
I should make one thing clear at once: these books are manuals, appearing in the now well-established tradition of presenting manuals as reference works by otherwise reputable publishers of computer science texts. The (anonymous) author(s) appear to assume that the reader knows what OSF/Motif is for, its advantages and limitations, and how it was developed; I shall do the same in this review.

At first glance, the paperback editions are an impressive sight: nearly two shelf-inches of paper. At closer inspection, we find, however, that they contain enough material for, say, about half-an-inch. It is clear that these books have been thrown together far too quickly, using the venerable troff(1) text formatter. All the usual text formatting errors occur: duplicated lines in different fonts, truly awful word layout on some lines (whatever happened to hyphenation?), and so on. The amount of text per page also varies wildly, with some pages having an immense amount of white space at the bottom. The *Style Guide* has 15 blank pages at the back.

The *User's Guide* assumes that the reader has practically no experience with any kind of interactive, mouse-driven, windowed user interface. This is a pity really, since this kind of system has been widely available on most computers for many years. Mercifully, the author(s) assume that the reader at least knows how to plug in and switch on their machine. This volume is really only for complete novices. The interface itself is in the modern “piano-playing” style, (You know, it goes: (Cntl)(Alt)(Shift) with the left hand, (click left mouse button) with the right hand, sustain pedal . . . once more, with feeling.) unless the user is prepared to wait for a pull-down menu to be painted, and then move the cursor to the desired item—very time-consuming.

The *Programmer's Guide* is the thickest volume—more than a shelf-inch on its own. This tells you things you actually need to know, assuming of course that you are an experienced C programmer who actually wants to implement something using OSF/Motif. This is in three sections, with five appendices at the back.

The first section describes the toolkit of “widgets” and “gadgets” available, and also includes a description of classes, and class-based inheritance, without mentioning the term “object-oriented” even once. This is quite remarkable, but presumably it gives the author(s) leeway to misuse terms which have widely-understood meanings in the object-oriented world. There is the usual obfuscation of the distinction between instances and classes. Worse still, the author(s) appear to believe that a superclass which should not be instantiated is called a “metaclass”, rather than an “abstract” (or “supporting”, or “private”) superclass. (A class is an *instance* of its metaclass, not a subclass of it.)

The second section describes the Motif window manager, and the way in which this can be customized by the user. The final section introduces the “User Interface Language”, a (slightly) higher-level language for describing the user interface part
of an application. This is a conventional textual language; why a graphical interface for building graphical interfaces is not provided is not clear.

The *Style Guide* is perhaps the best of the three books, containing a considerable amount of sensible advice on how to make new application interfaces consistent with (or perhaps merely indistinguishable from) existing applications. This is the best book to read if you actually want to know what Motif interfaces look like.

There is a (different) glossary in all three volumes—some 13 pages in the *Programmer's Guide*. The definitions are usually consistent (although not identical) between the three volumes.

In summary, these are definitely not books to settle and read; these are books to be studied if and only if you actually really need to know the fine details of design, implementation and use of Motif.

Trevor P. Hopkins

Computer Science Department

University of Manchester

Manchester, United Kingdom


When I was asked to review this book, I was told it was funny. Did this mean the specification style or the concepts used, I wondered. No, came the reply, the book was just funny.

And it is. The adventures of Rita Rigorous as she pursues her job as a programmer, over cups of tea and bowls of breakfast cereal, while dealing with such diverse people as Urmston: Upwardly-Mobile and Cynthia Cynical, transform what could be a rather dry subject into a lively and entertaining one.

This is a book not about a particular programming language but about the whole process involved in producing computer programs. There are four stages in this process: requirements, specification, design and implementation. Requirements are expressed informally in English. The specification expresses unambiguously the requirements using formal mathematical concepts. The design produces an algorithm which behaves as stated in the specification. The implementation converts the design into a program written in some given programming language.

The reader learns about the programming process via a series of case studies graduating from simple examples to substantial, nontrivial ones. Each case study is structured according to the above-mentioned programming process: requirements, specification, design, implementation.

Of course, it is necessary to provide vehicles for teaching this process; the specification and design method, VDM, and the programming language Pascal are