P A T C H W O R K n e w s l e t t e r

n° 2 - October 1995

This is the Newsletter for PatchWork. It it is composed of frequently Asked Questions and Update lists for PatchWork. It serves two purposes:

• to reduce the number of "often asked questions" about PatchWork, by posting the questions and answers in this document so that they are shared by all.

• to inform the community of PatchWork users about updates and improvements.

Please send your corrections, questions, and comments to the editor,
Marc Battier, at E-mail : ircam-doc@ircam.fr.

Please report all bugs to:
Mikhail Malt, Laurent Pottier, at:
E-mail : ircam-doc@ircam.fr
mail : Ircam, 31, rue Saint-Merri, F-75004 Paris (France)
fax : (1) 42 77 29 47

==========

**** TABLE OF CONTENTS ****

Environment

New Developments

This newsletter corresponds to the release of PatchWork 2.5. Changes are explained below, and in the `README` file that you will find in the PW 2.5 folder.

• PatchWork 2.5 runs on the two kinds of Macintoshes processors (68OxO and PowerPC).

Running PatchWork

Note to Apple Common Lisp 2.0.1 users

Please note that if you own Apple MCL 2.0.1., you should recreate a PatchWork image from scratch since that way you will be able to take advantage of the power of the MCL compiler.
To do this, after having installed PatchWork, first delete the file `PW 2.5:images:PW-music.image`. Then launch MCL 2.0.1, and, from there, load the file `PW 2.5:images:set-PW-image.fasl`. You will be asked to enter an absolute Macintosh directory name (typically `HD:PW 2.5` if HD is the name of your Hard Disk) which must correspond to the directory where PatchWork has been installed.
Then the process of recreating a PatchWork image file will begin. This should take from 5 to 15 minutes, depending on the type of your Macintosh. When it is done, MCL will quit and a new file `PW 2.5:images:PW-music.image` will appear on the finder. Double click it in order to launch PatchWork.

• Caution: don't try to load the file `set-PW-image.fasl` from within a PatchWork image, as PatchWork will crash in that case.

• A known problem that might happen when a new PatchWork image is NOT recreated is the following:

• MCL will not find the help file needed when you ask for the documentation of a PatchWork box (or any standard Common Lisp function). MCL will issue a dialog asking if the user wants to locate the help file. As stated in the PatchWork introduction manual you should answer No (due to a strange behaviour of MCL in this case). You will then be able to access PatchWork boxes' documentation but not Common Lisp functions documentation. A way to avoid all these problems, other than recreating the PatchWork image, is to have both the MCL 2.0.1 and PW 2.5 folders at the same level on your hard disk.

Note to Apple Common Lisp 2.0 users

If you have an installed version of Apple MCL 2.0 you will have to upgrade to MCL 2.0.1, as the re-creation of a PatchWork image won't work otherwise. If you don't recreate a new one, the PatchWork image distributed on the diskettes will work anyway, but some strange behaviours might happen.
It is unwise to have several versions of MCL on the same disk, or several PatchWork Images on the same disk.

Note on Config

When you run the menu command **PWoper->Save Current Config**, PatchWork saves the name of currently loaded libraries or abstraction folders in a file named `PW 2.5:PW-inits:config.init`. The next time you launch PatchWork, it will try to load these libraries or abstractions. You can reset this auto-load mechanism by running **PWoper->Delete Config**, which will reset the file `config.init`. If however there is a problem when launching PatchWork (like if one of your auto-loaded libraries has a bug), just remove the file `config.init` from the `pw-inits` folder. It will be re-created automatically next time you call **PWoper>Save Current Config**.

Using abstractions

1. When using the command **PWoper:load abstracts**, open the target folder by double-clicking it in the dialog box, then, when inside it, press **Select current folder**. Otherwise PatchWork will try to load the father directory, which could result in very long disk scanning (and an error message !).
2. When loading a patch containing compiled abstractions, be sure to load those compiled abstractions before loading the patch, otherwise loading the patch will produce an error message!

Updates: What's New in PatchWork 2.5?

PW-Script technology.

PatchWork is now compatible with the powerful AppleScript Technology (see the PW-Script Manual). This lets you build scripts that automatize tasks like creating and editing a patch. The scripting feature of PatchWork (we call it PW-Script for short) is also useful for communication between PatchWork and other Macintosh applications. PW-script is integrated in the PatchWork kernel. A new library "aac.lib" lets you handle Apple scripts inside a PatchWork box. See the new PW-Script manual (September 1995).

PW-Midi Manager

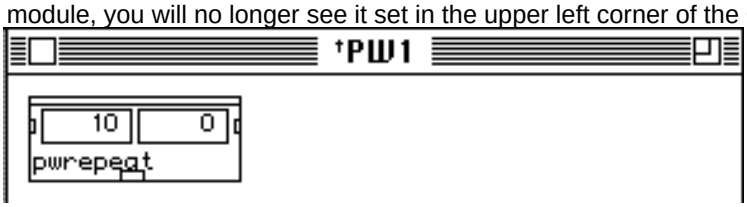
PatchWork 2.5 is now compatible with Apple Midi Manager, which means it can send and receive MIDI instructions to and from other applications running in the background as well as internal boards like Digidesign SampleCell. See the PW-Midi-Manager manual (September 1995).

Compatibility with OMS

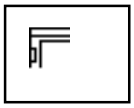
OMS contains Apple Midi Manager. You can use OMS as long as you allow "Non-OMS Applications" in the "OMS Midi Setup" window.

The loading of modules

When loading a new module, you will no longer see it set in the upper left corner of the OPW window.



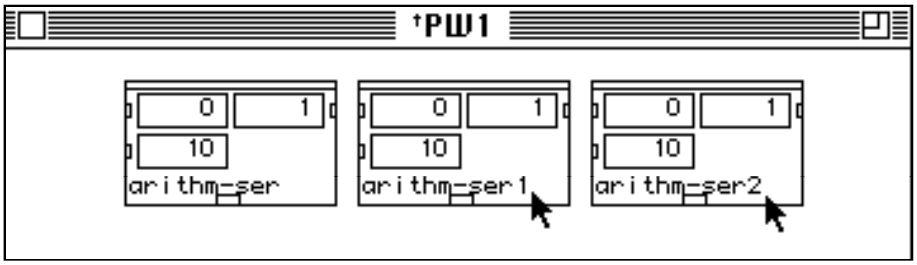
Instead, PW 2.5 will display this :



Drag then click at the desired location in the PW window.

Multiple instances of a module

When loading several identical modules, an index number is added to the name of the module:

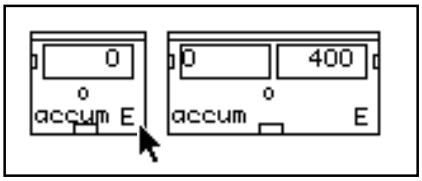


The numbering changes according to the addition or the suppression of a module.

Caution: the mere duplication of a module (using the key command -d for instance) does not create or increment an index.

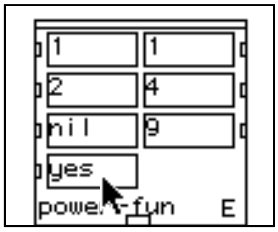
The module accum

The module **accum** becomes extensible. The number of elements accumulated is 400, by default, but it is possible to modify this value by the extensible input.



The module power-fun

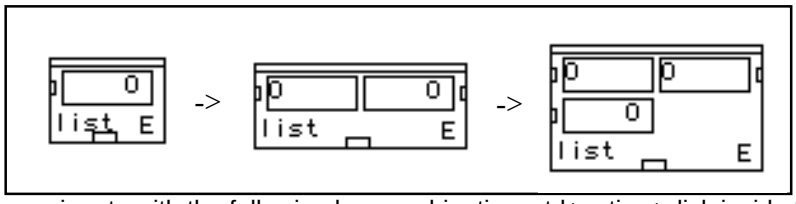
The module **power-fun** has gained an additional input to allow the display of the corresponding function:




```
? y = 1.0 x ^ 2.0
PW->#<Anonymous Function #xCC2D8E>
```

Extensible modules

You have now the ability to remove extensible module inputs. Take for example the module **list** (from **Kernel>lisp**).



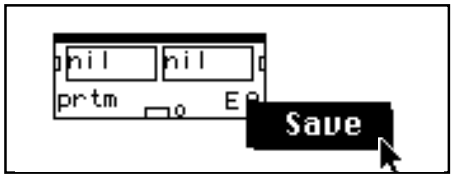
It is possible to remove inputs with the following key combination: ctrl+option+click inside the box.

The module play-object

the module **play-object** has an input for MIDI channels.
The convention now is :
-1 for MIDI channel 1 (it used to be 0 or 1 depending on what was connected to it).
- If you put 0, **play-object** will play using the MIDI channel setup of incoming objects, without changing it.

The module poly-rtm

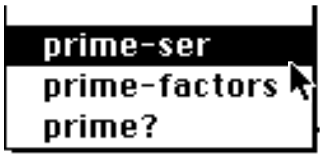
The module **poly-rtm** can now save musical data.



New modules in PW 2.5

New modules in Kernel>Num-series

Three modules have been added in the **Kernel>Num-series.**



New module: prime-ser

Returns a list of prime numbers between 0 and max.

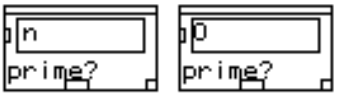
```
Pw->(1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97)
```

New module: Prime-factors

Factories *number* in prime factors. Ex: *number*=500

`? PW->((2 2) (5 3))` which means $2^2 * 5^3$

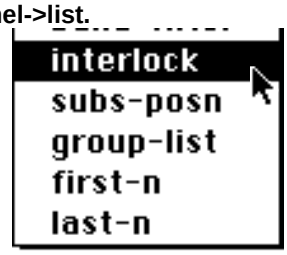
New module: Prime?



Checks if *n* is prime. If yes, then "t", else "nil".

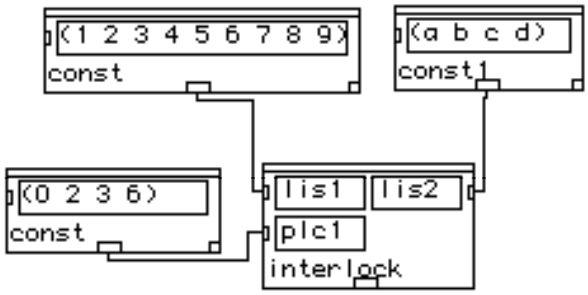
New modules in Kernel>list

Five modules have been added to **Kernel->list.**



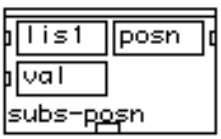
New module: interlock

Interleaving of *lis1* in *lis2* before elements placed at location *plc1*. Example:



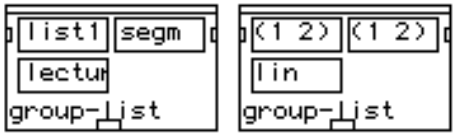
where
lis1 = (1 2 3 4 5 6 7 8 9)
lis2 = (a b c d)
plc1 = (0 2 3 6)
`? PW->(a 1 2 b 3 c 4 5 6 d 7 8 9)`

New module: subs-posn



Substitution of all the elements of *lis1* which are located at *posn* by successive elements from list *val*.

New module: group-list

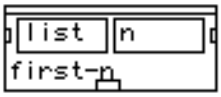


This module groups elements of list *list1* as sub-lists with lengths equal to elements of *segm*.
Example:
where
list1= '(0 1 2 3 4 5 6 7 8 9 10)
segm= '(2 4 3 2)

```
(group-list '(0 1 2 3 4 5 6 7 8 9 10)  '(2 4 3 2))
PW->((0 1) ( 2 3 4 5) ( 6 7 8) (9 10)).
```

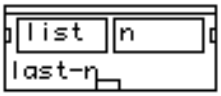
It is possible to choose linear or circular scanning of *list1*.

New module: first-n



Returns the *n* first elements of list *list*.

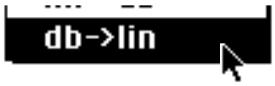
New module: last-n



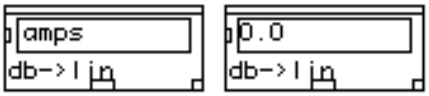
Returns the *n* last elements of list *list*.

New module in Music>Conv-approx

One new module has been added to **Music>Conv-approx.**



New module: db->lin



This module converts a value or a list of values expressed in dB into linear amplitudes.

Libraries updates

Kant 1.5

This new version of the Kant rhythm quantification library has several new interesting features. They are fully described in the addendum manual to the Kant documentation (September 1995).

Profile library

The Profile library has been written by Mikhail Malt and Jacopo Baboni-Schilingi to control melodic contours. A manual has been published.

Csound library

• The menus have been slightly modified, and some items have been placed in sub-menus:

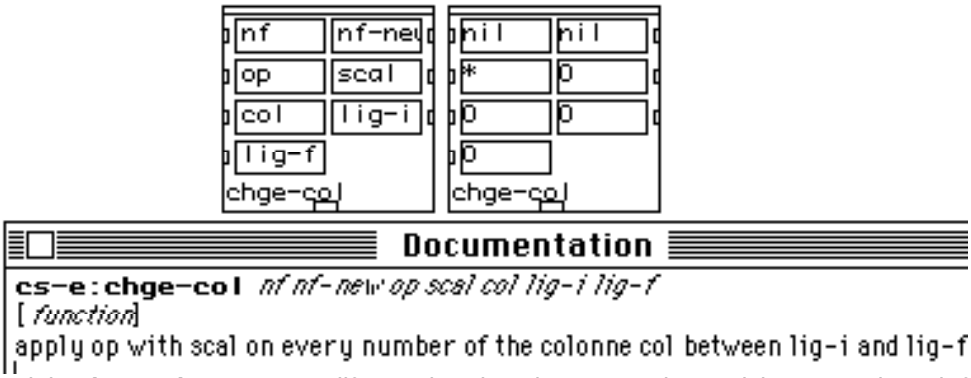
• Tutorials are now available for every module. To access the tutorial, select the module and type "t". Some tutorials have been improved.

• The scores now use "tab" separators, which enhance legibility. Parameters numbers appear as comments.

• The compiled function **echant-max** has been fixed.

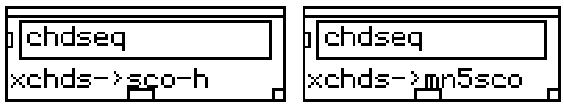
New module: chge-col

A new function is available: **chge-col**.



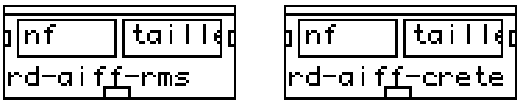
With the module **chge-col** you can modify an already written score by applying a PatchWork function on one of its columns.

New modules: xchds->sco-h and xchds->mn5sco



The new modules **xchds->sco-h** and **xchds->mn5sco** allow you to write a score from a **chordseq**.
With **xchds->sco-h**, the score is written horizontally (as a sequence of chords).
With **xchds->mn5sco**, the score is written vertically as a path of frequencies. It is then necessary to have chords with a similar number of notes, and not to have more than 24 chords (a Csound limit).

New modules: rd-aiff-rms and rd-aiff-crete



The two modules compute the amplitude envelope of mono AIFF soundfiles. The result can be displayed in a **BPF**. The *taille* parameter specifies the size in samples for the window which is used to scan the soundfile.

Midifile library

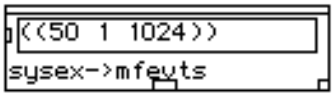
• The midifiles writing and reading functions have been improved.

• A bug on the writing and reading of pitchbends has been fixed.

• MIDI events time resolution has been improved, and is now at 480 divisions per beat.

• The Midi-File library has been fixed in order to generate Midifiles with a default tempo quarter-note=60.

New module: sysex->mfevts



The new module **sysex->mfevents** allows you to write MIDI system exclusive messages into Midifiles.

Chant library

• Chant is now compatible with PowerPC Macintoshes. There are now two versions, one for each type of Macintosh processor (680X0 and PowerPC). The right version is loaded automatically by the program. A new version has been released in October 1995 (v. 1.3).

• Stereo synthesis has been fixed.

• Synthesis behavior was sometimes erratic. This has been fixed.

• Filtering of sounds shorter than the result (output) sound is now allowed.

Libraries current versions

Here are the version numbers of the libraries:

3Dim-disp	v. 1.0	(April 1994)
AAC	v. 1.0	(July 1995)
ALEA	v. 1.0	(October 1993)
Chant	v. 1.3	(October 1995)
Chaos	v. 1.0	(June 1994)
Csound-edit-sco	v. 1.2	(October 1995)
Esquisse	v. 1.0	(October 1994)
Functionals	v. 1.0	(February 1994)
Instruments	v. 1.0	(October 1993)
Kant	v. 1.5	(September 1995)
Midi-file	v. 1.1	(September 1995)
Profile	v. 1.0	(October 1995)
Situation	v. 1.0	(February 1994)

Bug fixes

Bug fixes in version 2.5

In addition to the inormation listed above in the What's New in PatchWork 2.5" section, here are some bug fixes and improvements available • The interface to Finale 2.6 (menu **file->save as Enigma** in RTM editor) used to bug with chords. It has been fixed.

• All play routines have been improved and unified. You can play a closed **RTM** module by selecting it and pressing 'p' on the keyboard. To stop, press 's'.

• The menu **File>Save as Midi** file in **RTM** and **Chordseq** used to generate inconsistent Midifiles. This has been fixed, but only with the tempo quarter-note=60. Please check that your sequencer or music notation software are set at this very tempo.

• The **RTM** module has been fixed in order to generate the right onset time for chords, even after it has been edited by hand.

• The duration value of the notes of the module **RTM** is now by default 100% of the delay between notes attacks. This fixes some errors that happened when extracting duration using the module **RTM-dim**.

• The module **posn-order** has been fixed.

Bug fixes in version 2.1

• A new version of **Quantify** has been installed. It has an optional "precision" flag which is a float value between 0 and 1. Near 0 values favour notation simplicity. Near 1 values favours rhythmic precision.

• The **Save current config** and **Delete current config** functionalities have been improved and have a better compatibility with the Chant environment. Their behaviour has changed slightly : **Delete current config** does not erase the **User** menu instantly. Rather, it changes the file `Config.init` in order that the **User** menu will be empty the next time PatchWork is launched. Note that **Save current config** stores absolute library pathnames in `Config.init`. Thus, if you move PatchWork on the disk, or rename the disk, you MUST run **Delete current config**, then rebuild your library configuration by loading libraries, then run **Save current config**.

• The Chant Library has been recompiled in Think C and is much faster than the old one. In order to build a PW-Chant image, load the Chant library from your PatchWork image, then **Save Image** from the **PWoper** menu. This does not necessitate anymore the presence of MCL 2.0.1 on the disk.

• A new mechanism is being prototyped for auto-load example patches (command 't' in patch windows). When somebody designs a new library, he can put inside the library folder an example folder where the file `my-module.pw` for instance, will document the module "my-module". See the Chaos library for a model.

• The file `Chaos.lib` contains the Lisp instruction which informs the system that a new example directory is available ("`chaos-patches:chaos-tutorial`" here).