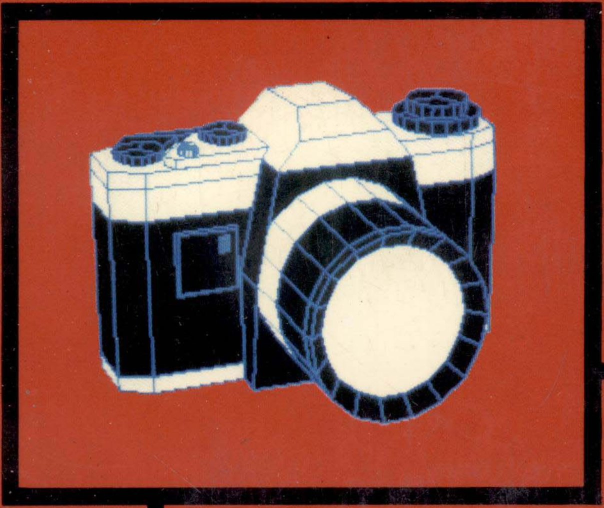
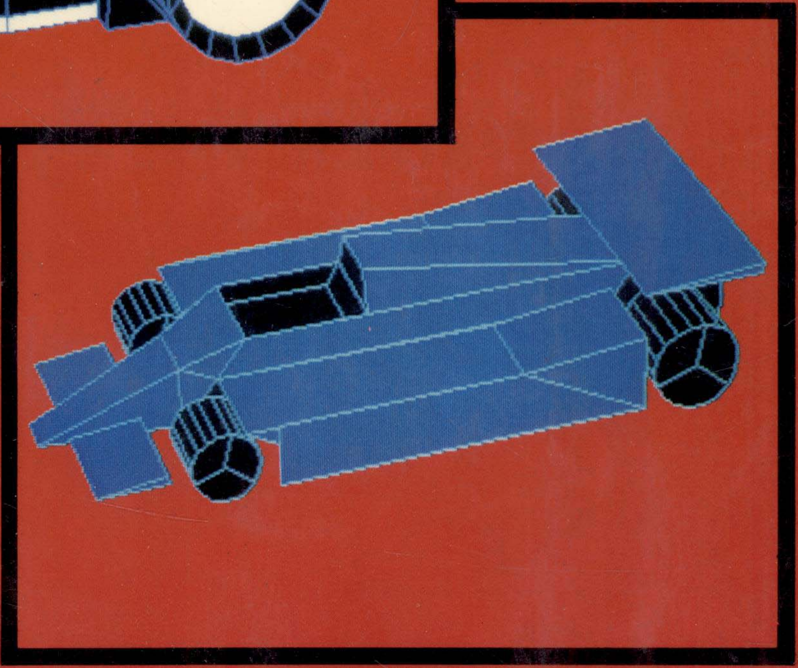
Advanced Graphics with the BBC Model B Microcomputer





I. O. Angell and B. J. Jones

Advanced Graphics with the BBC Model B Microcomputer

Ian O. Angell and Brian J. Jones

Department of Statistics and Computer Science
Royal Holloway College
University of London
Egham, Surrey

M

MACMILLAN PRESS
LONDON

First published by
THE MACMILLAN PRESS LTD
London and Basingstoke
Companies and representatives
throughout the world

Typeset by RDL Artset Ltd, Sutton, Surrey

Printed in England by Pindar Print Limited, Scarborough, North Yorkshire.

ISBN 0 333 35052 9 (book)
0 333 35053 7 (cassette 1)
0 333 36141 5 (cassette 2)

Preface

With the rapid advance of computer technology has come a substantial reduction in the price of computer hardware. In the coming years the price of peripheral devices will also tumble. This means that users with a limited budget, who previously had access only to the most elementary computing devices, will soon be able to afford the most sophisticated computers. They will also be able to escape from the limitation of tabular numerical output and buy microprocessor attachments for television monitors or inexpensive special-purpose colour graphics devices. Software, however, does not appear to be getting cheaper.

Because of the enormous capital expenditure that was required to set up graphical output in the past, both for machines and for software, the subject of computer graphics has so far been the preserve of large research groups. This inaccessibility has led to a mystique growing up around the subject and it has thus achieved a false reputation for difficulty. This book is an attempt to lay the ghost of complexity; it will also show that complicated (and hence expensive) software packages, which are naturally of great value in research organisations, need not frighten away the average computer user. For most purposes these packages are unnecessary. This book, as well as being an introduction to computer graphics, may be considered a (very inexpensive) software package: it is a lot cheaper than commercially available packages! Naturally, because of this fundamental approach, users have to achieve a reasonable understanding of their graphics device before pictures, other than those provided, can be drawn. This need not be a disadvantage; the amount of groundwork required will be seen to be very limited and, as a direct result, the user's knowledge grows along with the package and he is far less likely to misinterpret any of the graphical procedures. References are given and relevant further reading material is also recommended in order to expand the reader's horizons in the subject.

It is assumed that the reader has an elementary knowledge of Cartesian coordinate geometry (the authors recommend books detailed in Cohn, 1961, Coxeter, 1974 and McCrae, 1953 – see the references) and also the BASIC programming language (see the BBC User Guide – page numbers are not given because this excellent handbook is constantly being updated as the BBC micro is being extended). Many interesting programming exercises are proposed, and these should raise the standard of the reader's BASIC expertise. BASIC is a universally popular language that is available (in various guises) on all types of microcomputer, so the programs can be easily adjusted to run on micros other than the Model B: it is also a good medium for transmitting the algorithms that are used in computer graphics, so enabling readers to translate these ideas readily into any other computer language of their choice.

The concepts necessary for the study of computer graphics are organised as a combination of theory and worked examples; these are introduced as and when they are needed in the natural progression of the subject. Program listings that form part of the examples may be considered not only as algorithms that describe solutions to fundamental graphical problems, but also as a computer graphics software package in BASIC, or just as programs to draw patterns. Alongside the examples are a series of exercises that expand on these ideas. The practical problems that are implicit in programming the various concepts of computer graphics are often more a source of difficulty to the student than the concepts themselves. Therefore it is essential that readers implement many of the program listings given in the book in order to understand the algorithms, as well as attempt a large number of the exercises. As an extra learning aid, two companion audio-cassette tapes are being made available; these contain most of the larger program listings that are given in this book. If readers are frightened by the mathematics they should run the programs first before studying the theory.

This approach to the subject has been used with great success in teaching computer graphics to undergraduates and postgraduates at Royal Holloway College. Quickly producing apparently complex pictures results in the positive feedback of enthusiastic interest. The ability to construct pictures on line-drawing and colour interactive graphics VDUs makes a long-lasting impression on the student; and the step by step approach brings him very quickly to the level of very sophisticated computer graphics. That level is outside the scope of this book, but where necessary the reader will find relevant references to guide him into the more advanced topics.

This book is aimed at those who are competent BASIC programmers but who are complete beginners in graphics. It contains the elementary ideas and basic information about pixel and two-dimensional graphics which must be mastered before attempting the more involved ideas of character and three-dimensional graphics. This is followed by a section relating to character graphics and the display of data (in line drawings and colour) – probably the most important non-specialised, commercial use of computer graphics. Later chapters introduce the reader to the geometry of three-dimensional space, and to a variety of projections of this space on to the two-dimensional space of graphics devices. The related problems of hidden lines and hidden surfaces, as well as the construction of complex three-dimensional objects, are dealt with in detail. Finally we return to advanced ideas in BASIC programming and give a large worked example of a video game (to be found on cassette 2).

Graphics is one of the most rapidly expanding areas of computer science. It is being used more and more in the fields of Computer Aided Design (C.A.D.), Computer Assisted Management (C.A.M.) and Computer Assisted Learning

(C.A.L.). At one time it was only the big corporations such as aircraft and automobile manufacturers who used these techniques, but now most companies are realising the potential and financial savings of these ideas. What is more, not only is computer graphics profitable, it is fun! The BBC microcomputer is an ideal machine on which to learn the basics of computer graphics, and an excellent springboard up to the most sophisticated (and expensive) graphics devices.

We hope this book will display some of the excitement and enthusiasm for computer graphics experienced by us, our colleagues and students. To demonstrate just how useful computer drawings are for illustrating books and pamphlets, all the pictures in the following chapters were drawn by computer specifically for this book.

Ian O. Angell
Brian J. Jones

Introduction

This book may be read at a number of different levels. Firstly, it can be considered as a recipe book of graphics programs for those who simply want to draw complex pictures with their BBC microcomputer. We naturally hope that the reader, having drawn these figures, will be inspired to delve deeper into the book in order to understand how and why the programs were constructed. Secondly, some of the programs can be used as a package to produce and to label data diagrams (pie-charts, histograms and graphs) for business and laboratory applications. Finally, and the main objective in writing the book, it is an introductory text to computer graphics that leads the reader from the elementary notions of the subject to such advanced topics as character graphics, construction of three-dimensional objects and hidden surface (and line) algorithms.

The complex programs given later in the book are much too involved to be compiled as single listings; furthermore, there is a great deal of repetition in the use of elementary algorithms. Therefore the *top down* or *modular* approach is used in writing and in explaining programs. The solution to each major graphics problem is conceived as a series of solutions to subproblems. These subproblems can be further broken down into a set of problems to be solved (*modules*). Such modules are programmed in the form of BASIC procedures. Each is given an identifier (in lower case characters) and will solve a particular subtask. Submodules are then combined to solve the major graphics problem. The program listings present the algorithms that are needed for the solution of subtasks, and the naming of the procedures makes an understanding of the algorithms easier. We use lower case characters for procedure identifiers (and groupings of procedures in the text) only: all other program variables are in upper case characters to avoid confusion.

Two cassette tapes are available to accompany the text; they contain all the larger listings in the book, as well as the data for diagrams and character sets used in later programs (which would otherwise have to be constructed by the readers themselves – a rather time-consuming process). The first cassette consists of the two- and three-dimensional geometrical programs, and the second contains the character graphic manipulation, diagram construction and video games etc.

A list of complete programs is given at the end of each chapter, together with suitable data values, for those who want nothing more than to run these programs. In fact it is a good idea for everyone, including the more serious readers, to LOAD the relevant programs from the tape and run them before reading any particular chapter.

There are many REMarks in the program listings, however, and hence some of the programs approach the storage limits of the BBC micro. In these cases you should delete the REMarks before saving the programs. To make the listings easy to read we advise readers to LISTO1 the programs in MODE 7. We have placed a REMark in red before each procedure (on lines with numbers that end in 0) so that they stand out: all other REMarks are in green (on lines with numbers that end in 9). You may find that the latter REMarks take up too much store in which case you should strip them away by typing AUTO9, 10 and by holding the RETURN key down. Even then some of the programs are too big to fit into the store, in which case you must LOAD them after setting PAGE = &1100.

As an example of what to expect we give the program that is required to draw figure I.1, a drawing of a body of revolution in which all the hidden surfaces have been suppressed.

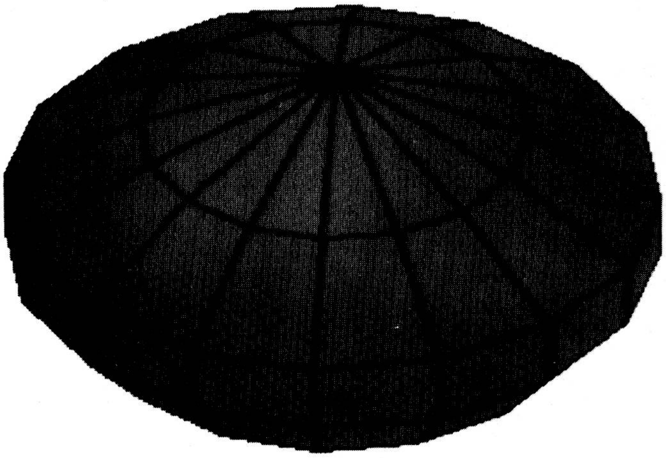


Figure I.1

The program requires the listings 2.1 ('start'), 2.2 (two functions FN X and FN Y), 2.3 ('setorigin'), 2.4 ('moveto'), 2.5 ('lineto') and 2.7 ('triangle'). This combination of procedures will be called 'lib1', and it was designed for drawing line figures on the television screen.

To 'lib1' must be added listings 3.3 ('angle'), 8.1 ('mult3' and 'idR3'), 8.2 ('tran3'), 8.3 ('scale3'), 8.4 ('rot3'), 9.1 ('look3') and 9.2 ('main program'). Procedures, which when combined we call 'lib3', are used for transforming and for observing objects in three-dimensional space.

Listing 10.5 ('revbod') is also needed, together with the 'scene3' procedure given in listing I.1.

Listing I.1

```
6000 REM scene3 / flying saucer
6010 DEF PROCscene3
6020 LOCAL I%
6030 DIM X(12),Y(12),XD(6),YD(6)
6040 DIM A(4,4),B(4,4),R(4,4)
6050 DATA 0,3, 3,2, 5,1, 5,0, 4,-1, 0,-3
6060 RESTORE
6069 REM INPUT horizontal data
6070 NUMV=5
6080 INPUT"NUMBER OF HORIZONTAL LINES",NUMH
6090 INPUT"INITIAL ROTATION",PHI
6099 REM READ definition set
6100 FOR I%=1 TO NUMV+1
6110 READ XD(I%),YD(I%)
6120 NEXT I%
6130 PROCidR3 : PROClook3
6140 PROCrevbod
6150 ENDPROC
```

Figure I.1 requires the data HORIZ = 12, VERT = 9, EX = 1, EY = 2, EZ = 3, DX = 0, DY = 0, DZ = 0, number of horizontal lines NUMH = 16 and initial rotation PHI = 0. Each value has to be typed in individually, when requested by the machine. Run the program with different data values. What happens if HORIZ = 6 and VERT = 4 and the other values stay the same? Set HORIZ = 16, VERT = 12, EX = 1, EY = −2, EZ = 3, DX = 1, DY = 0 and DZ = 0. Try NUMH = 20, PHI = 0.1. You will have to read up to and including chapter 10 to understand the details of what is happening.

This example illustrates the reasoning behind the layout of this book. Assuming that you are a fast typist, or that you have bought the accompanying cassettes, then a relatively complex three-dimensional picture can be constructed very quickly with a minimum of effort. Even one-finger typists (like the authors) will have little difficulty in implementing this and the other programs, before they go on to study the book in detail.

We hope that this example will inspire you to implement *all* the programs in this book, to try most of the examples, and then to go on and draw your very own computer graphics pictures.

Now read the rest of our book and we wish you many happy hours with your BBC microcomputer.

Contents

1 Graphics Operations of the BBC Model B Microcomputer

Throughout the course of this book it will be assumed that the BASIC programming language of the BBC micro is reasonably familiar to the reader. In this first chapter, therefore, we shall be looking at some of the BASIC commands — those concerned wholly or partly with graphics. The display capabilities of the micro will be explored by means of a series of example programs and simple exercises. In the chapters that follow we shall use this knowledge to develop a sound understanding, both practical and mathematical, of computer graphics.

Initially we shall consider the hardware and software facilities that are available for producing pictures. On the BBC computer there is a choice of eight different display MODEs numbered 0 to 7 (the last of which is the special TELETEXT mode which is discussed separately in chapter 13). All the modes produce television pictures by using *raster scan* technology; this is also true of most of the newer commercial mini and main-frame computers. An area of memory at least 1 K(ilo)byte long (1 Kbyte = 2^8 bytes = 1K for short), known as the *screen memory*, is reserved out of the available RAM (Random Access Memory the area available for programming use) to hold the display information for the screen. This memory is examined, bit by bit, as the electron beam sweeps across the raster screen. The display is composed of dots or *pixels* (from picture-cells) each of which, in the simplest case of modes 0, 3, 4 and 6, is represented by a single bit (a binary on/off switch) in the memory. Whenever a binary-on is detected during the raster scan, the beam is switched on for a short period, so producing a dot of light on the screen. In the other modes more than one bit corresponds to each pixel (see later). The screen can be considered in two ways; either as a grid of individual points that are addressed by *graphics* commands or as a grid of blocks in which characters can be placed by *text* commands.

The MODE Command

On the BBC micro there is a palette of sixteen different *actual* colours/effects (numbered 0 to 15) and the MODE command is used to decide how many different colours from this palette will be available and what type of display is used.

MODE N switches to display mode N and decides how much memory must be set aside for the screen memory. The number of pixels (known as the resolution) and TEXT characters available, as well as their physical size, alter with each MODE. The various modes are detailed in table 1.1.

Table 1.1

MODE	TEXT characters (column × row)	Graphics pixels (horizontal × vertical)	Number of colours	Memory used	Pixel
0	80 × 32	640 × 256	2	20K	2 × 4
1	40 × 32	320 × 256	4	20K	4 × 4
2	20 × 32	160 × 256	16	20K	8 × 4
3	80 × 25	–	2	16K	–
4	40 × 32	320 × 256	2	10K	2 × 4
5	20 × 32	160 × 256	4	10K	8 × 4
6	40 × 25	–	2	8K	–
7 TELETEXT	40 × 25	80 × 75	16	1K	–

In this chapter neither the TELETEXT mode (7), which is dealt with separately in chapter 13, nor the two text-only MODEs (3 and 6) will be considered. In the two-colour modes (0 to 4) each pixel is represented by one bit in the memory. This bit defines which *logical* colour is used for that pixel and initially it is set to display a white pixel for a logical 1 and a black pixel for a logical 0. It is possible to change these default assignments (see listing 1.9) so that any actual colour may be displayed for either logical colour. In the four-colour modes (1 and 5) two bits of memory are used to represent each pixel. These two bits represent, in binary notation, a number between 0 and 3 which is the numerical code of the logical colour displayed for that pixel. This allows us to distinguish between four types of pixel and to use a different colour for each type. In the sixteen-colour mode (2) four bits are used per pixel to make up a logical colour between 0 and 15. For all the standard graphics modes the number of pixels vertically is 256; however, since more memory is required to represent a sixteen-colour pixel compared to a four-colour pixel, the number of points available horizontally varies inversely with the number of colours used.

Screen Memory

This type of screen picture is referred to as a *memory mapped* display since it corresponds directly to the contents of an area of memory. On the BBC micro the memory used for the display, called the screen memory, starts at location

HIMEM (which is reset by the MODE command) and ends at location 32767 (the end of RAM). A simple exploration of how the screen is affected by changing the contents of the memory can be made with a program such as listing 1.1.

Listing 1.1

```
10 REPEAT
20 INPUT'" Which mode ",M
30 MODE M
40 ?HIMEM=137
50 UNTIL FALSE
```

This program uses the indirection operator '?' (see the user guide) to indicate the intention of placing a number (VALUE) in the memory location with address HIMEM. This is the first location of the display file and it holds the information for the top left-hand corner of the screen. Run the program and select mode 4 (or 0). Since each location, or *byte*, contains eight binary *bits*, the first eight pixels on the display are affected in such two-colour modes. These change to show a pattern of dots that is equivalent to the binary representation of the VALUE, in this case 10001001. Run the program again but now choose a four-colour mode (1 or 5). This time only four pixels will be affected since two bits are used per pixel. Try the same program with the sixteen-colour mode (2) and you will see that only two pixels are now affected since four bits per pixel are used. Table 1.2 shows how the eight bits in one byte are split up by different modes to represent the logical colours of the different number of pixels that use the specific value 10001001.

From the above we see that it is possible to construct a complete picture by storing various values in the locations of the display file. This is tedious for two-colour pictures, and extremely complicated for pictures with a greater number of colours. Obviously we need a simpler method for altering the contents of the display file. BASIC provides the graphics commands that deal precisely with this problem. The first command to be considered is PLOT, a very complicated command that offers many options, as we shall see later. For the time being, however, we shall limit ourselves to using just three options, PLOT 69, PLOT 4 and PLOT 5. Two of these are considered so important that they are given alternative names: PLOT 4 is MOVE and PLOT 5 is DRAW.

Because the number of pixels and their relative positions are MODE-dependent, a new object is defined for the BBC micro, the *addressable point* (or *point* for short). All the graphics commands treat the display as a grid of 1280 addressable points horizontally by 1024 addressable points vertically (1 310 720 in total). Each point is uniquely defined by a pair of integers such that point (X, Y) is X addressable points to the left and Y points above the screen origin (point (0, 0) at the bottom left-hand corner of the screen). We have already seen that the number of available pixels is mode-dependent; in fact each pixel is composed

Table 1.2

Two-colour MODES

	1	0	0	0	1	0	0	1	
pixel 1 =	B_7								= 1 = logical colour 1
pixel 2 =		B_6							= 0 = logical colour 0
pixel 3 =			B_5						= 0 = logical colour 0
pixel 4 =				B_4					= 0 = logical colour 0
pixel 5 =					B_3				= 1 = logical colour 1
pixel 6 =						B_2			= 0 = logical colour 0
pixel 7 =							B_1		= 0 = logical colour 0
pixel 8 =								B_0	= 1 = logical colour 1

Four-colour MODES

	1	0	0	0	1	0	0	1	
pixel 1 =	B_7				B_3				= 11 = logical colour 3
pixel 2 =		B_6				B_2			= 00 = logical colour 0
pixel 3 =			B_5				B_1		= 00 = logical colour 0
pixel 4 =				B_4				B_0	= 01 = logical colour 1

Sixteen-colour MODE

	1	0	0	0	1	0	0	1	
pixel 1 =	B_7		B_5		B_3		B_1		= 1010 = logical colour 10
pixel 2 =		B_6		B_4		B_2		B_0	= 0001 = logical colour 1

of a small block of addressable points (see the last column of table 1.1). This correspondence between points and pixels is all worked out by the computer and means that, since we are working with addressable points, we can switch between MODEs without having to change the programs. Any command that affects one point will actually affect the whole pixel that contains this point. The use of points is a great help when changing between MODEs since point (640, 512) always represents the middle of the screen, whereas if we counted in pixels then pixel (80, 128) is close to the left side of the screen in mode 0, one-quarter of the way across the screen in mode 1 and the middle of the screen for mode 2 only. The graphics commands help in constructing pictures by allowing us to control a *graphics pen*, which is initially positioned over point (0, 0).

PLOT 4, X, Y or MOVE X, Y moves the pen from its current position and places it above the pixel that contains the point (X, Y).

PLOT 69, X, Y moves the pen to the point (X, Y) and plots a pixel there.
PLOT 5, X, Y or DRAW X, Y draws a line from the pen's current position to the point (X, Y).

After the execution of these commands the pen remains over the last point that was visited, while awaiting the next command. Before examining the other more advanced graphics commands, a simple example and some exercises will serve to demonstrate what can be achieved with only the few commands that have been dealt with so far.

Example 1.1
PLOT 69 can be used to scatter pixel dots over the screen. The program in listing 1.2 illustrates the flexibility of addressing the pixels via the overlying grid of addressable points.

Listing 1.2

```
10 INPUT"Which mode",M : MODE M
20 REPEAT
30 PLOT 69,RND(1280),RND(1024)
40 UNTIL FALSE
```

Exercise 1.1
Alter listing 1.2 to DRAW lines, either between the random points as they are generated or from the middle (point (640, 512)) to each point.

Exercise 1.2
Write a program to calculate the position of lines that form a grid on the screen. DRAW them by using two FOR. . .NEXT loops (one for horizontal lines, the other for vertical lines).

Exercise 1.3
Write a program that accepts the INPUT of N pairs of addressable point co-ordinates from the keyboard, and then DRAWs an irregular polygon of N sides by joining the points in order. (Remember to join the last point to the first.)

PRINT, LIST and VDU

So far we have not discussed the most obvious method of changing the display, namely the text commands PRINT (PRINT TAB), LIST and VDU (consult the user manual for a full description of these commands). This is because these use character-size text blocks and are designed primarily for use with low-resolution graphics. This will be dealt with in chapter 5, but since the BBC micro allows