MACINTOSH C

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

VERSION 1.1

by
K. J. Bricknell

CodeWarrior Edition

MACINTOSH C:

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

CodeWarrior Edition — Version 1.1

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; customizing TextEdit; primary TextEdit data structures. Monostyled TextEdit: initializing 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 customizing 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; customizing 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. Customizing 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 optimization. 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 optimization; 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; organization 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 C: A Hobbyist's Guide to Programming the Mac OS in C

This book relies very heavily on information contained in the principal ten volumes of the Addison-Wesley publication **Inside Macintosh**. Some demonstration programs include the author's translations into C 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, has kindly consented to the author distributing Macintosh C on the Internet, on-line services, and bulletin boards as a free publication.

Origins and Purpose of Macintosh C

A few years back, 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.

Version 1.0 of Macintosh C was published on the Internet in early 1996. This version (Version 1.1) is the result of a major revision of Version 1.0. It incorporates corrections, refinements, and bug-fixes, and includes additional material.

The First Task — Learn the C Language

The main assumption made by Macintosh C is that you have already learned the C language. Accordingly, if you do not already know C, learning that language will be your first task.

Since the computer in question is a Macintosh, and the development system in question is Metrowerks CodeWarrior, there is probably no better way to for you to learn C than to work through the book Learn C on the Macintosh by Dave Mark. This book, together with its associated example programs, is included with the CodeWarrior package.

Other recommended books are Programming in C by Stephen G. Kochan (Hayden Book Company) and Teach Yourself C by Herbert Schildt (Osborne McGraw-Hill), possibly topped off with Part 1 of C — The Complete Reference by Herbert Schildt (Osborne McGraw-Hill).

As you peruse these works, do not spend too much time on the subject of console input/output, since this has limited application in the world of the graphical user interface. In addition, you can afford to gloss over file input/output at this stage, since Macintosh C examples utilise Macintosh system software routines, rather than C standard library routines, to effect file input/output. (Indeed, it is entirely possible that you will never need to use the C standard library routines.)

The Macintosh C Phase

When you have learned the C language, you are ready to open Macintosh C. As you move through this second phase of the journey, you will quickly discover that learning C was by far the easiest part!

Essentially, Macintosh C covers all of the territory which, in my judgement, 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.

General Structure of Macintosh C

The general structure of all but two chapters of Macintosh C 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.

[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 C!

The Metrowerks product Discover Programming For Macintosh includes full-featured CodeWarrior C/C++ tools for 680x0-based Macintoshes. The included C/C++ 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.

You should read the top section of the source code comments in each chapter before running each program. For most programs, this explains what to do, what to expect, and what to note.

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.

Sorry, No MacHeaders

If you already know C, you will be familiar with the concept of header files and with the lines of code at the top of a source code file which explicitly include the header files required. CodeWarrior relieves the programmer of the requirement to explicitly include the most commonly used header files by

---

[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/ .

automatically including a pre-compiled header file titled MacHeaders68K (680x0 projects) or MacHeadersPPC (PowerPC projects).

Ordinarily, these precompiled headers are summoned up by the file MacHeaders.h which, by default, appears in the Prefix File editable text item in the Project Settings dialog box (Language Settings/C/C++ Language section). MacHeaders.h has, however, been deleted as the Prefix File in all Macintosh C CodeWarrior projects, and all required header files are explicitly included in all source code modules. Although the deletion of this precompiled header adds to compilation time, there is a sound reason for this approach. Familiarising yourself with the contents of relevant header files should be considered to be an integral part of the process of learning to program the Macintosh in C. Accordingly, I would recommend that, every time you see a new header file at the top of a Macintosh C source code listing, you open that file and peruse its contents.

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 C 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 C 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.

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 Headers

You will see things in the Universal Headers 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 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 Headers 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 is apparent inconsistencies. For example, you will see the

[4]The Universal Headers are included in the CodeWarrior package.

relatively new data type `WindowRef` in Macintosh C 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.

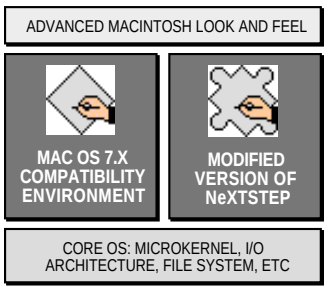

FIG 1 - BASIC RHAPSODY ARCHITECTURE

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.

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.

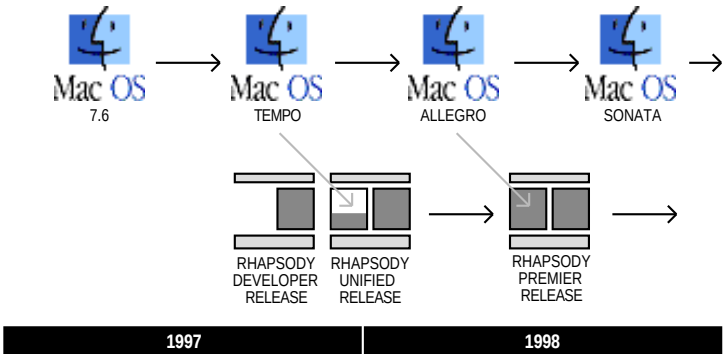

FIG 2 - PROJECTED OPERATING SYSTEM RELEASES

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 C 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.

- C 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

That part of the Mac OS known as the Toolbox uses Pascal calling conventions. Indeed, the fact that the official Mac OS reference work (Inside Macintosh) is Pascal-oriented reflects the fact that Toolbox routines are designed to be called from a Pascal program. It is inevitable, therefore, that certain Pascal terminology will appear in this book.

For example, those not familiar with the Pascal language may wonder why the term *procedure* is occasionally used in this book in a context where the term *function* would be expected. In C, a routine which returns a value and a routine which does not return anything are both called functions. In Pascal, a routine which returns a value is also called a function; however, a routine which does not return anything is called a procedure. Generally, this C-oriented book simply uses the generic term *routine* for what would be referred to in a Pascal-oriented book as either a function or a procedure. However, there are certain cases where use of the term *procedure* has been considered appropriate, hence the appearance of such terms as *hook procedure*, *click loop procedure*, *action procedure*, etc. This is simply a reflection of the apparently magnanimous acceptance by the C community of a term which is meaningless in C but which owes its origins to the historically cosy relationship between Pascal and the system software.

Another term that you may not initially recognise is the term *record*. This is the Pascal equivalent of the C data structure. Once again, this Pascal term is used in this C-oriented book because of accepted convention.

There are a few other terms (or, rather, words) in this book which, depending on your country of residence, may seem only vaguely familiar. Bear in mind that this book was compiled in Australia, a civilised land where spelling conventions equate with those of the country that invented the language. 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, I seek to offend nobody.

Towards Version 1.2

The author welcomes comments and suggestions on Macintosh C Version 1.1, which may be addressed to: brick@spirit.com.au

K. J. Bricknell
Canberra
Australia
January 1997