

Preface

After several years of programming, it became evident to us that traditional programming languages had many shortcomings that made programming inefficient and more difficult than it needed to be. This became even more obvious when we were teaching computer programming to naive students. Textual programming languages such as C or C++, while the current standard for corporate programming, possessed syntactical oddities that could baffle new programmers and experienced programmers alike. While these language constructs afforded powerful tools, they also opened the door to bug-prone application programs.

Then came Prograph. The first advertisements and reviews for Prograph made it look as much like a programming language as the Starship Enterprise looks like a Volkswagen Beetle. Programs were written by connecting graphical shapes with a mouse. Programming just couldn't be that simple! And yet it *was* with Prograph. Here was a new type of language -- a *visual* language -- that meshed with the graphical user interfaces of modern computers. Just as new computer users found graphical user interfaces easier to use than command-line interfaces, new computer programmers took to visual programming like a duck to water. Differentiating between different program elements like operations, constants, loops and classes with uniquely-shaped graphical symbols made program code easy to understand at a glance. Try doing that with a C program!

And it wasn't just new programmers who found Prograph to be revolutionary. Experienced corporate programmers learned that they could program in Prograph in a fraction of the time needed to program in conventional languages. Prograph was more than a language. It was an integrated program development environment for writing code, testing code and designing user interfaces that reduced the amount of code that needed to be written and minimized the possibilities of error in program code. Software developers could spend more time prototyping and experimenting with their code than worrying about its picky little details.

Join us for a detour into the future of programming -- a new programming paradigm that takes a bold new step in programming efficiency. Visual programming is here and Prograph is leading the way.

This book is intended for two audiences. The first is novice programmers who are computer literate but who have shied away from programming due to the daunting task of learning the complex syntaxes of conventional textual programming languages. The second is experienced programmers who need to improve their efficiency and rapidly produce polished bug-free prototypes or finished software. Both audiences have a lot to gain from visual programming and Prograph's integrated professional software development tools.

Acknowledgments

The authors would like to thank Dr. Marjan Bace of Manning Publications and Dr. Ted Lewis, the series editor, for their valuable assistance during the preparation of this book. We would also like to thank the reviewers of the manuscript at various stages in its development -- Dr. Takayuki Dan Kimura, Dr. Ephraim Glinert, Henry Cates, Mark Szpakowski of Prograph International, and our students at the University of Missouri-St. Louis and the West Kings District High School in Auburn, Nova Scotia.

Dr. Steinman is especially grateful for the assistance of his wife, Dr. Barbara Steinman, who served as reviewer, proofreader and eager student of the Prograph language. He is especially proud of her use of Prograph to program the psychophysical experiments for her Masters and Doctoral research projects.

Assumptions about the Reader

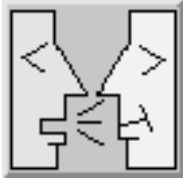
This book assumes that the reader is an experienced computer user who has programmed in at least one textual programming language. Wherever possible, the equivalent code in the C++ programming language will be shown to provide a point of reference for the reader. However, the reader does not need to be fluent in C++ to understand the Prograph programming language. The reader should also have read the Prograph CPX Tutorial and be well-versed in navigating through the Prograph CPX programming environment and in the mouse and menu operations needed for creating Prograph program elements. Therefore, in order to focus upon programming issues rather than the mechanics of dealing with the Prograph environment, we will not discuss these precise mouse and menu operations.

Programming Exercises

At several points in the book we include programming exercises for the reader that build upon the concepts covered in the previous section. These exercises are not mandatory, but may help the reader get a complete grasp of Prograph programming concepts. Programming is learned not just by reading -- it is also learned by *doing*.

Conventions Used in this Book

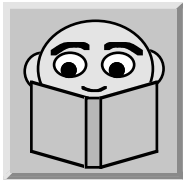
- The text of this book is printed in the Times font, with important concepts in *italic* text.
- Prograph primitive and external code names are printed in `Courier` text.
- Prograph class, class method and class attribute names are printed in `Helvetica` text.
- Special icons are used to draw the reader's attention to important points:



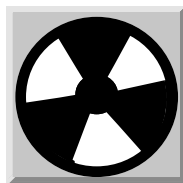
“By The Way...” icons are used to present additional information for the inquisitive reader.



“A Hint...” icons are used to show helpful facts or techniques that make Prograph programming easier.



“For More Information...” icons direct the reader to other books or documentation that cover material beyond the scope of this book.



“Warning!” icons present critical information that can prevent the reader from making costly mistakes.