Introduction to
Pointer Environment
&
Menu Extensions

Joachim & Nathan Van der Auwera
PROGS
PROfessional & Graphical Software © 1993

February 21, 1993

Contents

Chapter 1

Introduction

1.1 Motivation

We have written this manual as a seperate part because there are people who have already used programs which run under the Pointer Environment and Menu Extensions. These people don’t have to read this manual as they will probably know it already.

The Pointer Environment and Menu Extensions are the way ahead for the QL. Most of the better new software for the QL uses it. or at least is 100% compatible with it. The Pointer Environment has the advantage of offering true multitasking and gives you the possibility to write pointer driven software, so you can use a mouse if you want to, with easy menus which can be accessed by a key or with the pointer. Using the Pointer Environment in a program interface has the advantage of being user-friendly. You can easily get used to it. and once you know the Pointer Environment, you know how to operate any program which us it even before you open the package.

The Menu Extensions are actually just an extension to the Pointer Environment. to give standard windows to some common menus. The most important one is definitely File Select, which makes sure that you can select a file to load or merge with a directory, ao that you don’t have to remember any filenames.

1.2 Loading

The Pointer Environment and Menu Extensions are a set of resident extensions. This means that these routines should be loaded at start-up (power on or after a reset), and that they should remain in memory until the power is switched off or until the next reset. This also means that these extensions should be loaded into RESPR area. You can load the Pointer Environment and Menu Extensions by including the next lines somewhere at the start of your boot file.

```
base=RESPR(14534) : LBYTES flp1_ptr_gen,base   : CALL base
haae=RESPR(10360) : LBYTES flp1_wman,base      : CALL base
base=RESPR(11684) : LBYTES flpl_hot_rext,hase  : CALL base
base=RESPR(22432) : LBYTES flpl_menu_rext,base : CALL base
HOT_GO
```

Please note that the lengths of the files may be different, this depends on which version you use. If you have Toolkit II you can also use these lines.

```
LRESPR flpl_ptr_gen   : REMark load the Pointer Interface
LRESPR flpl_wman      : REMark load the Window MANager
LRESPR flpl_hot_rext  : REMark load Hotkey System II
LRESPR flp1_menu_rext : REMark load Menu Extensions
HOT_GO
```

This is actually easier as you don’t have to worry about file lengths.

The HOT_GO command should be included to make sure that you don’t loose the last line restore provided by toolkit 2 (alt_enter).

Please note that some of the files have to be loaded in a distinct order. You have to load wman after ptr_gen, and you have to load menu_rext after hot_rext. It is also possible that some other resident extensions have to be loaded before all these, in particular extension which change the screen driver like Lightning or SpeedScreen.

1.3 Concepts

window A window is a part of the screen. It can usually be recognised because there is a border around it. A job (=a running program) can have many windows in it, but all these windows must lie inside the *outline.* In most Pointer Environment programs the outline can be recognised by the shadow around it.

pointer Most Pointer Environment programs have a pointer. The pointer is the thing which moves when you move your mouse. If you don't have a mouse, it usually is the thing which moves when you press the cursor keys, but not always. The pointer may have the same shape and behaviour as the cursor, but a cursor can't be moved with a mouse. the pointer has the general advantage that it *can* have any desired shape and size. it is also possible to change the chape over time, thus creating something that. looks like a cursor, or a walking person or anything you want.

Most application programs don't allow the user to set the shape of the pointer, but they often change it themselves to show you what kind of operations the pointer can be used for at a given moment.

item An item is a part of the window. It usually contains some text or a small drawing. An item can be recognised because a border appears around it when the pointer is on it. This border is removed when the pointer is moved to another part of the screen.

Items can have three distinct statusses.

available	An item is available when it can be indicated by a hit or do, but isn't yet. Such items can be recognised because they usually fit into the general look of the window.
selected	An item can also be selected, which means that the accompanying action will be taken later or is going on. Selected items can be recognised because they are highlighted in some way.
unavailable	The last possible status is that an item can't be selected. This can be because a certain action can't be executed when the program is in a certain state of operation, or because the action isn't included in the program yet or similar. Unavailable items can be recognised because they usually aren't 100% clear.

hit Two of the most important types of input to a Pointer Environment program are a hit and a do. They are both ways to indicate items in a window. A hit is cause by pressing <SPACE> or the left mousekey. A hit changes the status of an item from available to selected or vice versa. Some items are immediatly invoked when hit (e.g. a sub-menu), but this depends on the nature of the item.

do A do is quite similar to a hit. except that it always changes the status of the indicated item to selected. Usually[1] a do also invokes the item. This usually makes a do equivalent to a hit on the item and a hit on a "DO" item in the window[2].

You can also invoke a do when you press **<ENTER>** or the right mousekey when the pointer is not on any item.

underline Most items in a Pointer Environment program can also be selected by another keypress than a hit or do. and this has the advantage that the pointer doesn't have to be on the item. These keys can usually be recognised because they are visualised with a little line under the letter in the command. For some items this is impossible. Those items often have an indication of the key which has to be pressed just in front or above them. This is mainly the case for items which can be calh'd by pressing the function keys. This method of selection is equal to a hit.

[1] *not all items support this, it depends on the programmer(s), in fact some items even treat a "do" exactly the same as a hit*

[2] *in some cases there is no "DO" item, but only an "Esc" or "OK" item. In these cases a do usually invokes the "ESC" reps. "OK" item.*

Chapter 2

Pointer Environment

2.1 Pointer Interface

This is the part of the Pointer Environment that makes sure that multitasking works as it should, making sure that a window is completely visible when printing or drawing in it. This part also makn sure that the pointer actually exists.

The pointer interface is contained in the ptr_gen file.

It is the Pointer Interface that controls the proper handling of jobs (programs) and their windows. There are several ways to switch jobs. The first - and traditional - method is by pressing **<CONTROL-C>,** and thus running through all available jobs. The other method - which only works if the program which is currently using the screen doesn't cover the entire area - is by moving the pointer to a program which is partly visible. This program can then be selected by a hit or do. If you select with a do then a wake is also executed (if it exists. see later). If the program you want to switch to is not partly visible, then you will have to use **<CONTROL-C>** or a special program (e.g. Pick in QPAC II).

2.2 Window MANager

The Window Manager is the most visible part of the Pointer Environment. This is what allows programmers to make their programs to look and feel like any other Window Manager program. This common look and feel is even stronger because most Pointer Environment programs also use the *same* colours for the various parts of the menus and the Menu Extensions.

2.2.1 General

Most menus are built from items and information objects. Information objects are there just to give the user some extra guidelines on how to use the program or some extra information like the position on a page, the name of the file which is edited or similar.

There are also (a lot of) items. Some stand on their own (loose items) and some may be grouped (in an application window). When items are grouped, it is possible that there are too many items to fit in the part of the window which is reserved for them. In that case the application window will become serollable or pannable (or both). There wili appear some arrows at the border of the application window. When you hit one of these arrow items, the window will be panned or scrolled by one item. If you "do" on such an item, the window will be panned or scrolled by as many items as fit in the window (minus one).

It is also possible that there is a pan or scroll bar apart from these arrows. Such a bar tries to indicate what part. of all the items is visible in the application window. By hitting somewhere in the bar, you can indicate which part should be made visible.

2.2.2 Standard keys & items

One of the main reasons why programs look and feel the same is because so much is done in the same way. For this, many standard operations have a common item.

move This item is use to move the entire window to another place on the screen.
The pointer will then get the same shape as the move item. You can now move the pointer and press <SPACE>, <ENTER> or any of the mousekeys. The window will then get the same relative movement as you gave the pointer. This command can be cancelled by pressing <ESC>.

Move can also be executed by pressing <CONTROL-F4>.

size This item can be used to change the size of the window. Depending on the program this item can be handled in two ways. ‘[he window can hr resized immediately. This usually happens when the program only has a couple of distinct sizes it can handle. In the other case the window usually has a variable size Then the pointer will get the shape of the item and the relative movement of the pointer indicates the change of size for the window. Please note that the bottom-right corner of the window remains in place. So moving the pointer to the left or top will increase the size of the window. Moving the pointer to the right or to the bottom will decrease the size of the window.

Size can also be executed by pressing <C0NTROL-F3>.

sleep This item can be use to make your job sleep. This means that your program will change appearance, becoming much smaller. This has two advantages: the job will use less memory, and there will be more space on your desktop (screen). The new appearance is called a *button.* it is a small menu which only contains the program name. If you hit on the item then nothing will happen (or if you don’t have QPAC II, you will probably be able to move the button).

When you do on the item, then the program should return to the same status as before you put the program to sleep.

Sleep can also be executed by pressing <CONTROL-F1>.

wake This item, which is not available in many programs, give you the option to of giving the program the option to refresh the screen. For instance, the directory will be re-read when you are in a window which shows a directory. Wake can also be executed by pressing <CONTROL-F2>.’

2.3. HOTKEY SYSTEM II

2.3 Hotkey System II

This part of the Pointer Environment implements the THING system. it is not important to know much about this unless you are an expert user. All you have to know is that it allows you to load general extension routines so that they can be accessed by any program. This is used for (amongst others) the Menu Extensions, the DATAdesign engine, some device drivers in the SMS2 system,

Hotkey System II also replaces the <ALT-ENTER> and altkey commands in toolkit 2 and adds some new ones. We will not give more details about all these options, as they are mostly used by people who already know the Pointer Environment quite well.

2.4 Config

This is a utility program, which is often used in combination with Pointer Environment programs to allow you to set some defaults in the program. This program can be called with a line like

```
      EXEC flp1_Config
```

The program is quite straightforward to use. It will first prompt for the name of the program you wish to configure. it will then ask you what you want to configure and the new values. These new values can usually be set by typing the new value, or by cycling through all the values with any key and confirming with <ENTER>. The program will clearly tell you what to do and you just have to follow the given guidelines.

Chapter 3

Menu Extensions

This part of this documentation is mainly written by Jochen Merz as a user guide to the Menu Extensions.

3.1 File Select

The **FILE SELECT** window is always shown when the user is required to enter or select a filename. Here you can enter the filename either directly or edit a suggested one by selecting the menu option directly beneath the request.

Beneath this are two menu options with which you can recall the contents of the HOTKEY buffer and all previous contents. Just select the menu option and the contents will he written to the “suggested” area- Confirm with OK and the input will be accepted by the system. You can also edit the name, of course. The rest of the window concerns the current drive.

Depending on the size of the window, one or two sub—windows are shown - If you only see one sub-window this will contain the filenames and sub-directories. If you see two windows, the right hand (larger) on will only show the filenames, the smaller one on the left only the subdirectories. In these windows all the files are sorted alphabetically. The files will be taken from the current drive and must all have the correct ending (if any). The endings for sub-directories are ignored, because subdirectories don’t really have endings Now you can edit the drive and/or the ending.

If you press <ENTER> at the directory menu option, a further window is overlaid, from which you can select pre-defined devices and sub-directories. If you just select a directory, the list of files will be updated. You can also “update” it with a wake.

If you press <ENTER> at the endings menu option, this will be deleted if it wsan’t already empty. This is easier and faster than having to select it and then deleting the four characters, If it was already empty, then *a* window overlay will show some suggested endings.

Above the current directory is a list of available devices, e.g. MDV or FLP**.** There are also drive numbers from 1 to 4 listed. To select RAM1, press <R> and 1, and the directory window displays RAM1_

Behind the directory name you will see an arrow “←“. By selecting this option you can retrace a step back along the subdirectory tree without having to edit anything. So if you’re in directory fl1_pau_texts and select this once, then you get to flp1_paul_. The next time you select it, you get flp1_. If the current drive has true subdirectories (e.g. Miracle’s Harddisk or the QL Emulator drivers) then you’ll find the subdirectories of the file names marked with a “→“. As already mentioned. subdirectories are always listed, the endings don’t have to correspond. If you select such a subdirectory, then you’ll get in it”, i.e. the name will be taken over for the direclory and the file list read in again.

But to get, back to the list of files: you can select any file you like. <SPACE> accepts the name as “suggested”. <ENTER> takes the name and carries out an OK automatically. If the window is too small to show all the suitable files, the normal scroll arrows will appear in order to scroll the next batch of names up the screen. You can also select files or directories by pressing the character which is in front of the name.

At the right you will see a scrolling bar. Move to this area, press <SPACE> and the area will be shown relative to the total area. Press <ENTER> and the window will split, enabling you to control the two parts indepently from each other. Move to the split, and press <ENTER> to join the window together again.

You can also preselect the eight, different subdirectories suggestions in the Directory Select menu. Make the necessary changes to the menusext file with Config.

If the program which calls the File Select extension has a window which is not big enough to show the File Select window, then the filename will be prompted with the Read String extension (see below).

3.2 Item Select

When you see a window with one to three menu option. you can make your selection by pressing <SPACE> or <ENTER>. You can also press the first letter of the option.

3.3 Read String

You are asked to enter a string or filename. Under certain circumstances you may be offered a suggested name. You can either press <ENTER> to accept the suggestion, edit it as usual using the cursor keys or just enter a new string.

3.4 List select

This window is in fact very similar with Item Select. The only differences are that there can be more than three items in the window, and that the window can be scrollable if not all items fit in the window.

3.5 Report Error

The only thing you can do here is indicate the “OK” item to show that you’ve taken notice of the reported error.

3.6 View file

This window enables you to view a tile. You can setoll one line with a “hit” in the view-window. You can scroll a page with a “do”. Waking the window lets you start again by viewing the file from the beginning. Selecting WRAP causes any line, which exceeds the permitted width, to hr continued on the next line, preceded by a →
.

3.7 Button

If a program is in Button mode, then you can wake this up by moving to the button area and pressing <ENTER>. If the button is not positioned inside the button frame, you can move the button by using <SPACE>.