2007,2008,2013 Claude Heiland-Allen <claude@mathr.co.uk>
 * Copyright (C) 2012 Martin Peach martin.peach@sympatico.ca
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, ↙
    ↳ USA.
 *
 */

/* various C stuff, mainly for reading files */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

/* we use Lua */
#include <lua.h>
#include <lauxlib.h>
#include <lualib.h>

/* we use Pd */
#include <pd/m_pd.h>
#include <pd/m_imp.h>
#include <pd/s_stuff.h>


/* BAD: support for Pd < 0.41 */

#if PD_MAJOR_VERSION == 0
# if PD_MINOR_VERSION >= 41
```

```
50  #  define PDLUA_PD41
    /* use new garray support that is 64-bit safe */
    #  define PDLUA_ARRAYGRAB garray_getfloatwords
    #  define PDLUA_ARRAYTYPE t_word
    #  define PDLUA_ARRAYELEM(arr,idx) ((arr)[(idx)].w_float)
55  # elif PD_MINOR_VERSION >= 40
    #  define PDLUA_PD40
    /* use old garray support, not 64-bit safe */
    #  define PDLUA_ARRAYGRAB garray_getfloatarray
    #  define PDLUA_ARRAYTYPE t_float
60  #  define PDLUA_ARRAYELEM(arr,idx) ((arr)[(idx)])
    # elif PD_MINOR_VERSION >= 39
    #  define PDLUA_PD39
    /* use old garray support, not 64-bit safe */
    #  define PDLUA_ARRAYGRAB garray_getfloatarray
65  #  define PDLUA_ARRAYTYPE t_float
    #  define PDLUA_ARRAYELEM(arr,idx) ((arr)[(idx)])
    # else
    #  error "Pd version is too old, please upgrade"
    # endif
70  #else
    # error "Pd version is too new, please file a bug report"
    #endif

    /* BAD: end of bad section */
75
    /** Global Lua interpreter state, needed in the constructor. */
    static lua_State *L;

    /** State for the Lua file reader. */
80  typedef struct pdlua_readerdata {
      int fd; /**< File descriptor to read from. */
      char buffer[MAXPDSTRING]; /**< Buffer to read into. */
    } t_pdlua_readerdata;

85  /** Lua file reader callback. */
    static const char *pdlua_reader(
      lua_State *L, /**< Lua interpreter state. */
      void *rr, /**< Lua file reader state. */
      size_t *size /**< How much data we have read. */
90  ) {
      t_pdlua_readerdata *r = rr;
      ssize_t s;
      (void) L;
      s = read(r->fd, r->buffer, MAXPDSTRING-2);
95    if (s <= 0) {
        *size = 0;
        return NULL;
      } else {
        *size = s;
100     return r->buffer;
      }
    }

    /* declare some stuff in advance */
105 struct pdlua_proxyinlet;
    struct pdlua_proxyreceive;
```

```
struct pdlua_proxyclock;

/** Pd object data. */
typedef struct pdlua {
  t_object pd; /**< We are a Pd object. */
  unsigned int inlets; /**< Number of inlets. */
  struct pdlua_proxyinlet *in; /**< The inlets themselves. */
  unsigned int outlets; /**< Number of outlets. */
  t_outlet **out; /**< The outlets themselves. */
  t_canvas *canvas; /**< The canvas that the object was created on. */
} t_pdlua;

/* more forward declarations */
static void pdlua_dispatch(
  t_pdlua *o, unsigned int inlet, t_symbol *s, int argc, t_atom *argv
);
static void pdlua_receivedispatch(
  struct pdlua_proxyreceive *r, t_symbol *s, int argc, t_atom *argv
);
static void pdlua_clockdispatch(
  struct pdlua_proxyclock *clock
);


/** Proxy inlet class pointer. */
static t_class *pdlua_proxyinlet_class;

/** Proxy inlet object data. */
typedef struct pdlua_proxyinlet {
  t_pd pd; /**< Minimal Pd object. */
  struct pdlua *owner; /**< The owning object to forward inlet messages to. */
  unsigned int id; /**< The number of this inlet. */
} t_pdlua_proxyinlet;

/** Proxy inlet 'anything' method. */
static void pdlua_proxyinlet_anything(
  t_pdlua_proxyinlet *p, /**< The proxy inlet that received the message. */
  t_symbol *s, /**< The message selector. */
  int argc, /**< The message length. */
  t_atom *argv /**< The atoms in the message. */
) {
  pdlua_dispatch(p->owner, p->id, s, argc, argv);
}

/** Proxy inlet initialization. */
static void pdlua_proxyinlet_init(
  t_pdlua_proxyinlet *p, /**< The proxy inlet to initialize. */
  struct pdlua *owner, /**< The owning object. */
  unsigned int id /**< The inlet number. */
) {
  p->pd = pdlua_proxyinlet_class;
  p->owner = owner;
  p->id = id;
}

/** Register the proxy inlet class with Pd. */
static void pdlua_proxyinlet_setup() {
```

```
  pdlua_proxyinlet_class = class_new(
    gensym("pdlua proxy inlet"),
    0,
    0,
    sizeof(t_pdlua_proxyinlet),
    0,
    0
  );
  class_addanything(
    pdlua_proxyinlet_class, pdlua_proxyinlet_anything
  );
}


/** Proxy receive class pointer. */
static t_class *pdlua_proxyreceive_class;

/** Proxy receive object data. */
typedef struct pdlua_proxyreceive {
  t_pd pd; /**< Minimal Pd object. */
  struct pdlua *owner; /**< The owning object to forward received messages to. */
  t_symbol *name; /**< The receive-symbol to bind to. */
} t_pdlua_proxyreceive;

/** Proxy receive 'anything' method. */
static void pdlua_proxyreceive_anything(
  t_pdlua_proxyreceive *r, /**< The proxy receive that received the message. */
  t_symbol *s, /**< The message selector. */
  int argc, /**< The message length. */
  t_atom *argv /**< The atoms in the message. */
) {
  pdlua_receivedispatch(r, s, argc, argv);
}

/** Proxy receive allocation and initialization. */
static t_pdlua_proxyreceive *pdlua_proxyreceive_new(
  struct pdlua *owner, /**< The owning object. */
  t_symbol *name /**< The symbol to bind to. */
) {
  t_pdlua_proxyreceive *r = malloc(sizeof(t_pdlua_proxyreceive));
  r->pd = pdlua_proxyreceive_class;
  r->owner = owner;
  r->name = name;
  pd_bind(&r->pd, r->name);
  return r;
}

/** Proxy receive cleanup and deallocation. */
static void pdlua_proxyreceive_free(
  t_pdlua_proxyreceive *r /**< The proxy receive to free. */
) {
  pd_unbind(&r->pd, r->name);
  r->pd = NULL;
  r->owner = NULL;
  r->name = NULL;
  free(r);
```

```
220 }

    /** Register the proxy receive class with Pd. */
    static void pdlua_proxyreceive_setup() {
      pdlua_proxyreceive_class = class_new(
225     gensym("pdlua proxy receive"),
        0,
        0,
        sizeof(t_pdlua_proxyreceive),
        0,
230     0
      );
      class_addanything(
        pdlua_proxyreceive_class, pdlua_proxyreceive_anything
      );
235 }


    /** Proxy clock class pointer. */
    static t_class *pdlua_proxyclock_class;
240
    /** Proxy clock object data. */
    typedef struct pdlua_proxyclock {
      t_pd pd; /**< Minimal Pd object. */
      struct pdlua *owner; /**< Object to forward messages to. */
245   t_clock *clock; /** Pd clock to use. */
    } t_pdlua_proxyclock;

    /** Proxy clock 'bang' method. */
    static void pdlua_proxyclock_bang(
250   t_pdlua_proxyclock *c /**< The proxy clock that received the message. */
    ) {
      pdlua_clockdispatch(c);
    }

255 /** Proxy clock allocation and initialization. */
    static t_pdlua_proxyclock *pdlua_proxyclock_new(
      struct pdlua *owner /**< The object to forward messages to. */
    ) {
      t_pdlua_proxyclock *c = malloc(sizeof(t_pdlua_proxyclock));
260   c->pd = pdlua_proxyclock_class;
      c->owner = owner;
      c->clock = clock_new(c, (t_method) pdlua_proxyclock_bang);
      return c;
    }
265
    /** Register the proxy clock class with Pd. */
    static void pdlua_proxyclock_setup() {
      pdlua_proxyclock_class = class_new(
        gensym("pdlua proxy clock"),
270     0,
        0,
        sizeof(t_pdlua_proxyclock),
        0,
        0
275   );
    }
```

```
/** Dump an array of atoms into a Lua table. */
static void pdlua_pushatomtable(
  int argc, /**< The number of atoms in the array. */
  t_atom *argv /**< The array of atoms. */
) {
  int i;
  lua_newtable(L);
  for (i = 0; i < argc; ++i) {
    lua_pushnumber(L, i+1);
    switch (argv[i].a_type) {
    case A_FLOAT:
      lua_pushnumber(L, argv[i].a_w.w_float);
      break;
    case A_SYMBOL:
      lua_pushstring(L, argv[i].a_w.w_symbol->s_name);
      break;
    case A_POINTER: /* FIXME: check experimentality */
      lua_pushlightuserdata(L, argv[i].a_w.w_gpointer);
      break;
    default:
      error("lua: zomg weasels!");
      lua_pushnil(L);
      break;
    }
    lua_settable(L, -3);
  }
}

/** Pd object constructor. */
static t_pdlua *pdlua_new(
  t_symbol *s, /**< The construction message selector. */
  int argc, /**< The construction message atom count. */
  t_atom *argv /**< The construction message atoms. */
) {
  lua_getglobal(L, "pd");
  lua_getfield(L, -1, "_constructor");
  lua_pushstring(L, s->s_name);
  pdlua_pushatomtable(argc, argv);
  if (lua_pcall(L, 2, 1, 0)) {
    error(
      "lua: error in constructor for `%s':\n%s",
      s->s_name, lua_tostring(L, -1)
    );
    lua_pop(L, 2); /* pop the error string and the global "pd" */
    return NULL;
  } else {
    t_pdlua *object = NULL;
    if (lua_islightuserdata(L, -1)) {
      object = lua_touserdata(L, -1);
      lua_pop(L, 2);/* pop the userdata and the global "pd" */
      return object;
    } else {
      lua_pop(L, 2);/* pop the userdata and the global "pd" */
      return NULL;
    }
  }
```

```
    }

/** Pd object destructor. */
static void pdlua_free(
  t_pdlua *o /**< The object to destruct. */
) {
  lua_getglobal(L, "pd");
  lua_getfield (L, -1, "_destructor");
  lua_pushlightuserdata(L, o);
  if (lua_pcall(L, 1, 0, 0)) {
    error(
      "lua: error in destructor:\n%s",
      lua_tostring(L, -1)
    );
    lua_pop(L, 1); /* pop the error string */
  }
  lua_pop(L, 1); /* pop the global "pd" */
  return;
}

#ifdef PDLUA_DEBUG
static void pdlua_stack_dump (lua_State *L)
{
    int i;
    int top = lua_gettop(L);

    for (i = 1; i <= top; i++)
    {  /* repeat for each level */
        int t = lua_type(L, i);
        switch (t)
        {
            case LUA_TSTRING:  /* strings */
                printf("`%s'", lua_tostring(L, i));
                break;
            case LUA_TBOOLEAN:  /* booleans */
                printf(lua_toboolean(L, i) ? "true" : "false");
                break;
            case LUA_TNUMBER:  /* numbers */
                printf("%g", lua_tonumber(L, i));
                break;
            default:  /* other values */
                printf("%s", lua_typename(L, t));
                break;
        }
        printf("  ");  /* put a separator */
    }
    printf("\n");  /* end the listing */
}
#endif

/** a handler for the open item in the right-click menu (mrpeach 20111025) */
/** Here we find the lua code for the object and open it in an editor */
static void pdlua_menu_open(t_pdlua *o)
{
    const char  *name;
    const char  *path;
    char        pathname[FILENAME_MAX];
```

```
    t_class     *class;

#ifdef PDLUA_DEBUG
    post("pdlua_menu_open");
#endif /* PDLUA_DEBUG */
    /** Get the scriptname of the object */
    lua_getglobal(L, "pd");
    lua_getfield(L, -1, "_whoami");
    lua_pushlightuserdata(L, o);
    if (lua_pcall(L, 1, 1, 0))
    {
        error("lua: error in whoami:\n%s", lua_tostring(L, -1));
        lua_pop(L, 2); /* pop the error string and the global "pd" */
        return;
    }
    name = luaL_checkstring(L, -1);
#ifdef PDLUA_DEBUG
    post("pdlua_menu_open: L is %p, name is %s", L, name);
#endif /* PDLUA_DEBUG */
    if (name)
    {
        if (name[strlen(name)-1] == 'x')
        {
            /* pdluax is a class, the particular file should loadable by name alone, we hope */
            sprintf(pathname, "%s", name);
            lua_pop(L, 2); /* pop name and the global "pd" */
        }
        else
        {
            lua_getglobal(L, "pd");
            lua_getfield(L, -1, "_get_class");
            lua_pushlightuserdata(L, o);
            if (lua_pcall(L, 1, 1, 0))
            {
                error("lua: error in get_class:\n%s", lua_tostring(L, -1));
                lua_pop(L, 4); /* pop the error string, global "pd", name, global "pd"*/
                return;
            }
            class = (t_class *)lua_touserdata(L, -1);
            path = class->c_externdir->s_name;
            sprintf(pathname, "%s/%s", path, name);
            lua_pop(L, 4); /* pop class, global "pd", name, global "pd"*/
        }
#if PD_MAJOR_VERSION==0 && PD_MINOR_VERSION<43
        post("Opening %s for editing", pathname);
#else
        logpost(NULL, 3, "Opening %s for editing", pathname);
#endif
          sys_vgui("::pd_menucommands::menu_openfile {%s}\n", pathname);
    }
}

/** Lua class registration. */
static int pdlua_class_new(
  lua_State *L /**< Lua interpreter state.
```

```
    * \par Inputs:
    * \li \c 1 Class name string.
    * \par Outputs:
    * \li \c 1 Pd class pointer.
    * */
) {
    const char *name = luaL_checkstring(L, 1);
    t_class *c = class_new(
      gensym((char *) name),
      (t_newmethod) pdlua_new,
      (t_method) pdlua_free,
      sizeof(t_pdlua),
      CLASS_NOINLET,
      A_GIMME,
      0
    );
    /* a class with a "menu-open" method will have the "Open" item highlighted in ↙
        ↳ the right-click menu */
    class_addmethod(c, (t_method)pdlua_menu_open, gensym("menu-open"), A_NULL);/* ↙
        ↳ (mrpeach 20111025) */
    lua_pushlightuserdata(L, c);
    return 1;
}

/** Lua object creation. */
static int pdlua_object_new(
    lua_State *L /**< Lua interpreter state.
    * \par Inputs:
    * \li \c 1 Pd class pointer.
    * \par Outputs:
    * \li \c 2 Pd object pointer.
    * */
) {
    if (lua_islightuserdata(L, 1)) {
      t_class *c = lua_touserdata(L, 1);
      if (c) {
        t_pdlua *o = (t_pdlua *) pd_new(c);
        if (o) {
          o->inlets = 0;
          o->in = NULL;
          o->outlets = 0;
          o->out = NULL;
          o->canvas = canvas_getcurrent();
          lua_pushlightuserdata(L, o);
          return 1;
        }
      }
    }
    return 0;
}

/** Lua object inlet creation. */
static int pdlua_object_createinlets(
    lua_State *L /**< Lua interpreter state.
    * \par Inputs:
    * \li \c 1 Pd object pointer.
    * \li \c 2 Number of inlets.
```

```
  * */
) {
  unsigned int i;
  if (lua_islightuserdata(L, 1)) {
    t_pdlua *o = lua_touserdata(L, 1);
    if (o) {
      o->inlets = luaL_checknumber(L, 2);
      o->in = malloc(o->inlets * sizeof(t_pdlua_proxyinlet));
      for (i = 0; i < o->inlets; ++i) {
        pdlua_proxyinlet_init(&o->in[i], o, i);
        inlet_new(&o->pd, &o->in[i].pd, 0, 0);
      }
    }
  }
  return 0;
}

/** Lua object outlet creation. */
static int pdlua_object_createoutlets(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd object pointer.
  * \li \c 2 Number of outlets.
  * */
) {
  unsigned int i;
  if (lua_islightuserdata(L, 1)) {
    t_pdlua *o = lua_touserdata(L, 1);
    if (o) {
      o->outlets = luaL_checknumber(L, 2);
      if (o->outlets > 0) {
        o->out = malloc(o->outlets * sizeof(t_outlet *));
        for (i = 0; i < o->outlets; ++i) {
          o->out[i] = outlet_new(&o->pd, 0);
        }
      } else {
        o->out = NULL;
      }
    }
  }
  return 0;
}

/** Lua object receive creation. */
static int pdlua_receive_new(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd object pointer.
  * \li \c 2 Receive name string.
  * \par Outputs:
  * \li \c 1 Pd receive pointer.
  * */
) {
  if (lua_islightuserdata(L, 1)) {
    t_pdlua *o = lua_touserdata(L, 1);
    if (o) {
      const char *name = luaL_checkstring(L, 2);
```

```
      if (name) {
        t_pdlua_proxyreceive *r =  pdlua_proxyreceive_new(o, gensym((char *) ↙
            ↳ name)); /* const cast */
560     lua_pushlightuserdata(L, r);
        return 1;
      }
    }
  }
565 return 0;
}

/** Lua object receive destruction. */
static int pdlua_receive_free(
570 lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd recieve pointer.
  * */
) {
575 if (lua_islightuserdata(L, 1)) {
    t_pdlua_proxyreceive *r = lua_touserdata(L, 1);
    if (r) {
      pdlua_proxyreceive_free(r);
    }
580 }
  return 0;
}

/** Lua object clock creation. */
585 static int pdlua_clock_new(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd object pointer.
  * \par Outputs:
590 * \li \c 1 Pd clock pointer.
  * */
) {
  if (lua_islightuserdata(L, 1)) {
    t_pdlua *o = lua_touserdata(L, 1);
595   if (o) {
      t_pdlua_proxyclock *c =  pdlua_proxyclock_new(o);
      lua_pushlightuserdata(L, c);
      return 1;
    }
600 }
  return 0;
}

/** Lua proxy clock delay. */
605 static int pdlua_clock_delay(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd clock pointer.
  * \li \c 2 Number of milliseconds to delay.
610 * */
) {
  if (lua_islightuserdata(L, 1)) {
    t_pdlua_proxyclock *c = lua_touserdata(L, 1);
```

```
    if (c) {
      double delaytime = luaL_checknumber(L, 2);
      clock_delay(c->clock, delaytime);
    }
  }
  return 0;
}

/** Lua proxy clock set. */
static int pdlua_clock_set(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd clock pointer.
  * \li \c 2 Number to set the clock.
  * */
) {
  if (lua_islightuserdata(L, 1)) {
    t_pdlua_proxyclock *c = lua_touserdata(L, 1);
    if (c) {
      double systime = luaL_checknumber(L, 2);
      clock_set(c->clock, systime);
    }
  }
  return 0;
}

/** Lua proxy clock unset. */
static int pdlua_clock_unset(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd clock pointer.
  * */
) {
  if (lua_islightuserdata(L, 1)) {
    t_pdlua_proxyclock *c = lua_touserdata(L, 1);
    if (c) {
      clock_unset(c->clock);
    }
  }
  return 0;
}

/** Lua proxy clock destruction. */
static int pdlua_clock_free(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd clock pointer.
  * */
) {
  if (lua_islightuserdata(L, 1)) {
    t_pdlua_proxyclock *c = lua_touserdata(L, 1);
    if (c) {
      clock_free(c->clock);
      free(c);
    }
  }
  return 0;
```

```
}

/** Lua object destruction. */
static int pdlua_object_free(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd object pointer.
  * */
) {
  unsigned int i;
  if (lua_islightuserdata(L, 1)) {
    t_pdlua *o = lua_touserdata(L, 1);
    if (o) {
      if (o->in) {
        free(o->in);
      }
      if(o->out) {
        for (i = 0; i < o->outlets; ++i) {
          outlet_free(o->out[i]);
        }
        free(o->out);
        o->out = NULL;
      }
    }
  }
  return 0;
}

/** Dispatch Pd inlet messages to Lua objects. */
static void pdlua_dispatch(
  t_pdlua *o, /**< The object that received the message. */
  unsigned int inlet, /**< The inlet that the message arrived at. */
  t_symbol *s, /**< The message selector. */
  int argc, /**< The message length. */
  t_atom *argv /**< The atoms in the message. */
) {
  lua_getglobal(L, "pd");
  lua_getfield (L, -1, "_dispatcher");
  lua_pushlightuserdata(L, o);
  lua_pushnumber(L, inlet + 1); /* C has 0.., Lua has 1.. */
  lua_pushstring(L, s->s_name);
  pdlua_pushatomtable(argc, argv);
  if (lua_pcall(L, 4, 0, 0)) {
    pd_error(
      o,
      "lua: error in dispatcher:\n%s",
      lua_tostring(L, -1)
    );
    lua_pop(L, 1); /* pop the error string */
  }
  lua_pop(L, 1); /* pop the global "pd" */
  return;
}

/** Dispatch Pd receive messages to Lua objects. */
static void pdlua_receivedispatch(
  t_pdlua_proxyreceive *r, /**< The proxy receive that received the message. */
```

```
    t_symbol *s, /**< The message selector. */
    int argc, /**< The message length. */
730 t_atom *argv /**< The atoms in the message. */
  ) {
    lua_getglobal(L, "pd");
    lua_getfield (L, -1, "_receivedispatch");
    lua_pushlightuserdata(L, r);
735 lua_pushstring(L, s->s_name);
    pdlua_pushatomtable(argc, argv);
    if (lua_pcall(L, 3, 0, 0)) {
      pd_error(
        r->owner,
740     "lua: error in receive dispatcher:\n%s",
        lua_tostring(L, -1)
      );
      lua_pop(L, 1); /* pop the error string */
    }
745 lua_pop(L, 1); /* pop the global "pd" */
    return;
  }

  /** Dispatch Pd clock messages to Lua objects. */
750 static void pdlua_clockdispatch(
    t_pdlua_proxyclock *clock /**< The proxy clock that received the message. */
  ) {
    lua_getglobal(L, "pd");
    lua_getfield (L, -1, "_clockdispatch");
755 lua_pushlightuserdata(L, clock);
    if (lua_pcall(L, 1, 0, 0)) {
      pd_error(
        clock->owner,
        "lua: error in clock dispatcher:\n%s",
760     lua_tostring(L, -1)
      );
      lua_pop(L, 1); /* pop the error string */
    }
    lua_pop(L, 1); /* pop the global "pd" */
765 return;
  }

  /** Convert a Lua table into a Pd atom array. */
  static t_atom *pdlua_popatomtable(
770 lua_State *L, /**< Lua interpreter state.
    * \par Inputs:
    * \li \c -1 Table to convert.
    * */
    int *count, /**< Where to store the array length. */
775 t_pdlua *o /**< Object reference for error messages. */
  ) {
    int i;
    int ok;
    t_float f;
780 const char *s;
    void *p;
    size_t sl;
    t_atom *atoms;
    atoms = NULL;
```

```
785     ok = 1;
        if (lua_istable(L, -1)) {
          *count = lua_objlen(L, -1);
          if (*count > 0) {
            atoms = malloc(*count * sizeof(t_atom));
790       }
          i = 0;
          lua_pushnil(L);
          while (lua_next(L, -2) != 0) {
            if (i == *count) {
795           pd_error(o, "lua: error: too many table elements");
              ok = 0;
              break;
            }
            switch (lua_type(L, -1)) {
800         case (LUA_TNUMBER):
              f = lua_tonumber(L, -1);
              SETFLOAT(&atoms[i], f);
              break;
            case (LUA_TSTRING):
805           s = lua_tolstring(L, -1, &sl);
              if (s) {
                if (strlen(s) != sl) {
                  pd_error(o, "lua: warning: symbol munged (contains \\0 in body)");
                }
810             SETSYMBOL(&atoms[i], gensym((char *) s));
              } else {
                pd_error(o, "lua: error: null string in table");
                ok = 0;
              }
815           break;
            case (LUA_TLIGHTUSERDATA): /* FIXME: check experimentality */
              p = lua_touserdata(L, -1);
              SETPOINTER(&atoms[i], p);
              break;
820         default:
              pd_error(o, "lua: error: table element must be number or string or ↙
                  ↳ pointer");
              ok = 0;
              break;
            }
825         lua_pop(L, 1);
            ++i;
          }
          if (i != *count) {
            pd_error(o, "lua: error: too few table elements");
830         ok = 0;
          }
        } else {
          pd_error(o, "lua: error: not a table");
          ok = 0;
835     }
        lua_pop(L, 1);
        if (ok) {
          return atoms;
        } else {
840       if (atoms) {
```

```
      free(atoms);
    }
    return NULL;
  }
}

/** Send a message from a Lua object outlet. */
static int pdlua_outlet(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd object pointer.
  * \li \c 2 Outlet number.
  * \li \c 3 Message selector string.
  * \li \c 4 Message atom table.
  * */
) {
  t_pdlua *o;
  int out;
  size_t sl;
  const char *s;
  t_symbol *sym;
  int count;
  t_atom *atoms;
  if (lua_islightuserdata(L, 1)) {
    o = lua_touserdata(L, 1);
    if (o) {
      if (lua_isnumber(L, 2)) {
        out = lua_tonumber(L, 2) - 1; /* C has 0.., Lua has 1.. */
      } else {
        pd_error(o, "lua: error: outlet must be a number");
        lua_pop(L, 4); /* pop all the arguments */
        return 0;
      }
      if (0 <= out && (unsigned int) out < o->outlets) {
        if (lua_isstring(L, 3)) {
          s = lua_tolstring(L, 3, &sl);
          sym = gensym((char *) s); /* const cast */
          if (s) {
            if (strlen(s) != sl) {
              pd_error(o, "lua: warning: symbol munged (contains \\0 in body)");
            }
            lua_pushvalue(L, 4);
            atoms = pdlua_popatomtable(L, &count, o);
            if (count == 0 || atoms) {
              outlet_anything(o->out[out], sym, count, atoms);
            } else {
              pd_error(o, "lua: error: no atoms??");
            }
            if (atoms) {
              free(atoms);
              lua_pop(L, 4); /* pop all the arguments */
              return 0;
            }
          } else {
            pd_error(o, "lua: error: null selector");
          }
        } else {
```

```
            pd_error(o, "lua: error: selector must be a string");
          }
900       } else {
          pd_error(o, "lua: error: outlet out of range");
        }
      } else {
        error("lua: error: no object to outlet from");
905   }
    } else {
      error("lua: error: bad arguments to outlet");
    }
    lua_pop(L, 4); /* pop all the arguments */
910 return 0;
}

/** Send a message from a Lua object to a Pd receiver. */
static int pdlua_send(
915   lua_State *L /**< Lua interpreter state.
    * \par Inputs:
    * \li \c 1 Receiver string.
    * \li \c 2 Message selector string.
    * \li \c 3 Message atom table.
920 * */
) {
    size_t receivenamel;
    const char *receivename;
    t_symbol *receivesym;
925 size_t selnamel;
    const char *selname;
    t_symbol *selsym;
    int count;
    t_atom *atoms;
930 if (lua_isstring(L, 1)) {
      receivename = lua_tolstring(L, 1, &receivenamel);
      receivesym = gensym((char *) receivename); /* const cast */
      if (receivesym) {
        if (strlen(receivename) != receivenamel) {
935       error("lua: warning: symbol munged (contains \\0 in body)");
        }
        if (lua_isstring(L, 2)) {
          selname = lua_tolstring(L, 2, &selnamel);
          selsym = gensym((char *) selname); /* const cast */
940       if (selsym) {
            if (strlen(selname) != selnamel) {
              error("lua: warning: symbol munged (contains \\0 in body)");
            }
            lua_pushvalue(L, 3);
945         atoms = pdlua_popatomtable(L, &count, NULL);
            if (count == 0 || atoms) {
              if (receivesym->s_thing) {
                typedmess(receivesym->s_thing, selsym, count, atoms);
              }
950         } else {
              error("lua: error: no atoms??");
            }
            if (atoms) {
              free(atoms);
```

```
955             return 0;
              }
            } else {
              error("lua: error: null selector");
            }
960       } else {
            error("lua: error: selector must be a string");
          }
        } else {
          error("lua: error: null receive name");
965     }
      } else {
        error("lua: error: receive name must be string");
      }
      return 0;
970 }

    /** Set a [value] object's value. */
    static int pdlua_setvalue(
      lua_State *L /**< Lua interpreter state.
975   * \par Inputs:
      * \li \c 1 Value name string.
      * \li \c 2 Value number.
      * \par Outputs:
      * \li \c 1 success (usually depends on a [value] existing or not).
980   */
    ) {
      const char *str = luaL_checkstring(L, 1);
      t_float val = luaL_checknumber(L, 2);
      int err = value_setfloat(gensym(str), val);
985   lua_pushboolean(L, !err);
      return 1;
    }

    /** Get a [value] object's value. */
990 static int pdlua_getvalue(
      lua_State *L /**< Lua interpreter state.
      * \par Inputs:
      * \li \c 1 Value name string.
      * \par Outputs:
995   * \li \c 1 Value number, or nil for failure.
      * */
    ) {
      const char *str = luaL_checkstring(L, 1);
      t_float val;
1000  int err = value_getfloat(gensym(str), &val);
      if (!err) {
        lua_pushnumber(L, val);
      } else {
        lua_pushnil(L);
1005  }
      return 1;
    }

    /** Get a [table] object's array. */
1010 static int pdlua_getarray(
      lua_State *L /**< Lua interpreter state.
```

```
    * \par Inputs:
    * \li \c 1 Table name string.
    * \par Outputs:
    * \li \c 1 Table length, or < 0 for failure.
    * \li \c 2 Table pointer, or nil for failure.
    * */
) {
  t_garray *a;
  int n;
  PDLUA_ARRAYTYPE *v;
  const char *str = luaL_checkstring(L, 1);
  if (!(a = (t_garray *) pd_findbyclass(gensym(str), garray_class))) {
    lua_pushnumber(L, -1);
    return 1;
  } else if (!PDLUA_ARRAYGRAB(a, &n, &v)) {
    lua_pushnumber(L, -2);
    return 1;
  } else {
    lua_pushnumber(L, n);
    lua_pushlightuserdata(L, v);
    return 2;
  }
}

/** Read from a [table] object's array. */
static int pdlua_readarray(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Table length number.
  * \li \c 2 Table array pointer.
  * \li \c 3 Table index number.
  * \par Outputs:
  * \li \c 1 Table element value, or nil for index out of range.
  * */
) {
  int n = luaL_checknumber(L, 1);
  PDLUA_ARRAYTYPE *v = lua_islightuserdata(L, 2) ? lua_touserdata(L, 2) : NULL;
  int i = luaL_checknumber(L, 3);
  if (0 <= i && i < n && v) {
    lua_pushnumber(L, PDLUA_ARRAYELEM(v, i));
    return 1;
  }
  return 0;
}

/** Write to a [table] object's array. */
static int pdlua_writearray(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Table length number.
  * \li \c 2 Table array pointer.
  * \li \c 3 Table index number.
  * \li \c 4 Table element value number.
  * */
) {
  int n = luaL_checknumber(L, 1);
  PDLUA_ARRAYTYPE *v = lua_islightuserdata(L, 2) ? lua_touserdata(L, 2) : NULL;
```

```
  int i = luaL_checknumber(L, 3);
  t_float x = luaL_checknumber(L, 4);
  if (0 <= i && i < n && v) {
    PDLUA_ARRAYELEM(v, i) = x;
  }
  return 0;
}

/** Redraw a [table] object's graph. */
static int pdlua_redrawarray(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Table name string.
  * */
) {
  t_garray *a;
  const char *str = luaL_checkstring(L, 1);
  if ((a = (t_garray *) pd_findbyclass(gensym(str), garray_class))) {
    garray_redraw(a);
  }
  return 0;
}

/** Post to Pd's console. */
static int pdlua_post(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Message string.
  * */
) {
  const char *str = luaL_checkstring(L, 1);
  post("%s", str);
  return 0;
}

/** Report an error from a Lua object to Pd's console. */
static int pdlua_error(
  lua_State *L /**< Lua interpreter state.
  * \par Inputs:
  * \li \c 1 Pd object pointer.
  * \li \c 2 Message string.
  * */
) {
  t_pdlua *o;
  const char *s;
  if (lua_islightuserdata(L, 1)) {
    o = lua_touserdata(L, 1);
    if (o) {
      s = luaL_checkstring(L, 2);
      if (s) {
        pd_error(o, "%s", s);
      } else {
        pd_error(o, "lua: error: null string in error function");
      }
    } else {
      error("lua: error: null object in error function");
    }
```

```
    } else {
      error("lua: error: bad arguments to error function");
    }
    return 0;
}

static void pdlua_setrequirepath( /* FIXME: documentation */
    lua_State *L,
    const char *path
) {
    lua_getglobal(L, "pd");
    lua_pushstring(L, "_setrequirepath");
    lua_gettable(L, -2);
    lua_pushstring(L, path);
    if (lua_pcall(L, 1, 0, 0) != 0) {
      post("lua: internal error in 'pd._setrequirepath': %s", lua_tostring(L, -1));
      lua_pop(L, 1);
    }
    lua_pop(L, 1);
}

static void pdlua_clearrequirepath( /* FIXME: documentation */
    lua_State *L
) {
    lua_getglobal(L, "pd");
    lua_pushstring(L, "_clearrequirepath");
    lua_gettable(L, -2);
    if (lua_pcall(L, 0, 0, 0) != 0) {
      post("lua: internal error in 'pd._clearrequirepath': %s", lua_tostring(L, -1));
      lua_pop(L, 1);
    }
    lua_pop(L, 1);
}

/** Run a Lua script using Pd's path. */
static int pdlua_dofile(
    lua_State *L /**< Lua interpreter state.
    * \par Inputs:
    * \li \c 1 Pd object pointer.
    * \li \c 2 Filename string.
    * \par Outputs:
    * \li \c * Determined by the script.
    * */
) {
    char buf[MAXPDSTRING];
    char *ptr;
    t_pdlua_readerdata reader;
    int fd;
    int n;
    const char *filename;
    t_pdlua * o;
    n = lua_gettop(L);
    if (lua_islightuserdata(L, 1)) {
      o = lua_touserdata(L, 1);
      if (o) {
```

```
      filename = luaL_optstring(L, 2, NULL);
      fd = canvas_open(o->canvas, filename, "", buf, &ptr, MAXPDSTRING, 1);
      if (fd >= 0) {
        pdlua_setrequirepath(L, buf);
        reader.fd = fd;
        if (lua_load(L, pdlua_reader, &reader, filename)) {
          close(fd);
          pdlua_clearrequirepath(L);
          lua_error(L);
        } else {
          if (lua_pcall(L, 0, LUA_MULTRET, 0)) {
            pd_error(
              o,
              "lua: error running `%s':\n%s",
              filename,
              lua_tostring(L, -1)
            );
            lua_pop(L, 1);
            close(fd);
            pdlua_clearrequirepath(L);
          } else {
            /* succeeded */
            close(fd);
            pdlua_clearrequirepath(L);
          }
        }
      } else {
        pd_error(o, "lua: error loading `%s': canvas_open() failed", filename);
      }
    } else {
      error("lua: error in object:dofile() - object is null");
    }
  } else {
    error("lua: error in object:dofile() - object is wrong type");
  }
  lua_pushstring(L, buf); /* return the path as well so we can open it later ↙
      ↳ with pdlua_menu_open() */
  return lua_gettop(L) - n;
}

/** Initialize the pd API for Lua. */
static void pdlua_init(
  lua_State *L /**< Lua interpreter state. */
) {
  lua_newtable(L);
  lua_setglobal(L, "pd");
  lua_getglobal(L, "pd");
  lua_pushstring(L, "_iswindows");
#ifdef _WIN32
  lua_pushboolean(L, 1);
#else
  lua_pushboolean(L, 0);
#endif
  lua_settable(L, -3);
  lua_pushstring(L, "_register");
  lua_pushcfunction(L, pdlua_class_new);
  lua_settable(L, -3);
```

```
    lua_pushstring(L, "_create");
    lua_pushcfunction(L, pdlua_object_new);
    lua_settable(L, -3);
    lua_pushstring(L, "_createinlets");
    lua_pushcfunction(L, pdlua_object_createinlets);
    lua_settable(L, -3);
    lua_pushstring(L, "_createoutlets");
    lua_pushcfunction(L, pdlua_object_createoutlets);
    lua_settable(L, -3);
    lua_pushstring(L, "_destroy");
    lua_pushcfunction(L, pdlua_object_free);
    lua_settable(L, -3);
    lua_pushstring(L, "_outlet");
    lua_pushcfunction(L, pdlua_outlet);
    lua_settable(L, -3);
    lua_pushstring(L, "_createreceive");
    lua_pushcfunction(L, pdlua_receive_new);
    lua_settable(L, -3);
    lua_pushstring(L, "_receivefree");
    lua_pushcfunction(L, pdlua_receive_free);
    lua_settable(L, -3);
    lua_pushstring(L, "_createclock");
    lua_pushcfunction(L, pdlua_clock_new);
    lua_settable(L, -3);
    lua_pushstring(L, "_clockfree");
    lua_pushcfunction(L, pdlua_clock_free);
    lua_settable(L, -3);
    lua_pushstring(L, "_clockset");
    lua_pushcfunction(L, pdlua_clock_set);
    lua_settable(L, -3);
    lua_pushstring(L, "_clockunset");
    lua_pushcfunction(L, pdlua_clock_unset);
    lua_settable(L, -3);
    lua_pushstring(L, "_clockdelay");
    lua_pushcfunction(L, pdlua_clock_delay);
    lua_settable(L, -3);
    lua_pushstring(L, "_dofile");
    lua_pushcfunction(L, pdlua_dofile);
    lua_settable(L, -3);
    lua_pushstring(L, "send");
    lua_pushcfunction(L, pdlua_send);
    lua_settable(L, -3);
    lua_pushstring(L, "getvalue");
    lua_pushcfunction(L, pdlua_getvalue);
    lua_settable(L, -3);
    lua_pushstring(L, "setvalue");
    lua_pushcfunction(L, pdlua_setvalue);
    lua_settable(L, -3);
    lua_pushstring(L, "_getarray");
    lua_pushcfunction(L, pdlua_getarray);
    lua_settable(L, -3);
    lua_pushstring(L, "_readarray");
    lua_pushcfunction(L, pdlua_readarray);
    lua_settable(L, -3);
    lua_pushstring(L, "_writearray");
    lua_pushcfunction(L, pdlua_writearray);
    lua_settable(L, -3);
```

```
    lua_pushstring(L, "_redrawarray");
    lua_pushcfunction(L, pdlua_redrawarray);
    lua_settable(L, -3);
    lua_pushstring(L, "post");
    lua_pushcfunction(L, pdlua_post);
    lua_settable(L, -3);
    lua_pushstring(L, "_error");
    lua_pushcfunction(L, pdlua_error);
    lua_settable(L, -3);
    lua_pop(L, 1);
}

/** Pd loader hook for loading and executing Lua scripts. */
static int pdlua_loader(
    t_canvas *canvas, /**< Pd canvas to use to find the script. */
    char *name /**< The name of the script (without .pd_lua extension). */
) {
    char dirbuf[MAXPDSTRING];
    char *ptr;
    int fd;
    t_pdlua_readerdata reader;
    fd = canvas_open(canvas, name, ".pd_lua", dirbuf, &ptr, MAXPDSTRING, 1);
    if (fd >= 0) {
        class_set_extern_dir(gensym(dirbuf));
        pdlua_setrequirepath(L, dirbuf);
        reader.fd = fd;
        if (lua_load(L, pdlua_reader, &reader, name) ||
            lua_pcall(L, 0, 0, 0)) {
            error(
                "lua: error loading `%s':\n%s",
                name,
                lua_tostring(L, -1)
            );
            lua_pop(L, 1);
            close(fd);
            pdlua_clearrequirepath(L);
            class_set_extern_dir(&s_);
            return 0;
        }
        close(fd);
        pdlua_clearrequirepath(L);
        class_set_extern_dir(&s_);
        return 1;
    } else {
        return 0;
    }
}

/** Start the Lua runtime and register our loader hook. */
#ifdef _WIN32
__declspec(dllexport)
#endif
extern void lua_setup(void) {
    char pd_lua_path[MAXPDSTRING];
    t_pdlua_readerdata reader;
    int fd;
    char *luaver1  = "lua 0.8 (GPL) 2013 Claude Heiland-Allen, merging";
```

```
  char *pdluaver = "pdlua 0.7.1 (GPL) 2011 Martin Peach, based on";
  char *luaver0  = "lua 0.6~svn (GPL) 2008 Claude Heiland-Allen";
  char compiled[MAXPDSTRING];
  snprintf(compiled, MAXPDSTRING, "lua: compiled for pd-%d.%d on %s %s",
      ↪ PD_MAJOR_VERSION, PD_MINOR_VERSION, __DATE__, __TIME__);
#if PD_MAJOR_VERSION==0 && PD_MINOR_VERSION<43
#define POST(S) post(S)
#else
#define POST(S) logpost(0, 3, S)
#endif
  POST(luaver1);
  POST(pdluaver);
  POST(luaver0);
  POST(compiled);
#undef POST
  pdlua_proxyinlet_setup();
  pdlua_proxyreceive_setup();
  pdlua_proxyclock_setup();
  L = lua_open();
  luaL_openlibs(L);
  pdlua_init(L);
  /* "pd.lua" is the Lua part of pdlua, want to keep the C part minimal */
  /* canvas_open can't find pd.lua unless we give the path to pd beforehand like
      ↪  pd -path /usr/lib/extra/lua */
  /* To avoid this we can use c_externdir from m_imp.h, struct _class: t_symbol
      ↪ *c_externdir; */
  /* c_externdir is the directory the extern was loaded from and is also the
      ↪ directory contining pd.lua */
  sprintf(pd_lua_path, "%s/pd.lua", pdlua_proxyinlet_class->c_externdir->s_name)
      ↪ ; /* the full path to pd.lua */
  fd = open(pd_lua_path, O_RDONLY);
  if (fd >= 0) {
    reader.fd = fd;
    if (lua_load(L, pdlua_reader, &reader, "pd.lua") ||
        lua_pcall(L, 0, 0, 0)) {
      error(
        "lua: error loading `%s':\n%s",
        pd_lua_path,
        lua_tostring(L, -1)
      );
      error("lua: loader will not be registered!");
      lua_pop(L, 1);
      close(fd);
    } else {
      close(fd);
      sys_register_loader(pdlua_loader);
    }
  } else {
    error("lua: error loading `%s': open() failed", pd_lua_path);
    error("lua: loader will not be registered!");
  }
}

/* EOF */
```

80 src/lua-help.pd

```
#N canvas 0 22 534 259 10;
#X obj 17 101 lua;
#X msg 44 58;
#X text 118 58 << more methods will come (maybe);
#X text 118 102 << global interface to pdlua;
#X text 118 20 << load and run a Lua file (searches Pd's path);
#X msg 17 21 load hello.lua;
#X text 17 191 See also:;
#X obj 87 216 luax hello;
#X obj 37 216 hello;
#X text 15 135 Side-effects:;
#X text 29 157 [lua] registers a loader that allows Pd classes written
in Lua to be loaded.;
#X connect 1 0 0 0;
#X connect 5 0 0 0;
```

81 src/luax-help.pd

```
#N canvas 0 22 447 171 10;
#X text 31 16 [luax foo] loads the source file "foo.pd_luax" on each
object creation. Less efficient than the loader ([foo] loads "foo.pd_lua"
just once for all time) but more flexible when developing or live-coding.
;
#X obj 33 88 luax hello;
#X obj 123 88 luax hello;
#X obj 213 88 luax hello;
#X obj 303 88 luax hello;
#X obj 134 144 hello;
#X obj 88 144 lua;
#X text 31 122 See also:;
```

82 src/Makefile.am

```
SUFFIXES = .@PDEXT@
PDEXT    = @PDEXT@
EXEEXT   = .@PDEXT@
LFLAGS   = @LFLAGS@
pdextra_PROGRAMS = lua
EXTRA_PROGRAMS = lua
lua_SOURCES = lua.c
lua_LDFLAGS = $(LFLAGS)
lua_CFLAGS = -fPIC
pdextra_DATA = pd.lua lua-help.pd luax-help.pd hello-help.pd hello.lua hello.↙
    ↳ pd_lua hello.pd_luax
pddoc_DATA = ../doc/*.txt ../examples/*.pd ../examples/*.pd_lua ../examples/*.↙
    ↳ pd_luax
```

83 src/pd.lua

```
--[[
pdlua -- a Lua embedding for Pd
Copyright (C) 2007,2008,2013 Claude Heiland-Allen <claude@mathr.co.uk>
Copyright (C) 2012 Martin Peach martin.peach@sympatico.ca

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
```

```
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
--]]

-- storage for Pd C<->Lua interaction
pd._classes = { }
pd._objects = { }
pd._clocks = { }
pd._receives = { }

-- add a path to Lua's "require" search paths
pd._setrequirepath = function(path)
  pd._packagepath = package.path
  pd._packagecpath = package.cpath
  if (pd._iswindows) then
    package.path = path .. "\\?;" .. path .. "\\?.lua;" .. package.path
    package.cpath = path .. "\\?.dll;" .. package.cpath
  else
    package.path = path .. "/?;" .. path .. "/?.lua;" .. package.path
    package.cpath = path .. "/?.so;" .. package.cpath
  end
end

-- reset Lua's "require" search paths
pd._clearrequirepath = function()
  package.path = pd._packagepath
  package.cpath = pd._packagecpath
end

-- constructor dispatcher
pd._constructor = function (name, atoms)
  if nil ~= pd._classes[name] then
    local o = pd._classes[name]:new():construct(name, atoms)
    if o then
      pd._objects[o._object] = o
      return o._object
    end
  end
  return nil
end

-- destructor dispatcher
pd._destructor = function (object)
  if nil ~= pd._objects[object] then
    pd._objects[object]:destruct()
  end
end
```

```
65  -- inlet method dispatcher
    pd._dispatcher = function (object, inlet, sel, atoms)
      if nil ~= pd._objects[object] then
        pd._objects[object]:dispatch(inlet, sel, atoms)
      end
70  end

    -- clock method dispatcher
    pd._clockdispatch = function (c)
      if nil ~= pd._clocks[c] then
75      local m = pd._clocks[c]._method
        pd._clocks[c]._target[m](pd._clocks[c]._target)
      end
    end

80  -- whoami method dispatcher
    pd._whoami = function (object)
      if nil ~= pd._objects[object] then
        return pd._objects[object]:whoami()
      end
85  end

    -- class method dispatcher
    pd._get_class = function (object)
      if nil ~= pd._objects[object] then
90      return pd._objects[object]:get_class()
      end
    end

    -- prototypical OO system
95  pd.Prototype = { }
    function pd.Prototype:new(o)
      o = o or {}
      setmetatable(o, self)
      self.__index = self
100   return o
    end


    -- clocks
105 pd.Clock = pd.Prototype:new()
    function pd.Clock:register(object, method)
      if nil ~= object then
        if nil ~= object._object then
          self._clock = pd._createclock(object._object, method)
110       self._target = object
          self._method = method
          pd._clocks[self._clock] = self
          return self
        end
115   end
      return nil
    end
    function pd.Clock:destruct()
      pd._clocks[self._clock] = nil
120   pd._clockfree(self._clock)
      self._clock = nil
```

```
end
function pd.Clock:dispatch()
  local m = self._target[self._method]
  if type(m) == "function" then
    return m(self._target)
  else
    self._target:error(
      "no method for `" .. self._method ..
      "' at clock of Lua object `" .. self._name .. "'"
    )
  end
end
function pd.Clock:set(systime)
  pd._clockset(self._clock, systime)
end
function pd.Clock:delay(delaytime)
  pd._clockdelay(self._clock, delaytime)
end
function pd.Clock:unset()
  pd._clockunset(self._clock)
end

-- tables
pd.Table = pd.Prototype:new()
function pd.Table:sync(name)
  self.name = name
  self._length, self._array = pd._getarray(name)
  if self._length < 0 then
    return nil
  else
    return self
  end
end
function pd.Table:destruct()
  self._length = -3
  self._array = nil
end
function pd.Table:get(i)
  if type(i) == "number" and 0 <= i and i < self._length then
    return pd._readarray(self._length, self._array, i)
  else
    return nil
  end
end
function pd.Table:set(i, f)
  if type(i) == "number" and type(f) == "number" and 0 <= i and i < self._length
    ↪ then
    return pd._writearray(self._length, self._array, i, f)
  else
    return nil
  end
end
function pd.Table:length()
  if self._length >= 0 then
    return self._length
  else
    return nil
```

```
  end
end
function pd.Table:redraw()
  pd._redrawarray(self.name)
end

-- receivers
function pd._receivedispatch(receive, sel, atoms)
  if nil ~= pd._receives[receive] then
    pd._receives[receive]:dispatch(sel, atoms)
  end
end
pd.Receive = pd.Prototype:new()
function pd.Receive:register(object, name, method)
  if nil ~= object then
    if nil ~= object._object then
      self._receive = pd._createreceive(object._object, name)
      self._name = name
      self._target = object
      self._method = method
      pd._receives[self._receive] = self
      return self
    end
  end
  return nil
end
function pd.Receive:destruct()
  pd._receives[self._receive] = nil
  pd._receivefree(self._receive)
  self._receive = nil
  self._name = nil
  self._target = nil
  self._method = nil
end
function pd.Receive:dispatch(sel, atoms)
  self._target[self._method](self._target, sel, atoms)
end

-- patchable objects
pd.Class = pd.Prototype:new()
function pd.Class:register(name)
  if nil ~= pd._classes[name] then    -- already registered
    return pd._classes[name]          -- return existing
  else
    self._class = pd._register(name)  -- register new class
    pd._classes[name] = self          -- record registration
    self._name = name
    if name == "lua" then
      self._scriptname = "pd.lua"
    else
      self._scriptname = name .. ".pd_lua"
    end -- mrpeach 20111027
    return self                       -- return new
  end
end
function pd.Class:construct(sel, atoms)
  self._object = pd._create(self._class)
```

```
235     self.inlets = 0
        self.outlets = 0
        if self:initialize(sel, atoms) then
          pd._createinlets(self._object, self.inlets)
          pd._createoutlets(self._object, self.outlets)
240       self:postinitialize()
          return self
        else
          return nil
        end
245   end
      function pd.Class:destruct()
        pd._objects[self] = nil
        self:finalize()
        pd._destroy(self._object)
250   end
      function pd.Class:dispatch(inlet, sel, atoms)
        local m = self["in_" .. inlet .. "_" .. sel]
        if type(m) == "function" then
          if sel == "bang"    then return m(self)           end
255       if sel == "float"   then return m(self, atoms[1]) end
          if sel == "symbol"  then return m(self, atoms[1]) end
          if sel == "pointer" then return m(self, atoms[1]) end
          if sel == "list"    then return m(self, atoms)    end
          return m(self, atoms)
260     end
        m = self["in_n_" .. sel]
        if type(m) == "function" then
          if sel == "bang"    then return m(self, inlet)           end
          if sel == "float"   then return m(self, inlet, atoms[1]) end
265       if sel == "symbol"  then return m(self, inlet, atoms[1]) end
          if sel == "pointer" then return m(self, inlet, atoms[1]) end
          if sel == "list"    then return m(self, inlet, atoms)    end
          return m(self, inlet, atoms)
        end
270     m = self["in_" .. inlet]
        if type(m) == "function" then
          return m(self, sel, atoms)
        end
        m = self["in_n"]
275     if type(m) == "function" then
          return m(self, inlet, sel, atoms)
        end
        self:error(
          "no method for `" .. sel ..
280       "' at inlet " .. inlet ..
          " of Lua object `" .. self._name .. "'"
        )
      end
      function pd.Class:outlet(outlet, sel, atoms)
285     pd._outlet(self._object, outlet, sel, atoms)
      end
      function pd.Class:initialize(sel, atoms) end
      function pd.Class:postinitialize() end
      function pd.Class:finalize() end
290   function pd.Class:dofile(file)
        return pd._dofile(self._object, file)
```

```
end
function pd.Class:error(msg)
  pd._error(self._object, msg)
end
function pd.Class:whoami()
  return self._scriptname or self._name
end
function pd.Class:get_class() -- accessor for t_class*
  return self._class or nil
end

local lua = pd.Class:new():register("lua")  -- global controls
function lua:initialize(sel, atoms)
  self.inlets = 1
  self.outlets = 0    -- FIXME: might be nice to have errors go here?
  return true
end
function lua:in_1_load(atoms)  -- execute a script
  self:dofile(atoms[1])
end


local luax = pd.Class:new():register("luax")  -- classless lua externals
function luax:initialize(sel, atoms)          -- motivation: pd-list 2007-09-23
  local f, pathname = self:dofile(atoms[1] .. ".pd_luax")
  if nil ~= f then
    self._scriptname = pathname .. "/" .. atoms[1] .. ".pd_luax" -- mrpeach ↙
        ↳ 20120201
    local atomstail = { }          -- munge for better lua<->luax compatibility
    for i,_ in ipairs(atoms) do
      if i > 1 then
        atomstail[i-1] = atoms[i]
      end
    end
    return f(self, atoms[1], atomstail)
  else
    return false   -- error message already output by dofile()
  end
end
```

84 TODO

```
todo
- fix Pd crash when a Lua error contains '{'
- documentation
- install target for make
- audit require support thoroughly:
  - make require() support compiled packages
  - make require() look next to Lua source before containing Pd patch
- add access to this script's and (containing/parent) Pd patch paths

done
- /usr/bin/ld: lua-lua.o: relocation R_X86_64_32 against `.rodata.str1.1'
  can not be used when making a shared object; recompile with -fPIC
- variable support (like [v foo])
- table support (like [tabread]/[tabwrite])
- send support
```

```
- add hook for object (post creation)
- receive support (preferably multiple receives)
- clock support
- audit possible bugs relating to self.f = f(self, ...) misuse
- write docs on foo.lua vs foo.luax
- change inlet/outlet numbering to start from 1 instead of 0
- add pd.dofile(file) that searches via Pd's path (actually obj:dofile(file))
- write docs implementation (eg foo._bar => internal, etc)
- add more features to lexpr
```