in

COLLABORATORS			
	TITLE : in		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 10, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 The Web for Music

==========

The Web for Music

==========

1993 October

Contents:

Concepts

Travelling the Web

Package Components

1.2 What and Why

What and Why
==========

'The Web' is a general environment for building up complex functions from simple elements, using a graphic 'block-diagram' metaphor. This version applies the scheme to the manipulation of MIDI events. (To be useful, it naturally needs a MIDI interface on your computer -- see

Prerequisites

.) The current set of modules is devoted exclusively to 'real time' performance (no recording or playback). In addition, 'System Exclusive' message sequences are at the moment just discarded. Many more modules -- to handle areas such as these -- are planned, but this is a first release!

You will probably be able to take best advantage of this package if you have some kind of multitimbral, multichannel MIDI setup. Development of these particular modules has undoubtedly been considerably affected by my own current equipment -- just a single YPR-20 Electronic Piano. This can handle up to twelve simultaneous notes (total) over up to five channels, each with its own voice; in multitimbral mode it seems to be quite intelligent about dropping early notes when it reaches its limit, rather than blocking new ones. On the other hand, it doesn't have such 'Synth' facilities as keyboard split.

With these modules I can nicely supplement the basic repertoire of my setup with things like keyboard split (multichannel if desired) and 'fattened' sounds -- adding layers, transposing notes, and delaying them. I can split a MIDI stream into multiple paths (perhaps segregated by key range or function) and do different operations on these, combining them at the end to drive the several channels of my instrument. Also, the 'Instrument' module that plays the Amiga's internal four sound channels -- with the collection of 8SVX instrument samples I have acquired -- makes a nice enhancement to my basic set of piano sounds. Although there are only a dozen types of element so far, multiple instances of them can be combined in quite complex ways.

As your setup will surely be quite different from mine, I suggest experimentation... Let me know if you find any really cool things you can do! (Also I'd be grateful for suggestions on other useful modules. There's a lot I want to do, but I'm sure I haven't thought of all the possibilities.)

A fuller outline of the concepts behind the Web is elsewhere in this

guide (
Concepts
), but briefly:

Using the Web's graphic interface. you can link instances of different types of elements into an overall configuration that performs complex operations on the data passing through it. The element functions may be quite limited in number, and individually fairly primitive, but are designed to be good building blocks for the more complex functions.

The 'Web' itself is a a configuration management program that gives the user the interactive 'block-diagram' interface; it manages all the various 'Modules', which are themselves individual -- and semi-independent -- programs that provide for all the actual handling of MIDI events. Modules usually have their own 'Control Panels' in addition to the Web interface itself. Note the distinction between 'Modules' and 'Elements': you only start up one instance of each module program, but this can manage as many elements of that type ('blocks' in the diagram) as you wish. Yet another component is the 'Traveller', which is responsible for carrying data between the elements; one Traveller process is created for each path (linear chain of elements) in the configuration.

Startup of all these components is normally done by a supplied script, invoked by an icon. Custom icons can be used to start with a specific configuration loaded (though you must create these yourself with the usual WorkBench tools).
Startup Script and Icons
For a quick introduction to running the MusicWeb, look at
Getting Started
.

1.3 Distribution

Distribution
==========

This suite of programs and its documentation is copyright, all rights reserved. It is ShareWare -- *not* Public Domain. If you find it useful, *please* show your appreciation of the effort that has gone into it by an appropriate donation -- at least $US35 is suggested.

This package is freely distributable, however, provided that it is kept intact as initially provided, and that no charge is made for its distribution aside from that necessary to cover duplication and other reasonable overheads.

Future developments will depend to a major extent on a positive reaction to this release, so please don't freeload...

On the other hand, if you feel that the package is not worth the suggested amount in its current form, let me know about that too. Any and all comments will be taken into account. (And donations of any amount will be gratefully accepted!)

Contact:

Pete Goodeve
3012 Deakin St #D
Berkeley, CA 94705
USA

phone: (510) 848-2092

Internet: pete@violet.Berkeley.edu

1.4 Getting Started

```
 Getting Started
=================
```

A full discussion of prerequisites and operation is in other sections of this guide, but if you want to try it out immediately, provided that you are running version 2.04 (or later) of the operating system and have the supplied package available with all its components in some directory of your system (if it's on floppy, having this in a drive will do), just follow these steps below. To do any more than look at it, you will of course also need a standard MIDI interface connected to the serial port of your machine. Depending on your setup, you may also have to match the MIDI channels, in some of the more complex example configurations, to your instrument.

Double-click on the 'Install Libs' icon. This will copy the required library file ('ppipc.library' -- size about 3K) to your 'LIBS:' directory. (This only has to be done once, the very first time you run. It will remain in place unless you rebuild your System disk or otherwise delete it.)

Double-click on 'MIDI DEMO'. This will start the Web Master and all the supplied modules, then load a trivial configuration. KeyofC Demo You may play with this as you desire, but you will want to look to the other sections of this Reference for further explanations. To bring in one of the other demo diagrams, first use the 'New' item (to clear the screen), and then 'Load Config', from the 'Project' menu; you will get a file requester from which you can choose another configuration.

For more on the Demo configurations, see: Guide to the Examples

To close down the Web when you are done, use the 'QUIT' item of the 'Project' menu. This shuts everything down, removing all components from memory (except the ppIPC library -- which obeys the usual automatic flushing behaviour).

1.5 Prerequisites

```
                Prerequisites
===============
```

The purpose of this package is to manipulate MIDI events, so one basic requirement is a MIDI interface box plugged into the Serial Port of your Amiga. Any of the available makes should work fine.

The Web program and associated modules need the 2.0 environment or later, and ppipc.library must be available in 'LIBS:'. In the package there is an icon ('Install Libs') that will copy this special library to LIBS: for you, or you may do it yourself. (It is in the 'Libs' subdirectory.)

The system should run on any Amiga. It will. however, not show its full speed potential on a basic 68000 machine (!). The minimum memory needed has not been directly checked, but it probably should run in 1MB.

The Web Music system comprises quite a number of component parts (see

```
                Package Components
```

) but if all directories are present as supplied and in their correct relationship, and you use the 'Run_WEB' icon (or the associated CLI script), everything should get started properly. The system should be runnable from floppy, hard disk, or RAMdisk.

A caution for those of you who habitually run Enforcer in the background: avoid the habit when using the MusicWeb! Not that you should get any hits -- by all means run it to verify that the Web is behaving properly, but expect a few missed and stuck notes while you are doing so. I don't know the reason, but I regard it as an incompatibility rather than a bug. (:-)) No other incompatibilities are known (except the obvious ones, such as other programs that would want access to the Serial Port, which will be locked out properly if they obey the rules).

1.6 Drawing Diagrams

```
 Drawing Diagrams
==================
```

The effect the Web has on MIDI data passing through it is determined by the configuration diagram cuurently built on the screen. This may have been read from an existing Configuration File, or you may construct it directly in the window. Whatever its source, it is never immutable: you can rebuild part or all of it at any time -- even while data is flowing [although in the present case it may be tough to do this, with one hand on the mouse while the other is playing your MIDI Keyboard...!]. Of course you can also save the current diagram and parameters to a file, for use at another time.

Configurations are built up from a limited set of basic elements. In general you can place as many elements of a desired type in the diagram as you wish. In the present system, only 'MIDI In' and 'MIDI Out' are

restricted to single instances. The placed elements are linked together in the sequence that you want to have the data follow; a linear chain of such elements forms a 'Path'. The head element of the path must be a 'Source' to provide the data that will flow. Paths can be branched off from other paths (forming an independent processing chain) in which case the branch point is itself a source for one of the two forks (arbitrarily chosen; the original path continues on down the other fork).

Different types of elements may have different connections allowed. The majority of them are 'Filters' at which data arrives, is processed, and is passed on. These have only one output connector, but inputs may be attached to any or all of the other three sides; all incoming data is merged onto the single output link. (If there is no output attached, that filter is the end of that path.) 'Sources' only have the output connector; data comes from elsewhere -- such as the Serial Port in the case of 'MIDI In'. A third category is 'Multiconnector': these are elements in which each connection has a distinct function; they may be both a Filter and a Source, have multiple outputs, or whatever the function decrees. Inputs cannot be merged at a Multiconnector element; this must be done, if desired, at an earlier point in the path.

Each element you place in a diagram is given a unique default name at that time (shown when you click on it in 'Select' mode, and in its control panel). If you're going to keep the diagram around, you will probably want to change it to something more mnemonic (again using 'Select').

For detailed instructions on how to build diagrams, refer to other sections, but you may be able to determine most of what you need to know by playing with the supplied demo configurations. As a single example of the innumerable possibilities, consider constructing a 'keyboard split' that feeds two different MIDI channels. The 'best' way to proceed is entirely a matter of personal preference: for example, you could place all the elements first, then link them in the desired way, or you could start by placing the ends of the main path (MIDI In and Out), linking these -- so you can check that your MIDI system is working -- and then elaborating into the desired configuration by inserting the required elements and branches. We'll take the first course here.

First click on the 'Place' button and select 'MIDI In' from the list on the right of the screen; click on the desired spot -- preferably to the left side -- to place that element. Select 'MIDI Out' and place it well to the right. Select 'Key Range' and place two instances, one above the other, a little to the right of 'MIDI In'; then place two instances of 'Set Channel' to the right of these. Now click on the 'output knob' of 'MIDI In'; you will see the mode switch automatically to 'Link'. Then click on the left side of the top 'Key Range' element; a link will be drawn between the two points. In the same way, link the output of that 'Key Range' to the input of the top 'Set Channel', and the output of that to 'MIDI Out' (use the top side of this, if you like). Then go back to the first link you made, and click somewhere along it: a marker to show the branch point should appear; link this to the lower 'Key Range' and then complete the rest of that path as before, ending up on a different side of 'MIDI Out'.

This completes the basic configuration, but you need some parameter settings to make it work. Change the mode to 'Param' by clicking on that button, then click on the first 'Key Range' element; its control panel

will pop up. Move the panel to some convenient part of the screen so it won't be obscured and click on the other 'Key Range' to get its control panel as well; you can open the 'Set Channel' panels at the same time if you like. Returning to the first 'Key Range', move its 'Low limit' slider to, say, 60 (middle C); this will allow it to pass middle C and all notes above. Set the 'High Limit' slider on the *other* 'Key Range' to 59 (all notes below middle C). Set the top 'Set Channel' to some MIDI channel recognized by your instrument, and the lower one to another channel. Now you should have your split keyboard.

From that point, you can elaborate it as you like, with still other parallel paths, maybe with delays in them and so on. However it is much easier to do it that to try to describe it it words, so I suggest you go ahead and play. Look at the supplied demo configurations for ideas.

1.7 Overview of Master Screen

Brief Overview of the Master Screen
==========

The buttons on the right represent the running modules, those at the bottom the operations that can be performed. There is a 'Project' menu that contains a fairly standard set of choices: 'New', 'Load Config', 'Save Config', 'About', and 'QUIT'.
Project Menu
(The 'Direction' Menu over on the right-hand side is not one you ↩ are
ever likely to need, but it can be used to change the orientation of future elements to be placed on the screen. (Elements cannot be rotated once placed.) There is also a 'Modules' menu which is also not particularly useful in the system as it stands.)
Other Menus
Each module appears as a button on the right-hand panel. Clicking ↩ on one
of these will select that module for relevant operations, such as 'Place' or 'Insert'. The name on the button is often not quite the same as the filename of the Module it represents (the program 'MIDILink' supplies both the 'MIDI In' and 'MIDI Out' buttons, for example). MIDI Modules

An operation mode selected from the lower buttons remains in effect until another is picked -- except for a couple, like 'Cancel', that are momentary (and the particular case of 'Place' and 'Link', which can flip automatically between each other according to where the user clicks on the diagram.). A description of each button's function is in
Control Buttons
Other windows will appear on the Web Master screen as well in the ↩ course of
a session. In 'Select' mode, clicking on a part of the diagram causes the appearance of one or two temporary panels: one gives information about the item that was selected, the other -- if present -- presents a selection of possible actions to be taken on that item. Most modules, too, have Parameter Control Panels, which can be displayed for an element by

selecting in ‘Param’ mode or using the ‘Param’ action in ‘Select’;
these will remain until explicitly closed.

1.8 Control Buttons and Actions

```
 The Control Buttons and their Actions
======================================
```

Place	When this button is engaged, an instance of the current module (determined by whichever of the right-hand panel buttons is selected) will -- if possible -- be placed in the main window at any point at which the (left-hand) mouse button is clicked. This is the default initial mode if no file was specified in the command line. If, after placing an element, you click on an output connector (see ‘Link’), the system will switch automatically into ‘Link’ mode. As described next, you can also switch back automatically into ‘Place’ mode.
Link	In this mode, elements (instances of modules) already placed on the screen may be connected into a chain by data paths. Click first on the "output connector" (small dot on the right hand side of the element, usually), and then on one of the sides other than the output of the element you wish to connect it to. If you click on intermediate (empty) points in the window between these two actions, the path will pass through those points as well. You may also begin a new path from an intermediate point on another (making a ‘Branch’); simply click first on the desired branch point rather than on an output connector. (This action will be refused if the ‘MIDIBranch’ program is not running.) If, instead of clicking on an output connector or an existing link to start a path, you click on an empty area of the window, the system will switch into ‘Place’ mode and an instance of the current element will be placed at that point.
Insert	This is similar to ‘Place’, except that the element can only be inserted into an already existing path. A single click at the desired point in the path will insert a new instance of the selected module there.
Param	In this mode, clicking on any element that has a control (‘Parameter’) panel will bring that panel up. This panel remains on the screen until explicitly closed. (The screen element the panel refers to will remain highlighted while the panel is visible.

Replace	In this mode, clicking on an element will, if *possible*, replace it with an instance of the currently selected module. It will fail if the two elements are not compatible (trying to replace a filter in a data path with a source, for example).
Elide	Is essentially the reverse of 'Insert'. Clicking on an element in a path will remove it but leave the path intact. It can, of course, only operate on elements that have exactly one input and one output connected. (Or, for convenience, on elements that aren't connected at all.)
Delete	Clicking on an element will remove that element *together* with any flow links currently connected to it (unless one of these links itself cannot be removed, in which case the diagram will not be changed).
CUT Link	Click on the flow you wish to remove. (For compatibility with earlier versions, you may instead click twice -- first on the connector at one end of the path then on the other.) Note that you cannot remove a link that leads to a Branch: you must delete one of the branch links first.
Restore Link	['Momentary action' button] *If* neither of the elements previously connected by the last link you cut has been moved or replaced in the meantime, clicking this button will restore the connection. (This button may be used at any time -- the current mode remains unchanged.)
Move	Click on an element you wish to move, then on the new position you wish it to have. Any data paths connected to the element will be suitably readjusted (to the program's satisfaction, at any rate! You may want to re-route them yourself). You can move branch points too.
Route	Allows you to specify points you wish a given existing path to be adjusted to follow. Click first on the output connector that is the source of the path, then on the points, in order, that you want it to go through, then finally on the (existing) input connection of that path. Note that you cannot change either of the connection points themselves with this command; this can only be done by cutting and re-Linking.
Select	In this mode, you can click on any part of the diagram to bring up an information window about that point; the window remains visible until you either select another item or click on some empty space (or 'CANCEL'). The contents of the information window depend on whether you have clicked on an element, a connecting link, a connection point, or a branch point. (The selected diagram element is highlighted.)

If the selected item is an element, its current name is shown within a string gadget, which you may edit as you desire; a unique default name is initially assigned by the program ('SRCn', 'FLTn', or 'ELEMn' as appropriate). (This name will be stored when a configuration file is saved. It is also passed to the modules for their own use -- they normally display that name in any associated control panels or other windows.)

If appropriate, a window of applicable buttons will also pop up; the button labels correspond to mode selections in the main control panel, but only affect the selected item.

Buttons available when a placed element is selected are some or all of:

'Param', 'Move', 'Elide', 'Delete'

Buttons for inapplicable operations are suppressed, except for 'Param' (because this is managed by the module itself).

A selected Link, can be 'Cut' (if this is possible). A selected BranchPoint can be 'Move'd.

For a connector on a normal source or filter, only an information panel appears, but if it is a multi-connector element, you will also get a panel of buttons designating the possible choices for that connection (with the current choice highlighted).

'Select' mode is engaged initially if a file was specified in the command line (as is actually the case with both demos).

CANCEL — This button will abort any uncompleted action, such as a Link, Move or Route. It will NOT undo any operation you have actually completed. (The program does not currently have an UNDO as such. Sorry.)

1.9 Project Menu

The Project Menu
==========

New — Clears any current configuration from the screen.

Load Config — Brings up a standard File Requester to select a previously saved Configuration File for loading. It does not clear the screen first; you will normally have to use 'New' if a configuration already is on-screen, unless you are loading

```
                    parameters only.
                    After loading a file, the mode is automatically
                    set to 'Select', because this is virtually always
                    what you want to do at that point.
                    (See
                 Saving and Loading Configurations
                    Save Config as...   Brings up a Standard File Requester to ↩
                       specify
                        a file into which the configuration is to be saved.
                        The resulting Configuration File contains full
                        information about all the elements in the diagram
                        -- position, connections and current parameters.
                        (See
                 Saving and Loading Configurations
                    Save Params...      Allows current control settings of  ↩
                       selected
                        elements to be saved.  When this item is selected,
                        a 'Done' window will appear; click on the elements
                        you wish to record, then on the 'Done' button to
                        write the file (a file requester will appear).
                        A second click on an element will remove it from
                        the set; clicking 'Cancel' (either in the window
                        or the main button) will abort the operation. The
                        file can be read in 'over' an existing diagram
                        (with matching elements) to change their settings.
                        (See
                 Saving and Loading Configurations
                    About Web       Version and Copyright information (same as  ↩
                       initial
                    panel).

    QUIT Web        Shuts down any running configuration and exits the
                    program.
```

1.10 Other Menus

```
 Other Menus
=============

The 'Set Dir'n' Menu

    UP              Determines the direction in which the output side
    DOWN            of placed elements will face.  The default is 'RIGHT'.
    LEFT            (You shouldn't need to bother with this.  Inserted
    RIGHT           elements for example automatically choose a suitable
                    direction.  You can't rotate an already placed item.)


The 'Modules' Menu

    Close Current   (At present the only entry) Removes the module
                    represented by the currently selected button in
                    the module window.  Although the button will
                    immediately disapper from the window, the module
                    program itself will not terminate until all the
```

active instances have also been deleted.
(This information is only here for completeness.
You are very unlikely to need this menu.)

1.11 Temporary Windows

```
 Temporary Windows
==================
```

If you move a temporary window (such as the message panel or the 'Select' information window) to another location while it is open, the next time it pops up it will again be at the new location. (This also mostly applies to windows managed by the modules themselves, such as parameter panels.)

1.12 Saving and Loading Configurations

```
                  Saving and Loading Configurations
===================================
```

A Configuration File is a text file that describes the sequence of steps needed to recreate a diagram, together with any parameter settings for individual module instances.

A configuration that is being read expects all the modules that it needs to be running. If the read operation encounters a request to place an element whose module is not yet running, it displays the message "Waiting for Module xxxx", and waits for this to register itself. If the module is not already on its way in, you have the choice of either giving an appropriate CLI command or icon-click to start that module, or aborting the whole load by clicking 'CANCEL'. The Brancher process is treated here just like any other module. (You should never have to worry about this if you start from one of the supplied icons, as these always initiate all the available modules.)

A Load operation will also abort if there is any conflict between the file being loaded and what is already on the screen. Such conflicts include overlapping an already existing element or having the same name as an existing one. You will want to clear the screen ('New' in the Project Menu) before reading a fresh configuration. (In the general case, it is possible for more than one configuration to be in the system at a time; however a MIDI diagram will usually have both 'MIDI In' and 'MIDI Out', and there can only be one of each of these, so individual configurations will *always* conflict.)

There is a particular type of Configuration file (produced by 'Save Params...', see below) that only contains parameter settings for elements already in the diagram. This of course has the reverse restriction to the above: an error will be flagged if elements of the specified names are *not* on screen already.

The 'Load' and 'Save As...' menu operations use a standard ASL File Requester for selection. You will be queried if you try to save a

file of the same name as an existing one: a panel will pop up to
give you the choice of overwriting, choosing a new name, or aborting.

Each invocation of the requester remembers the context from the previous
time. The Web command line option 'DIR' may be used to specify the initial
default directory for the requester. For this version, the initial default
is 'CONFIGURATIONS'. (This is set by the startup script; if you prefer a
different path, you can easily change it there.).
                CLI Command Line
               A subspecies of configuration file, generated by the 'Save Params ↩
                  ...'
menu item, contains only parameter settings for specific named elements.
The resulting file is a Configuration File without any position or
connection information -- just the parameters of the selected elements.
This is intended to be read when a diagram containing those elements
is already in place, to change the settings of some or all of them.

1.13 CLI Command Line

                CLI command line for the Web
==============================

The Web can be invoked from the CLI with command-line parameters.

The switch parameter 'LACE' (or 'INTERLACE' or '-i') sets Interlace mode
for the Web screen.

The parameter 'DIR dirname' sets the initial directory for configuration
files.

Any configuration files named in the line (optional keyword 'FILE') will be
read immediately on start up. (More than one name can appear, but there
will probably be conflicts between elements in the files unless all but
the first are just parameter settings.)

Two other keywords are recognized, but they are for more specialized use.
'TRAV' controls the number of Travellers that will be started for each
Source element; don't use it unless you are in an experimental frame
of mind. 'TPRI' set the task priority at which Traveller processes
will be started. The startup script for the Music Web uses this to
ensure that events get processed rapidly, but giving them the high
priority currently used lets you create a configuration that can lock
out user control of the machine. See
                Travelling the Web
                for
more details.

There are no tooltypes recognized by Web as yet. In most cases anyway you
will probably want to start the whole environment -- with its modules -- as
a unit, so a script is recommended ('Run_WEB' is supplied). Use Xicon or
iconX to run that script from the Workbench.
                Startup Script and Icons

1.14 Startup Script and Icons

```
 The Startup Script and Icons
==============================

Because so many components must be running for the Web to function,
the only reasonable way to start it up is with a command script.  You
will probably usually invoke this from an icon (via the Xicon tool
--or even IconX if you prefer), but it may equally well be run with
a Shell command.  It is a standard AmigaDOS script.

The standard script 'Run_WEB' is provided for normal use, but you
can modify it or create an alternative if your needs are different.
It will have to be extended for instance if new modules are added.
There is nothing particularly tricky about it, but there are perhaps
a few points that should be mentioned.

Three command-line parameters have been allowed for; these are passed on
to the Web itself.  This number is arbitrary but should be enough for most
use. Typical parameters you might want are 'LACE' (provided here in the
icon invocations) and an initial configuration file.

The first action of the script is to make the all-important 'Traveller'
resident, so that instances of it may be quickly activated as required.
Then the Web itself is run, with an initial directory specified, and
a Traveller priority setting of 30.  Note that this RUN command has
a continuation line; this of course will be executed when the Web
terminates, and is a short script ('End_Services') that cleans things
up by removing Traveller from residence and sending 'QUIT' messages to
the utilities that are not shut down automatically with the Web.

After this, the required modules are simply started in sequence.  The
only special action is taken for 'MIDILink' -- the MIDI interface itself
-- which is run at a high priority of 31.

The supplied icons use Xicon to invoke the Run_WEB script, because the
options available from that program are useful (for example the same
script can be used by multiple icons with suitable command parameters),
but IconX will also work if you prefer.  To ensure that it is available,
Xicon is included in the 'C' subdirectory of the package.  You can adjust
this if you already have it elsewhere.

You may want to use the Workbench menu item 'Icons>Information' to
examine or change the controlling ToolTypes of thes icons.  You may
need to look up Xicon documentation (not included, but widely available)
but briefly:

    'MODE=nowindow|noscript' means 1) don't open a window for console
    messages; and 2) don't look for an attached file as the script to run.
    (You should always include the 'noscript' part because of the next
    ToolType, but the 'nowindow' is optional.)

    'EXECUTE=run_WEB LACE' (or some variant) means perform the Shell
    command 'execute run_WEB LACE', thus invoking that script with that
    parameter.
```

The ‘MIDI DEMO’ icon simply adds the ‘KeyofC’ filename to the command line
to give you that as an initial screen. It also *does* provide a window
so that you can see things happening to some extent.

(The ‘Install Libs’ icon has a much more involved ToolTypes sequence
that will not be dealt with here.)

1.15 Concepts

```
                Concepts
==========
```

There are many situations in which a user would like to be able to
build up combinations of simple basic functions to perform more complex
operations on data. It would be very convenient to be able to put the
basic modules together and change them around "on the fly", even while
data is being processed.

The Web is a graphic environment that gives a user the ability to
do just this. This version is designed to work exclusively on MIDI
data, but with different sets of basic modules there are many other
areas to which it might be applied. With other modules incorporated,
the Web is also presently in use for Scientific Data Acquisition and
Processing.

Rather than being a single program, the Web should be thought of as
a configurable, easily extensible suite of programs for the integrated
management of operations on data. The one invariant is the Web Master
control program, giving the user a visual "flow diagram" representation of
the configuration that he or she can manipulate with ease -- even while
data is flowing.

The underlying concept is of a series of data processing elements,
beginning with a data ‘source’ -- here for example, ‘MIDI In’ -- and
subsequent ‘filters’ that act on the data. The elements are connected
by paths along which the data flows; each element takes some specific
action with respect to the data, then passes it along the path (perhaps
modified, perhaps not).

A path may be more elaborate than just beginning at a source and passing
through some filters in straight sequence. It may contain branches, where
the data flow is duplicated into separate streams. More than one stream
may converge on an element: filters can accept several input connections,
although the data output from a simple filter all flows on through a single
path. (‘MIDI Out’ is just another filter, with the restriction that there
can only be one in a configuration; multiple paths may converge on it,
however.)

In addition to the simple Sources and Filters exemplified by most of the
modules here, it is possible for a single element to have multiple input
and output connections, each with a different function. ‘MIDIDelay’ is
a module of this kind.

Naturally the system takes full advantage of the Amiga’s multitasking:
several flow paths can be working on their own data simultaneously.

On the other hand, the design is such as to avoid unnecessary task-switching; typically a single process handles most of the data manipulation algorithms that are chained into one path (see

Travelling the Web

). [It is probably worth pointing out that only the Amiga has this capability to handle real time data with multiple tasks so efficiently. This system is unlikely to be ported to any of its competitors...]

The data to be processed is purely the concern of the Modules 'plugged in' to the Web. The Master program itself has no interest in, or control over, the data format: it solely determines the paths over which the data will flow. The only inherent assumption is that the data can be maintained as 'packets' in some way, so that each 'station' on a path can work on a packet at a time. There is complete freedom as to what form a packet should take, and what may happen to it on its travels. Each piece of data that is flowing has an identifier specifying its nature that travels with it; quite different types of data can flow along the same path and be treated properly -- or ignored, as appropriate -- by the filters that they encounter.

Although the modules included here are directed toward a particular application, the system is designed to be adaptable to a wide range of needs. 'Source' or 'Sampler' modules can be written to acquire data from any available source that can connect to the Amiga; output of data to any suitable destination could also easily be arranged. The packet structure is also totally flexible, and can be designed to suit the purpose.

(You may by chance think you notice a slight kinship with the design of a certain other unique music environment for the Amiga. In fact, though, the more generalized block-diagram approach of the Web gives quite a different set of capabilities from those of B&P, and was truthfully not inspired by the other. [Though I'm not averse to adopting a few ideas...!] The Web principle was originally developed for general data management, in particular scientific data processing, and this application is a natural extension. On the other hand, because it is more general, it probably will never approach some of B&P's areas of excellence.)

1.16 Travelling the Web

```
 Travelling the Web
====================
```

In the 'Concepts' section, data was referred to as being 'passed' from element to element. In actual fact the data doesn't move. There is a separate process -- known as the 'Traveller' -- associated with each path that manages the packet, calling functions from the modules to operate on it in the sequence specified by the path itself. Any necessary changes are made to the data in place, without copying or moving it. For most Filters the Traveller calls the function directly, without any task switch. Where this isn't possible, the Traveller sends a message to the module to request the function.

There is (in the normal mode) one Traveller created for each Source
or Branch in the configuration, with that process serving all the other
elements chained into the path from that source.

The reason for this approach is of course speed. It is far faster than
any mechanism that would involve copying the data from element to element.
Where necessary, for example at branch points, data does still have to
be copied, but the need for this is minimized.

There are some consequences of the 'Traveller' concept that you should be
aware of, to avoid misconceptions. In the usual case there is just one
Traveller process for each complete, linear path. For this to 'circulate'
properly, the path must have a Source, where the data is placed in the
packet, and a Termination -- an element with no output connected serves
this purpose. The Traveller takes each new packet from the Source along
the complete path, visiting each element in turn, finally returning to
the Source when it finishes at the termination point. You can't have any
circularity in the path, because then the Traveller will simply keep going
forever, never returning to its Source for new data. Actually, things
aren't quite as bad as this, because the Traveller has a 'cut-out' that
will break a loop if it continues too long, but you can still get odd
effects when buffers in some of the modules get filled to overflowing.

If you do get such a loop circulating, you can just cut one of its links
to break it. Things may take a few seconds to come back to normal if
the output buffer has overflowed or something, but you should be able
to recover.

The 'circulating Traveller' problem does *not* apply to loops where
data is buffered in some element and transferred to another Traveller
to continue. The 'Delay' element does exactly this -- storing the
MIDI events internally before tossing them out again at a later time --
so if you loop from the 'Delayed' output of the element back to an
earlier point in the path, data will happily cycle around. Forever.
You do have the 'Cutout' button to use when you get tired of this,
but it still is probably not a particularly practical pastime. May
be fun to play with for a while, though. (The Delay element actually
works a little better if it runs at high priority too, but it is then
so efficient that it is easy to totally lock up a loop with it; if
you're willing to risk this, change the startup script suitably.)

1.17 Package Components

```
                  Package Components
====================
```

Whatever modules are in the set (Sources, Filters and so on) certain
program files must be available for the Web system to function. One
is the main 'Web' program itself, which is normally started at the
beginning of a session and remains running until the end. Then there is
the 'Traveller' program, which performs all the data-transfers along the
links of the Web, and must be available -- either as a resident module, or
in the path for the CLI from which Web was invoked -- while Web is running.
Things will start up faster -- and memory consumption will be *much* less --

if it is made resident; one Traveller process is started (automatically)
for each Source element in the Web. Thirdly a program is required to
manage the forking of data at branch points; this should be started when
other modules are, before any configuration is built or loaded; a custom
version is used here ('MIDIBranch') to handle the demanding speed
requirements of MIDI events. Underlying the whole thing of course is the
'ppIPC' protocol (ppipc.library), details of which can be found elsewhere
(Fish 290 or on amiNet).

For details of the modules refer to The MIDI Modules .

The Web program should be started first, followed by the brancher and
all the modules; the order of these is unimportant, except that it will
be reflected in the ordering of the buttons on the Master Screen. The
Traveller need not be started, provided it is present: the Web invokes
these processes as needed.

When the Web Master is eventually shut down, all the Module and Traveller
processes also terminate. If the Traveller module has been made resident
(and you don't use the standard scripts, which remove it at the end) you
may clear it from memory with the usual 'RESIDENT traveller REMOVE'.

The standard (shell) script 'run_Web' (used by all the icons) performs
all these required startups properly, and, when the Web is terminated,
calls a sub-script 'End_Services' to remove the Traveller from residence
and otherwise tidy up. This is the most convenient way of starting the
system, but the components can be started directly and individually if
preferred, provided that the sequence of startup is observed.
                scripts
               A few example configuration files are included in the package (↩
                  directory
'CONFIGURATIONS') but you will no doubt quickly want to create others
suited to your needs.

Some convenience icons (with Xicon as their Tool) are also present.
'Install Libs' should be used once initially to place 'ppipc.library'
in your LIBS: directory (unless you have done this yourself).
Don't be confused when you look in the main directory: the executable
icons (.info files) don't necessarily have an associated program or
data file of the same name; Xicon allows commands and script files
to be invoked directly from ToolTypes within the icon.

'MIDI DEMO' runs the system (via startup script) and loads a simple
configuration, so you can try things out; you can try others in the
set via the 'New' and 'Load Config' menu items. 'Run_WEB' just invokes
the standard startup script without any initial configuration: load
the one you want, or build your own.

By the way, unlike the Web in the Scientific Data Demo distributed
a while back, this is not a 'limited' version (you can save configurations
for example). However, it *is* restricted to use only with the MIDI
modules: it will not work with the others.