MORE COLOR COMPUTER™ APPLICATIONS

High-Resolution Graphics

JOHN P. GRILLO
D. ROBERTSON

MORE COLOR COMPUTER APPLICATIONS

John P. Grillo and J. D. Robertson
Bentley College

A Wiley Press Book
John Wiley & Sons, Inc.
New York • Chichester • Brisbane • Toronto • Singapore

Publisher: Judy V. Wilson
Editor: Theron Shreve
Managing Editor: David Sobel
Composition & Make-Up: Cobb Dunlop, Inc.

Library of Congress Cataloging in Publication Data

Grillo, John P.
More color computer applications.

Includes index.
1. Computer graphics. 2. TRS-80 Color computer—Programming. I. Robertson, J. D. (James Douglas), 1943- . II. Title.
T385.G755 1984 001.64'43 84-2321
ISBN 0-471-86923-6

Dedicated to Walt Disney;
he had more color applications.

PREFACE

Some books are written to entertain and some to educate. Although we wrote this book for both of these reasons, our first thought was to give you some ideas, to stimulate your thinking. Each one of the eleven chapters explores a fundamentally different area of high resolution graphics, and in each chapter we prod you to expand on the ideas we have introduced.

In the first color graphics book we wrote, *Color Computer Applications* (Wiley, 1983), we concentrated on low-resolution graphics techniques. Our purpose in that book was also to provide programs that demonstrated what graphics can do to enhance computer output. The differences between the two books, aside from their programs, are the result of the compromise a programmer has to exercise. Low resolution may give you less detail, but it is richer in colors. High resolution detail expands the range of programs at the expense of some colors.

All of the programs in this book will run on the Radio Shack's TRS-80® Color Computer and Color Computer 2. Our Color Computer had 24K of memory, but we found that most of the programs would run on a 16K system also. Of course, if you have a 64K Color Computer 2, you should have a great time with these programs. Because of their modularity, you can attach parts of one program to another, expanding a program with ease.

We used a disk drive to store programs only, so that a person with a cassette system could still use all of the programs. If you have a disk drive, you should consider acquiring the Convenience Diskette that accompanies this book and its predecessor. Even if you have a Color Computer 2 with disk, this diskette is readable because of Radio Shack's foresight in producing the two systems to be completely compatible.

All programs from both books are found on the one Convenience Diskette, exactly as written in the books. They have been tested thoroughly and they make the job of program alteration much simpler, because they greatly reduce the likelihood of keying errors when entering the programs.

The TRS-80 Color Computer and Color Computer 2 are long-lived machines. By that we mean that the design principles they incorporate are based on industry standards, such as the use of Microsoft BASIC as a graphics language. This choice insures a wide base of users who can program this machine with relative ease, and whose skills in graphics programming can be used on many more advanced systems. We expect that your experience with this book and with the TRS-80 will become more and more valuable as the awareness of this exciting form of programming increases.

MORE COLOR COMPUTER APPLICATIONS

OVER THREE MILLION PEOPLE HAVE LEARNED TO PROGRAM, USE AND ENJOY MICRO-COMPUTERS WITH WILEY PAPERBACK GUIDES AND WILEY PROFESSIONAL SOFTWARE. LOOK FOR THEM AT YOUR FAVORITE BOOK OR COMPUTER STORE.

TRS-80 Books

Albrecht, *TRS-80 Color BASIC*
Albrecht, Inman & Zamora, *TRS-80 BASIC*
Bove & Finkel, *The TRS-80 Model III User's Guide*
Finkel & Brown, *TRS-80 Data File Programming*
*Gratzer & Gratzer, *Fast BASIC: Beyond TRS-80 BASIC*
*Grillo & Robertson, *Color Computer Applications*
Grillo & Robertson, *Subroutine Sandwich*
Grillo & Robertson, *More Subroutine Sandwich*
Inman & Conlan, *Problem Solving on the TRS-80 Pocket Computer*
Inman, Zamora & Albrecht, *More TRS-80 BASIC*
Kohl, Karp & Signer, *The Genie in the Computer: BASIC Programming on the TRS-80*
Konkel, *Guide to TRS-80 Model 100 Utility Programs*
*Lewis, *The TRS-80 Means Business*
*Valentine, *CoCo Extravaganza*

TRS-80 Software
Bibbero, *Stock Selection: Modern Portfolio Management*

*Convenience disk available

CONTENTS

INTRODUCTION

In the early 1970s, the personal computers available for purchase were usually kits with limited capabilities. How is it, then, that now in the mid 1980s there can be such a proliferation of excellent and reasonably priced personal computers?

A great many important advances in technology had to occur in order for the hardware to become so improved. Computer memory chips have increased in capacity by a factor of over 100 (from 2K to 256K bits) and still their price has stayed relatively constant. The microprocessor units that act to control all the computer's functions have improved also, though perhaps to a lesser extent than memory. However, it is not the improvements in the internal electronics of the personal computers that has so dramatically affected the marketplace. Rather, it is the major advances in the computer's peripheral devices, those items of hardware that act to communicate with the user.

The output of a computer is what is important to us, because it presents to us the evidence of its work. The traditionally accepted output device, the printer, is still in use and will remain so as long as the computer's users need reports and the computer's programmers need listings of their programs. The screen of a monitor or TV set is becoming more and more important as an output device because of its relatively low cost. For immediate and temporary output, it can't be beat.

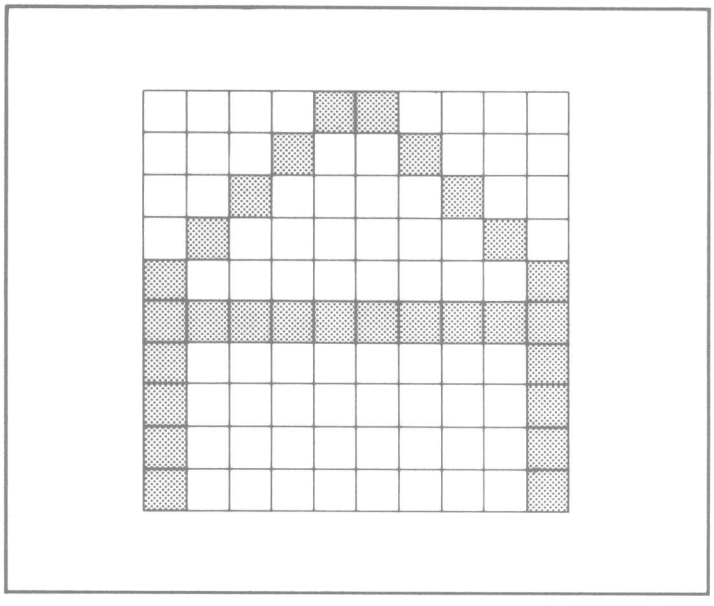
The screen as an output device has another very important advantage over paper. Because of the way images are formed on the screen as patterns resulting from individual spots arranged electronically, the screen can act as a two-dimensional palette to display pictures with a wide variety of both detail and color. These computer-generated screen

images are made of a combination of black-and-white pre-formed patterns (letters, digits, characters, blocks of light), or from many tiny colored spots no bigger than a small pinhead. The size of the smallest program-controlled image determines the computer's *resolution*.

A computer's screen image is produced using an array of blocks of light, and each block occupies a given row and column position on the screen. Imagine a screen with 10 horizontal positions and 10 vertical positions. Each individual block of light would occupy one hundredth of the area of the screen. You could program this computer to produce large, choppy-looking characters, and perhaps some simple images. This screen would have a limited number of individually addressable areas of the screen, each addressable by row and column number from 1 to 10. The images produced would have little detail, or in terms of computer graphics, would lack resolution.

Of all the advances in computer technology, those that affect the computer's output are most noteworthy, primarily because they are so noticeable. During the last dozen or so years, the personal computer industry has reaped the benefits of this vast improvement in screen graphics resolution. The arcade games were first to capitalize on this technology, and fast on its heels were such companies as Apple, Atari, Commodore, IBM, and Radio Shack.

The Radio Shack personal computer that uses high-resolution graphics to best advantage is the TRS-80 Color Computer, both in its older version and in the newer TRS-80 Color Computer II. It is interesting to



note in passing that this computer can be programmed to produce low-resolution graphics, with 64 columns and 32 rows of individual rectangular blocks, or blips, of color, or it can increase its resolution to as many as 256 columns and 192 rows. However, by producing such high-resolution graphics, it reduces the number of colors it can display. The high-resolution displays on the TRS-80 Color Computer are achieved at the added expense of an improved BASIC, more memory, and limited color.

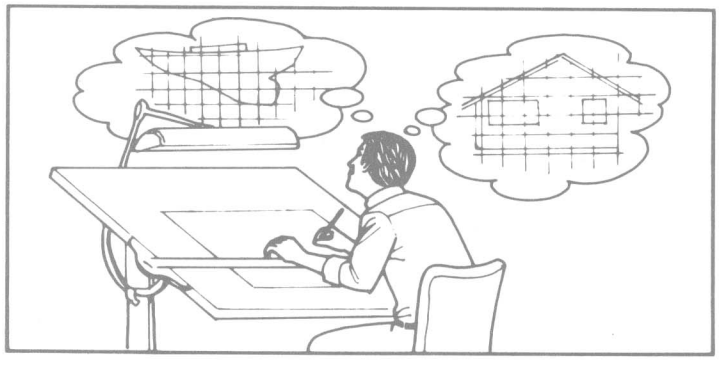
So we want to encourage you to use high-resolution color graphics. We choose to do so with the TRS-80 Color Computer because it is an intriguing machine. It is inexpensive; it is most definitely a personal computer intended for home use; it is in some ways inflexible and

demanding. But at the same time, it is as fast or faster than many small business microcomputers, and it has a powerful BASIC language with surprisingly good graphics instructions.

We wrote the programs in this book on a TRS-80 Color Computer with 24K memory, extended BASIC, and one disk drive. This computer was also marketed in stores (other than Radio Shack) as the TDP System 100. You should be able to use all the programs with no change on the British clone to the Tandy machine, the Dragon 32. And you should be able to modify all programs with relative ease to run on an IBM PC, because the BASIC is very similar. This is perhaps one of the most important contributions that the Microsoft company has made to personal computing—providing a common, standard, flexible, easily transportable BASIC for a wide variety of machines.

When we took on the challenge of writing a book about this computer and how to make use of it, we decided immediately to produce a book of programs for it. This way, you can experience the full range of features that this computer boasts. Because there is such a large difference between the low-resolution standard BASIC and the more advanced language with its high-resolution features and language constructs, we divided the chore into two separate books. This is the second one, and it concentrates on high-resolution graphics only.

We assume that you are familiar with BASIC programming. The programs included in this book are intended to demonstrate and develop ideas in graphics programming, not to teach you the syntax, or the rules, of the language. You should rely on the Color Computer manuals and on other sources to help you with problems in syntax. This book's purpose is to stimulate you, to expand your ideas concerning what computers can do so that you can explore the topic of graphics programming in whatever area of application interests you.

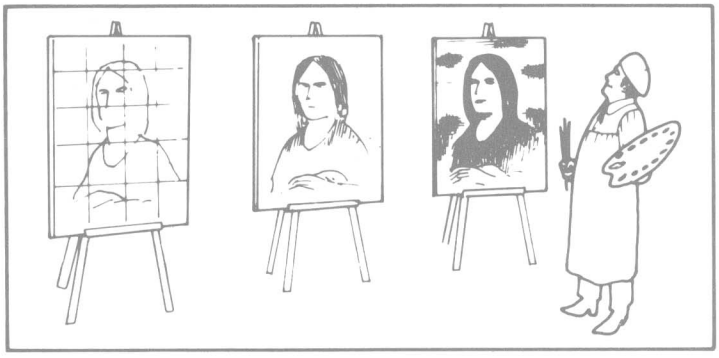


Chapter Organization

Each chapter of this book has a set pattern, with a single program as its centerpiece. This major program is used as an example for the several features that we wish to highlight in the chapter. The chapters start with a statement of objectives, outlining in brief fashion the reasons for the chapter's existence. What will you get out of it? What new techniques will you learn? What new instructions in BASIC will you see in this program?

The second part of every chapter is a statement of the specific problem that the program attempts to address. For example, the problem statement in Chapter 2 explains what the Alphabet Generator program does and how that problem is solved by using a program on the TRS-80 Color Computer.

The third part of every chapter is what we call the program's "soft design," samples of the program's input, output, and dialog. These help you understand how the program must be organized. When you write a program, you are faced with the task of translating a problem statement into a set of instructions for the computer. The transition from problem statement to program is not an easy one for you to make unless you outline it carefully in progressively more detailed steps.



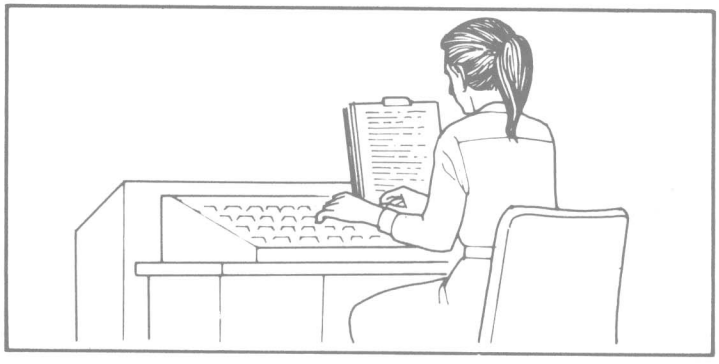
In the soft design, we list the outputs the computer must produce, as much like the final screen image as possible. Then we itemize all the user's responses or data inputs. If the problem is best answered by a running dialog, a question-and-answer session between the user and the computer, we plan that mix of inputs and outputs by designing sample screen images of that dialog.

Once you have a clear understanding of the program's communications—its input/output, or I/O—you can attack the process that produces these interactions. This is the next step in our program design, and it is the next section of every chapter. The process is outlined in what is called *pseudocode*, which is a programmer's shorthand plan for the program's logic. We have chosen short English phrases as our pseudocode, simply because this way there is nothing new for you to learn.

Our pseudocode program outlines show the logic of the program—its method of solution, or *algorithm*. Using this form of an algorithm allows any one to write a program in any language on any computer. Of course, we understand that the computer and its language must be capable of doing what is requested in the problem statement. Therefore, the pseudocode algorithm must reflect the hardware's features and limitations that will be a part of the program's input and output. For example, a program design for the TRS-80 Color Computer that uses only its low-resolution graphics should specify in the algorithm what features will be used.

The fifth part of every chapter is the program itself. We have purposely kept our programs relatively free of remarks, because they are so carefully documented in the rest of the chapter. We are quite aware that this practice leaves us wide open for heavy criticism, as remarks are so often absolutely necessary for any listing. We feel justified in our method of documentation because these programs are so tightly tied to the book.

Although the programs have a minimum of internal documentation, they are highly structured. That is, they are written in a highly modular fashion using top-down design principles. Each process is isolated in its own subroutine, and the entire program is menu-driven for maximum flexibility. This modular approach to programming allows you to modify existing modules or add new modules in the form of subroutines with a minimum of effort.

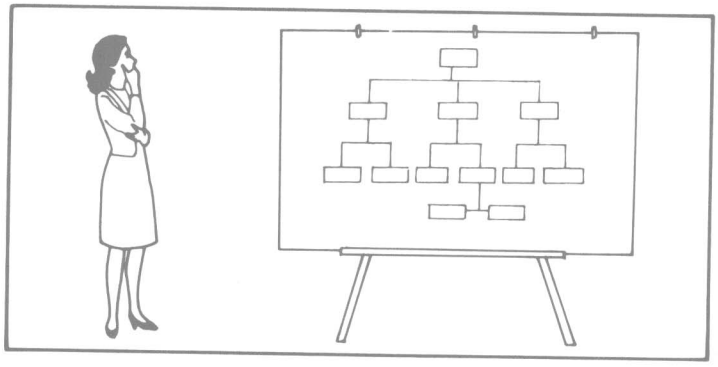


Finally, we give sample output from each program. Unfortunately, it is quite impossible to illustrate the output in every case with lots of color pictures of the screen, or else this book would cost five times as much as it does now. Instead, we use black-and-white pictures where necessary and printer output where appropriate or possible. In any case, this last section of each chapter cannot do justice to the program's purpose, because the programs were written to produce output on the color computer's screen. If you are serious about exploring any given program further, you should code it into your own computer, or you should invest in the companion diskette that contains all programs together from both books.

Programming Style

You have seen many programs in print, from the listings in the Color Computer's manual to those included in magazines. Many published program listings are almost unreadable because they have been stripped of all remarks and spaces and they are crowded with as many statements as can be squeezed on each line. There is some justification for publishing such listings: They occupy the least amount of space on the printed page, and they crowd into a minimum amount of memory in your computer. But they are almost impossible to key in—to copy into your computer accurately—because they are so terse. And if you make the slightest mistake when you copy the program, you often have a hard time discovering the source of your error because these programs are usually unstructured.

As we mentioned before, we made only a few brief remarks about the listings in this book. However, we use a programming style designed to improve their readability. We insert spaces where necessary to help readability and organize line numbers to isolate subroutines from the



main program. In terms of overall organization, we segment each program into several major modules, each of which can be changed independently of the others. This clear separation of function is one of the strong features of modular, structured code.

The first part of most programs is what we call its *driver*. The driver is the main program, also sometimes called the *mainline* program or, in some cases, the *menu driver*. Its function is to select one activity from several possible activities the program can perform. You are presented with a list of all activities and select from this "menu" the desired one, and the mainline program branches to the chosen activity.

Each activity is a separate subroutine called by this menu driver. If we can incorporate an activity into one subroutine, so much the better, as long as that subroutine is not longer than two dozen or so lines. This way, we can understand, expand, or modify each activity without affecting any other. If an activity is so complex that it must be further subdivided into other subroutines, that activity should become a driver, perhaps with a menu display of its own.

The segmentation of a program, which is called its *modularity*, is extremely valuable when that program needs to be changed or expanded. Because each chapter has a program in somewhat simple form begging for expansion, its modularity is important. Our intent with these programs is to provide you with a beginning that works and is easily expandable into a variation that suits you better.

System Limitations

When we coded these programs into the TRS-80 Color Computer, we discovered some disturbing shortcomings of the computer's BASIC language that prevented us from using some features of structured programming that we feel are helpful to the reader of any program. Three limitations were particularly bothersome: First, BASIC automatically deletes leading spaces in a line, so that it is not possible to indent several lines to make a segment stand out. Second, there is no such thing as a line-feed character; so a statement longer than 32 characters, the width of the screen, cannot be broken at a convenient place to force the next line on the screen to start at a phrase. For example, consider the line

```
3110 IF X < A OR X > B THEN PRINT "ALL DONE"
     ELSE GOSUB 5000
```