Advanced Graphics in C: Programming and Techniques

Nelson Johnson

Advanced Graphics in C:
Programming and Techniques

Nelson Johnson

Osborne **McGraw-Hill**
Berkeley, California

Osborne **McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations and book distributors outside of the U.S.A., please write to Osborne **McGraw-Hill** at the above address.

A complete list of trademarks appears on page 661.

Advanced Graphics in C: Programming and Techniques

1234567890 DODO 8987

ISBN 0-07-881257-7

Jeffrey Pepper, Acquisitions Editor
Herbert Schildt, Technical Reviewer
Fran Haselsteiner, Project Editor
Nancy Leahong, Text Design

Acknowledgments

A book is the product of experience, and experience is never gained in a vacuum. The efforts of a host of people go into the education of an author. The author owes much to those who have been kind enough to provide that education.

I owe a great deal to Bryan Higgins, a superior programmer, who helped me understand 8086 assembler and opened my eyes to the C language. More directly, he provided me with an incredible text editor, EMAX, that was used to write this book. The sophisticated buffering and fast editing characteristics of EMAX made possible the rapid development of GRAPHIQ. I would also like to acknowledge the friendship and support of Emil Flock, of Computer Handholding, who suggested that I might be able to write a book on graphics in C language. Without the kind guidance of Tony Crispino and Jim Gordon, of Number Cruncher Microsystems, I would probably still be working in one big subdirectory. They gave me an understanding of the dynamics of DOS. Until Liz Fisher and Jeff Pepper of

Osborne/McGraw-Hill helped me understand the editing process I didn't fully realize how valuable and necessary professional editing can be.

The origin of much of my work over the last 12 years can be credited to Lou and Genevieve Katz, who, at Columbia University, decided that architectural students could and should know something about computers. Their foresight and imagination guided this student to graphics programming and the creation of the MicroCAD software.

—Nelson L. Johnson
January 22, 1987

Introduction

Years ago, computer graphics were considered unnecessary, impractical, or too expensive.Not so today. Hardware and software are beginning to be fast enough and powerful enough to make graphics not only feasible but essential. The emergence of desktop publishing integrates graphics with text in modern computing. Soon database applications will routinely include images as well as text.

The kinds of tasks we use computers to perform are changing. At one time computers were used primarily for accounting, but now they are used for everything you can imagine. Analytical software depends on graphics to reveal everything from flaws in metals to weather patterns to the properties of geologic formations. Graphics systems are used in graphic arts, medical, scientific, robotics, security, and quality control applications. Drafting, business planning, and design applications are flourishing.

This book is designed to give you the general concepts you need to program graphics in C. The source code for a practical graphics system called GRAPHIQ, written specifically for this book, follows in Appendix A. Each of the program's functions is explained individually. Together they provide a **toolkit** of functions that help you program for graphics on the IBM Enhanced Graphics Adapter (EGA) and the AT&T Image Capture Board (ICB). Toolkits are particularly valuable for an open-ended language like C. They add an ever-expanding functional richness to the language.

This book also contains many techniques that will increase your understanding of the complex field of graphics software development. Algorithms used in graphics programming, graphics editing, hard-copy creation, user access, and communications, as well as the design of effective documentation, are covered in detail. When appropriate, these areas are related to specific functions in GRAPHIQ.

In addition to detailed graphics functions, you will find useful tools needed for sophisticated graphics programming and production. These include general-purpose functions that support serial and parallel input/output, pop-up menus, and graphics text. A complete graphics text font is provided in source form. *Advanced Graphics in C* also includes interfacing techniques for many devices, including printers, plotters, digitizers, and mice.

Appendix A presents all the functions of GRAPHIQ as one integrated program. Because most of the functions and tools presented in this book are used in GRAPHIQ, by studying this program you can learn how they can be made to work together. GRAPHIQ also demonstrates how graphics programming is intimately involved with the entire computer system. Even if you are not primarily interested in graphics programming, you will find useful information in this book.

The approach taken was intended to be generous, open, and sharing, rather than protective. It is hoped that presenting a graphics program from the inside out, in source code form, will cause many of the mysteries of graphics programming to disappear.

HOW MUCH DO YOU KNOW ABOUT C?

If you are just starting to program in C, you may find some of this book hard to understand. It is not intended to teach fundamentals. It is, rather, intended for the intermediate- to advanced-level C programmer who wants to know how to program for graphics.

If you are just beginning, however, this book may prove useful to you. By studying the complete program written in C language, GRAPHIQ, you may learn a great deal. As your knowledge of C increases, you will be able to understand how the functions presented here are constructed, as well as how to modify them when writing your own graphics programs.

HOW TO USE THIS BOOK

The information in this book is presented in two ways. You can read detailed explanations in the chapters, or you can read complete source code listings in the appendixes.

The first two chapters deal with graphics programming from a stylistic and conceptual point of view. You will find in these chapters some general rules for programming in C and for designing graphics software.

Chapter 3 introduces the GRAPHIQ program itself. This chapter explains the purpose behind GRAPHIQ and describes the program's general features.

Chapters 4 and 5 describe algorithms used in graphics programming. You will find algorithms for changing the colors of pixels, drawing lines, drawing circles, filling areas, and many other functions that are routinely used in graphics programming.

Chapters 6 and 7 discuss graphics editing (where most of the drawing actually takes place) and the creation of text. Chapter 7 describes how graphics text fonts can be created and stored.

Chapter 8 describes how graphics hardware and software can be used to create hard copy. You will see how to send graphics information to printers and plotters.

Chapters 9 and 10 are concerned with the way the computer system communicates with the user. Pop-up menus and locator crosshairs are described in detail. You will see how to move graphics and text quickly onto and off of the display surface.

Chapters 11 and 12 deal with the use of your computer's hardware for communication with the outside world. Detailed information is provided regarding serial and parallel ports. You will see how to save drawings on a floppy disk or hard disk and create command files to run your system by remote control, as it were.

Chapter 13 discusses the all-important but too often overlooked subject of documentation. The use of icons and commands as communication tools is discussed.

Chapters 14 and 15 show you how to use the Microsoft C compiler and linker. General techniques for compiling and linking are covered. In addition, the specific requirements for compiling and linking GRAPHIQ are presented.

The appendixes contain the complete source code listing for GRAPHIQ, including a command summary and helpful hints on using your C compiler to help you program in assembler. The new field of video programming is discussed, including some useful routines you can use with the ICB to transfer images between it and the EGA.

If you wish to use the functions in this book on hardware other than the IBM PC with an EGA, you can do so with relative ease. Because the toolkit in this book is provided in source code form, you will be able to make the necessary changes.

New standards are emerging daily. In graphics, "standards" are so profuse they can hardly be called standards at all. Because there is no current consensus as to which standard hardware and software to use, it is impossible to provide and maintain complete compatibility across all devices.

The toolkit source code that was developed as part of this book is available to you on disk, including an executable copy of GRAPHIQ and a library of object modules. If you are interested in ordering a copy, please use the coupon that follows, or send $21.95 (California residents add 6 1/2% sales tax) to

Imagimedia Technologies
P.O. Box 210308
San Francisco, California 94121-0308

Contents

1

Graphics Programming

Programming can be accomplished by intuition or by plan. The best method for approaching a programming project is to apply your intuition under control of a well-conceived structure. Graphics programs tend to be more complex than non-graphics programs. For this reason they can be harder to create, harder to standardize, and harder to use. By beginning with a clear concept in mind, however, you will go a long way toward minimizing the confusion graphics software can generate.

In this book you will find detailed information about programming for graphics in the C language. Because this book is oriented toward intermediate to advanced C programmers, many terms are assumed to be understood without the need for elaborate definition. No attempt will be made to educate you in the fundamentals of the C language, except as they pertain to intermediate and advanced graphics concepts. If, for example, you wish a clear definition of what "function," "offset," or "argument" means, you should consult your favorite text on the C language.

PRINCIPLES OF SUCCESSFUL GRAPHICS PROGRAMMING

The principles of successful graphics programming are not fundamentally different from the principles of successful programming in general. Systems that are based strictly on text benefit from the fact that the C language, like the program product, is used in text form. Graphics applications are different. You will be using code in the form of text to draw points, lines, and other graphical constructions that are not text. Because of the confusion this can create, you will need to emphasize structure in your code even more than you would if your product's output were limited to text.

Be Persistent The first principle of successful programming is persistence. If you are interested in what you are doing, it will not be hard to muster the passion to stick to the task. Graphics problems are less obvious than problems involving only text. It can take longer and be more frustrating to debug graphics programs.

Structure Your Programs The second principle of successful programming is the wise use of **structure.** The computer program is built from the general to the specific. Like a building, it begins with a simple concept that is expressed in greater and greater detail until you are able to walk around within a real structure of concrete, wood, or steel.

Of course, a building could have been built in many ways. You might have a pile of wood in your backyard and nail something together using scraps from this and parts from that, improvising as you go along. The result might be a structure built from the "bottom up" and might be expected to look that way. It would stand a good chance of being unsafe as well, because no thought was given to how the lower part of the building would support the upper part. In short, it would have been *built* but not *designed.*

Design is the process of bringing something into being through the use of a plan, of marking out or defining something. The German concept of "Das Ein," meaning "therein-ness," expresses another nuance of design. In designing something you ask yourself what it wants to be.

Divide and Conquer The third principle of successful programming is to "divide and conquer." In the fourteenth century William of Ockham, a British scholastic philosopher, when he wasn't trying to turn lead into gold formulated a principle that has come to be known as "Occam's Razor." (The spelling has been corrupted over the years.) According to Occam's Razor, any problem can be divided into parts in such a way that by solving all of the parts separately you will solve the entire problem. The design of the C language is basically driven by this concept. In C you are encouraged to break your program down into a collection of more-or-less isolated functions that are designed and tested separately. This feature is fundamental to the implementation of the **top-down** approach to programming.

THE TOP-DOWN PHILOSOPHY

The programs in this book are designed with the philosophy of top-down programming in mind. The top-down approach works well for graphics because it helps organize very abstract, nonverbal concepts.