

MACINTOSH PASCAL

A Hobbyist's Guide
to Programming the
Mac OS in Pascal

VERSION

1.1

by
Koryn Grant
and
K. J. Bricknell

Adapted by Koryn Grant
from the book MACINTOSH
C: A Hobbyist's Guide to
Programming The Mac OS
in C by K. J. Bricknell

For CodeWarrior

MACINTOSH PASCAL:

A Hobbyist's Guide to Programming the Mac OS in Pascal

Version 1.1

©1997, K. J. Bricknell

Macintosh Pascal was adapted from the book **Macintosh C: A Hobbyist's Guide to Programming the Mac OS in C**. Portions of **Macintosh C** were adapted from the **Inside Macintosh** series of books and **develop** magazine, © Apple Computer, Inc. All rights reserved. Used with the permission of Apple Computer, Inc.

Apple, the Apple logo, LaserWriter, and Macintosh are trademarks of Apple Computer Inc., registered in the United States and other countries.

Classic is a registered trademark licensed to Apple Computer Inc.

Finder and QuickDraw are trademarks of Apple Computer Inc.

Metrowerks is a registered trademark of Metrowerks Inc. CodeWarrior is a trademark of Metrowerks, Inc.

PostScript is a trademark of Adobe Systems incorporated, which may be registered in certain jurisdictions.

No warranty or representation is made, either express or implied, with respect to this manual, its quality, accuracy, or fitness for a particular purpose. As a result, this manual is distributed "as is", and you, the distributee, are assuming the entire risk as to its quality and accuracy. In no event will the author be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect or inaccuracy in this manual.

CONTENTS

1 System Software, Memory, and Resources

System software overview: the Toolbox; the Operating System; location of system software routines. Memory: the system partition; the application partition; nonrelocatable and relocatable blocks; heap fragmentation, compaction and purging; master pointer tag byte; temporary memory; virtual memory; addressing modes, Memory Manager errors. Resources: resources and files; resources and the application; resource types and IDs; creating resources; resource attributes; template resources and definition resources; reading in resources; purgeable resources; releasing resources; Resource Manager errors. System Software Development Implications - Memory.

2 Low-Level and Operating System Events

The main event loop. Processes and events. Categories of events. Low-level events and Operating System events. Obtaining information about events. Handling events: mouse events; keyboard events; update events; activate events; disk-inserted events; null events; suspend and resume events; mouse-moved events. Handling events in alert boxes and dialog boxes. The 'SIZE' resource.

3 Menus

Types of menus. Pull-down menus: menu definition procedures and menu bar definition functions; the menu bar, menus and menu items; the Apple menu; the File menu; the Edit menu; the Help menu; the Application menu; Font menus. Pop-up menus: pop-up control definition function; use of Control Manager routines; type-in pop-up menus. Hierarchical menus. Menu records, menu IDs, item numbers and menu lists. Creating menus. Changing the appearance of items in a menu. Adding items. Handling menu choices. Accessing menus from alert and dialog boxes.

4 Windows

Standard window elements. Active and inactive windows. Types of windows. Window definition IDs. Window type usage. Window regions. The window list. Graphics ports. Window records. Creating windows. Positioning windows. Managing multiple windows. Handling events in windows. Moving, zooming, resizing and closing windows. Hiding and showing windows.

5 Controls

Standard controls: buttons; checkboxes; radio buttons; pop-up menus; scroll bars. Custom controls. Visual feedback. Active and inactive controls. Hiding and showing controls. The control definition function. Creating and displaying controls. Handling mouse events in controls. Determining and changing control settings. Moving and resizing scroll bars. Scrolling operations with scroll bars.

6 Dialogs and Alerts

Types of alert: alert sound; note alert, caution alert, and stop alert boxes. Types of dialog boxes: modal dialog box; movable modal dialog box; modeless dialog box. Items in alert and dialog boxes. Creating alerts: resources. Creating dialog boxes: the dialog record; resources. Default buttons. Enabling and disabling items. Editable text items. Manipulating items. Adding items. Drawing the default button bold outline. Displaying alert and dialog boxes. Adjusting menus. Handling events. Event filter functions. Closing dialog boxes.

7 Finder Interface

The Finder. Resources, the catalog file and the desktop database. Application signature, creator, and file types. Creating icon resources for the Finder. The file reference resource. The bundle resource. How and when the Finder launches an application. Missing application name string and application missing string resources. Version resources. Using Finder information in the catalog file: Finder flags. Supporting stationery pads. Providing balloon help. Using aliases. Using the System folder and its related directories.

8 Required Apple Events

Apple events: attributes and parameters; interpreting attributes and parameters. Data structures within Apple events. Handling Apple events: extracting and checking data; performing the requested action and returning a result. Required Apple events: contents and required action.

9 QuickDraw Preliminaries

QuickDraw and imaging. Versions of QuickDraw. Graphics ports: bitmaps and pixel maps; printing graphics ports; offscreen graphics worlds. Basic QuickDraw's eight-colour system. Color QuickDraw routines available to Basic QuickDraw. Colours in Color QuickDraw: device-independent colour; influence of the video device; indexed colour and direct colour. Graphics devices and `GDevice` records. Other graphics managers.

10 Basic QuickDraw

Mathematical foundations of QuickDraw: the coordinate plane; points; rectangles; regions. The basic graphics port. Drawing in basic graphics ports: the graphics pen; bit pattern; boolean transfer modes; lines; rectangles, ovals, arcs, and wedges; polygons, regions, and pictures. Drawing text. manipulating rectangles and regions. Copying bits between graphics ports.

11 Color QuickDraw

RGB colours. Colour graphics ports. Differences between basic and colour graphics ports. Pixel maps. Translation of RGB colours to pixel values: indexed devices; direct devices. Colours on grayscale screens. Pixel patterns: pen pixel pattern; fill pixel pattern; background pixel pattern. Testing for the existence of Color QuickDraw. Working with Color QuickDraw: creating colour graphics ports; drawing with different foreground colours; drawing and filling with pixel patterns. Copying pixels between colour graphics ports: distinguishing between bitmaps and pixel maps; boolean source modes with colour pixels; arithmetic transfer modes. Highlighting. Color QuickDraw and Text.

12 Offscreen Graphics Worlds, Pictures, Cursors, and Icons

Offscreen graphics worlds: creating an offscreen graphics world; setting the graphics port; preparing to draw; copying an offscreen image to a window; updating and disposing of offscreen graphics worlds. Pictures: picture formats; the Picture record; opcodes; colour pictures in basic graphics ports; 'PICT' files, resources and scrap format; the Picture Utilities; creating pictures; opening and drawing pictures; saving pictures; gathering picture information. Cursors: cursor movement, hotspot, visibility and shape; creating custom non-animated cursor resources; changing cursor shape and hiding cursors; creating an animated cursor. Icons: icons and the Finder; other icons (icons, colour icons and small icons); icons in windows, menus, and dialog boxes; drawing and manipulating icons; icon families, suites and caches.

13 Printing

The Printing Manager. Printer drivers: types and characteristics; QuickDraw printer drivers; PostScript printer drivers; background printing, deferred printing, and spool files; printer drivers and Picture comments. Printer resolution. Page and paper rectangles. Job dialog box. Style dialog box. The `TPrint` record. The printing graphics port. Print status dialog boxes and idle procedures. The printing loop. Getting and setting printer information. Text on the screen and the printed page. Altering the style or job dialog box. Printing from the Finder.

14 Files

Macintosh files. Characteristics of files: file forks; file size; file access. The hierarchical file system: directories and directory ID; root directory; mounted volumes; parent directory and parent directory ID; aliases. Identifying files and directories. General File menu and required Apple events handling strategy. Creating a document record and a new document window. Opening a file and reading in data. Saving a file. Reverting to a saved file. Closing a file. Customized open and save dialog boxes.

15 More on Resources

Search path for resources: current resource file; default search order; setting the current resource file; restricting the search to the current resource file. Detaching and copying resources. Creating, opening and closing resource forks. Reading and manipulating resources. Writing resources. Partial resources. Preferences files.

16 Scrap

The Scrap Manager and the desk scrap: scrap data formats; location of the desk scrap; getting information about the desk scrap; using the desk scrap; the Clipboard; transferring the desk scrap to disk. Private scrap. Copying data between private scrap and the desk scrap. TextEdit, dialog boxes and the scrap.

17 Text and TextEdit

More on text: characters; character sets and codes; glyphs; typefaces; styles; fonts; font families; system font and application font; the Font Manager and QuickDraw. Aspects of text editing: caret position; text offsets; selection range; insertion point; highlighting. Keyboards and text. Introduction to TextEdit: editing tasks performed by TextEdit; TextEdit options; caret position and movement in TextEdit; automatic scrolling; TextEdit private, null, and style scraps; text alignment; customising TextEdit; primary TextEdit data structures. Monostyled TextEdit: initialising TextEdit; creating and disposing of a monostyled edit record; setting the text of an edit record; responding to events; cutting, copying, pasting, inserting, and deleting text; setting the selection range or insertion point; enabling, disabling, and customising automatic scrolling; saving and opening TextEdit documents. Multistyled TextEdit: style runs, text segments, font runs, and character attributes; additional data structures; creating a multistyled edit record; setting the text; cutting, copying, pasting, inserting, and deleting text; scrolling text; setting and checking text attributes; saving and opening multistyled TextEdit documents. Formatting and displaying dates, times, and numbers: the Text Utilities and international resources; date and time value representations; obtaining date-time values and records; converting date-time values into strings; converting date-time strings into internal numeric representation; numbers and number format specification strings; integers; converting between floating point numbers and numeric strings.

18 Lists and Custom List Definition Functions

Appearance and features of lists: cells; cell font; cell highlighting. Scroll bars and size boxes. Selection of cells using the mouse: multiple cell selection using the default cell-selection algorithm; customising the cell-selection algorithm. Selection of cells using the keyboard: moving the selection using arrow keys; extending the selection using arrow keys; type selection. Creating lists: the list record and other data types; drawing borders; adding rows and columns; disabling and enabling automatic drawing mode. Responding to events. Getting and setting list selections. Scrolling a list. Storing, adding to, and clearing cell data. Searching a list. Changing the current list. Customising the cell-selection algorithm. Custom list definition procedures.

19 Custom Control Definition Functions and VBL Tasks

Control definition functions: declaration; default dragging and custom dragging; responding to message parameter values. Vertical blanking (VBL) tasks: VBL tasks and the Vertical Retrace Manager; Types of VBL tasks; VBL task rules; VBL tasks and foreground/background switching; installing and removing a VBL task.

20 Floating Windows and Custom Window Definition Functions

Floating windows: front-to-back ordering of screen objects; appearance of floating windows; implementation considerations; substitute and supporting routines. Custom window definition functions: resource IDs; general requirements; responding to messages.

21 Sound

Introduction to sound: audio hardware; sound-related system software; sound input and output capabilities; basic and enhanced sound capabilities; sound data; sampled sound; sound components; sound resources and sound files. Sound production: sound channels; sound commands; synchronous and asynchronous sound; playing sound resources and files. Sound recording: recording sound resources and sound files; recording quality; checking for sound recording capability. Speech: generating speech from a string; checking for speech capabilities.

22 Miscellany

Code segmentation and heap space optimisation. Status bars and scanning for a Command-period event. Notifications from applications in the background: the need for the Notification Manager; examples of notifications; elements of a notification; suggested notification strategy; creating a notification request; installing and removing a notification request. Soliciting a colour choice: colour models; the Color Picker; invoking the Color Picker. Ensuring compatibility with the operating environment: getting operating environment information using the `Gestalt` function; determining whether a trap is available. Coping with multiple monitors: image optimisation; window zooming.

23 Porting to the Power Macintosh

The 68LC040 emulator. The Mixed Mode Manager: mode switches; intervention in mode switching; creating a routine descriptor; effect of the routine descriptor; routines requiring routine descriptors. The PowerPC native environment: fragments; categories of fragments; fragment storage and loading; code fragment resource; fat applications; accelerated resources; fat resources; calling conventions; organisation of memory; demise of the A5 world; accessing global variables from detached code; data alignment. Source code changes — Chapters 1-22 demonstration programs.

PREFACE Version 1.1

MACINTOSH PASCAL: A Hobbyists Guide to Programming the Mac OS in Pascal

This book was adapted from the book **Macintosh C: A Hobbyist's Guide to Programming the Mac OS in C**. Macintosh C relies very heavily on information contained in the principal ten volumes of the Addison-Wesley publication **Inside Macintosh**. Some demonstration programs in Macintosh C include the author's translations of Pascal code examples in that publication. In addition, parts of Chapters 20 and 21 rely on information contained in Issues No 11 and 15 of **develop** (The Apple Technical Journal). Apple Computer, Inc, which holds the copyright to those publications, kindly consented to the author distributing Macintosh C on the Internet, on-line services, and bulletin boards as a free publication. That consent has been extended to include this, the Pascal variant of Macintosh C.

From the Author of Macintosh C - Origins and Purpose of Macintosh C

Some time ago, I decided to teach myself to program my recently acquired Macintosh in the C language. Given that my previous foray into the world of hobbyist programming occurred in the early eighties, it was not long before I became acutely aware that times had changed — and with a vengeance! First came the development system purchase, then three introductory books on programming the Macintosh in C, then a book documenting a thing called a resource editor, then an on-line guide to the system software, then a book on user interface guidelines, and then an interactive tutorial. After wading through all that, it became increasingly obvious that I would never get myself to where I had to be unless I purchased no less than ten volumes of the Inside Macintosh series, the monumental Addison-Wesley publication that documents the Macintosh system software. Then, because all of the code examples in those ten volumes of Inside Macintosh are in Pascal rather than C, I had to buy a book on Pascal and learn enough about that language to be able to work out exactly what those code examples were telling me. By this time, my wife was looking somewhat askance at what I continued to insist was nothing more than a simple hobby aimed at the constructive utilisation of odd moments of my spare time.

Discussion with other hobbyists suggested that, while many beginners are quite willing to do what it takes to learn to program their machines in this era of increasingly sophisticated system software, the cost and volume of all that material, coupled with the Pascal-to-C translation task, was a major turn-off for the struggling amateur. Professionals, I concluded, need Inside Macintosh, but the beginning hobbyist needs a gentler (and, above all, cheaper) introduction to all this complexity.

Having arrived at that conclusion, I decided to turn my notes into a full-blown manual in the belief that I just might be able to save other amateurs from what many would regard as cruel, unusual, and pocketbook-depleting punishment. Macintosh C, then, represents my attempt to provide an easier and more economical entry point to Macintosh programming for the beginning hobbyist.

About four months after Macintosh C was published on the Internet, Koryn Grant approached me with the idea of adapting the book and demonstration program package to the Pascal language. I readily agreed to this excellent idea, especially since Koryn volunteered to do all the work! Koryn's work

proceeded in parallel with my work on a major revision of Version 1.0 of Macintosh C, the result being that the first version of Macintosh Pascal is the equivalent of Version 1.1 of Macintosh C.

K. J. Bricknell
Canberra
Australia
January 1997

From the Co-Author of Macintosh Pascal

Part of K.J.'s opening paragraphs in Macintosh C, reproduced above, hint at my motivation for translating his book into Pascal:

"Discussion with other hobbyists suggested that ...the cost and volume of all that material, coupled with the Pascal-to-C translation task, was a major turn-off for the struggling amateur."

My reasoning was that, instead of performing the Pascal-to-C translation task, why not simply learn to program the Macintosh in Pascal in the first place? After all, in an ideal world we would all be programming in Pascal! However reasonable this may sound, there are some practical difficulties. Much of the Pascal code in Inside Macintosh is now out-of-date and does not work with the latest Universal Interface files as it is written; C seems to be established as some sort of de-facto standard in the programming world; and the majority of "introduction to Macintosh programming" books (certainly all the ones I have) are in C.

On the other hand, there are advantages in having an introductory Mac OS programming book written in Pascal. Pascal code is usually more readable and understandable than C code, a definite advantage when you are trying to get to grips with the Macintosh Toolbox. The Toolbox itself was originally written in Pascal (although new Apple technologies are now being written in C), putting Pascal on a somewhat distinguished footing among programming languages when it comes to learning the current Mac OS. Finally, and this was probably the most important consideration to me, it seems that there are a large number of hobbyist programmers who have learnt to program in Pascal but not in C. It makes sense that they should be able to learn to program the Mac in a language they already know.

In summary, this translation of Macintosh C is aimed fairly and squarely at those hobbyist programmers who are comfortable with Pascal and want to learn how to program the Macintosh without the added complication of learning a new language. And since Metrowerks CodeWarrior provides what is arguably the best Pascal development environment, the demonstration programs that accompany Macintosh Pascal are CodeWarrior projects.

The remainder of Macintosh Pascal, including the following sections in this preface, is adapted directly from Macintosh C. It is to K.J.'s credit that he was able to write a book on programming the Macintosh in C that kept the Macintosh aspect sufficiently disjoint from the C implementation that the text from Macintosh C was able to be reproduced almost verbatim in Macintosh Pascal. It is also very much to his credit that he has been more than willing to allow his book to be adapted for the benefit of, firstly, the Pascal programming community and, ultimately, I hope, the Macintosh community at large, and for this I thank him.

Koryn Grant
Canterbury
England
January 1997

OverView of Macintosh Pascal

Essentially, Macintosh Pascal covers all of the territory which, in the judgement of the authors, needs to be covered before you write your first serious application. This includes, for example, how to create and manage all elements of the user interface (menus, windows, controls, dialogs, alerts, lists, etc.), how to ensure that your application observes the house rules of the Macintosh graphical user interface

and cooperative multitasking environment, how to perform file input/output, how to print files, how to draw text and graphics, and so on.

Considerable thought has been given to the sequence in which each topic is introduced, the content of most chapters relying to some extent on a full understanding of what has gone before. Accordingly, you should note that Macintosh C is not intended to be a randomly-accessed reference work; rather, it should be regarded as more in the nature of a tutorial in which each chapter should be worked through in sequence.

The general structure of all but two chapters of Macintosh Pascal is the same: first comes the information, then a list of constants, data types and routines relevant to the subject of that chapter, then the source code listing of a demonstration program related to the subject of that chapter, and, finally, line-by-line comments which explain the workings of the source code.¹ Some chapters also include instructions on how to create the associated demonstration program's resources.

The book itself is supported by the CodeWarrior project files, source code files, and resource files for all demonstration programs.

What You Will Need

Development System

Apart from Macintosh C you will, of course, require a development system. This edition of Macintosh C assumes that that system will be Metrowerks CodeWarrior.

The Metrowerks product Discover Programming For Macintosh includes full-featured Pascal tools for 680x0-based Macintoshes. The included 680x0 compiler, which produces code which will run on 680x0-based Macintoshes (and in emulation on PowerPC-based Macintoshes), will be sufficient for Chapters 1 to 22. The significantly more expensive CodeWarrior Gold, which, amongst other things, adds a compiler capable of producing code which will run native on PowerPC-based Macintoshes, could be useful when you get to Chapter 23 — Porting to the Power Macintosh; however, it is by no means essential.²

On-Line Reference

An on-line reference enables you to quickly and easily access information relating to the system software, and is thus quite indispensable. You can choose between THINK Reference³ (which is to some extent out-of-date but still very useful) and Apple's CD-ROM-based Macintosh Programming Toolbox Assistant.

Resource Editor

A resource editor allows you to create resources for programs and files. A copy of the resource editor ResEdit, including the manual, is included with the CodeWarrior package.

Other Tools

Another useful tool is ZoneRanger, a dynamic memory inspection tool that allows you to investigate how effectively and efficiently your application uses memory. ZoneRanger is included with the CodeWarrior package. You will also find a programmer's calculator very useful for converting between decimal, hexadecimal and binary values, the nicely-presented shareware program CalcWorks being ideal for that purpose.

¹Note that the marginal line numbers are included in the source code listings only to facilitate referencing from the comments section. This is not some strange line-numbered version of Pascal

²Specially-priced academic versions of CodeWarrior Gold are available for students. Information on Metrowerks CodeWarrior products, including system requirements, is available at <http://www.metrowerks.com/>

³THINK Reference was originally marketed by Symantec but is now available on a CD-ROM produced by MacTech magazine. See the MacTech CD-ROM section at <http://web.xplain.com/mactech.com/>.

Demonstration Programs

All of the demonstration programs may be run from within CodeWarrior with the exception of the program that accompanies Chapter 8 — Required Apple Events. By its nature, this program should be run as a built (that is, double-clickable) application. The demonstration program at Chapter 14 — Files may be run within CodeWarrior, although certain aspects of the program can only be explored by running it as a built application. Only two programs (one at Chapter 9 — QuickDraw Preliminaries and one at Chapter 11 — Color QuickDraw) will not run on black-and-white Macintoshes such as the Classic.

As far as is possible, each demonstration program avoids making calls to system software routines that are only explained in a later chapter. However, achieving that ideal has not been possible in the demonstration programs associated with the earlier chapters. For example, the demonstration program associated with Chapter 1 must, of necessity, make calls to system software routines relating to windows (the subject of Chapter 4) and drawing in a graphics port (the subject of Chapter 10). Where this occurs, you should simply accept, on faith, that the associated source code does as is stated in the demonstration program comments section. The important thing is to concentrate on that part of the source code pertaining to the subject of the chapter with which the program is associated.

System Software Assumptions

One of the banes of the programmer's existence is the necessity to ensure that a program will run successfully under various versions of the system software. Macintosh Pascal addresses the matter of compatibility; however, in order to avoid endless digressions to account for what must surely be a very, very small percentage of the overall Macintosh population, Macintosh Pascal contains no material explaining or demonstrating the measures required to accommodate versions of the system software earlier than System 7.0.

Coping With Change

The hobbyist programmer lives in difficult times. Until comparatively recently, learning to cope with the complexities of the Macintosh system software was challenge enough. Then along came the Power Macintosh, with its PowerPC microprocessor, to add to that challenge. . And now, looming on the near horizon at the time of writing (January 1977), is the reality of significant new developments in the Macintosh system software arena.

Coping With the Power Macintosh

So far as coping with the Power Macintosh is concerned, the approach taken by Macintosh C is to stay firmly and exclusively lodged in the world of the 680x0 microprocessor (whether it be implemented in hardware (680x0 Macintoshes) or in software (the emulator in PowerPC-based Macintoshes)) for the first 22 Chapters. Then, at Chapter 23, the consequences of the PowerPC microprocessor are addressed, including an explanation of the modifications which must be made to the source code of previous demonstration programs if that code is to be compiled as native PowerPC code. (The folder chap23cw_demo contains the revised versions of those source code files requiring such modification.)

However, even if you had no intention of writing or modifying source code for compilation as PowerPC code, or if Macintosh C did not contain Chapter 23, the reality is that you could not escape the influence of the Power Macintosh. The culprits in this regard are the so-called Universal Headers files, which were introduced at the same time as the Power Macintosh and which, amongst other things, enable you to write source code capable of being compiled as either 680x0 code or PowerPC code — hence the term "Universal".⁴

Influence of the Universal Interfaces

You will see things in the Universal Interfaces which appear to be inconsistent with information in such references as THINK Reference and Inside Macintosh, and with various source code examples you may have seen or will see. Do not despair; the reasons for these apparent inconsistencies will be explained

⁴The Universal Headers are included in the CodeWarrior package.

as you go along. Your first major encounter in this regard will be in a dissertation on the well-known system software routine `TrackControl` in the demonstration program comments section at Chapter 5 — Controls.

At the time of their introduction, the Universal Interfaces not only accommodated the reality of the Power Macintosh and its PowerPC processor but also looked ahead to intended developments in the system software. Once again, the result was apparent inconsistencies. For example, you will see the relatively new data type `WindowRef` in Macintosh Pascal in contexts where THINK Reference, Inside Macintosh, and older source code examples would lead you to expect the `WindowPtr` data type. All this results from the intended introduction of opaque data structures with Mac OS 8, an avenue in the development of the system software which has now been abandoned.

Because Mac OS 8 was abandoned, opaque data structures did not become a reality. This means that the `WindowRef` data type is now little more than the legacy of a failed endeavour. In these circumstances, all that is required is to remember that, whenever your compiler sees `WindowRef`, it thinks it is seeing `WindowPtr`. The same applies to, for example, `MenuRef` and `MenuHandle`, `ControlRef` and `ControlHandle`, `DialogRef` and `DialogPtr`, and `ListRef` and `ListHandle`.

Future System Software

Until comparatively recently, Mac OS 8 was the future of the system software. Then, in late 1996/early 1997, Apple abandoned Mac OS 8, acquired NeXT Software, Inc, and announced a bold new direction in system software development, the essence of which is that, for the next several years, Apple will develop and support *two* operating systems (OSs):

- **Mac OS.** The first system will be the Mac OS, which will continue to be upgraded and improved.
- **Rhapsody.** The second system will be a new OS based on NeXT Software's operating system technologies. This new OS is currently code-named Rhapsody.

Apple will continue to advance the Mac OS until its customers transition to Rhapsody. Apple expects that transition to take several years.

Mac OS Compatibility Environment. In addition to leveraging the NeXT technologies, Rhapsody is designed to run Mac OS applications through a Mac OS compatibility environment. This environment will be a complete implementation of the current Mac OS hosted on the modern operating system infrastructure provided by Rhapsody. The basic architecture is shown at Fig 1.

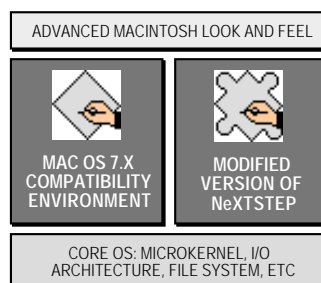


FIG 1 - BASIC RHAPSODY ARCHITECTURE

The foundation of Rhapsody will be a modern microkernel designed to provide pre-emptive multitasking, protected memory, and other modern operating system capabilities.

Apple will support the ability to boot either Mac OS or Rhapsody on a single Mac OS-compatible computer. Rhapsody's user interface will combine elements from both the Mac OS and NeXTSTEP, but will be closer in look and feel to the Mac OS Finder.

The intended Rhapsody/Mac OS development path at the time of writing is as shown at Fig 2.

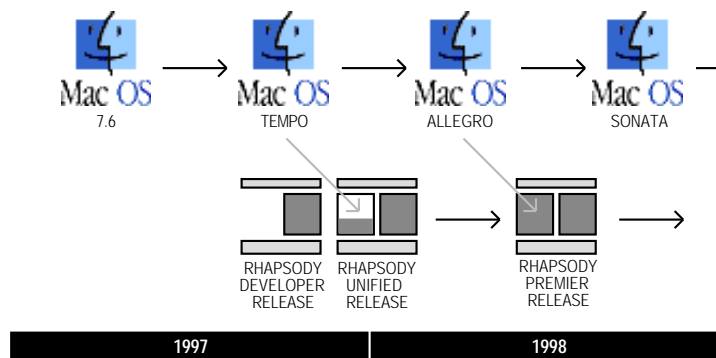


FIG 2 - PROJECTED OPERATING SYSTEM RELEASES

Although this book is concerned with programming the Mac OS, not Rhapsody, it is nonetheless of significance to the hobbyist programmer that Apple intends to allow developers to use the programming languages of their choice for new application development in Rhapsody. Objective C is the native language and will offer some advantages over other languages; however, in conjunction with partners such as Metrowerks, Apple expects to offer Java, C, C++, and Pascal as viable languages.

The upshot of all this for the Pascal hobbyist is as follows:

- Learning to program the Mac OS at this stage in the history of the Macintosh will not constitute wasted effort. Clearly, the Mac OS will be running on hundreds of thousands of Macintosh computers for many years to come. It will be the only OS capable of running on the huge installed base of 680x0-based machines. Furthermore, the knowledge and experience acquired in learning to program the Mac OS, more particularly the graphical user interface (GUI) aspects, should be of at least some assistance when the time comes to learn to program Rhapsody.
- Pascal will be a viable programming language for Rhapsody, as it continues to be for the Mac OS.

The upshot of all this for the author is that someday there will have to be two versions of this book — one targeted at the Mac OS (as now) and one targeted at Rhapsody!

Terminology and Other Sorrows

There are a few terms (or, rather, words) in this book which, depending on your country of residence, may seem only vaguely familiar. Bear in mind that Macintosh C was originally compiled in Australia, a civilised land where spelling conventions equate with those of the country that invented the language, and adapted to Pascal by a New Zealander in the United Kingdom. Hence the word *colour* is generally spelled with a *u*. That said, the *u* has been removed where appropriate — for example, when reference is made to a component of the system software known, officially, as Color QuickDraw. In this way, and at the risk of being accused of inconsistency, the co-authors seek to offend nobody.

Towards Version 1.2

The authors welcome comments and suggestions on Macintosh Pascal Version 1.1. Pascal-specific comments should be addressed to <kdg2@ukc.ac.uk> (before September 1998) or koryn@stephens.manawatu.gen.nz (after September 1998). Comments of a more general nature should be addressed to <brick@spirit.com.au>, <kdg2@ukc.ac.uk> (before September 1998), or koryn@stephens.manawatu.gen.nz (after September 1998).

Koryn Grant
Canterbury
England
January 1997

K. J. Bricknell
Canberra
Australia
January 1997