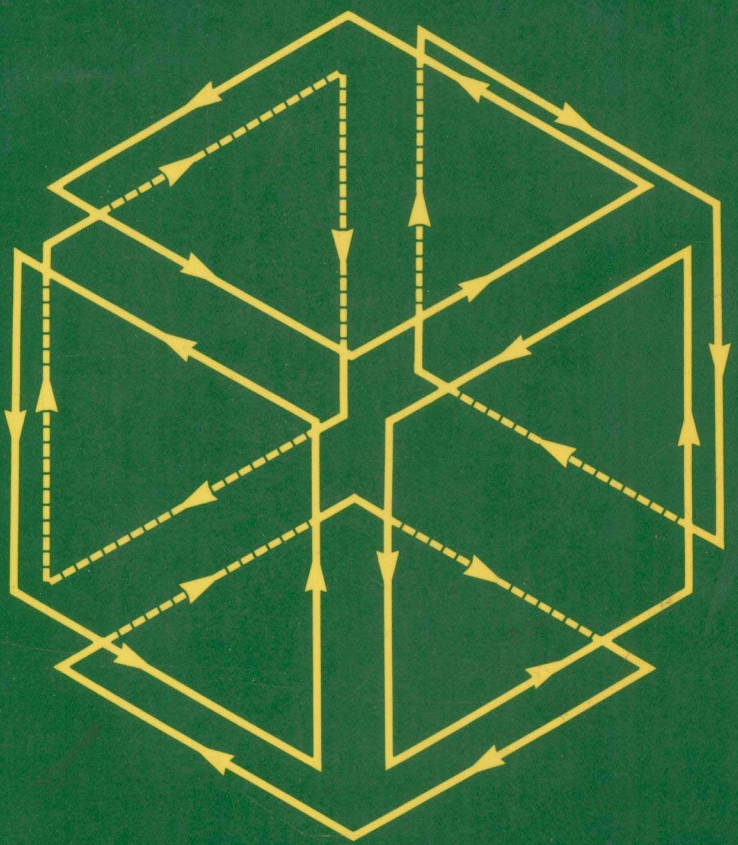
Computer Graphics and CAD Fundamentals

BBC Micro Version

Noel M Morris



Pitman

Computer Graphics and CAD Fundamentals

BBC Micro Version

Noel M Morris

Pitman

Pitman Publishing Limited
128 Long Acre, London WC2E 9AN

A Longman Group Company

First published 1986

British Library Cataloguing in Publication Data

Morris, Noel M.
Computer graphics and CAD fundamentals :
BBC Micro version.
1. Computer graphics 2. BBC Microcomputer
— Programming
I. Title
006.6'86 T385

ISBN 0 273 02517 1

Printed at The Bath Press, Avon

Preface

Computer graphics are, without doubt, the most exciting application of computers. Not only can they be used to present facts and figures in an informative and interesting way, but they also allow representation of complex facts and equations.

This book is intended to fill the gap between elementary 'how to draw lines and shapes' books and advanced books on computer-aided design. The level of mathematics in the first eight chapters does not go beyond school trigonometry, and each step is explained as it is reached. The final chapter on Computer Graphics Mathematics gently leads to an appreciation of fascinating and exciting areas of mathematics that can be used in program development.

The book is particularly suitable for all levels of study, including GCSE, 'O' and 'A' level computer-based topics (including Computer-Aided Learning or CAL), BTEC courses (ONC, HNC and HND), and also degree courses (including computer science, information technology, electronics, mechanical engineering, etc). It also serves as a sourcebook of ideas for projects, and is well suited to the needs of the computer 'buff' who wishes to extend his/her knowledge and experience in graphics and CAD.

Whilst a knowledge of programming in the BASIC language is assumed, the book begins at a level which anyone can understand. As far as is possible with any microcomputer, the programs have been written so that they are reasonably 'transportable' between microcomputers.

Eighty program listings are contained within the covers of this book, the twenty-four most important and complex programs being available on a disc that can be obtained from Pitman Publishing. It is estimated that the cost of the programs on the disc, if purchased from a software house, would be many times the cost of the book!

Chapter 1 introduces aspects of modular programming and top-down program design. Many features of the microcomputer are introduced in this chapter, including the process of merging programs and writing menu-driven programs. Also included are timing of program operations, 'error' trapping, program 'trace' facility, and speeding up of BASIC programs. One of the limitations of most personal microcomputers (the BBC micro being no exception) is that large programs can run out of memory space. A method of 'overlaying' programs from a disc is described in Chapter 1; using this technique, it is possible to handle very large programs indeed.

Chapter 2 introduces fundamental drawing features, including the drawing of 'wire frame' shapes and specifying user-defined characters. The

progrmaming of 'soft' keys or user-defined keys is introduced, and the chapter shows how they can be incorporated into a program. The operation of a graphics 'ruler' is described, explaining how to make measurements on the screen. Finally, text and graphics 'windows' are introduced.

Chapter 3 deals with a very important feature of graphics, namely colour and the filling of shapes with colour. The way in which a microcomputer handles colour is demonstrated by several programs, including the way in which 'logical' operations are carried out on colours. The chapter concludes with an animation program that uses features developed in the chapter.

The use of text on diagrams and drawings is very important in CAD programs. Chapter 4 includes routines that introduce the techniques involved in printing enlarged text at any position on the screen and at any angle, or even around the arc of a circle!

The presentation of data in art, commerce, industry and engineering is vital to the everyday running of organizations (even small clubs find that the graphical presentation of data is invaluable). Chapter 5 introduces concepts associated with drawing graphs, histograms or bar charts, and pie charts. A number of practical effects are covered in the chapter, including different methods of drawing graphs, as well as three-dimensional bar charts, and drawing pie charts with a 'slice' displaced from the chart.

Chapter 6 introduces a line-drawing program which makes it possible to use the keyboard to draw diagrams constructed from straight lines and dots; this is the basis of many computer-aided drawing (CAD) programs. Many important features are dealt with, including methods of 'rubbing out' mistakes on the screen. The use of icons or images is incorporated in the program; in this chapter they are positioned on the screen by a method known as 'picking and placing'.

An area of considerable prominence in graphics and CAD is that of drawing three-dimensional shapes. A number of programs that are developed in Chapter 7 allow the keyboard to be used to manipulate both 'wire-frame' and 'solid' three-dimensional shapes. The basic program, which is extended as the chapter unfolds, incorporates perspective, zooming, translation and rotation. You are shown how to configure your own shapes, a rocket or space vehicle being used as an example. The program is developed to display stereoscope 'wire-frame' three-dimensional shapes which can be made to look as through they 'come out of the screen'! The mathematics of vectors is brought to life in Chapter 7 by extending the program to draw a 'solid' three-dimensional shape including the features of perspective, zooming, translation and rotation. The chapter includes a program which allows a character to be 'embossed' (a letter from the alphabet is used to illustrate the principle) on one face of a 'solid' three-dimensional object, and it concludes with a discussion of the elimination of 'hidden' lines.

An extensive and practical computer-aided drawing program is developed in Chapter 8. The program further extends the use of the user-defined keys — over thirty different effects are achieved by this means. The program also leaves more than ten other 'soft' keys available

for the user's own special effects. Features included in the program are: 'rubber-band' drawing of circles, polygons and rectangles; moving rubber-banded shapes around the screen; deleting rubber-banded shapes; drawing lines and arcs; centring the cursor; using a 'ruler' to measure lengths and angles on the screen; using a 'rubber' or 'eraser'; global erasure of the diagram on the screen; adding text to the diagram (at an angle if needed); drawing dots, 'blobs' and arrowheads (the latter at an angle if needed); 'filling' and shading shapes; drawing a graticule on the screen; centring the cursor; 'picking and dragging' icons; 'saving' a diagram either on tape or on disc; 'loading' a diagram on tape or disc onto the screen.

Chapter 9 opens the door not only to the basic mathematics associated with graphics, but also to some of the mathematics used in advanced CAD programs. This chapter deals with trigonometry, compound angles, complex numbers and matrices — mathematical techniques that are vital to the manipulation of lines and shapes in graphics programs and are widely used in art, commerce and industry.

This book could not have been written without the help of many people and organizations, and it is difficult to name everyone who has contributed to it in some way. In particular, I would like to thank the students on whom many of the programs were 'tested', and also Mr F. W. Senior, MSc, and his son Andrew. I am indebted to the Computer Services Unit of the North Staffordshire Polytechnic for advice and assistance, in particular to Mrs C. Faul and her colleagues. I would also like to thank the staff of *Beebug* magazine, and I am grateful to Acorn Computers for permission to reproduce data relating to the BBC microcomputer.

As ever, I gratefully acknowledge my wife's assistance, help and encouragement during the writing of this book.

Developing the programs in the book was not without its moments of difficulty: Mr L. A. Meredith, B.A., reports the following graffiti on the wall of a university computing laboratory. We should reflect on its sentiments!

I'm fed up with this computer,
I wish that they would sell it.
It never does what I want,
Only what I tell it!

North Staffordshire Polytechnic

Noel M. Morris

Introduction

The twin topics of computer-aided drawing and computer-aided design are emerging in a variety of fields of study including science, engineering, computing, architecture and many others. It is perhaps unfortunate that both computer-aided drawing and computer-aided design have been described as CAD! The book begins by introducing the reader to the general concepts of the former and, in later chapters, leads towards the latter, namely the use of graphics in the design process.

The book aims to provide students and teachers with material which explains the principles of CAD programming, and is highly appropriate to any course of study involving design technology with a computing element.

It is possible that the reader may not have a background of mathematics or geometrical drawing. This possibility was borne in mind when the book was written and, to this end, the more complex aspects of mathematics are dealt with in the final chapter. This prevents the mathematics from getting in the way of understanding the graphics techniques.

Since it is recognised that only a few readers will have access to sophisticated support devices such as a 'mouse' or a 'bit-stick', the book has been written around a stand-alone BBC microcomputer with, perhaps, a dot matrix printer as a 'hard copy' output device. It is hoped that, in this way, the book will appeal to any reader with an interest in computer graphics.

As any computer user knows, the documentation provided with the computer is often difficult to read and even more difficult to understand! For this reason, a minimum amount of information about the BBC micro has been included in the text, consistent with the reader being able to understand what he is doing. To enable the reader to make full use of the micro, one or two specialized techniques are also discussed in the text. Among these is the use of the 'overlay' method, which overcomes many of the difficulties associated with the limited amount of memory of a standard personal computer.

It is hoped that between the covers of this book, the reader will find much that is interesting and exciting in the field of mathematics, for it is this subject which permeates every field of study.

Some important statistical methods of treating experimental data are touched on; one such application is a method of producing the 'best' straight line from a set of experimental data. Many items of commercial software make a feature of this technique, and some go to the trouble of extending it into 'curve fitting' software which produces a mathematical equation which best fits the results. It is hoped that the bok will lead readers to develop their own techniques for handling these problems.

Fascinating areas of graphics, including the design and manipulation of three-dimensional shapes, and the production of engineering and architectural drawings, are introduced. The mathematical techniques involved are kept fairly low-key in the early chapters, but those with a mathematical bent can use some of the matrix methods in the final chapter to produce some dramatic effects!

Since BASIC is the most popular programming language, it has been used as a vehicle to develop the programs in this book. However, if trigonometrical ratios are involved in a graphics program, they have the effect of slowing down its operation. For example, a standard (but easily understandable) circle drawing program is used in the early chapters of the book, but it has the drawback that it needs to compute many trigonometrical values. Later in the book a faster (but not so easily understandable) method is introduced; although the speed increase is not dramatic, it serves to illustrate that there is always an alternative method of solving a given problem.

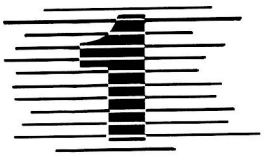
Each computer has its own dialect of the BASIC language, and the BBC micro is no exception. So far as is possible, the progrmas have been written so that they are structured, logical and understandable. It should be possible for a computer-literature programmer to convert the programs into the BASIC dialect of another computer.

At various points in the book, certain problems are left unsolved, and the reader is invited to think about possible solutions. In most cases, the problems are returned to at a later stage, and a solution is offered. In this way, the book offers an educational challenge to those who wish to test their knowledge.

As with most aspects of life, education is a process of increasing perception, and the book leads the reader gradually forward to a deeper understanding of the relationship between art or graphics and the associated mathematics.

The programs on the disc have run successfully on the Model B, the Model B+ and the Master series, with the exception that for the Master series in program P8 it is necessary to change the calling address for the operating system in the screen saving and screen loading routines. The information relating to the addresses is available in the User Guide for the Master series.

Contents



Introduction to Programming Techniques

1.1 Modular Programming or 'Top-down' Programming

An advantage of the BBC BASIC language over many other forms of BASIC is that it contains many features which allow highly structured programs to be written. A technique known as **modular programming** using a **top-down** method can be adopted which enables programs not only to be easily understood but also quickly debugged so that errors can be removed.

A program written on these lines employs a relatively short **executive program** whose function it is to "call" a sequence of **procedures** or modules into use. In turn, each of these procedures may call other procedures, and so on; this is known as **nesting** the procedures.

A textbook is a good example of a top-down design. When you first open the book you see the list of contents; this is the executive program of the book which presents you with the list of chapters in the book (these can be thought of as the procedures or modules in the book). It is in the chapters that the real detail and activity takes place. Each chapter may be self-contained or it may call on other chapters for further information, in much the same way that one procedure in a program may call on another.

Modular programming offers a number of advantages over a large single program as follows:

1 Each procedure or module can be individually tested.
2 Errors within a procedure can be easily located.
3 Testing any one procedure can proceed independently of the others.
4 The program can easily be extended by adding new procedures as necessary.

1.2 A Simple 'Top-down' Designed Program

We now consider the process of writing a program in BBC BASIC which calls upon two simple procedures, one using the graphics facility of the computer and the other using the SOUND facility. The graphics procedure simply draws a rectangle, and the SOUND procedure produces a siren-like noise. Initially, we will see how each procedure is written and tested using its own executive program. Finally (in section 1.4) we will merge the two procedures into a single program.

The graphics program is given in *listing 1.1*. The executive program is in lines 10 to 50 inclusive, the purpose of each line being as follows:

10 defines the program name.
20 gets rid of the screen jitter.

30 puts the computer into screen MODE 1 which gives reasonable graphics display combined with a 40-character text width.
40 calls the procedure PROC_rectangle into use, which causes a rectangle to be drawn on the screen.
50 ENDs the executive program.
60 the colon acts as a separator between the end of the executive program and the begining of PROC_rectangle.

Listing 1.1

```
   10  REM****PROGRAM"P1.1"****
   20  *TV0,1
   30  MODE 1
   40  PROC_rectangle
   50  END
   60 :
10000  DEF PROC_rectangle
10010  CLS
10020  MOVE 100,100
10030  DRAW 100,900:DRAW 1100,900
10040  DRAW 1100,100:DRAW 100,100
10050  ENDPROC
10060 :
```

The procedure PROC_rectangle for drawing the rectangle commences in line 10000.There are good reasons for writing procedures which commence with a high line number; this is explained in section 1.3.

The first line in every procedure is a DEFinition of its name, e.g. DEF PROC_rectangle. Since the BASIC programming language has a number of *reserved names* which cannot be used in defining procedure names (these are names involved with program instructions such as MOVE and DRAW,

you are most strongly advised to use lower-case letters for procedure names.

For example you can, if you wish, call a procedure PROC_draw, but you cannot call it PROC_DRAW. The purpose of the lines in PROC_rectangle is as follows:

10010 clears the screen.
10020 MOVEs the graphics cursor to a point close to the bottom left-hand corner of the screen.
10030 and 10040 DRAWs the rectangle.
10050 ENDs the PROCedure (DO NOT USE an END instruction for this purpose, otherwise the computer thinks it has reached the end of the executive program).

Run the program and see the rectangle drawn on the screen. If you have made an error in writing the program, it is a relatively easy matter to put it right at this stage.

The program which produces the SOUND effect is *listing 1.2*. The executive program again commences at line 10, and the only purpose of the executive program is to bring PROC_siren into use. Once again the procedure commences at line 10000; the procedure causes the computer to output data to the sound chip (integrated circuit) which gives rise to a siren-like sound for a period of time. Once again, the procedure can be tested independently and any errors quickly corrected.

Listing 1.2

```
   10  REM****PROGRAM "P1.2"****
   20  PROC_siren
   30  END
   40 :
10000  DEF PROC_siren
10010  FOR C=1 TO 5
10020    FOR P=70 TO 100:SOUND 1,-15,P,1:NEXT P
10030    FOR P=100 TO 70 STEP -1:SOUND 1,-15,P,1:NEXT P
10040    NEXT C
10050  ENDPROC
10060 :
```

1.3 *SPOOLing and *EXECing Programs

In the previous section you encountered and tested two independent modules or procedures which we will now merge into one complete program. When you have written *any* program you can SAVE it on tape or disc (the latter being much quicker). However, when you save a program using the SAVE instruction, it is stored on tape or disc in what is known as a *tokenized* or *compressed* form. That is, certain words are stored in the form of a single character; for example, the basic word PRINT is stored as the token P. However, when you are storing a program or procedure which is to be merged or joined with another program, it is necessary to store each word in exactly the same form as it is typed into the computer. That is, the word PRINT needs to be stored as five characters or five ASCII characters (American Standard Code for Information Interchange).

The operating system command *SPOOL is used to store programs and procedures in ASCII or uncompressed form as follows. Suppose that you wish to *SPOOL the procedure PROC_rectangle so that it can be used as part of a larger program at a later stage (see section 1.4). You now need to retain in the memory of the computer only the part of the program you wish to SPOOL; it is therefore necessary to DELETE lines 10 to 60 of program *P1.1* using the DELETE 10,60 command, leaving only lines 10000 onwards in the memory.

You must give the file to be SPOOLed a name; let us call it "RECTANG". To avoid confusion betwen files that have been SAVED and files that have been SPOOLed, the author has found it useful to precede the filename with "S". Thus, we will call the *SPOOLed version of PROC_rectangle "S.RECTANG". To SPOOL the file you simply type *SPOOL "filename", that is

*SPOOL "S.RECTANG"

and press the Return key. This has the effect of "opening" a file. Next you must type

LIST

and, as the program is listed on the screen, it is also written into a file named S.RECTANG. You must then "close" the file and terminate SPOOLing by typing

*SPOOL

The latter operation is very important, otherwise the file remains open. You can then verify that the file has been saved by typing *CAT and verifying that the program S.RECTANG is stored.

When SPOOLing a procedure, it is advisable to leave the ":" separator which follows each procedure (see line 10060 in *listings 1.1* and *1.2*). The reason for this is that, when several PROCedures are merged using the *EXEC command, each procedure will be separated from the next one by a ":" symbol, allowing the start of each procedure to be easily identified.

Once a file has been SPOOLed onto tape or disc, it can be returned to memory by typing *EXEC "filename" followed by ⟨RETURN⟩. In this case you type

*EXEC "S.RECTANG"

Whilst the program is being *EXECed, it appears line-by-line on the screen of the monitor as though you were typing it at high speed. However, a SPOOLed program CANNOT be LOADed into the memory of the computer using the LOAD command; if you try to LOAD "S.RECTANG", the computer will reply with "Bad Program".

When you have *EXECed a program, the computer terminates the display by printing "Syntax Error". This arises because of the > symbol which precedes the *SPOOL instruction; you can ignore the syntax error statement.

If you wish to look at the listing of a SPOOLed file, simply type *LIST "filename", that is

*LIST "S.RECTANG"

The *LIST command causes the computer to LIST a named ASCII file; in this case the quotation marks around the filename are optional and may be omitted.

1.4 Merging Programs

Suppose that you wish to merge the procedures in programs *P1.1* and *P1.2* into a single program called *P1.3*. Firstly, you must SPOOL PROC_rectangle in the manner described in section 1.3; that is, LOAD program *P1.1* into the memory of the computer and DELETE lines 10 through 50. Next, type the following (pressing ⟨RETURN⟩ after each line):

```
*SPOOL "S.RECTANG"
LIST
*SPOOL
```

You must then LOAD program *P1.2* into the memory and DELETE lines 10 through 40, and type the following:

```
*SPOOL "S.SIREN"
LIST
*SPOOL
```

At this stage the programs S.RECTANG and S.SIREN are SPOOLed onto disc or tape.

You now need a new executive program which calls the two procedures into use. A simple example of this is given below (however, you must make sure that the memory is initialy cleared by typing NEW).

```
10 REM****PROGRAM "P1.3"****
20 *TV0,1
30 MODE 1
40 PROC_rectangle
50 PROC_siren
60 END
70:
```

Next, you must merge the two SPOOLed procedures onto the end of the new executive program. PROC_rectangle is merged by typing

```
*EXEC "S.RECTANG"
```

As the program is EXECed, you will see it appear line-by-line on the screen; remember, ignore the "Syntax Error" message printed by the computer. To complete the merging process type

```
RENUMBER
```

followed by ⟨RETURN⟩. This command renumbers the complete program including the EXECed PROCedure. To verify that this has been done, LIST the program to see what it looks like. The reason for RENUMBERING the program is that is you *EXEC a number of PROCedures, each commencing at line 10000, then each EXECed PROCedure is written over the top of the earlier program, and deleted by it.

You must now append PROC_siren by typing

```
*EXEC "S.SIREN"
```

followed by RENUMBER. The complete program appears as shown in