effect of the books, and the very high quality of graphic design, layout, printing and production that has gone into them. No library should be without these books, no course on human–computer interaction should ignore them and every aspiring software designer should use them as meditation aids when contemplating how graphics can be used to represent information and interaction, choice and effect.

MATHAI JOSEPH
University of Warwick

ALISTAIR GEORGE AND MARK RICHES
Advanced Motif Programming Techniques. Prentice-Hall. 1994. ISBN 0-13-219965-3. £24.95. 406 pp. softbound.

The X Window System has developed rapidly in the last few years. A fundamental tool which is needed in order to program easily and effectively in X is a *toolkit*—this is a high-level library supporting a set of *widgets*. Several toolkits have been created, but only in the last couple of years has *Motif* begun to emerge as a *de facto* standard.

The concepts on which Motif is based are relatively simple and several good introductions to Motif are available. These all suffer from a common problem: Motif contains a very large (and somewhat complex) collection of routines and even simple GUI design can be effected in a variety of ways. *Advanced Motif Programming Techniques* contains advice and warnings to help a competent Motif programmer to design and write GUIs more effectively and efficiently.

As the title suggests, this is not a book for the novice X programmer. Although the book commences with a short discussion of the concepts underlying X and Motif, this is too brief to serve as a tutorial. It is assumed that the reader can write programs using Motif and has a good working knowledge of C. Discussion of Motif is supplemented with many code fragments which can be obtained either via FTP or on disk (for the latter a nominal sum is charged).

The book is divided into two parts. In the first, the authors discuss the Motif widgets and explain the rationale for them. Down-to-earth advice about good (and bad) techniques for building complex GUIs is given. These are illustrated by numerous examples of commonly-encountered situations. In the second part of the book other relevant topics are looked at, including: drawing using *Xlib*, use of colour, interaction between Motif and window managers, resources, and low-level X event handling. Some of the paragraphs are annotated with warning triangles, indicating traps for the unwary programmer. I liked this approach. The authors' style is very clear and I had no difficulty following their discussions.

Advanced Motif Programming Techniques is not a book to be read from cover to cover in one sitting; nor is it a reference manual. It is a volume to be consulted for good ideas and for common-sense. It is a valuable complement to your favourite Motif reference books.

MIKE JOY
University of Warwick

ZHAOHUI LUO
Computation and Reasoning. Oxford University Press. 1994. ISBN 0-19-853835-9. £30.00. 228 pp. hardbound.

Type theory concerns the formal study of the idea that entities in common use by computer scientists can be organized into collections with uniform properties (called *types*) together with rules for computing members of types. This gives rise to a conceptual framework for computer science. The book *Computation and Reasoning* is a comprehensive account of (the meta-theory and application of) one particular type theory, the *Extended Calculus of Constructions* (ECC), developed by Chaohui Luo in his recent PhD thesis.

The ECC provides a very rich type system together with an associated language of terms (programs). The syntax of the ECC is introduced, along with computational rules for evaluating terms to values (the results of programs). The formal presentation of the syntax is very clear and the informal explanations of the type theory guide the reader through the intended meaning of the ECC. Results about the meta-theoretic properties of the ECC are proved, such as principal typing and subject reduction. The ECC is shown to be strongly normalizing, i.e. every well typed program computes to a value in a finite number of steps. The method of proof is based on a well known technique (Girard–Tait reducibility) which is adapted in an interesting and novel way for the ECC; this proof will be of interest to the experts. The ECC contains a higher order logic, within which one may reason about programs; there is a concise explication of the logic and the proof of its consistency. A set-theoretic model of the ECC is described, which uses ideas from recursion theory. At first sight, the syntax of the ECC will (probably) seem quite complicated and this semantic model will aid the reader's understanding. However, to understand the model will require some knowledge of category theory. Examples of how to express computational theories in the ECC are given, and there is a very abstract account of how to specify and implement programs in the ECC. The book concludes with a lengthy account of a very general type-theoretic framework in which other type theories (such as the ECC) may be presented.

I found this to be a well written, thorough and enjoyable account of the ECC. As well as a detailed exposition of the meta-theory of the ECC, there are some lengthy philosophical discussions about type theory in general and its role in computer science. The author compares and contrasts the ECC with other well known type theories, in particular showing *how* the ECC extends the Calculus of Constructions and why the ECC may be