

mathwise

Courseware Management System
Version 2.1



Author's Guide

Contents

Chapter 1	<i>Introduction to MathWise v2</i>	3
	An Overview of the MathWise System	3
	The Components of MathWise v2	3
Chapter 2	<i>Installation of MathWise</i>	5
	[1] Create a Directory Structure	5
	[2] Copy the Files	5
	[3] Edit the Initialisation File	5
	[4] Add Mathwise to your Windows' System	8
	[5] Run Mathwise	8
Chapter 3	<i>Indexing Your Module</i>	9
	The Index Entry / Descriptor	9
	Building the Index File	11
	Using the Index Editor	11
	Using the Index Convertor	12
	Adding the Index File to the System	12
Chapter 4	<i>Navigation</i>	14
	Navigation Within Items	14
	Navigation Between Items and MWGoDesc ()	14
	Variants on MWGoDesc ()	17
	Navigation In Practice	18
	Returning From Items	18
Chapter 5	<i>Palettes</i>	19
	Using the Palette Editor	19
Appendix 1	<i>Using Mathwise v1 Material with Mathwise v2</i>	21
Appendix 2	<i>What Happens When Mathwise is Started and Closed</i>	22
Appendix 3	<i>Customisation Using mathwise.exe Command Line Options</i>	23
Appendix 4	<i>Miscellaneous Mathwise functions</i>	24
Appendix 5	<i>Mathwise Initialisation Files</i>	27
Appendix 6	<i>Windows NT Precautions</i>	32

Chapter 1

Introduction to Mathwise v2

An Overview of the Mathwise System

Mathwise v2.0 is an open courseware management system for Windows. By the use of **index files** which describe and define courseware items it is able to register, organise and run material produced using any Windows based authoring system¹. The indexing protocol used is simple enough to permit authors and system administrators to quickly apprehend and utilise the system, but sophisticated enough to allow some powerful and comprehensive organisational structures to be imposed on courseware material and to enable a wide variety of search and match methods to be used in locating material. In addition Mathwise provides a customisable floating **resource palette** which enables rapid access to system-wide functions and utilities and also module specific functions. Control over memory use is available and the system is customisable in a number of other important regards to allow system administrators as much flexibility as possible in setting up a Mathwise installation for their specific purposes. For authors and programmers a comprehensive set of functions is available that allow sophisticated control and interaction between courseware items.

The Components of Mathwise v2

The main components of the Mathwise system are as follows.

- **Mathwise Courseware Manager**

This is the functional core of Mathwise that runs and controls the navigation between courseware items, as well as providing various utilities such as the Mathwise palette. The Courseware Manager takes the form of a Windows dynamic link library (DLL) called `mw.dll`.

- **Mathwise Launcher**

This is a small program called `mathwise.exe` which the user runs to start the Mathwise system. Since the Courseware Manager is in the form of a DLL it is possible for any program which links to it to start the system. However, this requires the start-up program to issue a few commands to the DLL in order to properly initialise it. This requires some programming skill and a full understanding of the Mathwise system. Consequently `mathwise.exe` is provided to carry out these essential tasks. In addition to initialising the system it tells the Courseware Manager which courseware item to run and use as the entry point into the system (by default the standard entry-unit `entry.exe`) and it displays a user-defined bitmap as a 'welcome-style' splashscreen until the system is properly loaded. It also permits a number of command-line options which allow multiple system set-ups to be run from different icons providing a powerful but simple means for system administrators to customise Mathwise to their particular requirements (see *Appendix 3*).

- **Index Files**

These are the essential link between the actual courseware items and the Mathwise system. An index file contains the requisite information that the courseware manager uses to organise, search, locate and run particular courseware items. Index files are recognised by the suffix `.mw` and are produced and modified using the Index Editor.

- **Entry-unit**

This acts as the central, on-screen user-interface to the system. It is from here that users can select which modules they wish to study, browse through the leaflets associated with the modules, search for material by keyword and find out basic information about the Mathwise system. Mathwise is provided with a default entry-unit called `entryu.exe`, but the system allows for this to be replaced by a user-supplied entry-unit if desired.

- **Mathwise Initialisation File**

This is a standard Windows INI file which, in addition to providing certain basic initialisation parameters for the system, tells the Courseware Manager where to find index files. The default initialisation file is called `mw.ini`, but it is possible to override this in order to permit a variety of customisable system set-ups.

¹ This includes not only explicit courseware authoring systems such as Multimedia Toolbook, Authorware and Director, as well as future authoring packages, but any material that can be run or viewed with a 'reader' program such as Windows Help files, Word documents, Excel spreadsheets, Media Player files etc..

- **Mathwise Index Editor**

This is a utility program called `mwindex.exe` which is used to create and edit Mathwise index files.

- **Mathwise Index Converter**

Mathwise index files may also be written in ASCII form using a text editor. This utility, called `indxconv.exe`, is used to convert from ASCII to the `.mwx` index files that the courseware manager can read.

- **Mathwise Palette Editor**

This is a utility program called `mwpaled.exe` which is provided to allow the creation and modification of Mathwise palette files.

- **Mathwise Customiser**

This utility program, `mwcustom.exe`, is intended to simplify the creation of multiple Mathwise set-ups. It can be used to select different sets of courseware material appropriate to different years or courses for example, and furthermore can automatically create either Program Manager items (for Windows 3.1) or Start Menu items (for Windows 95/NT) to run the different set-ups it creates.

- **Palette Files**

These files define the form of the Mathwise palette. A system palette defined by `mw.mwp` is the default but this may be overridden for particular modules. Palette files are generated and edited using the Palette Editor and are recognised by the extension `mwp`.

- **Mathwise Toolbook System-books**

One Toolbook system-book, called `ukmcc.sbk`, provides the interface between Toolbook v1.53 books and the Courseware Manager, as well as providing many basic handlers and functions for Mathwise style books. It is also used to allow material authored in Toolbook under v1 of Mathwise to work with v2 (see *Appendix 1*). There are also a number of other system books such as `ukmcc_.tbk`, which provides authoring tools and `tkstompn.sbk` which provides graphics facilities. There are also versions of these system books for Toolbook v3 and Toolbook v4.

Chapter 2

Installation of Mathwise

Installation of Mathwise is straightforward. If a custom installer has been supplied with your copy of Mathwise then run it now. If not then the steps below should make the manual process clear. Note that more sophisticated set-ups are possible to allow for example for multiple configurations of courseware modules.

The installation of Mathwise to a network is much the same as to a standalone machine. The one aspect that must be paid close attention to is that any drives, directories and paths that have to be set in the Mathwise initialisation file (see section [3] below) are those as seen from a client machine, not the server. There are also special precautions that should be observed when installing Mathwise to a Windows NT network which are outlined in *Appendix 6*.

[1] Create a Directory Structure

All the Mathwise system files should reside in the same directory. We recommend that the module material and any ancillary support material, such as runtime Toolbook and Authorware should reside in the same directory as the Mathwise system, although there is no absolute requirement for this. Here is an example directory structure :

```
C:\
|
|--MATHWISE <== Mathwise system files and all runtime files in here
|   |
|   |--MODULE1 <== MODULE1 module and index files in here
|   |--MODULE2 <== MODULE2 module and index files in here
|   |--MODULE3 <== MODULE3 module and index files in here
|   |--UTILS <= Other utility files in here
```

If you wish to run the material directly from the CD-ROM then that is acceptable, however we would recommend that for best performance the system files at least should be copied to a local hard disk. As section [3] in this chapter makes clear, you can organise your material in more or less any fashion you choose, as long as you correctly edit the paths/locations of material in the Mathwise initialisation file.

[2] Copy the Files into the Appropriate Directories

The Mathwise system files are those contained in the root directory of the CD-ROM. If you intend to install the system on a hard drive (recommended) then simply copy the contents of this root directory into your chosen Mathwise system directory (`C:\MATHWISE` in the above example).

[3] Edit the Initialisation File

The initialisation file `mw.ini` contains a number of essential entries that you will probably need to edit to allow the system to work properly as well as some optional settings that you might want to alter to suit your setup. To edit the `mw.ini` file simply open it using a text-editor such as Windows Notepad. Alternatively you can run the Mathwise Customiser (`mwcustom.exe`) to do this using a friendlier interface. For full information on the contents of the `mw.ini` file you should read *Appendix 5*.

If Mathwise is being installed on a network then you should note that any paths that are specified in the various section of the initialisation file are those as seen from a client machine. This only really applies to paths that are on a drive other than that on which the Mathwise system is located.

Firstly we will deal with the essential initialisation entries.

- **Runtime Associations** The `[associations]` section of `mw.ini` contains the

information Mathwise requires to decide how to run a particular courseware item. It does this by associating the file-suffix (e.g. .tbk for Toolbook files, .hlp for Windows Help files etc.) with the runtime program or **reader** required to run it. It is this section of `mw.ini` that tells Mathwise about these associations. The format is simple. Each association requires three entries in this section, numbered consecutively from 1. The first entry `AssocXName` (where X is the association number) should be set to a description of the association. The second entry `AssocXPath` tells Mathwise the file name and location of the reader required to run the material. If this reader application resides in sub-directories below the Mathwise directory (as is recommended) then a full path is not required. Neither is it required for reader applications that are on the path, as would be the case for `winhelp.exe` for example. None may also be entered here, although this only makes sense for 'self-running' courseware items, i.e. .exe files. The final association entry `AssocXExt` simply gives the suffix for the files that require this reader application to run. Here is an example [associations] section for a system that will be running Toolbook, Windows Help, Executables and Winword courseware material:

```
[associations]
Assoc1Name=Toolbook
Assoc1Path=tbbook.exe
      ; In the Mathwise system directory
Assoc1Ext=tbk
Assoc2Name=Windows' Help
Assoc2Path=winhelp.exe
      ; In Windows directory so on system path
Assoc2Ext=hlp
Assoc3Name=Winword files
Assoc3Path=c:\word\winword.exe
      ; Not in sub-dir or on system path so full path needed
Assoc3Ext=doc
Assoc4Name=Executable
Assoc4Path=none
Assoc4Ext=exe
```

There is an optional fourth entry that you may require to use. This takes the form `AssocXTrack` and may be set to 0 or 1. If it is omitted a value of 1 is implicitly assumed (as for all the associations in the example above). Setting this entry to 0 for a given association will turn off Mathwise's attempts to track any courseware items using this association i.e. they become 'launch and forget' items as far as Mathwise is concerned. We have found that for a few applications² the tracking procedures Mathwise uses can cause problems, this entry allows those problems to be avoided at the cost of losing control over associated items after they are launched.

- **Module References**

The [modules] section of `mw.ini` is where you inform Mathwise what courseware material you wish it to run. This is done simply by inserting references to the index files that describe that material. The path requirements for these references work in the same way as for the associations described above. For example the [modules] entries in `mw.ini` for a system with just two modules, that also requires to run Excel, may look like this:

```
[modules]
EntryUnit=ent_unit.mwx
      ; In Mathwise system-dir so no path needed
ModuleA=moda\modulea.mwx
      ; Module A is in sub-dir
ModuleALeafs=moda\modulea1.mwx
      ; Module A's leaflets
ModuleB=modb\module2.mwx
ModuleBLeafs=modb\moduleb1.mwx
Excel=c:\excel\excel.mwx
      ; Not in sub-dir or on system path so full path needed
```

² These have all been Microsoft applications to date: Windows Help system in Windows NT, and Internet Explorer v3 and later.

For further information on this see *Adding the Index File to the System* in the *Indexing Your Module* chapter later in this manual.

- **Palette References** The information required to define a palette is contained within `.mwp` palette files. The system default (as defined in `mw.mwp`) will run automatically unless it is overridden for specific modules. This is done by defining a new palette file, placing it in the relevant module directory and referring to it in this section. For full details on how to produce override palettes see *Chapter 5*. As an example, say Module A in our example above requires an override palette as defined in `moda.mwp`. In this instance the `[palettes]` section of `mw.ini` would look like this:

```
[palettes]
ModuleA=moda\moda.mwp
```

Now we shall deal with the optional initialisation entries. These all occur in the `[general]` section of `mw.ini`. You are unlikely to need to alter most of these at this stage. Full explanations of these entries are given in *Appendix 5*. A brief overview of each entry follows here.

StartupDesc This should be set to the descriptor query (see *Navigation Between Items* in the *Navigation* chapter later in this manual) appropriate for defining which courseware item you wish a Mathwise session to start with. In most instances this will be the Mathwise standard entry-unit in which case the following entry would be seen:
`StartupDesc=EUNT,*,*`

StartupSplash This locates a Windows bitmap (`.bmp`) file that will override the default Mathwise splashscreen that displays while the system is initialising. The logo of your institution could be appropriate here e.g. `StartupSplash=my_logo.bmp`

EUAuthorsDesc This is set to a descriptor query for the item that is run when the About the Authors button in the standard entry-unit is clicked.

EUWelcomeDesc This is set to a descriptor query for the item that is run when the Using Mathwise button in the standard entry-unit is clicked.

MaxSimultaneousItems This enables control over memory usage. This is unimportant on modern machines but for institutions with older equipment, perhaps relying on disk swapping for virtual memory, problems, especially of speed, can easily arise when multiple items are open simultaneously. This setting allows you to control the number of courseware 'levels' that co-exist in memory at any one time. The value to which this item is set may range between 2 and 5. Once the specified limit is exceeded items are temporarily closed, thus freeing memory, and they are reopened at the point at which they were left, when it is necessary to return to them. This process is transparent to the user and can result in substantial speed improvements on systems where memory is at a premium. However, on powerful systems limiting the number of simultaneous items can have the opposite effect - longer time delays while items are reloaded unnecessarily when they could have just remained in the ample memory available. Consequently some experimentation is advised in order to determine if you need to use this feature at all (a setting of 5 will effectively disable it), and if you do then at what level to set the limit.

ErrorMessageLevel This controls the complexity of any error messages that the system displays. The possible settings are: 0 = no messages, 1 = student level, 2 = admin level.

ConfirmDuplicateItems Ignored.

- **Font References** The `[fonts]` section of `mw.ini` contains references to any custom TrueType fonts you may wish to use. These fonts will be temporarily installed into Windows for the duration of the Mathwise session. Mathwise uses three custom fonts as standard so the `[fonts]` section looks like this by default:

```
[fonts]
```

```
font1=tkmath.fot  
font2=ukmcsb.fot  
font3=ukmcsp.fot
```

Simply add your own fonts after these, numbering additional entries sequentially.

[4] Add Mathwise to your Windows System

Depending on the version of Windows you are running you should now add Mathwise to your Program Manager (for Windows 3.1) or add the appropriate short-cut to the Start Menu or Desktop (for Windows 95/NT) following the instructions supplied with your Windows system if in doubt how to do this. In either case it is the `mathwise.exe` program that you should run to enter Mathwise. For an authoring installation you should also add the four author utilities - Index Editor (`mwindex.exe`), Palette Editor (`mwpaled.exe`), Customiser (`mwcustom.exe`) and the Index Convertor (`indexconv.exe`).

It is important to the smooth running of Mathwise that you take care when creating Program Manager items / Win32 shortcuts that you specify a working directory for the former or 'starts in' directory for the latter. This applies particularly for network installations.

[5] Run Mathwise

Assuming the preceding steps have been correctly followed then you should now be able to run Mathwise by the method appropriate to your Windows version.

Chapter 3

Indexing Your Module

Index files (with suffix `.mw \times`) are at the heart of the Mathwise system. It is these that the Courseware Manager reads in order to construct the basic navigational 'architecture' within which all the components of Mathwise can be located. The indexing system allows great flexibility in the cross-referencing of Courseware items and provides powerful searching tools to utilise this to the full. However, as an author you do not need to understand the indexing system in anything other than its basic detail in order to properly index your module. This chapter explains the essential elements of the index entries of which index files are comprised, how to construct index files for your module and how to add your index (and hence module) to the Mathwise system. Throughout we shall use a sample module – M4BASE *Concepts of Differentiation*, to illustrate the procedures involved.

The Index Entry / Descriptor

Each component of your module requires an entry in an index file. These **index entries** (also known as **descriptors**) comprise a number of elements (**descriptor fields**) that contain essential information for the Courseware Manager. In brief outline these elements are:

- A **filename**, identifying the particular file which contains this part of the courseware e.g. M4BASE_1.TBK
- A **type**, identifying where in the hierarchy this item should be placed e.g. leaflet
- A **module code**, identifying the module to which this item 'belongs' e.g. M4BA
- An **item code** (for leaflets a **leaflet code**, for learning units a **unit code** etc.) which acts with the module code as a unique identifier e.g. cvar
- For leaflets only a **leaflet type** and a **user type** e.g. Definition and Mathematician
- A **locator**, identifying numerically 'where' an item begins on first opening it e.g. 1
- A **title** which gives a name to the item that will be seen in the title bar etc.. e.g. "Changing Variables"
- A **keyword string** which contains a list of keywords applicable to the item e.g. "differentiation variables"

We shall now look at these various descriptor fields in more detail.

Filename This is simply the DOS filename of the file that contains the courseware. Only the filename is required since the path is appended dynamically by the Courseware Manager at run-time.

Type Mathwise manages courseware in an hierarchical fashion. The hierarchy imposes an implicit structure upon the courseware through only permitting items to call other items *lower* than themselves in this hierarchy. The levels of the hierarchy are intended to be used for different purposes and in descending order are as follows:

ENTRY-UNIT	- used as a common entry point to Mathwise
MODULE	- used as the control point for a broad subject area
LEARNING UNIT	- used as the control point for a topic within a module subject
LEAFLET	- used for specific, individual topics, tests, reference etc.
UTILITY	- used for tools, general references, external applications etc.

A standard entry-unit is provided with Mathwise (although it is possible to override this or write your own) and need not concern authors. This field is sometimes referred to more fully as the **descriptor type**.

Module and item codes The single most important part of these individual index entries are the various codes. The principle of these is very simple: each courseware item (apart from modules) is indexed with two codes. The first of these is the **module code**: this designates which module 'owns' the particular item, thus for a given module all components will possess the same module code³. The second code, known as the **leaflet code** when referring to leaflets, the **unit code** when referring to learning units, and the **utility**

³ Note that this notion of ownership does not limit in any way the accessibility of a given item. A given item can be indexed several times in different index files if desired, giving it several 'owner' modules. Furthermore the calling procedures for leaflets allow for module codes to be ignored if required.

code when referring to utilities, in conjunction with the module code serves as a unique identifier for the particular courseware item being indexed.

All these codes take the form of a four-character, case-sensitive string⁴. For the module code this string should be the first four characters, in uppercase, of your module number. So for our sample module, the module code is M4BA. As for learning units and leaflets you are free to choose the unit and leaflet codes as you feel appropriate (there are a few exceptions to this, explained later). However, in the aid of consistency across modules it is recommended that you restrict yourself to 'descriptive' codes (e.g. 'lims' for a leaflet on 'Limits') as opposed to 'systematic' codes (e.g. 'lft1', 'lft2' etc).

The exceptions to the free choice of codes are with regards to standard resources such as your 'About This Module' file and your module's bibliography. The module code is unaltered but the item code should be one of the standard codes shown in Table 1.1 below⁵.

Standard Item	Item Type	Item Code
About This Module	LEAFLET	abtm
Bibliography	LEAFLET	bibl
Glossary	UTILITY	gls
Module Map	UTILITY	mapm

Table 1.1 *Standard Item Codes*

Example: The following table shows the various codes for our example module. Note how the module code remains constant and how the item codes reflect the contents of each courseware item, albeit in a highly abbreviated manner.

Item Title	Item Type	Module Code	Item Code
Concepts of Differentiation	MODULE	M4BA	
Introduction	LEARNING UNIT	M4BA	intr
The Idea of a Slope	LEARNING UNIT	M4BA	idea
Basic Definition	LEARNING UNIT	M4BA	defn
The Derived Function	LEARNING UNIT	M4BA	derv
Polynomial Functions	LEARNING UNIT	M4BA	dpol
Log Functions	LEARNING UNIT	M4BA	clog
Trig Functions	LEARNING UNIT	M4BA	dtrg
Notation	LEARNING UNIT	M4BA	notn
Changing Variables	LEAFLET	M4BA	cvar
Functional Form	LEAFLET	M4BA	func
Higher Order	LEAFLET	M4BA	d2dx
Definition	LEAFLET	M4BA	defn
Limits	LEAFLET	M4BA	lims
Notation	LEAFLET	M4BA	notn
Rate of Change	LEAFLET	M4BA	rofc
Slope of a Line	LEAFLET	M4BA	line
Standard Derivatives	LEAFLET	M4BA	stan

Table 1.2 *Sample Codes for M4BASE Module*

Leaflet and User Type This descriptor field is applicable only to leaflets and exists to allow selection of leaflets according to the nature of their content and their intended audience. This facility is not fully exploited in the current version (v2.0) of Mathwise but will be properly supported in later releases. Thus you should still provide values for these fields despite their current redundancy. It is suggested that when in doubt use a leaflet type of DEFINITION and a user type of MATHEMATICIAN.

Permitted leaflet-types are as follows: APPLICATION, DEFINITION, EXAMPLE, NOTE,

⁴ The reason for limiting the code to 4 characters is that it is internally stored and manipulated as a 64 bit /4 byte long-integer. This is for reasons of both speed and memory conservation.

⁵ The reason for using these standard codes is that it ensures that certain 'short-cut' navigation functions (such as 'send AboutThisModule') and the scripts for some standard buttons (such as the 'Bibliography' button) work correctly. You would still be able to call these resources if their codes were non-standard; it would just be necessary to use the general navigation function MWGoDesc() rather than resource specific functions.

PRINTABLE NOTE, QUESTION

Permitted user-types are as follows: MATHEMATICIAN, ENGINEER

Locator This descriptor field is an integer and is used to indicate the point at which the courseware item should begin when it is first opened. In the case of a Toolbook book for example the locator is used as a page number, in Authorware as a screen number, and so on. In most cases you will probably set the locator at 1. The reason for permitting other values, effectively allowing a jump into the middle, or even the end of a Toolbook book say, is that it allows an author to combine several leaflets for example in a single file.

Title This is a text string (limited to 32 characters maximum) and should give a description of the courseware item suitable for display in the item's title-bar, in the browsing list-boxes in the entry-unit etc. It is sometimes referred to as the comment field.

Keywords This is a text string (limited to 64 characters maximum) which should be used for a list of keywords (separated with spaces) which appropriately describe the contents of the courseware item. The Mathwise system allows navigation on the basis of keyword searches as a complement to the code system described above, and it is this descriptor field that is the basis of it.

Building the Index File

Once you have devised a coding scheme for your module it is time to actually create your index file to contain this information. Index files are unlimited in their capacity, so in theory you can place all the required index entries for your module into a single file. However we recommend that the index entries for your leaflets are kept in a separate file since it is in the nature of leaflets that they are 'portable' and liable to be used separately from your module and learning units; the separate index file aids this portability. We also recommend that you name your index files after the module number thus: for the example module, learning-units and utilities, the index file is called M4BASE.MWX and for the example leaflets the index file is called M4BASE_L.MWX.

To build the index file you have a choice. The recommended method is to use the **Index Editor** (mwindex.exe) that is supplied with Mathwise; this builds .mwx files directly. Alternatively it is possible to write an index file using a text editor in the form of an .ini file. This is then converted to .mwx form using the **Index Converter** (indxconv.exe).

Using The Index Editor

The Index Editor (mwindex.exe) is largely self-explanatory in use once the Mathwise descriptor system discussed above has been understood. The Editor comes with an integral Windows Help file if difficulties are encountered.

Start the Editor in the normal fashion, either from its Program Manager icon or from the File Manager. Once it has loaded you will see a window containing a menu bar and an empty, multi-column list-box that will display your index entries as you add them.

To add a new entry select **New** from the **Edit Index** menu. This will bring up the Index Entry Editor dialog-box. It is here that you enter the details of each required field of an index entry. The file name may be typed directly, or selected using a file browser invoked with the **Select** button. Type, and for leaflets, leaflet and user type, are selected from the lists in drop-down combo-boxes. The remaining fields (module and item codes, locator, title/comment and keywords) should be typed in the edit-boxes. When the entry is complete return to the main window with the **OK** button. The **Cancel** button returns to the main window without making any alterations.

To edit an existing entry either select it in the index entry list-box and then choose **Edit** from the **Edit Index** menu or simply double-click on it. Either action will invoke the Index Entry Editor dialog-box with the values for this entry for you to edit as required.

To delete an existing entry, select it in the index entry list-box and then choose **Delete** from the **Edit Index** menu.

Once your index is complete then you may save it by selecting **Save as...** from the **File** menu. Existing indices may be opened for examination or editing using the **Open...** option on the same menu.

Using The Index Converter

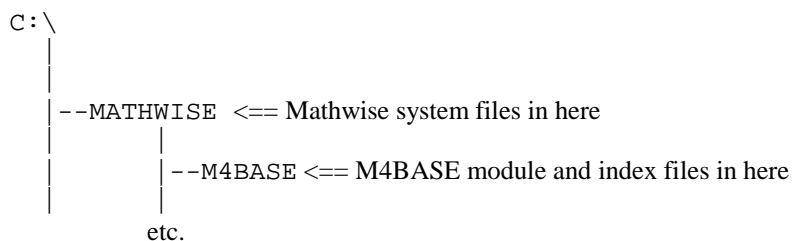
Some authors may prefer to eschew the Index Editor and write their indices using a text editor. This approach may offer some advantage in writing large indices of similar items (such as a glossary) in that details may be readily copied and pasted before the slight alteration for each item is made. The format of an ASCII index file is basically that of a typical Windows .INI file. It is divided into sections (one per index entry) with a variety of entries and their values within each section that correspond to the component fields of the index entry.

After an ASCII index has been written it **must** be converted into the standard .mw x form before Mathwise will understand it. This conversion is accomplished using the Index Converter (`indxconv.exe`). Operation of the converter is self-explanatory. Conversions can be made both ways between the two forms of index files. The **.mw x to .ini** conversion carried out on an existing index is a handy way of generating a 'template' .INI index file in a guaranteed correct format.

Please note that the current version of the Index Converter is not very forgiving of **.ini** files that are not of the correct format, so please take care to check your resulting **.mw x** file after conversion using Index Editor in case gibberish has been produced.

Adding the Index File to the System

Once your index files have been produced it is a simple matter to add your module to the Mathwise system. Firstly you should ensure that all your module files plus your index-files are resident in the same subdirectory beneath the Mathwise system directory e.g.



Next you need to make some small alterations to the Mathwise initialisation file `MW.INI`⁶. Open this using a text editor (such as Windows Notepad) and locate the `[modules]` section. To add your module to the system simply requires you to add references to your index files in this section. For our example module (which has separate index files for the module/learning units and leaflets) it would look like this:

```

[modules]

M4BASE=m4base\m4base.mwx
M4BASE_LEAFS=m4base\m4base_1.mwx

etc

```

Note two things. Firstly the actual name you put before the '=' sign is immaterial, provided that it differs from any of the other identifiers in the `[modules]` section. Secondly, you only need to provide a partial pathname to your index files; Mathwise assumes that these paths lie below the Mathwise system directory. Although you are at liberty to give a full pathname (ie. with a drive letter) please avoid this if possible. On stand-alone machines it would work without difficulty, but on networked machines using remote servers etc. drive-names are frequently ambiguous and so full pathname references can cause problems.

Once you have amended `MW.INI` you should start (or restart) Mathwise. Your index files will then be read by the system, however the individual index entries will only be added to Mathwise's internal index should the files referred to actually exist. Consequently, if you wish your navigational links to work (see next chapter) you should ensure that at the very least there are 'skeleton' files in place.

⁶ Or any .ini file you may be using to override the defaults using the `-I` command line option of `mathwise.exe` (see *Appendix 3*).

Chapter 4

Navigation

Navigation Within Items

Navigation between individual screens/pages within a single courseware is accomplished using the arrowed navigation buttons. The Toolbook version is shown here:



The plain arrows move backwards and forwards respectively between screens/pages (the exact details of how this is done varying according to the authoring environment) and should 'grey-out' and become redundant at the start or end of the item. The bent arrow 'returns' from an item to the one which initially called it. There are more details on this later in this chapter.

Navigation Between Items and MWGoDesc()

Movement between courseware items in Mathwise is carried out by the Courseware Manager in response to requests from items. These requests take the form of **descriptor queries** to the Courseware Manager's internal index that allow searches for appropriate items using a range of different criteria. It is possible to search on the basis of type, module and item codes, leaflet type and keywords, or any combination of these. In addition wildcards are permitted and, in the case of the keyword search, boolean operators, to fully refine a search. All these search types are carried out using a single function call to the Courseware Manager, `MWGoDesc()`. In the event of more than one match being found to the specified search conditions, a window, called the **browser** will be invoked, from which the desired item can be selected.

As far as authors are concerned it will rarely, if ever, be necessary to use the full searching potential that `MWGoDesc()` allows for. By far and away the most common form of navigation is from a module to a learning unit, or from a learning unit to a leaflet. In these instances the use of `MWGoDesc()` is very straightforward, and in the case of Toolbook or Authorware authors, even easier, since pre-configured buttons are provided in the author kits which obviate the need to understand the use of `MWGoDesc()` at all. However, a brief description of the function is useful even if you don't use it directly.

The syntax of `MWGoDesc()` is as follows:

```
Pascal: function MWGoDesc(h : HWnd; s : PChar) : integer;
C:      int MWGoDesc(HWND h, LPSTR s)
```

Where:

h Window handle of calling item. This is necessary to inform the Courseware Manager which courseware item is actually issuing the request. In the case of Toolbook this parameter is equal to the `sysWindowHandle` system property. In the case of Authorware it is given by the `WindowHandle` system variable.

s descriptor query string : "[t],[m],[x],[l],[keyword-query]" consisting of a series of optional fields corresponding to the various index fields it is wished to search on.

Returns :

The number of matches made if successful, 0 if no matches made.

A negative return indicates a syntax error or other problem in evaluating the query.

The algorithm by which Mathwise evaluates a descriptor query is designed to be as fast and efficient as possible and has a bearing on the manner in which authors should frame descriptor queries. To understand this algorithm you should regard the descriptor query as, in effect, a logical expression of the form: `t AND m AND x AND l AND keyword-query`. Mathwise only makes a match between the descriptor query and a given descriptor when this expression evaluates to TRUE. This will clearly only

occur when the match of each field evaluates to TRUE. Mathwise takes advantage of this fact by evaluating each field in turn, starting with `t`, only proceeding to evaluating the next field for as long as the 'running result' equals TRUE (any unspecified fields are assumed to evaluate to TRUE). Thus the lesson for framing a descriptor query is basically to be as precise as possible, as early as possible.

Parameters:

`t` type code. The standard codes are defined as:

```
EUNT : entry-unit
MODL : module
UNIT : learning unit
LEAF : leaflet
UTIL : utility
```

The reserved value `INTL` may also be specified here. This is a special code which indicates the Mathwise system is to execute an internal routine, specified by the values of the other descriptor-query fields, and is required primarily for use with the palette/resource browser. The functions available using this special type code are not yet documented.

`m` module code. The reserved code `crnt` should be used to indicate the current module code.

`x` 'x' code. For learning units, leaflets and utilities this is the unit code, leaflet code and utility code respectively

`l` leaflet type. The standard codes are defined as:

```
APPL : application
DEFN : definition
EXMP : example
NOTE : note
PRNT : printable note
QSTN : question
```

All these parameters may be a maximum of four characters long and are case sensitive. Since leaflet typing has been largely ignored by authors of Mathwise material it is perhaps best to ignore this parameter altogether when framing a descriptor query.

The '*' character may be used as a wildcard, either as a single character wildcard (e.g. "`LEAF,mod1,lft*`" would match all leaflets for module-code `mod1` that have leaflet-codes `lft1`, `lft2`, `lftA` etc.) or as a 'wildfield' (eg. "`*,*,test`" would match all item types for all modules that have a unit/leaflet/utility code of `test`).

An empty field is treated as a wild-field e.g.

```
", ,lft1" ≡ "*,*,lft1" ≡ "***** ,***** ,lft1"
```

keyword-query

This parameter is used to search the keyword section of each index entry. The user may specify up to ten keywords in this query and these keywords may be combined using the standard logical operators NOT, AND and OR. The symbols '~' (tilde), '&' (ampersand) and '+' may also be used. NOT has the highest precedence, OR the lowest. Parentheses '(' and ')' are also supported to any depth. A match is only made when the entire query evaluates to TRUE. The search is both case and whole word insensitive.

There is a special meaning to this field if the first character is an underscore '_'. Any text that follows the '_' character is taken to be command-line parameters to be passed to the requested item.

Examples:

```
"LEAF,mod1,lft1"
  match leaflet lft1 of module mod1
"MODL,abc*"
  match all modules with module codes e.g. abc1, abc2, abcX etc.
```

```
"UNIT,fred"
    match all learning-units for module fred
"UTIL,,calc"
    match all utilities with utility code calc
"LEAF,crnt"
    match all leaflets for the current module
"LEAF,mike*,*,calculus and mechanics"
    match all leaflets for module mike that have both calculus and mechanics
    in their keyword fields.
",mod*,,,matrices + not(vector and~complex)"
    match all items with module code mod* (eg mod1,modW etc.) with
    matrices or complex but not vector in their keyword fields.
"UTIL,excl,excl,,_data2.xls"
    If Excel is indexed with mod-code excl and util-code excl then this will run the program and
    pass data2.xls as a command-line parameter eg. 'c:\excel\excel.exe data2.xls'
```

Variants on MWGoDesc()

There are several variations on the basic MWGoDesc() function, that will accomplish different ends using the same or similar syntax.

MWCanGoDesc(h,s)

Pascal: function MWCanGoDesc(h : HWND; s : PChar) : integer;

C: int MWCanGoDesc(HWND h, LPSTR s)

This function takes exactly the same parameters as MWGoDesc() but will not run any matching items. It is simply used to determine how many matches there are to a query – this is the return value.

MWGoDescBrowser(h,s)

Pascal: function MWGoDescBrowser(h : HWND; s : PChar) : integer;

C: int MWGoDescBrowser(HWND h, LPSTR s)

This function takes the same parameters as MWGoDesc() but will always show the browser, even if only one match is made (MWGoDesc() would automatically run the single matching item). This might be used for example on a button labelled 'Available Leaflets about Eigenvalues', which uses a query 'LEAF,,,,eigenvalues'. A user clicking this button would expect to see a list of these leaflets – even if there is only one match found they wouldn't expect to be launched straight into the leaflet and so in this instance MWGoDescBrowser() should be used rather than MWGoDesc().

MWGetDescCount(h,s)

Pascal: function MWGetDescCount(h : HWND; s : PChar) : integer;

C: int MWGetDescCount(HWND h, LPSTR s)

This function takes the same parameters as MWGoDesc(). Instead of running items it returns the number of items matching the descriptor query. The item data may then be retrieved as a string by using the MWGetDescItemAsText() function described below. This pair of functions can be used to build custom lists of courseware material within your modules.

MWGetDescItemAsText(nItem,fDelim)

Pascal: function MWGetDescItemAsText(nItem : integer; fDelim : char) :
Pchar;

C: LPSTR MWGetDescItemAsText(int nItem, int fDelim)

This function is used in conjunction with MWGetDescCount() to retrieve the descriptor information for the items found by using MWGetDescCount(). The parameter nItem is the number of the matching item you wish to retrieve and fDelim is the character to be used for delimiting the fields of the return string. The return string contains the descriptor fields for item nItem in the following order: file name, type (as integer 0 to 4), module code, x-code, leaflet type, user type, locator, comment/title and keywords, delimited by the fDelim character.

MWGoDest(h,fs)

Pascal: function MWGoDest(h : HWND; fs : PChar) : integer;

C: int MWGoDest(HWND h, LPSTR fs)

This function takes a file name as its second parameter fs. It will find all descriptors that refer to this file. It is used for backwards compatibility with version 1 of Mathwise.

MWGoGlossary(h,gs)

This function is simply a shortcut form of MWGoDesc(). The second parameter, gs, is a glossary search string. The function assembles the descriptor query 'UTIL,crnt,glss,,_gs' (run the glossary for the current module, sending gs as a command-line parameter) and uses MWGoDesc() to execute it. If the glossary is already active then it will be brought into focus and notified that it has a new word to look up using the MW_UPDATEGLOSSARY custom message. It can then retrieve this new value using the MWGetCmdLine() function which returns a PChar/LPSTR string. Note that you should use the Windows API function RegisterWindowMessage() or your authoring package's equivalent to register and respond to this message if you intend to write your own glossaries without using the provided templates. If the search fails it retries with 'UTIL,,glss,,_gs'. This will bring up the browser with a list of all the glossaries on the system for the user to select from.

MWGoBiography(h,bs)

Similar to MWGoGlossary() but for searching biographies.

Navigation In Practice

For nearly all authorial eventualities use of MWGoDesc() is extremely simple. For the most part you will be calling items, either learning units or leaflets, that belong to your module. Less frequent will be the calling of 'external' leaflets, ie. those authored by someone else and thus belonging to a different module. In the case of your own module's items you should specify the module code as crnt (ie. 'use the current module code') and then use the unit/leaflet code you assigned when indexing the module. When calling external leaflets you should if possible use the appropriate module and leaflet codes if known. If these are not known, perhaps because the leaflet you require has not yet been written, then you should use the keyword search facility with an appropriate search string. Note that use of the keyword search may be less exact than precisely specifying codes; for example a rather vague search string could produce multiple matches and thus invoke the browser window, possibly confusing the student. Conversely an overly stringent search string might make no matches at all, despite there being an appropriate leaflet available.

Here are examples of how to use MWGoDesc() for these common situations:

- Calling your own learning unit (unit code mylu):
MWGoDesc(h, 'UNIT,crnt,mylu')
- Calling your own leaflet (leaflet code mylf):
MWGoDesc(h, 'LEAF,crnt,mylf')
- Calling an external leaflet (on subject 'addition of Vectors', module code AlXX, leaflet code veca)
If the module and leaflet codes are unknown:
MWGoDesc(h, 'LEAF,,,,addition & vector')
- If the codes are known:
MWGoDesc(h, 'LEAF,AlXX,veca')

For Mathwise Toolbook authors the situation is simpler still since the 'Style Book' provided contains a leaflet button that automatically prompts for a leaflet code when pasted and sets the button's script as follows (where xxxx is the code you have entered):

```
to handle buttonUp
  get MWGoDesc(sysWindowHandle, 'LEAF,crnt,xxxx')
end buttonUp
```

and there are similar buttons to automatically deal with learning units and external leaflets.

Returning From Items

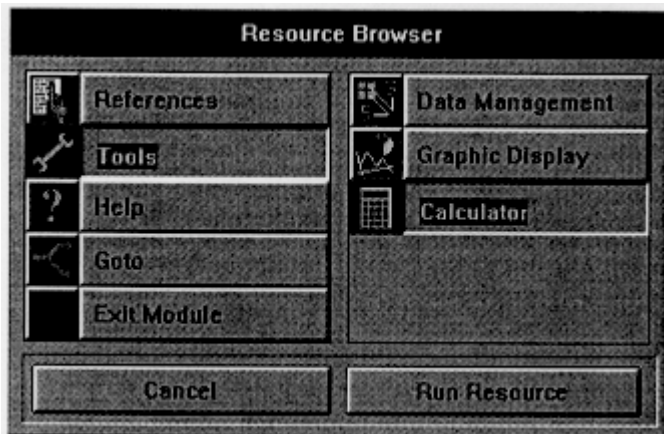
Once a leaflet has been read by a student, returning to the learning unit that called it is accomplished by clicking the return arrow. It may appear that this simply closes the leaflet directly. However this is not so and as an author you should be aware of what is really happening. What this button actually does is call the Courseware Manager function MWCloseByHandle() with the current item's Windows handle as an argument. This then enables the Courseware Manager to identify and then remotely close the item (By sending it the WM_CLOSE

Windows message). The reason for preferring this manner of closing items over self-closing is that it enables the Courseware Manager to keep close track of the current state of the Mathwise system. Mathwise is still able to recognize when items do self-close but it can be possible that it remains in a state of ignorance for some time, during which it may give misleading error messages to the user.

Chapter 5

Palettes

The Mathwise **palette**, also referred to as the **resource-browser**, provides a means of accessing system-wide resources and tools, as well as module specific resources, via a simple easy to operate interface. It takes the form of a floating modal window consisting of two list-boxes, and **Cancel** and **Run Resource** buttons, with which selections of resources may be made and then run. This is done by associating descriptor queries with palette items which are sent to `MWGoDesc()` and executed in the usual manner. Depending on configuration, selections in the left-hand list-box may run resources directly, but more usually bring up sub-lists in the right-hand list-box from which resources may be selected and run. As can be seen from the example view of the palette below, selections are clearly indicated by highlighting both text and icon:



There is a system wide default palette which is what the user will see unless an author decides to override it with a module specific palette of their own devising. Most authors will have no need of this facility but some modules, particularly application modules, require special tools to be available (such as a spreadsheet) just for that module, and an amendment to the palette is an appropriate way of making these tools more readily accessible to the user.

Mathwise palettes are defined in **palette files** which may be recognised by their `.mwp` extension. They are created, or edited, by using the Mathwise **Palette Editor**, `mwpaled.exe`. The standard default palette is defined in the `mw.mwp` file located in the Mathwise system directory.

Using the Palette Editor

The Palette Editor is largely self-explanatory in operation. Since you are unlikely to want to build a new palette from scratch what follows here is a brief guide to amending the default palette to suit your module's special requirements.

Firstly run the Palette Editor in the normal manner. Once loaded you will see a window with a menu bar and an empty list-box. Now you need to load the default palette to have something to edit. Select **O**pen... from the **F**ile menu. Navigate to the Mathwise directory (probably the same directory in which `mwpaled.exe` is located) and select `mw.mwp`. Click **O**K to load it.

You will now see that the initially empty list-box now contains some information. Most of these entries will have text showing only in the *Comments* column of the list-box and nothing in the *Descriptor Query* column (which indicates the command that a palette entry should carry out). This is because clicking these entries on the palette will bring up sub-menus in the right-hand list-box.

Click on the topmost entry (it should say 'References' in the Comment column) with your mouse. A second list-box should now appear beneath the first containing several entries. Unlike the upper list-box these entries should all have some text in the *Descriptor Query* column. This is because these entries will actually execute commands. Try the other entries in the upper list-box and you will see they will each bring up a different submenu listing in the lower list-box with the exception of the last entry ('Exit Module').

Before proceeding you should first test the palette to assure yourself that it is indeed the default palette. Simply select **Test Palette** from the menu and the palette will appear as it will when summoned from Mathwise. To dismiss the palette you can either click the **Cancel** button or select an entry and click the **Run Resource** button, in which case a message-box **will** appear giving the descriptor-query that is associated with that palette item.

Now it is time to edit the palette. Let us say that your module (with module-code M1AB say) requires to run a Cycloid Plotting program called `cyclplot.exe`, and that you wish this tool to be available to students throughout your module. Assume that you have prepared an index-file, or an entry within the module index-file, for the Cycloid Plotter, in which it is categorised as a Utility, with your module-code and utility-code `cycp`, then this is how you customise the palette:

[1] In the upper-list box of the Palette Editor select the entry with the comment 'Tools'. This will bring up the 'Tools' sub-menu in the lower list-box.

[2] Now select **New Sub Item** from the **Edit Sub** menu. This brings up the *Palette Entry Editor* dialog-box which is where you will enter the data for this item.

[3] In the *Descriptor* field you should enter an appropriate descriptor-query for the item. In this instance you should enter `UTIL, crnt, cycp`. That is 'run the utility with code `cycp` for my module'. You could also enter `UTIL, M1AB, cycp` of course, the result being the same.

[4] In the *Comment* field you should enter the text that will be seen on the palette for this item. Enter 'Cycloid Plotter' here.

[5] Now you need to add an icon for this item. You may select from Mathwise's own range of icons (click the **Select Internal Icon** button and browse the choice with the arrows) or import one from outside. Since it would be neater to use the same icon that the Cycloid Plotter would display in Program Manager then we shall do just that. Click the **Select External Icon** button. This will bring up an *Open File* dialog-box which may be used to import icon-files (`.ico`) or in this instance extract icons from executables. Navigate into the directory where the Plotter is located, select `cyclplot.exe` and click **OK**. This will bring up a dialog with the Cycloid Plotter icon displayed in it (for executables containing multiple icons you can browse through them with the arrow buttons at this point). Click **OK** and you will return to the *Palette Entry Editor* dialog-box where the new icon will now be displayed.

[6] Click **OK** to add this new entry to the palette. This would be a good point to test your new palette as explained above.

[7] Finally you need to specify that this amended palette is to be used only with your module. This is achieved by 'branding' the palette file with your module code. To do this simply select **Set Code** from the menu and enter the code of your module in the dialog-box, in this instance . Click **OK**.

[8] Your customised palette-file is now complete and all that remains is to save it. Select **Save As...** from the **File** menu and navigate to your module directory. Save your palette with an appropriate name, say `alab_pal.mwp`. To get the palette working next time you run Mathwise you also need to add a reference to it in the Mathwise initialisation file. To see how to do this see [3] *Edit the Initialisation File in Chapter 2*.

Appendix 1

Using Mathwise v1 Material with v2

There are some very great differences between version 1 and version 2 of Mathwise, not just in the author interface but at a fundamental 'design philosophy' level. This naturally presents difficulties in running material authored using Mathwise v1 with the v2 system. This Appendix contains details on the accommodations that Mathwise v2 makes for v1 material.

Toolbook v1 Material

It is Toolbook authored modules which constitute the large bulk of v1 material. There are two possible ways of running this material under v2 of Mathwise.

The first, and preferred method, is the direct conversion of the v1 function calls to v2 function calls. This will guarantee correct operation of the module and of course opens up authorial possibilities that the greater power of v2 offers. However, since this involves examination of every script of every object on every page of every Toolbook book in order to spot v1 function calls then clearly it can be a time-consuming and tedious process, even with an automated search.

The alternative is to use the Toolbook system book, `ukmcc.sbk`, which Mathwise v2 includes as a replacement for the similarly named file in v1. The use of this system book means that no alterations to any of the v1 material are necessary since its purpose is to intercept all v1 function calls and messages and translate them, as far as is possible, to the v2 protocols. The only additional work that is required is the creation of a v2 Index File for the v1 module. The index file(s), `ukmcc.sbk` and its sister v1 DLL, `ukmcc.dll` (which provides several screen effects such as zooming pop-ups which cannot easily be translated) should be placed in the v1 module directory and the module reference(s) added to the Mathwise initialisation file as usual.

How the Replacement UKMCC.SBK Works

Learning Units and Leaflets Under v1 all learning-unit and leaflet calls are made on the basis of file names whereas v2 uses descriptor codes. The Courseware Manager supports a function `MWGoDest()` which takes as an argument a file name and then searches the courseware index for a descriptor entry for this particular file name. If this is found then the data included in descriptor entry is then used to form a descriptor query which is used by `MWGoDesc()` in the usual manner. In the unusual event of the `MWGoDest()` search finding more than one match, then the Browser is invoked before `MWGoDesc()` is invoked. Since the Courseware Manager has effectively to conduct two searches rather than the single optimised search that `MWGoDesc()` carries out, and since the initial file name match search is by its nature fairly intensive, then you cannot expect these modified leaflet and learning unit calls to work with the same speed that proper v2 calls do.

Enquiries and Searches The two v1 functions `SearchIndex()` and `MakeEnquiry()` conduct keyword searches of the v1 leaflet index. These are readily translated to v2 by extracting the search string and placing it in the keyword-query field of a descriptor query, which is then sent to the Courseware Manager thus: `MWGoDesc(h, 'LEAF,,,keywordQuery')`. The `searchFrom` parameter of `SearchIndex()` has no meaningful parallel in v2 so is ignored.

Glossary Searches The v1 function `SearchGlossary()` is also easily translated to v2. The `Txt` search term is examined in order to find the actual word that `iHit` indicates. This word is extracted and used by the v2 function `MWGoGlossary()` directly. The `searchGlossary_` message is dealt with similarly.

Resources The notion of resources in v1 has no direct equivalent in v2 since all the possibilities v1 resources offer are easily accommodated by the same descriptor system that supports module, leaflet etc. navigation. This makes a direct translation very tricky without undue modification and extension to the v2 Courseware Manager. Consequently the v2 handling of the two possible calls in v1 involving resources is

limited.

In the case of `ExecResource()` it is assumed that the string argument name contains a file-name. This is extracted and any additional parameters in the string discarded. The file-name is then sent to `MWGoDest()` as with learning-unit and leaflet calls, assuming initially that it is indexed as a leaflet and then, if that search fails, it is assumed to be a utility. The message `CallResource` is dealt with similarly.

`CallResource()` poses even more problems since it uses 'resource names' rather than file-names. In the case of system resources it is quite straightforward to intercept and redirect. The following system resources are spotted by the translator `ukmcc.sbk` and the v2 equivalents are run: `biograph/biographer`, `handbook`, `glossary`, `map`, `authors`, `index` and `calculator`.

However, the case of user-defined resources (those defined in the `[resources]` section of a module `.ini` file, other than overrides of standard resources which the translator does deal with) is impossible to satisfactorily resolve. Consequently any calls to `CallResource()` within a module which use these user-defined resources will simply be ignored.

Other

All 'return' related functions and messages (`quit`, `exit`, `return`, `return_`, `returnToControl~` are intercepted are replaced with a simple `MWCloseByHandle()` call.

The 'history' effected by the Toolbook 'stack' in v1 is dealt with by the Courseware Manager in v2 by means of material using the `MWChangePosByHandle()` function every time a page or screen is changed. This informs Mathwise of the current position of a particular item. Carrying out this procedure with v1 material without altering the actual book's scripts proves to be possible only by placing `MWChangePosByHandle()` inside the handler for the v1 `SetCaptionText` message.

The two v1 palette functions `SupportPalette()` and `ZoomObjSupportPalette()` are both replaced by a call to `MWPalette()`.

All other v1 calls and messages remain as before.

Appendix 2

What Happens When Mathwise is Started and Closed

The Mathwise system can be regarded at heart as a combination of a database/search engine and an extension to the Windows operating system. It is designed to be as versatile in use as possible, enabling it to be used in situations very different from the specific requirements of the Mathwise courseware. To allow this flexibility it operates in a rather abstract fashion. This is technically complicated, so for the benefit of authors what follows in this Appendix is a brief explanation of the start-up and close-down procedures that might give some insight into the underlying structure of the system without delving too deeply into technical matters.

Starting Mathwise

- [1] The system is started by running the Mathwise system boot program `mathwise.exe`. The purpose of this boot program is threefold: it provides a fixed 'anchor' for the Mathwise system, it carries out the requisite initialisation of the system, and it displays a splashscreen to keep the user occupied whilst these processes take place. As explained in *Chapter 1* it is possible to replace `mathwise.exe` with a boot program of your own devising in which case you should consult the documentation for system start-up in *Appendix 4*.
- [2] The boot program displays the splashscreen specified by `startupSplash` in the initialisation file or given in the `-s` command-line option of `mathwise.exe`, which remains in view until the system has been fully initialised.
- [3] It initialises the Courseware Manager (`mw.dll.dll`) by first linking to it and then calling the `MWSetUpDLL()` function with the name of the default initialisation file (`mw.ini`) unless it has been overridden by the `-I` command line parameter of `mathwise.exe`.
- [4] The Courseware Manager now reserves memory for itself, carries out other initialisation tasks and then opens the initialisation file specified in [3].
- [5] It firstly reads in the `[general]` options, then any palette overrides specified in `[palettes]` and finally builds up a table of file-suffix / reader-program associations from the `[associations]` section of the initialisation file.
- [6] It now examines the `[modules]` section of the initialisation file and attempts to locate the assorted index files specified therein. Once an index file has been found, it is opened and the contents read by the Courseware Manager, conditional on the courseware files referred to actually existing.
- [7] Once all the module indices have been read, the Courseware Manager then carries out a variety of sorts on the index data to speed up access and search times.
- [8] Control returns to the boot program which, with system initialisation complete, removes the splashscreen and minimises itself. It finally calls the `MWSetStartupItem()` function with the descriptor query for the entry-unit as specified in the initialisation file or in the `-D` command-line option.
- [9] The Courseware Manager now runs the specified entry-unit to which focus is transferred.
- [10] The start-up of Mathwise is now complete.

Stopping Mathwise

A Mathwise session may be ended in a variety of ways, either using the `palettte` option, or by clicking the return navigation buttons until the Courseware Manager is visible. Then it is possible to use the Exit option. It may also be closed by closing the Mathwise icon that displays at the bottom of the screen whilst a Mathwise session is active.

In all these cases the system will firstly close down any currently open courseware items. It will then free the memory and Windows resources it has used and finally will close down the minimised boot-program, simultaneously ending its own instance.

Appendix 3

Customisation Using `mathwise.exe` Command Line Options

The Mathwise launcher program provided, `mathwise.exe`, will take a number of command-line options which allow easy and flexible customisation of the Mathwise system.

Here is the syntax for using `mathwise.exe` in this mode:

```
mathwise.exe [-Iini_file] [-Dstartup_desc] [-Ssplashscreen_file]
```

where

`-Iini_file` overrides the default system initialisation file `mw.ini` with the initialisation file specified in `ini_file`. For example `mathwise -Iyear1.ini` will launch Mathwise with the setup specified in `year1.ini` rather than `mw.ini`. In the absence of a full path for the override file it is assumed to reside in the Mathwise system directory.

`-Dstartup_desc` overrides the descriptor defined by the `StartupDesc` entry in the initialisation file (be it the default `mw.ini`, or an override given in the `-I` option), which tells the Courseware Manager which courseware item to start a Mathwise session with.

`-Ssplashscreen_file` overrides the splash-screen defined by the `StartupSplash` entry in the initialisation file with the BMP file given in `splashscreen_file`.

Use of these parameters will vary according to whether Windows 3.1 or Windows 95 is being used.

Windows 3.1 Create a new program-manager item in the desired group for `mathwise.exe`. Bring up the Properties of the item and add the desired command-line options to the calling string.

Windows 95 Create a short-cut to `mathwise.exe`. Bring up the Properties of the short-cut and tab to Shortcut. Add the required command-line options to the end of the string in the Target edit-box. The short-cut may be placed on the desktop or in the Start Menu as required.

A number of customisation possibilities are introduced by using these overrides.

For example system administrators may wish to make different choices of Mathwise modules available for different years or courses. To do this they can assemble several initialisation files which contain different lists of modules in the `[modules]` section. Optionally different startup-screens can be specified in these files as well. Name these files `year1.ini`, `year2.ini`, `year3.ini`, `year_pg.ini`, say. Create several new program-items / short-cuts as explained above – ‘`mathwise.exe -Iyear1.ini`’, ‘`mathwise.exe -Iyear2.ini`’ etc., and label these ‘Year 1’, ‘Year 2’ etc. Students can now get access to only those modules that concern them.

Alternatively by using the `-D` start-up item descriptor override with a descriptor for a module (eg. ‘`mathwise.exe -DMODL,A5ME`’ for the Mechanical Engineering module) a system-administrator can set-up an array of program-manager items / short-cuts for individual modules. When any one of these is then run Mathwise will go straight to the specified module. This in effect allows an administrator to use Windows itself as the system entry-unit.

Appendix 4

Miscellaneous Mathwise Functions

The Mathwise system provides a large range of functions. Some of the more useful functions for authors are documented below.

Note for Toolbook users. All functions that return strings have a special form for Toolbook use, since Toolbook requires strings to be passed to it in the form of a handle to global memory. These Toolbook specific functions use the same syntax as their standard forms but are named MWTBK* eg. the Toolbook form of MWGetData() is MWTBKGetData().

System Start-up and Close-down

If desired you may create your own Mathwise boot program to use instead of mathwise.exe. To do so you need to call just two functions. Firstly link to mwdll.dll, then start the Mathwise system by calling MWSetupDLL() (procedure MWSetupDLL(iniFileName : Pchar) or void MWSetupDLL(LPSTR iniFileName)) with the initialisation file-name as the parameter. By default this should be mw.ini. You then need to tell the Mathwise system your window handle in order that it can close the boot program down if possible. Use MWSetParent() (procedure MWSetParent(h : Hwnd) or void MWSetParent(HWND h)) with your window handle as the parameter. Finally, if necessary, you can use MWCloseDownDLL() (procedure MWCloseDownDLL or void MWCloseDownDLL()) to close the Mathwise system, probably in response to your boot program receiving WM_CLOSE.

Data Transfer.

Three functions, MWPutData(s), MWClearData() and MWGetData(), allow for the exchange of data between courseware items. Regard these as giving write and read access respectively to a 'data-repository' within Mathwise.

MWPutData() simply takes a zero-terminated string as its argument, which may be of any length, and stores the string in the repository. It then issues a user-defined message MW_UPDATEDATA to all top-level Mathwise item windows. This enables an active exchange of information between two Mathwise items. To respond to the message your program will have to:

- [1] Register the message using RegisterWindowMessage('MW_UPDATEDATA') and
- [2] Define some method of responding to this message that retrieves the data.

MWGetData() returns the current contents of the repository as a zero-terminated string. Since the data-transfer facility uses strings of any size it is clearly possible to use structured string formats of your own devising to share complex data if that is required. Also an application can modify the stored data by getting the string, modifying as required, then putting the string back to Mathwise.

MWClearData() will set the current data to a null string. Use this instead of MWPutData('') if you wish to clear the data repository. MW_UPDATEDATA is **not** sent when this function is used.

Sub-directories and system defines.

There is an optional section in a Mathwise initialisation file (mw.ini by default) entitled [directories]. Entries in this section are in the form tag=value. The second part of one of these entries (ie. value) may be retrieved into an application as a string by using the function MWGetSubDir(tag). The original purpose of this facility was to allow custom directory paths to be defined globally, for example to specify the location to store assessment results (an entry of assess=c:\data\mw-score, for example), but it can equally be used to specify system-wide defines for any purpose.

Path retrieval

The function MWGetPathByHandle(h), takes an item's window handle as an argument and returns the path to that item. It is intended to allow an application to determine its location. The function MWGetMWPPath() returns the path where the MW system (ie. mwdll.dll) resides.

Window-handle override

When Mathwise runs an item it attempts to determine the item's window handle automatically. As Microsoft themselves admit there is no 100% certain method of doing this, so although Mathwise uses a hybrid of the best

methods available it will occasionally fail, particularly when an item does not properly follow the Windows standard start-up procedures. This is important as an item's window handle is the key identifier that Mathwise uses to control items. In this instance Mathwise provides two functions to override its automatic procedure, `MWRegisterHandle()` and `MWResetRunItemHandle()`.

Video Playing

The Mathwise function `MWPlayAVI(h, filename, x, y, w, h)` is provided as a one-shot routine for playing AVI (Video for Windows) movie files in frameless, pop-up windows (ie. overlaid). `h` is the calling window handle, `filename` the name of the AVI file, `x` and `y` the location of the top-left corner, and `w` and `h` the width and height of the video window respectively.

Appendix 5

Mathwise Initialisation Files

The Mathwise system requires a system initialisation file to specify how the system should be configured. By default this is called `mw.ini`, although this default may be overridden (see *Appendix 4*) for purposes of customised setups for different years, courses etc. In addition, the standard Mathwise entry-unit, `entryu.exe`, also uses an initialisation file called `entryu.ini`. This Appendix explains how these files should be written.

System Initialisation File

This uses the standard Windows .INI format, ie. it is divided into sections each of which contains a range of entries.

[general]

This section contains entries that specify basic setup parameters for the system. These are as follows.

StartupDesc

Specifies the descriptor query of the item that you want to start a Mathwise session at. This will generally be the entry-unit, e.g. `StartupDesc=EUNT,*,*`, although you can set it to anything you like. For example you might want to start directly in a particular module, e.g. `StartupDesc=MODL,A1DI`. Alternatively you can use the command line version of a descriptor query with the standard entry-unit (`entryu.exe`) to override the entry-unit's own initialisation file, e.g. `StartupDesc=EUNT,**,_c:\mw\eu2.ini`

StartupSplash

Specifies the splash screen to use when starting Mathwise, e.g. `StartupSplash=c:\mw\mwscrn_1.bmp`

EUAuthorsDesc

Specifies the descriptor query for the 'About Mathwise' button in the standard entry-unit, e.g. `EUAuthorsDesc=UTIL,mwis,auth`

EUWelcomeDesc

Specifies the descriptor query for the 'Welcome to Mathwise' button in the standard entry-unit, e.g. `EUWelcomeDesc=UTIL,mwis,welc`

MaxSimultaneousItems

Specifies the maximum number of items Mathwise will keep open at any one time, e.g. `MaxSimultaneousItems=4`. Mathwise will temporarily close items not currently being used to keep within the value specified here. Only reduce this value if your memory is very low and there are serious performance problems.

ErrorMessageLevel

Determines the level of error message reporting. Set it to a number as follows: 0=No messages, 1=student level, 2=administration level.

ConfirmDuplicateItems

If an item of Mathwise material has been authored incorrectly there may be navigational links that attempt to open an item at the same level as the one being used, e.g. calling a module from a module. Setting this entry to YES will give a warning if a user attempts such a navigation. Otherwise it should be set to NO.

SetupName

This entry is for the benefit of the customisation utility `mwcustom.exe`. It should be set to a string which describes the contents of the initialisation file, e.g. `SetupName=STANDARD`. This will only be relevant in situations where your installation of Mathwise uses multiple set-ups for different years, courses etc.

debugMode

If set to 1 then prior to running items Mathwise will display a dialog which specifies the full command string being used to run the item. The option is given in the dialog to continue with running the item or to abort. Any other value will have no effect. As the name of this entry implies, this is for debugging purposes, especially on networks where path names need to be checked.

[fonts]

Entries in this section are for specifying any fonts to be dynamically installed when Mathwise runs. They will be de-installed on ending the Mathwise session. This allows authors to use non-standard typefaces in their material without needing to install these in Windows. Mathwise uses three custom fonts with entries as follows:

```
font1=tkmath.fot
font2=ukmcsb.fot
font3=ukmcsp.fot
```

Additional fonts may be added in a similar format after these, numbering them sequentially and specifying a font information file (*.fot). Note that the corresponding font data file (*.ttf) must also be present for this to work.

[directories]

This section contains entries that may be used to specify system-wide definitions for the use of authors. An entry takes the form tag=value, where both tag and value are any strings you care to use, e.g. assess dir=c:\mw\results. An author may retrieve a value by calling the MWGetSubDir() function, with tag as the argument. As the name of the section and the associated function imply this facility was originally designed for specifying directories into which, for example, assessment results could be stored.

[autoindex]

This section is used to add files automatically to the Mathwise system without the necessity of writing Mathwise index files to reference them; it does so by generating virtual index files when Mathwise is run which describe the files as utilities (UTIL) with a module code of AUTO (this may be overridden – see code below). It is therefore necessarily limited and is intended to be used to allow tutors to easily add material such as course notes that students may need to access. Authors should not add material using this option, instead they should follow the steps outlined in *Chapter 3*.

The autoindex entries are as follows.

useAutoIndex

Setting this to 1 will enable the autoindexing facility. Any other value will disable it.

addToPalette

This entry allows you to specify if you wish this material to be available from the Mathwise resource browser (see *Chapter 5*). This means that students will be able to access the material from anywhere within Mathwise courseware. A value of 1 will add a button to the main-menu part of the resource browser (the left-hand column of buttons) that when clicked will dismiss the resource-browser and bring up a modal dialog which lists all the autoindexed files. You are recommended to use this if there are more than six autoindexed files. A value of 2 will add the main-menu button. In this case however this button will bring up the autoindexed files as a sub-menu, ie. each autoindexed file will have its own button in the right-hand column of the resource browser. Use this for a small number of autoindexed files. The caption on the main button is set with the name entry (see below). Any other value for the addToPalette entry will not show the autoindexed material in the resource browser.

name

This entry specifies the text that will appear on the main button in the resource browser if that facility is enabled (see addToPalette above). You will probably want to set this to the name of your institution, e.g. name=Brainsville. It will also be added to the keyword field for each autoindexed file. If this value is omitted the word Special is used by default.

code

By default the autoindexed files are given a module code of AUTO. Use this entry to override that code, although you shouldn't need to.

The other entries in this section are the files you wish to autoindex. A file entry should be in the form description=file-path, where description is the text that describes the material and which will be displayed in Mathwise references to this file, and file-path is the full path of the actual file. Here are some examples.

```
Course notes : 1st Year=d:\mw\own\cnotes1.txt
Course notes : 2nd Year=d:\mw\own\cnotes2.pdf
Sample exam papers=d:\mw\own\exams\papers.doc
```

University Web site=s:\www\index.htm

Note that any material you add here should be runnable by Mathwise. If you add files here of a type not supported by Mathwise default file types, you will have to add references to the appropriate reader applications in the [associations] section of this initialisation file. This would be the case for example for the cnotes2.pdf Adobe Acrobat file or the papers.doc Word file above.

[palettes]

This section allows for overriding the default Mathwise resource palette for particular modules. Custom palettes may be designed and assigned to a particular module code using the Palette Editor (mwpaled.exe) as described in *Chapter 5*. They can then be included in the system by referencing them in this section. An entry in [palettes] takes the form name=filename where name is any text, which should be unique within the section, and filename is a path to the custom palette file, e.g. My palette=modules\mymod\mypal.mwp

[modules]

The modules section is where Mathwise is notified of the actual courseware material it will be using. This is done by including references to the Mathwise index files that describe that courseware. Entries in this section have the format name=file-path. name can be any text, as long as it is unique within the section, and you are advised to make it meaningful for your own convenience in maintenance. file-path is the path to a Mathwise index file. This path can be a full path (including drive letter) or a partial path. A partial path that starts with the \ character will be assumed to be on the same drive as the Mathwise system. A partial path that starts with a sub-directory name will be assumed to be a sub-directory of the Mathwise system directory. If a path is omitted altogether then the index file is assumed to reside within the Mathwise system directory. Here are some examples.

```

alastr=s:\modules\alastr\alastr.mwx      ; These indices are on another
alastr leafs=s:\modules\alastr\alastr_1.mwx ; drive from Mathwise
elcplx=\moremods\elcplx\elcplx.mwx      ; These indices are on the same
elcplx leafs=\moremods\elcplx\elcplx_1.mwx ; drive as Mathwise
a5mech=modules\a5mech\a5mech.mwx        ; This index is in a sub-dir
                                           ; below the Mathwise system dir
mwtools=mwtools.mwx                     ; This index is in the MW
                                           ; system directory

```

[associations]

This section is where Mathwise is told how to run the courseware material, which it does by association of a filename extension and a reader application. You will only need to edit this section if you are adding your own material to Mathwise of non-standard file-types (perhaps by autoindexing) or you wish to change the default reader application for standard file types (*.txt files for example). The section consists of 'triplets' of entries in the following format:

```

AssocXName=name
AssocXExt=extension
AssocXPath=file-path

```

There is also an optional entry:

```
AssocXTrack=0 or 1
```

For each of these entries X is a number. Triplets must be numbered sequentially, starting at 1; e.g. Assoc1Name, Assoc1Ext, Assoc1Path is followed by Assoc2Name, Assoc2Ext, Assoc2Path and so on.

The AssocXName entry takes an argument name. This is simply some text that describes the file type being associated, e.g. Assoc5Name=Toolbook 1.53

The AssocXExt entry takes an argument extension. This is the file extension to which the association applies and it can be up to 3 characters long, e.g. Assoc5Ext=tbk

The AssocXPath takes the argument file-path. This points to the application that will be used to run files with the extension given in AssocXExt. As with file-path in [modules] entries it can be a full or partial path depending on where the application is located; e.g. Assoc5Path=d:\tbk\tbook.exe refers to the application required to run *.tbk files being located on a different drive from Mathwise, whilst

`Assoc5Path=tbook.exe` would be used if the application is in the Mathwise system directory (as it is by default). Note that there is a special value of `none` that can be used for the `AssocXPath`; this means that no reader application is required to run the file-type. Basically this only applies to executable files (`*.exe`) and has been defined in the standard `mw.ini` file, so it should not concern you.

The `AssocXTrack` takes an argument of 0 or 1. Unless this is being set to 0 there is no need to include it. Setting this entry to 0 will prevent the Mathwise system from 'tracking' the item once it is launched. Mathwise will make no attempt to work out the window handle of any items of this type launched when `AssocXTrack` is set to 0, making it effectively a 'launch and forget' option. The reason for including this option is because certain applications (such as Microsoft Internet Explorer v3 and later, and Windows Help on NT) appear to cause a system crash when Mathwise tries to work out the window handle of items launched with them.

To get a better idea of how this section is used in practice you are advised to examine the `mw.ini` file that is supplied on the Mathwise CD-ROM.

Entry-unit Initialisation File

This initialisation file is required by the standard Mathwise entry-unit program, `entryu.exe`, and is called `entryu.ini` by default. Note that `entryu.ini` can be overridden by passing a different file name as a command-line parameter when running `entryu.exe`. If you wish to do this for some reason then note that it should be done by using the `_` command line passing form of the `StartupDesc` in the system initialisation file (see above).

The entry-unit initialisation file uses the standard Windows `.INI` format, ie. it is divided into sections each of which contains a range of entries.

[general]

This section contains some general setup parameters for the entry-unit.

`showCodes`

When module names are displayed in the entry-unit's list boxes (under Courseware | Modules, Courseware | Leaflets etc.) then you can optionally specify whether the code for the module is displayed alongside the name. For example the astronomy module can be displayed as 'Applications in Astronomy' (no code) or as 'A1ASTR Applications in Astronomy'. If `showCodes` is set to 1 then this code will be shown; any other value and it will not be shown. The code text is defined in the [moduleCodes] section described below.

`codesColumn`

If `showCodes` is set to display codes then the value of this entry determines where they are shown. If 0 they are shown to the left of the module name, e.g. 'A1ASTR Applications in Astronomy'; if 1 they are shown to the right, e.g. 'Applications in Astronomy A1ASTR'

`columnSpace`

The value of this entry determines the spacing between the columns of module code and module name in pixels, e.g. `columnSpace=50`

`showPrompt`

The entry-unit has a prompt field which shows help text whenever the mouse pointer is moved over an active control (button or list-box). Set `showPrompt` to 1 to enable this or to any other value to disable it.

`euBitmap`

The bitmap displayed in the entry-unit when it is first run is `entryu.bmp` by default. This can be overridden with this entry if desired. Set this entry to point to a BMP file, e.g. `euBitmap=mybitmap.bmp`

`EUWelcomeDesc` and `EUAuthorsDesc`

These entries may be used to override the descriptor queries used to run the help material shown when the 'Using Mathwise' and 'About the Authors' buttons on the Help screen of the entry-unit are clicked. You should not alter these.

`EUAssessDesc` and `EUSysResDesc`

These entries may be used to override the descriptor queries which retrieve the item names listed in the entry-unit under Courseware | Assessments and Resources | System resources respectively. You should not alter these.

`[special]`

This section is used to define the material that is available via the 'Special' menu button of the entry-unit. It allows you to make your own material available to users and/or to specify particular Mathwise courseware items that you want students to have direct access to from the entry-unit. A particular use may be to list material you have added to Mathwise using the auto-index feature described in the System Initialisation File section above. The entries in this section are as follows.

`useSpecial`

To enable the 'Special' button set this entry to 1. Any other value will disable the feature.

`buttonName`

This entry can be used to override the text on the 'Special' button. Set the entry to a string describing what the section will contain. For example if you are at Brainsville University and this section is set up to give access to material specific to that institution then use `buttonName=Brainsville`. Omit this entry and the text will default to 'Special'.

The remaining entries in this section consist of a series of descriptor queries that will determine the items displayed on the Special screen. An entry takes the form `descx=descriptor query`, where `x` is a number and `descriptor query` is a standard Mathwise descriptor query. Multiple entries must be numbered consecutively ie. `desc1, desc2, desc3`, etc. The results of each search will be concatenated together in the order that the queries are listed and the title for each item will then be shown on the Special screen. You will typically use this in conjunction with the auto-index feature to make material specific to your institution available, in which case the first (and possibly only) entry would take the form `desc1=UTIL,AUTO`. Here is a more complex example of entries:

```
desc1=UTIL,AUTO           ; Include autoindexed material
desc2=LEAF,MINE           ; Include leaflets authored by yourself that
                          ; have been given module code MINE
desc3=LEAF,A1AS          ; Include leaflets from the A1ASTR module
desc4=UNIT,,,integration ; Include learning units keyworded with
                          ; integration
```

`[modulecodes]`

This section is used in conjunction with the `showCodes`, `codesColumn` and `columnSpace` entries in the `[general]` section described above. If `showCodes` is set to 1 (ie. codes are shown) then this section describes what code text is actually shown alongside the module name. An entry takes the form `modCode=module code`, where `modCode` is the **four** character module code of the module and `module code` the **six** character code by which Mathwise modules have been organised. If this sounds confusing then don't worry – you don't need to edit this section!

Appendix 6

Windows NT Precautions

Mathwise is a 16-bit application and as such can experience problems when installed on NT based systems. Most 16-bit applications experience these but they usually cause few difficulties to users. However, since Mathwise has to manage and run a great number of files and materials, potentially distributed in many locations, the problems can become more obvious.

The problems arise from 4 causes.

[1] The use of long (more than 8 characters) directory names and directory names containing characters (such as <space>) disallowed under Windows 3.1.

[2] The possibility of mapping network drives to long drive names (aka Universal Naming Convention - UNC) under NT.

[3] The sophisticated methods of setting privileges and sharing permissions under NT.

[4] Problems that some of the third party software used by Mathwise to run courseware material.

Solutions

We have found that Mathwise will work best under NT if you observe the following precautions.

[1] Do not install the Mathwise system or courseware material on paths that contain long (greater than 8 characters) or 'forbidden-character' bearing directory names.

The permitted characters are the following fourteen special characters:

`_ ^ ! # % & - { } () @ ` '`

as well as the letters A to Z and the digits 0 to 9.

[2] Do not connect to the network drive containing Mathwise using an NT long drive (UNC) name. Instead map it to a single letter drive name. For example if, from a client machine, Mathwise is located in the following NT network directory:

```
\\NTServer_1\Apps\Mathwise
```

then you should map this drive to a single letter drive thus:

```
N:\Apps\Mathwise
```

The short cut you set up to run Mathwise should then point to `N:\Apps\Mathwise\mathwise.exe`, and the Start In directory set to `N:\Apps\Mathwise`.

[3] Ensure that all Mathwise file and directory permissions are set to Read Only (RX) apart from the assessment directories to which write access should be given. Do this first. Then set Share Permissions to the same.

[4] Ensure that `mw.ini` is edited correctly for the 'view' from the client machine. This applies in particular to the `[modules]` and `[associations]` sections. See *Appendix 5* and `custom.doc` for more information on editing `mw.ini`.

As a brief example if you install Mathwise to the C: drive of the server, say into `C:\apps\mathwise`, and the modules are copied into the `\modules` sub-directory of this, then the `[modules]` section of `mw.ini` will contain entries such as this:

```
m1grph=c:\apps\mathwise\modules\m1grph\m1grph.mwx
etc.
```

If Mathwise is then run from a client, where the network drive has been mapped to N:, then Mathwise will not be able to find the modules. The entries should therefore be altered to read:

```
m1grph=n:\apps\mathwise\modules\m1grph\m1grph.mwx  
etc.
```

or

```
m1grph=modules\m1grph\m1grph.mwx  
etc.
```

You should also make sure that entries in the [associations] section of mw.ini for text and help files point to the correct Windows directory for the client machine.

[5] The courseware authoring system Toolbook which much of the Mathwise material is authored in is known to have difficulties with fragmented NTFS partitions under NT4. This is apparently due to Toolbook's incremental loading of data which is frustrated by disc fragmentation. The only solution to this is to ensure that any NTFS partition to which Mathwise is installed is properly defragmented (this requires a third party utility since a defragmenter is not included with NT4). Of course ensuring that server drives are defragmented regularly should be a standard procedure for any network administrator in order to maximise performance.

[6] Under NT, if one 16-bit application crashes, all other 16-bit applications also fail. It is, however, possible to force the application to run in its own 'virtual machine' so that if it crashes, the others do not. This is done by selecting the shortcut's properties and checking the 'run in seperate memory space' check-box in the 'shortcut' tab.