MACINTOSH PASCAL

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

Version 1.2 (Frozen)

Koryn Grant and K. J. Bricknell

MACINTOSH PASCAL:

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

Version 1.2 (Frozen)

CONTENTS

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 patter; 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 1.2 (Frozen)

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.

Origin and Purpose

Macintosh Pascal: A Hobbyist's Guide to Programming the Mac OS in Pascal is a translation to Pascal by Koryn Grant of the book and demonstration program package **title Macintosh C: A Hobbyist's Guide to Programming the Mac OS in C** by K. J. Bricknell. Macintosh Pascal and Macintosh C represent an attempt to provide a fairly comprehensive entry point to Macintosh programming.

Version 1.1 of Macintosh Pascal was published on the Internet in early 1997. Version 2.1, which brought things up-to-date with Mac OS 8.5, was published in April 1999.

This version (Version 1.2 (Frozen)) is intended only for those who, for one reason or another, must remain in the era of System 7 minus the Appearance Manager. It is essentially Version 1.1 with one or two corrections, with the demonstration program projects files updated for Version 3.0 of the Metrowerks CodeWarrior IDE, and with some minor changes to the source code files to make the code compatible with Version 3.2 of the Universal Interfaces.

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, is 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 Pascal you will, of course, require a development system. This edition of Macintosh Pascal 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 Reference[3] (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.

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.

---

[1] Note 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

[2] Specially-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/

[3] THINK 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/ .

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.

So far as coping with the Power Macintosh is concerned, the approach taken by this version of Macintosh Pascal 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.

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 Pascal 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.

Koryn Grant
Palmerston North
New Zealand
April 1999

K. J. Bricknell
Canberra
Australia
April 1999