Advanced Graphics with the IBM Personal Computer

Other Macmillan titles of related interest

Advanced Graphics with the Acorn Electron

Ian O. Angell and Brian J. Jones

Advanced Graphics with the BBC Model B Microcomputer

Ian O. Angell and Brian J. Jones

Advanced Graphics with the Sinclair ZX Spectrum

Ian O. Angell and Brian J. Jones

Assembly Language Programming for the Acorn Electron

Ian Birnbaum

Assembly Language Programming for the BBC Microcomputer (second edition)

Ian Birnbaum

Advanced Programming for the 16K ZX81 Mike Costello

Using Your Home Computer (Practical Projects for the Micro Owner)

Garth W. P. Davies

Microchild – Learning through LOGO

Serafim Gascoigne

Beginning BASIC Peter Gosling

Continuing BASIC Peter Gosling

Practical BASIC Programming Peter Gosling

Program Your Microcomputer in BASIC Peter Gosling

More Real Applications for the ZX81 and ZX Spectrum Randle Hurley

The Sinclair ZX81 – Programming for Real Applications

Randle Hurley

Programming in Z80 Assembly Language

Roger Hutton

Beginning BASIC with the ZX Spectrum

Judith Miller

Microprocessor and Microcomputer Technology Noel Morris

The Commodore 64 in Action – Sound, Graphics and Filing

M. M. Novak

Using Sound and Speech on the BBC Microcomputer

Martin Phillips

Understanding Microprocessors B. S. Walker

Computer Literacy: A beginners' guide Vincent Walsh

Assembly Language Assembled – for the Sinclair ZX81 Anthony Woods

Advanced Graphics with the IBM Personal Computer

Ian O. Angell

*Department of Computer Science
University College
University of London
England*

M
MACMILLAN

© Ian O. Angell 1985

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provisions of the Copyright Act 1956 (as amended).

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1985

Published by
Higher and Further Education Division
MACMILLAN PUBLISHERS LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

British Library Cataloguing in Publication Data

Angell, Ian O.

Advanced graphics with the IBM personal computer.

1. Computer graphics 2. IBM Personal Computer
—Programming

I. Title

001.64'43 T385

ISBN 978-0-333-37063-6

ISBN 978-1-349-07338-2 (eBook)

DOI 10.1007/978-1-349-07338-2

Related Software Diskette: ISBN 978-0-333-38854-9

To Florence

Contents

<i>Preface</i>	x
<i>Acknowledgements</i>	xiii
Introduction	xiv
Aims of the book. Motivation and format. How to approach the contents.	
Three levels: example programs, a program package, or a textbook.	
Deleting remarks. A substantial example to illustrate what can be drawn after reading the contents of this book and MERG(E)ing together various listings	
1 Graphics Commands on the IBM Personal Computer	1
Switching between colour monitor and monochrome display. Display modes and colour. Text and graphics output. BASIC commands for drawing with characters and pixels. Colour and area-filling commands. CIRCLE and aspect ratio. GET and PUT. Logical plotting. Simple animation. WINDOW and VIEW	
2 From Real Coordinates to Pixels	27
Routines for mapping real two-dimensional space into screen pixels. Scaling factors, moving origin, drawing lines and colouring polygonal areas given in real coordinates. Windows on space. Patterns as a first step in two-dimensional graphics	
3 Two-Dimensional Coordinate Geometry	48
Two-dimensional coordinate systems — origin, axes, points, vectors, lines, curves and areas, and their properties. Functional representation and parametric forms. Polygons and convex areas: inside and outside, orientation. Intersection of convex polygons. Hatching polygons	
4 Matrix Representation of Transformations on Two-Dimensional Space	70
Explanation of matrices. Translation, rotation and scaling (reflection) transformations. Three by three matrix representation of two-dimensional space. Using matrices to transform points. Inverse transformations. Combining transformations. Positions. Construction and views of two-dimensional scenes	

5 Character Graphics and Pixel Blocks	92
Characters on the IBM PC. Screen memory locations. Redefining character sets. Programs for generating and editing characters. Vertical printing. Applications for games etc. Tessellated patterns. Pixel block generation. Recursive programs	
6 Data Diagrams	117
Construction of diagrams. Cursors. Labelling. Drawing axes. Histograms. Pie-charts and simple hatching. Discrete and continuous graphs	
7 Three-Dimensional Coordinate Geometry	141
Three-dimensional coordinate systems. Vector representation of points, lines and planes. Properties of these objects — intersection of lines and planes. Representation of surfaces. Sides of a surface. Orientation of two-dimensional triangles	
8 Matrix Representation of Transformations on Three-Dimensional Space	159
Four by four matrix representation of translation, rotation and scaling (reflection) transformations on three-dimensional space. Inverse transformations. Combining transformations. Rotation about an arbitrary axis	
9 Orthographic Projections	170
Set-up (and storage) of simple objects — vertices, lines, facets and colours. Introduction to projections. Orthographic projection. Positions (SETUP, ACTUAL and OBSERVED). Maintaining the vertical. Definition of scenes. Bodies of revolution (rotation)	
10 Simple Hidden Line and Hidden Surface Algorithms	189
Orientation of three-dimensional triangles. Discussion of general problem of hidden line and surface elimination. A simple algorithm for convex solids — an implementation where objects are not stored (for example, body of revolution). Animation. Shading. The ‘back to front’ method of hidden surface elimination. An algorithm for drawing special mathematical surfaces	
11 Perspective and Stereoscopic Projections	207
Theory of perspective. Drawing simple objects in perspective. Extension of previous algorithms to the perspective case. Theory of stereoscopic views, plus examples	
12 A General-Purpose Hidden Surface and Hidden Line Algorithm	221
Overlapping facets. Network representation of given three-dimensional view. Topological sorting, linked lists and stacks. A hidden surface algorithm to	

deal with the general case of a perspective view of a stored three-dimensional scene that has no special properties

13 Advanced Techniques for Programming the Text-Only Mode	235
Screen buffer. Text pages. WIDTH. Simple animation methods. Scrolling. Two video games. Multiscreen animation. A text picture editor	
14 Projects	251
Ideas for extended programs in computer graphics	
<i>References and Further Reading</i>	255
<i>Appendix: Contents and Availability of Related Software</i>	257
<i>Index</i>	258
<i>Where to Find Routines referred to in Text</i>	271

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 to only the most elementary computing devices, can now afford sophisticated computers like the IBM Personal Computer. They are able to escape from the limitation of tabular numerical output and buy inexpensive special-purpose colour graphics devices to produce hardcopy, or simply photograph the screen. 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 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 business or scientific computer user. For most purposes these packages are unnecessary. This book, as well as being an introduction to computer graphics, may be considered also as 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 the graphics potential of the IBM Personal Computer 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 users' knowledge grows along with the package and they are far less likely to misinterpret any of the graphical routines. References and relevant further reading material are also recommended in order to expand the reader's horizons in the subject.

In explaining some of the techniques of computer graphics it is assumed that the reader has an elementary knowledge of Cartesian coordinate geometry (the author recommends the books by Cohn (1961), Coxeter (1974) and McCrae (1953) — see References and Further Reading at the end of the book), and also of the BASIC programming language on the IBM Personal Computer. The program listings, however, are written in such a way that complete beginners can run them and produce useful displays without having any knowledge in either of the aforementioned areas. Many interesting programming exercises are proposed, and these should raise the standard

of readers' BASIC expertise. BASIC is a universally popular language that is available (in various guises) on all types of microcomputer, so the programs may be easily adjusted to run on micros other than the IBM: it is also a good medium for transmitting the algorithms used in computer graphics, 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 for describing 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 which 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 user 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, a companion diskette is available, which contains all of the 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 the University of London, as well as in commercial tutorials and seminars. 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 students; and the step by step approach brings them quickly to the level of very sophisticated computer graphics. That level is outside the scope of this book, but where necessary readers will find relevant references to guide them into the more advanced topics (see Newman and Sproull (1979), Foley and van Dam (1982), Harrington (1983)).

This book is aimed in general at those who are competent BASIC programmers but who are complete beginners in graphics. It starts with a general introduction to the graphics features of the IBM Personal Computer. Then follows the section on high-resolution graphics; this contains the elementary ideas and basic information about pixel and two-dimensional graphics, which must be mastered before the more involved ideas of three-dimensional graphics are attempted.

Some of these ideas are then used in a section on low-resolution graphics, which relates to the construction and use of characters and blocks of pixels, and which includes a short excursion into recursive programs. This in turn leads to a chapter on the display of data (in line drawings and colour) — histograms, pie-charts etc. — probably the most important non-specialised, commercial use of computer graphics.

The reader is then introduced to the geometry of three-dimensional space, and to a variety of projections of this space onto 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,

text-only graphics and the construction of a screen editor and video animation are discussed.

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 IBM Personal Computer is an ideal machine on which to learn and use the basics of computer graphics, and an excellent spring-board up to the most sophisticated (and expensive) graphics devices.

It is hoped that this book will display some of the excitement and enthusiasm for computer graphics experienced by myself, my colleagues and students. To demonstrate just how useful computer drawings are for illustrating books and pamphlets, all but three of the pictures in the following chapters were drawn on the IBM Personal Computer specifically for this book.

*University College
London*

IAN O. ANGELL

Acknowledgements

I should like to thank IBM (UK) for their loan of an IBM Personal Computer during the production of this book. I am particularly grateful to Benn Goulding and Leslie Banks for all their advice, help and encouragement.

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 IBM Personal Computer. Naturally it is hoped that the reader, having drawn these figures, will be inspired to delve more deeply 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 purposes. 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 through 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 given 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 explaining programs. The solution to each major graphics problem is conceived as a series of solutions to subproblems. These subproblems may be further broken down into a set of problems to be solved (*modules*). Such modules are programmed in the form of BASIC subroutines. Each is given an identifier (in lower case characters and referred to in the text inside quotes, such as 'scene3') and will solve a particular subtask. Then the submodules are combined to solve the required graphics problem. The program listings present the algorithms that are needed for the solution of these subtasks, and the naming of procedures in this way makes the understanding of the algorithms easier. Lower case characters are used solely for comments and routine identifiers (and groupings of routines in the text): all other program variables are in upper case to avoid confusion.

To aid the more serious readers, a companion diskette to this book is available. This contains all the listings in the book, as well as the data for pixel blocks and character sets used in later programs (which would otherwise have to be constructed by the readers themselves — a rather time-consuming job).

For those who merely want to run these programs, a complete list of programs is given at the end of each chapter, together with suitable data values. In fact it is a good idea for everyone, including the more serious readers, to LOAD and/or MERGE the relevant listings from diskettes and run the programs before approaching the text of each chapter.

There are plenty of REMarks in the program listings. However, some of the programs approach the storage limits of the IBM Personal Computer; in these cases you should delete the REMarks before resaving the programs on another disk. A REMark is placed before each routine (on program lines with numbers ending in 0) so that it indicates the start of the routine: all other REMarks are on program lines with numbers not ending in zero. Most of these REMarks are on lines ending in 9, with two major exceptions. Because subroutines do not have input and output parameters, program lines ending in 1 are reserved for ****in**** REMarks which give a list of all input parameters needed by the subroutine that are not generated as output parameters from internal calls to other routines. Program lines ending in 2 are reserved for ****out**** REMarks which give a list of all output parameters from a subroutine. You may find that all these REMarks take up too much store and slow down the execution time of the program, in which case you should strip them away with the help of listing I.1. All programs except listing I.1 begin at program lines not less than 100. So MERG(E)ing listing I.1 into an existing program listing will place this program at the front of the code. Running the combined program will delete every line with line number not ending in zero from the original program (all the REMarks) and request the name of a disk file where the deREMed program, minus listing I.1, can be stored for future use. You will have to know how BASIC programs are stored on the Personal Computer before you will understand listing I.1. Note that locations &H30 and &H31 of the BASIC segment hold the low and high bytes of the address of the start of the BASIC program.

Listing I.1

```

1 'MERGE this program to the front of a target program which
  contains REMarks to be deleted. All target lines with numbers
  ending in zeros are executable. The rest are REMarks.
2 DEF SEG
3 'INPUT the name of the file which will hold the deREMed listing.
4 PRINT " Type name of disk file for deREMed listing ";
  "e.g. B:NOREMS.BAS " : INPUT FILES
5 'locations PTR% and PTR%+1 of the BASIC segment hold the lo- and hi-
  byte values of the location of a program line. Initially the
  first program line. Store these two bytes.
6 PTR%=&H30 : STOL%=PEEK(PTR%+1) : STOO%=PEEK(PTR%)
7 '129 signifies the END of the derem program in the memory.
  Program lines that follow are from the target program.
8 IF PEEK(PTR%+4)<>129
  THEN PTR%=PEEK(PTR%+1)*&H100+PEEK(PTR%) : GOTO 8
9 PTR%=PEEK(PTR%+1)*&H100+PEEK(PTR%)
10 'TARGETPTR% and TARGETPTR%+1 hold the start of target program.
  PROGPTR% (and +1) will be used to over-write the target program
  locations, with program lines that are not REMarks.
11 PROGPTR%=PTR% : TARGETPTR%=PTR%
12 'PTR% (and +1) point to the present line under consideration,
  NEXTPTR% (and +1) point to the next program line.
13 NEXTPTR%=PEEK(PTR%+1)*&H100+PEEK(PTR%)
14 'NEXTPTR% is zero at the end of target program.
15 IF NEXTPTR%=0 THEN GOTO 36
16 'find the line number of the program line being considered.
17 LINENUM%=PEEK(PTR%+3)*&H100+PEEK(PTR%+2)
18 PRINT "considering line number";LINENUM%
19 'if LINENUM% doesn't end in a zero it is a REMark.
20 IF LINENUM% MOD 10 <> 0 THEN GOTO 33

```

```

21 'program lines are moved up the memory, over-writing other
    lines of the target program that have already been checked.
22 LINESIZE%=NEXTPTR%-PTR%
23 'store the address of the next program line. That is add the
    size of the program line to the present address.
24 POKE PROGPTR%,(PROGPTR%+LINESIZE%) MOD &H100
25 POKE PROGPTR%+1,INT((PROGPTR%+LINESIZE%)/&H100)
26 'move up the locations holding program line.
27 FOR I%=2 TO LINESIZE%-1
28 POKE PROGPTR%+I%,PEEK(PTR%+I%)
29 NEXT I%
30 'go on to next line in deREMed program.
31 PROGPTR%=PROGPTR%+LINESIZE%
32 'go on to next line in the target program.
33 PTR%=NEXTPTR%
34 GOTO 13
35 'mark the end of the program with two zeros.
36 POKE PROGPTR%,0
37 POKE PROGPTR%+1,0
38 'the start of program pointers show the deREMed program.
39 POKE &H30,TARGETPTR% MOD &H100
40 POKE &H31,INT(TARGETPTR%/&H100)
41 'save the deREMed program.
42 SAVE FILES$
43 'reset the start of program pointers.
44 POKE &H30,ST00% : POKE &H31,ST01%
45 END

```

The variable names in the programs will sometimes be rather long-winded, and therefore inefficient. Also the programs themselves may seem over-explicit in the number of routines involved. This is because the programs are meant to convey the ideas behind an algorithm, and the speed of execution is of secondary concern. The statements themselves are generously spaced out to aid clarity. Readers who wish to use the programs on a regular basis are advised to slim down the variable names and reduce the number of spaces, and also cannibalise the programs to make them more efficient. For example, calls to small subroutines (for example, 'lineto') can be replaced by explicit lines of code in the programs.

Listing I.2 is given as an example of what graphics to expect after reading this book. This listing is used to draw figure I.1, a drawing of a wire-body of revolution — in this case a goblet.

The program requires listings 1.1 ('colour' and 'monochrome' display routines, usually called 'lib0'), 2.1 ('start', two functions FN X and FN Y), 2.2 ('setorig'), 2.3 ('moveto'), 3.3 ('lineto' and 'clip') and 3.4 ('angle'). If you have release 2.0 of Advanced BASIC then the four listings 2.1, 2.2, 2.3 and 3.3 may be replaced by the more efficient equivalent listings 2.1a, 2.2a, 2.3a and 2.4a. This combination of routines is part of a second library, called 'lib1', which was designed for drawing lines and areas on the colour monitor.

Listings 8.1 ('mult3' and 'idR3'), 8.2 ('tran3'), 8.3 ('scale3'), 8.4 ('rot3') and 9.1 ('look3') must also be MERGED into the program. These routines form the 'lib3' library, which is used for transforming and observing objects in three-dimensional space.

The program also needs listing 9.9 ('revbod') to be MERGED into the 'scene3' main program given in listing I.2.

Figure I.1

Listing I.2

```

100 'scene3 / goblet
110 GOSUB 9700 'start
120 DIM X(24),Y(24),XD(12),YD(12)
130 DIM A(4,4),B(4,4),R(4,4)
140 DATA 0,-8, 4,-8, 4,-7.5, 0.5,-7.5, 0.5,-1, 3,-0.5,
        4,1, 7,7, 6.5,7, 3.5,1, 2.5,0, 0,0
150 RESTORE 140
159 'generate definition set.
160 NUMV%=11 : N1%=NUMV%+1
170 INPUT "Type number of horizontal lines ",NUMH%
180 INPUT "Type initial angle ",PHI
190 FOR I%=1 TO N1%
200 READ XD(I%),YD(I%)
210 NEXT I%
220 GOSUB 9300 'idR3
230 GOSUB 8200 'look3
240 GOSUB 60000 'colour monitor
250 XMOVE=HORIZ/2 : YMOVE=VERT/2
260 GOSUB 9600 'setorigin
270 GOSUB 6500 'revbod
280 GOSUB 60100 'monochrome monitor
290 END

```

To run on the colour monitor the program needs variable **MODE%**, **PALETTE%**, **BACKGROUND%**, **FOREGROUND%** and **PAPER%**. Try 1, 1, 0, 0, 3 respectively. The production of figure I.1 requires the further data **HORIZ = 36**, **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 on request by the machine. Run the program with different data values: what happens if **HORIZ = 60** and the other values stay the same? Set **HORIZ = 16**, **EX = 1**, **EY = -2**, **EZ = 3**, **DX = 1**, **DY = 0** and **DZ = 0**. Try **NUMH% = 20**, **PHI = 0.1**. Also run with

MODE% = 2 and PAPER% = 1. You will have to read up to and including chapter 9 to understand the details of what is happening.

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

It is hoped 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 to draw your very own complex computer graphics pictures.

Now you can go on and read the rest of this book, and I wish you many happy hours with your IBM Personal Computer.