£1.00

ELBUG FOR THE ELECTRON

Vol 1 No 4 March 1984

GAMES
*** KILLER DICE**
*** BLOCK BLITZ**
PLUS
*** SOUND ENVELOPE EDITOR**
*** SPIDER AND FLY**
*** CONVERTING BBC MICRO PROGRAMS**
PLUS
*** REVIEWS**
*** SOFTWARE LIST**
*** POSTBAG**
*** HINTS & TIPS**
And much more

MOVING CHEQUER BOARD

EDITORIAL

THIS MONTH'S MAGAZINE

This month we publish two excellent games, KILLER DICE and BLOCK BLITZ. We believe that they compare well with commercially available games despite being written entirely in Basic. If you enjoy games, then you will find the time spent entering these two programs into your Electron, well worth while.

Starting this month is a new series which will look at the problems likely to be encountered when trying to convert programs from the BBC micro to run on the Electron. Several letters have already been received from ELBUG readers seeking advice on this topic, and with so much BBC Micro software already around this is a potentially fruitful source of programs for the Electron.

We have also compiled a list for this issue of all the currently available commercial software for the Electron. As the quantity of software for the Electron is likely to grow rapidly we do not intend to maintain this as a regular feature of the magazine.

POSTBAG

We have been pleased to receive growing numbers of letters from ELBUG members and again some of these are printed in the Postbag section of the magazine. We do like to hear from you, and your comments and opinions on the magazine and the Electron are always welcome. We will also try to answer technical queries that you may have about the use of the Electron, but please enclose an SAE with your letter.

Mike Williams

TICE BOARD NOTICE BOARD NOTICE BOARD NOTICE BOA

ADD-ONS FOR THE ELECTRON

We have just received our review sample of the SIR expansion board add-on for the Electron. Quick tests with a few of the ROMs currently available for the BBC Micro (View, BCPL, Toolkit, Exmon and GROM) were promising, but some ROMs will need to be re-written before they work correctly on the Electron. We will be publishing a full review in a later issue of ELBUG.

3D ROTATION

In the production of the January/February issue of ELBUG, the very last part of this program (the procedure PROCshift) got lost and is reprinted in full below. Despite this, the main rotation and zoom features of the program should have worked with no problems at all.

```
2230 DEF PROCshift
2240 REM MOVE CUBE UP/DOWN/LEFT/RIGHT
2250 hor%=hor%+hmove%
2260 vert%=vert%+vmove%
2270 VDU 29,hor%;vert%;
2280 ENDPROC
```

ELBUG MAGAZINE

GENERAL CONTENTS

PAGE	CONTENTS
2	Editorial
4	Electron Add-ons
5	Killer Dice
8	Postbag
9	Electron Software List
10	The Spider and The Fly
11	Points Arising
12	Electron Graphics (Part 4)
17	How to tell an Electron from a BBC Micro
18	Moving Chequer Board
20	Sound Envelope Editor
24	Graphics Software for The Electron
27	Using BBC Micro Programs on an Electron (Part 1)
30	More New Books for Electron Users
32	Block Blitz

PROGRAMS

PAGE	CONTENTS
5	Killer Dice Game
10	The Spider and The Fly
12	Electron Graphics Examples
18	Moving Chequer Board
20	Sound Envelope Editor
32	Block Blitz Game

HINTS & TIPS

PAGE	CONTENTS
16	TAB
16	*SAVE Extension
16	Colour Video Output
19	Extra Function Keys
19	PLOT Equivalence
23	Cursor Controls
23	WIDTH
23	Getting Rid of ? on INPUT
26	Alternative Escape Actions
26	Disabling the Escape Key & Destroying Programs
26	* Commands
26	Lower Case Basic

ELECTRON ADD-ONS – THE LATEST POSITION

by David Graham

As we reported last month, there appear to be three main contenders in the race to produce add-ons for the Electron: Sir Computers, Solidisk Technology, and Acorn themselves. Of these the latter two have not formally launched their product. Acorn's box of tricks, we are informed, will probably provide an analogue interface (for joysticks etc.), a printer port, and two ROM sockets. No firm details of production dates or price have been announced, and all that a spokesman for Acorn would reveal was that the unit would probably cost under £100. It is likely, we think, to be some considerable time before the Acorn add-on sees the light of day.

Solidisk Technology demonstrated a prototype of their "General Purpose Interface" for the Electron to us last December. But the production unit will be different from the prototype we understand, and the functions of their add-on seem still to be in a state of flux. To the printer port, non-proportional joysticks and ROM sockets incorporated in the prototype, Solidisk are now considering adding a disc interface. But as with Acorn the product still seems to be at the planning stage, and no firm dates or prices are available.

By contrast, Sir Computers appear to have two products poised for launch at the time of writing: a 12-ROM Board, and a Printer and Joystick Interface. Both boards should be available by the time that you read this article. They cost £40 and £45 respectively, exclusive of VAT; and the boards are said to use a connection system which allows one to be plugged into the other. Both boards will be individually housed in a matching casing.

SIR 12-ROM BOARD

As its name suggests, this board will take up to twelve ROMS (or Read Only Memories). The purpose of such devices is to give the Electron instant access to languages other than Basic, such as Pascal for example, or applications software such as a word processor. ROMs (or EPROMs) containing such programs are plugged into the ROM board, and may be called up instantly whenever they are required. Such a ROM system is implemented on the BBC micro - there are three free ROM sockets inside the Beeb - and ROM software has proved to be very popular with BBC users. In fact some of the ROMs currently available for the BBC micro may be used without modification on Sir's Electron ROM Board, it is claimed. Examples given are the two word processors View and Edword, and Computer Concept's new Graphics ROM. There will doubtless be a flow of custom built ROMs for the Electron once Sir's board has been released. Our own BBC micro ROMs, TOOLKIT (a Basic programmer's aid) and EXMON (a machine code monitor), will be released for the Electron shortly.

SIR PRINTER AND JOYSTICKS INTERFACE

Again the title of this board defines its function. It gives you a printer interface to the so-called Centronics standard. This is the same as the parallel printer interface on the BBC micro, and will allow almost any modern printer to be connected to the Electron. This allows text and programs to be listed out. If in addition you want to reproduce graphics screens on your printer you will need an extra piece of software called a screen dump; and we shall be publishing a selection of these for the most popular printers in due course.

This second board of Sir's also contains a joystick interface. This is, we are informed, a standard analogue interface which will allow BBC type joysticks or other analogue voltage sources to be used with the Electron.

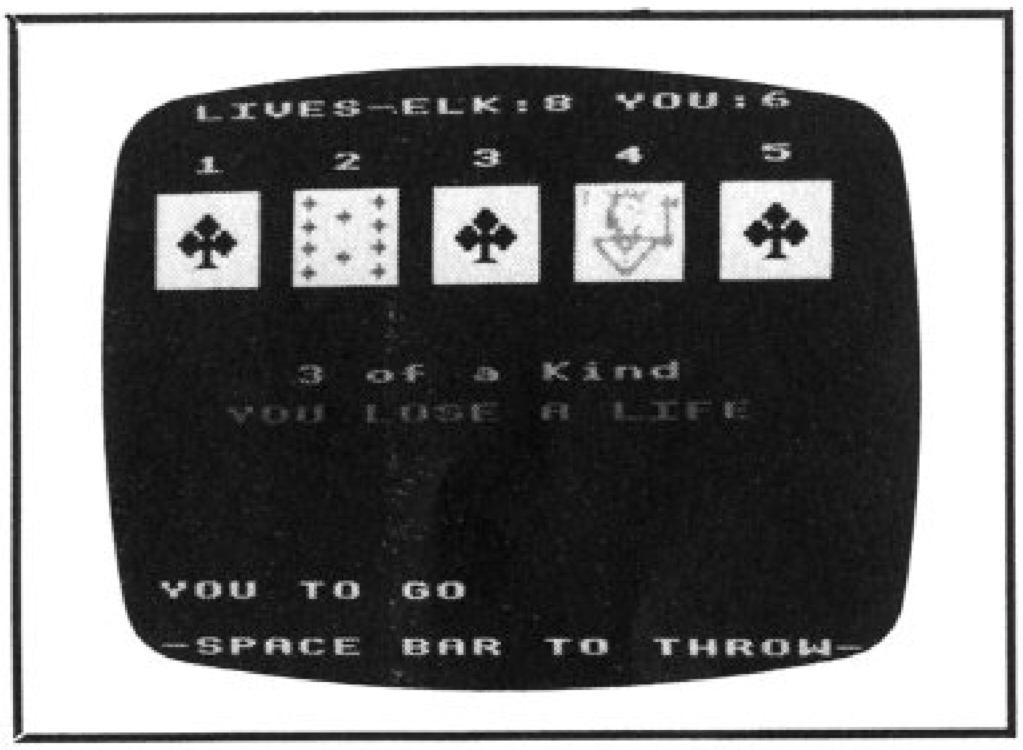
If either of Sir's boards are on schedule, we will be reviewing them in the next issue. But if you want to find out more before then, telephone SIR on 0222 621813. ●

KILLER DICE

by D. Jackson

Why not treat yourself to our excellent poker dice game, called "Killer Dice"? This is a gambling game which involves more thought and skill than you might at first think. Like all true gamblers you will soon find yourself having just one more game, and one which you are bound to win this time!

For those of you who won't own up to having played with poker dice before, we start with some basic information. There are five identical dice, each with six faces representing the 9, 1Ø, Jack, Queen, King and Ace. When the dice are thrown, it is the uppermost faces which count, and it is these which are displayed on the screen. You are playing against the computer, taking it in turns to better the score of each other until one of you loses. At the start of the game, both players have 9 lives and the contest finishes when one of the players has no more lives left.



```
LIVES-ELK:8 YOU:6
 1   2   3   4   5

   3 of a Kind
 YOU LOSE A LIFE

YOU TO GO
-SPACE BAR TO THROW-
```

Except when you are throwing first at the start of a new game, you have the option of retaining any of the dice on the table and just throwing the remainder. This is the course of action to follow if some of the dice on the table are particularly high scoring. In this computer version, you can 'hold' any of the dice 1 to 5 by pressing any of the number keys 1 to 5 before pressing the space bar for your next go. If you make a mistake, pressing the key marked Ø clears any of the dice currently in a hold state and you can start again.

Different combinations of dice have different scoring values. Here are the winning combinations in order (highest value first):

1. 5 of a kind	6. 3 of a kind
2. 4 of a kind	7. Two pairs
3. Full House	8. One pair
4. High run	9. Ace high
5. Low run	

If a player throws 5 of a kind, his partner is given 5 throws to equal or better the hand. A full house is when a player gets 3 of a kind and a pair together.

As you get more experienced at playing Killer Dice, you will find yourself working out various strategies to defeat the computer. You will have to try hard to succeed and you will also need a certain amount of luck as well.

PROGRAM NOTES

The program is well structured and therefore relatively easy to follow, but you will need to take great care with the mass of data statements at the end. These are cunningly used to define the characters that represent each of the dice on the screen. The card number (9 to 13) indexes to a DATA statement which gives the colour followed by the 15 character definitions that are then poked in turn directly into the character buffer. Each character is dynamically defined as ASCII 224 and placed in a text window 3 wide and 5 deep to produce the dice face. Look at the procedure PROCroll at line 238Ø to see the detail of this.

One consequence of this technique is that renumbering the program in any way may well result in a corrupted display because of the way the RESTORE command is used in the program.

```
  1Ø REM PROGRAM KILLER DICE
  2Ø REM VERSION EØ.1
  3Ø REM AUTHOR  D.JACKSON
  4Ø REM ELBUG   MARCH 1984
  6Ø :
  7Ø Prog$="KILLER DICE"
  8Ø ON ERROR GOTO311Ø
  9Ø :
1ØØØ MODE6:PROCintro
1Ø1Ø REPEAT:MODE2:PROCinitz
1Ø2Ø REPEAT:VDU17,3,31,Ø,26
1Ø3Ø IF T% PROCyou:ELSE PROCbbc
1Ø4Ø PROCthrow:PROCscore
1Ø5Ø IF B% T%=-T%-1:ELSE PROClose
1Ø6Ø UNTIL D%:PROCdead
1Ø7Ø UNTIL E%:MODE6:PROCend
1Ø8Ø END
1Ø9Ø :
11ØØ DEFPROCintro
111Ø *FX225,128
112Ø DIMD(5),N(6),P(6),L(1)
113Ø Z%=RND(-TIME):T%=RND(2)-2
115Ø VDU31,12,9:PRINT Prog$
116Ø VDU31,5,17
117Ø PRINT"PRESS SPACE BAR TO CONTINUE"
118Ø VDU31,17,18
119Ø REPEAT:PROCinkey:UNTIL I%=32
12ØØ ENDPROC
121Ø :
122Ø DEFPROCinitz
123Ø VDU23,1,Ø;Ø;Ø;Ø;
124Ø L(Ø)=9:L(1)=9
125Ø B%=Ø:F%=Ø:V%=1:E%=Ø:D%=Ø
126Ø VDU17,3,31,1,1
127Ø PRINT"LIVES-ELK:9 YOU:9"
128Ø VDU17,5,31,1,4
129Ø FORI%=1TO5:PRINT;I%;SPC(3);:NEXT
13ØØ ENDPROC
131Ø :
132Ø DEFPROCyou
133Ø PRINT"YOU TO GO";
134Ø Z%=B%OR(-F%>ØAND-F%<5)
135Ø IF Z% VDU17,1:PRINT"-INPUT HOLD"
136Ø VDU17,3,31,Ø,29
137Ø PRINT"-SPACE BAR TO THROW-"
138Ø REPEAT:PROCinkey:I%=I%-48
139Ø IF Z%:IF I%>Ø AND I%<6:PROChold
14ØØ IF I%=Ø:PROCcancel
141Ø UNTIL I%=-16
142Ø ENDPROC
143Ø :
144Ø DEFPROCbbc
145Ø PRINT"MY TURN":PROCdelay(3)
146Ø IF B%=Ø AND F%=Ø:ENDPROC
147Ø X%=Ø:Y%=Ø
148Ø FORI%=6TOV%STEP-1
149Ø IF N(I%)>X%:X%=N(I%):Y%=I%
15ØØ NEXT
151Ø IF X%=1 AND F%=Ø:ENDPROC
152Ø FORI%=1TO5
153Ø IF D(I%)=Y% PROChold
154Ø NEXT
155Ø ENDPROC
156Ø :
157Ø DEFPROCthrow
158Ø FORI%=1TO6:P(I%)=N(I%)ANDB%:NEXT
159Ø PROCdelay(2)
16ØØ FORI%=1TO5
161Ø IF D(I%)<8 PROCdice:VDU12
162Ø NEXT
163Ø VDU28,Ø,31,19,15-(B%*2),12,26
164Ø FORI%=1TO5:Z%=D(I%)
165Ø IF Z%<8 PROCdice:PROCdelay(1):PRO
Croll
166Ø NEXT:VDU26:PROCcancel
167Ø ENDPROC
168Ø :
169Ø DEFPROCscore:PkOCdelay(2)
17ØØ VDU28,Ø,17,19,15,12,26
171Ø X%=Ø:Y%=Ø:Z%=Ø
172Ø FORI%=1TO6
173Ø IF P(I%)=Ø:X%=X%+1
174Ø IF N(I%)=Ø:Y%=Y%+1
175Ø IF N(I%)>Z%Z%=N(I%)
176Ø NEXT:X%=X%ANDB%
177Ø IF X%=1:IF P(1)=Ø OR P(6)=Ø:X%=4
178Ø IF Y%=1:IF N(1)=Ø OR N(6)=Ø:Y%=4:
Z%=5
179Ø IF F%=Ø AND Y%=5:F%=-6:V%=D(5)
18ØØ IF F% PROCfive:ENDPROC
181Ø PROCprompt:PROCdelay(1)
182Ø IF Y%>X%:B%=TRUE:ENDPROC
183Ø IF X%>Y%:B%=FALSE:ENDPROC
184Ø B%=1:X%=16
185Ø REPEAT:Y%=6
186Ø REPEAT
187Ø Z%=(P(Y%) OR N(Y%)) AND X%
188Ø IF Z%>Ø AND N(Y%)>P(Y%) B%=TRUE
189Ø IF Z%>Ø AND N(Y%)<P(Y%) B%=FALSE
19ØØ Y%=Y%-1:UNTIL Y%=Ø OR B%<1
191Ø X%=X%DIV2:UNTIL X%=Ø OR B%<1
192Ø IF B%=1:B%=FALSE
193Ø ENDPROC
194Ø :
195Ø DEFPROClose
196Ø IF F% ENDPROC
197Ø PROCdelay(3):VDU17,1,31,2,17
198Ø IF T% PRINT"YOU";:SOUNDØ,-1Ø,1Ø,1
Ø ELSE PRINT" I";:SOUND1,-1Ø,1Ø1,8:SOUN
D1,-1Ø,81,8
199Ø PRINT" LOSE A LIFE"
2ØØØ L(-T%)=L(-T%)-1
2Ø1Ø VDU17,3,31,(-T%*6)+11,1,48+L(-T%)
2Ø2Ø IF L(-T%)=Ø:D%=TRUE
2Ø3Ø ENDPROC
2Ø4Ø :
2Ø5Ø DEFPROCdead
2Ø6Ø VDU31,4,22,17,9
2Ø7Ø IF T% PRINT"YOU ARE";ELSE PRINT"
I AM";
```

➽

```
2080 PRINT" DEAD"
2090 VDU17,3,31,0,28
2100 PRINT"SPACE BAR TO REPLAY"'
2110 PRINT"OR RETURN TO FINISH"
2120 REPEAT:PROCinkey:UNTILI%=32ORI%=13
2130 IF I%=13:E%=TRUE
2140 ENDPROC
2150 :
2160 DEFPROCend
2170 VDU12,17,3,31,5,12:PRINT"<BYEEEE";
2180 *FX225
2190 ENDPROC
2200 :
2210 DEFPROChold
2220 IF F%=-5 ENDPROC
2230 D(I%)=D(I%)+8
2240 VDU17,1,31,(I%*4)-3,12,72
2250 ENDPROC
2260 :
2270 DEFPROCcancel
2280 FORI%=1TO5
2290 D(I%)=D(I%)AND7
2300 VDU31,(I%*4)-2,12,127
2310 NEXT
2320 ENDPROC
2330 :
2340 DEFPROCdice
2350 Y%=(I%-1)*4:VDU28,Y%,11,Y%+2,6
2360 ENDPROC
2370 :
2380 DEFPROCroll
2390 SOUND1,-15,200,1
2400 N(Z%)=N(Z%)DIV2:Z%=RND(6):D(I%)=Z%
2410 IF N(Z%)=0 N(Z%)=1:ELSE N(Z%)=N(Z
%)*2
2420 RESTORE (Z%*50)+2750:READC%
2430 VDU17,C%,17,135
2440 FORJ%=1TO15:READX%,Y%:!&C04=X%:!&
C00=Y%:VDU224:NEXT
2450 VDU17,128
2460 ENDPROC
2470 :
2480 DEFPROCfive
2490 B%=0:Z%=-F%<6 AND Y%=5 AND D(5)>=
V%
2500 IF Z% V%=1:F%=0:T%=-T%-1:ENDPROC
2510 F%=F%+1:IF F%=0:V%=1:ENDPROC
2520 IF F%=-5:T%=-T%-1
2530 VDU17,2,31,0,15
2540 PRINT"Five of Kind To Beat"'
2550 PRINT"Or Equal In ";-F%;" Throws";
2560 IF -F%=1:VDU127
2570 ENDPROC
2580 :
2590 DEFPROCdelay(X%)
2600 TIME=0:REPEAT:UNTIL TIME=X%*50
2610 ENDPROC
2620 :
2630 DEFPROCinkey
2640 *FX15,1
2650 I%=INKEY(9)
2660 ENDPROC
2670 :
2680 DEFPROCprompt
2690 VDU17,2,31,4,15
2700 IF Z%=1 AND Y%=1 PRINT" Ace High"
2710 IF Y%=2 PRINT"One Pair"
2720 IF Z%=2 AND Y%=3 PRINT"Two Pairs"
2730 IF Z%=4 AND Y%=3 PRINT "3 of a Ki
nd"
2740 IF Z%=5 AND N(6)=0 PRINT"Low Run"
2750 IF Z%=5 AND N(1)=0 PRINT"High Run"
2760 IF Z%=4 AND Y%=4 PRINT"Full House"
2770 IF Z%=8 AND Y%=4 PRINT"4 of a Kin
d"
2780 ENDPROC
2790 :
2800 DATA0,&107C7C7C,&38100000,0,0,&10
7C7C7C,&38100000
2810 DATA&7C7C3810,&00000038,0,0,&7C7C
3810,&00000038
2820 DATA&10380000,&0038107C,&0038107C
,&7C7C3810,&10380000,&0038107C
2830 DATA&38000010,&387C7C7C,0,0,&3800
0010,&387C7C7C
2840 DATA&00001038,&7C7C7C10,0,0,&0000
1038,&7C7C7C10
2850 DATA1,&10387C38,&10100000,0,0,&10
387C38,&10100000
2860 DATA&7C381010,&10,&101038,&7C3810
10,&7C381010,&10
2870 DATA&38101000,&101038,0,0,&381010
00,&101038
2880 DATA&10000000,&1010387C,&1010387C
,&38101000,&10000000,&1010387C
2890 DATA&1010,&387C3810,0,0,&1010,&38
7C3810
2900 DATA4,&20202021,&21720000,&C2C2FE
FF,&55540000,&1E0A0800,&800000
2910 DATA&1010101,&1010060,&C0C48082,&
8288D8C0,&8080808,&80A1E3E
2920 DATA&F0F0000,&10101,&FFFFFE64,&64
E6C2C0,&FCFC0C08,&8080808
2930 DATA&1010303,&6060C0C,&93831111,&
38386C6C,&8080,&C8C87C7C
2940 DATA0,0,&3838,&7C7CC6D6,0,0
2950 DATA2,&20505051,&51220000,&8282FE
FF,&55540000,&3E1C0800,&800000
2960 DATA&3030303,&3030111,&1111111,&4
56D0101,&9CAAAA88,&9CBE3636
2970 DATA&F0F0000,&7070707,&FFFFEE44,&
83839329,&FCFC0C08,&C8C8C8DC
2980 DATA&1010303,&6060C0C,&93831111,&
38386C6C,&8080,&C8C87C7C
2990 DATA0,0,&3838,&7C7CC6D6,0,0
3000 DATA1,&51506061,&51520000,&182FEF
F,&55540000,&8080800,&800000
3010 DATA&3030303,&1010151,&29111111,&
456D0101,&88888888,&8080808
3020 DATA&F0F0000,&3030303,&FFFFFE7C,&
```

»

```
C7833901,&FCFC0C08,&BE888888
 3030 DATA&1010303,&6060C0C,&93831111,&
38386C6C,&8080,&C8C87C7C
 3040 DATA0,0,&3838,&7C7CC6D6,0,0
 3050 DATA0,0,0,0,0,0,0
 3060 DATA0,0,&7E7E7E3C,&3C181800,0,0
 3070 DATA&3070707,&3030101,&FFFFFFFF,&
DBDBBDBD,&C0E0E0E0,&C0C08080
 3080 DATA0,&10103,&7E3C1818,&189999DB,
0,&8080C0
 3090 DATA0,0,0,0,0,0
 3100 :
 3110 ON ERROR OFF
 3120 MODE6:IF ERR<>17 REPORT:PRINT" at
line ";ERL
 3130 END
```

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

ADJUSTING SCREEN DISPLAYS

Dear Sir,

The magazine, of which we have now received two copies, is excellent. We have been impressed with the variety of programs and, as we are all new to home computers, it has been specially gratifying to be able to load the short programs and to quickly see the results of our handiwork. I have also had reason to be grateful for the Rescue program in issue two, when I inadvertently loaded a second program over part of a previous one!

Please can you tell me how I can go about solving a problem I have with the Electron. I am losing the top line of print on my TV screen which can be very annoying. Is there anyway of correcting this? R.Pendell

Reply: Thanks for the compliments Mr Pendell. Unfortunately there is no way that you can change the position of the picture on the TV screen from your Electron. However, it should be possible to make an internal adjustment to your TV. This should only be carried out by a competent technician.

CONVERTING PROGRAMS FOR THE ELECTRON

Dear Sir,

In your first issue of ELBUG you stated that Electron programs were tested on an Electron, then loaded into a BBC micro for printing.

Can we assume then that any Electron program will load into a BBC, and more importantly, can some BBC programs be loaded into an Electron?

Some guidance on this would, I think, be appreciated by all your readers. P.M.Beecham

Reply: Most Electron programs will run alright on the BBC micro, unless they access directly some hardware feature that is Electron specific. You may find that the results are different, for example faster games, different sounds etc. Because of its additional features, it is much less likely that a program designed for the BBC micro will run properly on the Electron without some modification. Because there is already plenty of software for the BBC micro, many Electron users will be interested in trying to convert some of these programs and we have started this month a short series of articles which examines this topic in some detail.

TV/MONITOR RESOLUTION

Dear Sir,

Being one of the few people who actually own an Electron, I am now wondering what type of display to buy for it. At the moment I use it mainly with a black and white portable TV, but was thinking of buying a colour portable.

One of the main reasons I decided on the Electron was its excellent high resolution display. However, it wasn't until after I bought the computer that I read that most colour TVs can only show about 350 by 200 pixels. It seems that the Electron's display is being wasted on an ordinary TV and even most monitors can't match the Electron's high resolution mode. D.E.Gordine

Reply: The maximum resolution of the Electron is 640 horizontally by 256 vertically, though the apparent drawing area seems to be to a higher resolution - see this month's article on Electron Graphics. The Electron's medium resolution of 320 by 256 thus reasonably matches the figure which you quote. In practice you will get a much sharper picture from either a monitor or a TV/monitor with an RGB input. The main reason for the improved picture quality is that picture degradation in the UHF IF circuitry of the TV is avoided in a monitor or TV/monitor.

ELECTRON SOFTWARE LIST

Compiled by Alan Webster

We thought that it would be helpful to Electron owners to publish a comprehensive list of all the software, currently available for the Electron, that we could find. You should be able to buy Electron software through computer shops, chain stores such as W.H.Smith, and by mail order as advertised in various magazines. The list of software that can be run on the Electron will no doubt grow rapidly. Indeed, there may already be new titles released by the time you read this. We shall continue to review the most interesting software in ELBUG. Prices include VAT.

Title	Type	Price	Code
3D Maze	GM	7.50	I1
Airline	GM	6.95	C1
Bandits at 3 O'clock	GM	7.95	P1
Bugblaster	GM	7.95	A1
Caterpillar	GM	7.50	I1
Chess	GM	9.20	A2
Chess	GM	7.95	P1
Corn Cropper	GM	6.95	C1
Creative Graphics	UT	9.20	A2
Cybertron Mission	GM	7.95	P1
Dallas	GM	6.95	C1
Draughts and Reversi	GM	9.20	A2
Draw	UT	9.95	P1
Escape From Moonbase Alpha	GM	7.95	P1
Felix in the Factory	GM	7.95	P1
Felix and The Fruit Monsters	GM	7.95	P1
Flags	ED	7.50	I1
FORTH	L	16.10	A2
French Tutor	H/ED	9.95	S1
Fruit Machine	GM	5.95	A1
Galactic Commander	GM	7.95	P1
Golf	GM	7.95	S1
Graphics Pack	UT	24.95	S1
Graphs	ED	9.95	S1
Graphs and Charts	UT	9.20	A2
Hyperdrive	GM	7.50	I1
Intergalactic Trader	GM	8.95	P1
Invaders	GM	7.50	I1
Killer Gorilla	GM	7.95	P1
Labyrinths of LaCoshe	GM	7.95	P1
LISP	L	16.10	A2
Meteors	GM	9.20	A2
Monsters	GM	9.20	A2
Moonraider	GM	7.95	P1
Personal Money Managment	H	9.20	A2
Pontoon and Patience	GM	7.50	I1
Positron	GM	6.95	P1
Primary Art	ED	7.95	A1
Primary Time	ED	7.95	A1
Scribe II	UT	9.95	A1
Snake	GM	4.00	B1
Starship Command	GM	9.20	A2
Stratobomber	GM	7.50	I1
Swoop	GM	7.95	P1
Tree of Knowledge	ED	9.20	A2
Vectors	ED	9.95	S1

SOFTWARE SUPPLIERS CHECKLIST

Code	Supplier
A1	Alligata, Superior Systems, 178 West Street, Sheffield S1 4ET.
A2	Acornsoft, 4a Market Hill, Cambridge CB2 3NJ.
B1	BEEBUGSOFT, P.O. Box 109, High Wycombe, Bucks HP11 2TD.
C1	Cases Computer Simulations, 14 Langton Way, London SE3 7TL.
I1	IJK Software, Unit 3c, Moorfields Moor Park Ave., Bispham, Blackpool, Lancs FY2 0JY.
P1	Program Power, 8/8A Regent Street, Chapel Allerton, Leeds LS7 4PE.
S1	Salamander Software, 117 Norfolk Road, Brighton, East Sussex BN1 3AA.

Program Classification

ED - Educational
GM - Game
UT - Utility
H - Home use
B - Business
L - Computer Language

THE SPIDER AND THE FLY

by Anthony Sykes

The amusing animated display that is provided by the program presented here is a good example of the screen displays that can be produced with user-defined characters on the Electron.

First of all, carefully type the program into your Electron before saving on cassette. When you run the program, it shows a spider dangling on the end of a web, waiting for the moment when the lid on the jar opens, enabling it to consume the large fly inside which buzzes about trying unsuccessfully to escape. Every so often the fly pauses for a while before continuing its futile activity. Inevitably, the spider always does catch the fly in the end.

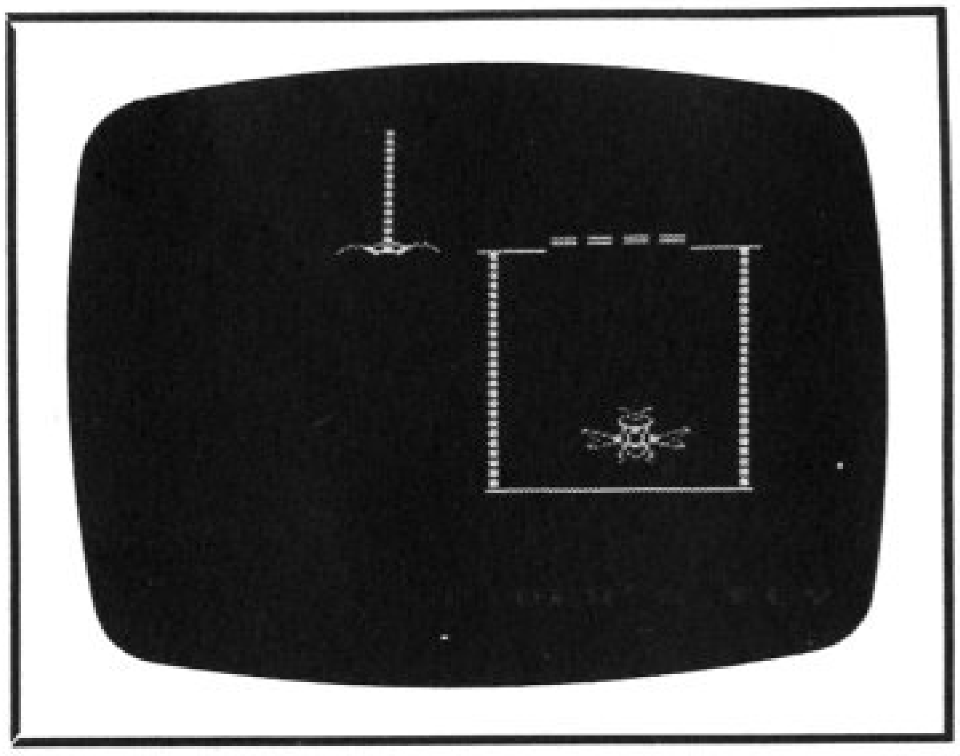
PROGRAM NOTES

The spider is displayed in two positions initially, once in the flashing black/white (colour 8), and just below that in flashing white/black (colour 15) in lines 5010 and 5020. This gives the impression of the spider moving up and down. The fly is moved around randomly inside the jar by first displaying the fly in green and then again in the background colour, so making it disappear (lines 5040 to 5090). Occasionally the fly has a rest, and then restarts after a short delay which may be terminated early by pressing any key. The number of rest periods allocated to the fly depends on the random value generated in line 4010.

The program is well structured using a number of procedures.

Line	Name	Function
1000	PROCspifly	Defines characters and sets up variables
2000	PROCfly	Displays the fly
3000	PROCspider	Displays the spider
4000	PROCinit	Sets up the screen
5000	PROCaction	Controls animation
6000	PROCending	The spider gets its meal!

PROCspifly sets up the user-defined characters for the spider and the fly. Note how the character strings fly$ and spider$ are built up in a manner similar to that described in our series on Electron graphics. PROCfly and PROCspider respectively display the two creatures in the required colours. PROCinit draws the jar and initialises the variables ready for use in PROCaction, which looks after the main animation. PROCending is called once the jar is opened, and the spider eats the fly!



```
   10 REM Program SPIDER
   20 REM Version E0.2
   30 REM Author A.SYKES
   40 REM ELBUG MARCH 1984
   50 REM Program subjct to copyright
   60 :
  100 ON ERROR GOTO 190
  110 MODE 2
  120 PROCspifly
  130 PROCinit
  140 REPEAT
  150 PROCaction
  160 UNTIL FALSE
  170 END
  180 :
  190 ON ERROR OFF:MODE 6:REPORT:PRINT
" at line ";ERL:END
  200 :
 1000 DEF PROCspifly
 1010 VDU 23,240,0,0,0,20,43,65,128,0
 1020 VDU 23,241,24,24,24,102,129,153,2
55,66
 1030 VDU 23,242,0,0,0,40,212,130,1,0
 1040 VDU 23,243,195,36,24,102,153,129,
195,126
```

»

```
 1050 VDU 23,244,192,176,76,35,17,33,78
,112
 1060 VDU 23,245,90,90,165,165,165,165,
126,90
 1070 VDU 23,246,3,13,50,196,136,132,11
4,14
 1080 VDU 23,247,66,165,165,153,66,129,
0,0
 1090 spider$=CHR$(240)+CHR$(241)+CHR$(
242)
 1100 fly$=CHR$(243)+CHR$(10)+CHR$(8)+C
HR$(8)+CHR$(244)+CHR$(245)+CHR$(246)+CH
R$(10)+CHR$(8)+CHR$(8)+CHR$(247)
 1110 ENDPROC
 1120 :
 2000 DEF PROCfly(col%,A%,B%)
 2010 COLOUR col%:PRINTTAB(A%,B%)fly$
 2020 ENDPROC
 2030 :
 3000 DEF PROCspider(col%,A%,B%)
 3010 COLOUR col%:PRINTTAB(A%,B%)spider$
 3020 ENDPROC
 3030 :
 4000 DEF PROCinit
 4010 T%=0:P%=RND(6):A$=CHR$95+CHR$95
 4020 B$="|"+STRING$(6,CHR$32)+"|"
 4030 COLOUR 14:PRINT TAB(4,24)"SPIDER
& FLY"
 4040 COLOUR 7:PRINT TAB(6,20)SPC(8)
 4050 VDU 23,1;0;0;0;0;
 4060 COLOUR 7:PRINT TAB(6,6)A$;"====";
A$
 4070 FOR T%=7 TO 18:PRINT TAB(6,T%)B$:
NEXT
 4080 MOVE400,415:DRAW872,415
 4090 COLOUR 7:FOR H%=0 TO 6:PRINT TAB(
3,H%)"|":NEXT
 4100 ENDPROC
 4110 :
 5000 DEF PROCaction
 5010 PROCspider(8,2,6)
 5020 PROCspider(15,2,7)
 5030 VDU5:MOVE195,828:GCOL0,7:PRINT"|"
:VDU4:VDU 23,1;0;0;0;0;
 5040 X%=RND(4)+7:L%=RND(4)+7
 5050 PROCfly(2,X%,L%)
 5060 TIME=0:REPEAT UNTIL TIME>5
 5070 PROCfly(0,X%,L%)
 5080 T%=T%+1:SOUND 1,-15,L%,3
 5090 IF T%=60 THEN SOUND 1,-14,5,3 ELS
E 5040
 5100 PROCfly(10,10,15)
 5110 A=INKEY(300):P%=P%-1:IF P%=1 THEN
 PROCending
 5120 PROCfly(0,10,15)
 5130 T%=0:GOTO 5010
 5140 ENDPROC
 5150 :
 6000 DEF PROCending
 6010 COLOUR 7:PRINT TAB(6,6)A$;SPC(4);
A$
 6020 A=INKEY(100)
 6030 PROCspider(0,2,6):PROCspider(0,2,
7):PROCspider(3,9,14)
 6040 COLOUR 0:FOR H%=0 TO 6:PRINT TAB(
3,H%)"|":NEXT
 6050 PROCfly(0,10,15)
 6060 FOR Q%=50 TO 1 STEP -1:SOUND 1,-1
5,Q%,1:NEXT
 6070 COLOUR 5:PRINT TAB(6,20)"YUM YUM!"
 6080 A=GET
 6090 PROCinit
 6100 ENDPROC
```

POINTS ARISING

It seems as though some of the gremlins escaped from 'Return of the Diamond', published in issue 2 of ELBUG, and instead got up to mischief in the listings of two other ELBUG programs in previous issues, causing some minor but annoying bugs. The correct details are listed below. We would like to thank those readers who wrote to us about these problems and we hope that we have now rounded up all the gremlins and returned them to where they will cause no more harm.

ASTAAD (issue No.2)
A semi-colon was lost from the end of line 350.

```
350 VDU23,1,0;0;0;0;
```

Hedgehog (issue No.1)
The procedure PROCGO was incorrectly called PROC_GO in line 1050.

```
1050 REPEAT:N%=N%+1:PROCGO:UNTIL N%=4
```

Most of you will probably have spotted this last error and corrected it already. We shall double check in future to try and avoid any further errors cropping up in our program listings.

ELECTRON GRAPHICS (Part 4)

by Mike Williams

Our introductory series on Electron graphics has so far concentrated on user-defined characters. This month we begin to look at the use of the DRAW and PLOT commands that will enable us to display a variety of lines and shapes on the screen.

The Electron has five graphics modes (modes Ø, 1, 2, 4 and 5). In these modes we can use the commands MOVE, DRAW and PLOT to create straight and curved lines and a variety of geometric shapes. In any graphics mode, we can think of the screen as a drawing area which is 128Ø points wide and 1Ø24 points high. Initially, the bottom left hand corner of the screen is the point (Ø,Ø), the top left hand corner is the point (Ø,1Ø23), the bottom right hand corner is (1279,Ø) and the top right hand corner is (1279,1Ø23). Notice that the numbering both vertically and horizontally starts at zero. This makes the bottom of the screen the 'x' axis and the left hand side of the screen the 'y' axis. The bottom left hand corner (Ø,Ø) is the origin.

Now, although we can create any drawing based on the measurements given above, the Electron is actually capable of displaying far fewer points on the screen, the number depending on your choice of mode. The relationship between the actual number of points and the measurements above is called resolution, a larger number of points producing higher resolution graphics, and a smaller number of points lower resolution graphics. In practice, a straight line drawn with high resolution graphics will normally appear as a thinner line on the screen than one drawn in low resolution graphics.

On the Electron, three different resolutions are available (low, medium and high), and this choice is also coupled with the number of colours that you can use and the amount of memory taken up, as shown in table 1.

Table 1

Graphics Mode	Number of Points	Number of Colours	Memory Used	Resolution
Ø	64ØX256	2	2ØK	High
1	32ØX256	4	2ØK	Medium
2	16ØX256	16	2ØK	Low
4	32ØX256	2	1ØK	Medium
5	16ØX256	4	1ØK	Low

Resolution in different graphics modes.

In a graphics mode, the GCOL command, described briefly last month, is used to select the drawing colour, while CLG is used to clear the graphics screen. As with text, the default is to draw in white on a black background. The MOVE command, which was also used last month, can be used to move to any point on the graphics screen while the DRAW command is used to draw a straight line from the last point specified (in a MOVE, DRAW or similar command) to the new point. When starting with any graphics mode, the starting point is always the origin (Ø,Ø).

DRAWING SQUARES AND TRIANGLES

Suppose we want to draw a square in the centre of the screen with a side of length 2ØØ. The following short program will do this.

```
100 MODE 4
110 MOVE 54Ø,412
120 DRAW 54Ø,612
130 DRAW 74Ø,612
140 DRAW 74Ø,412
150 DRAW 54Ø,412
160 END
```

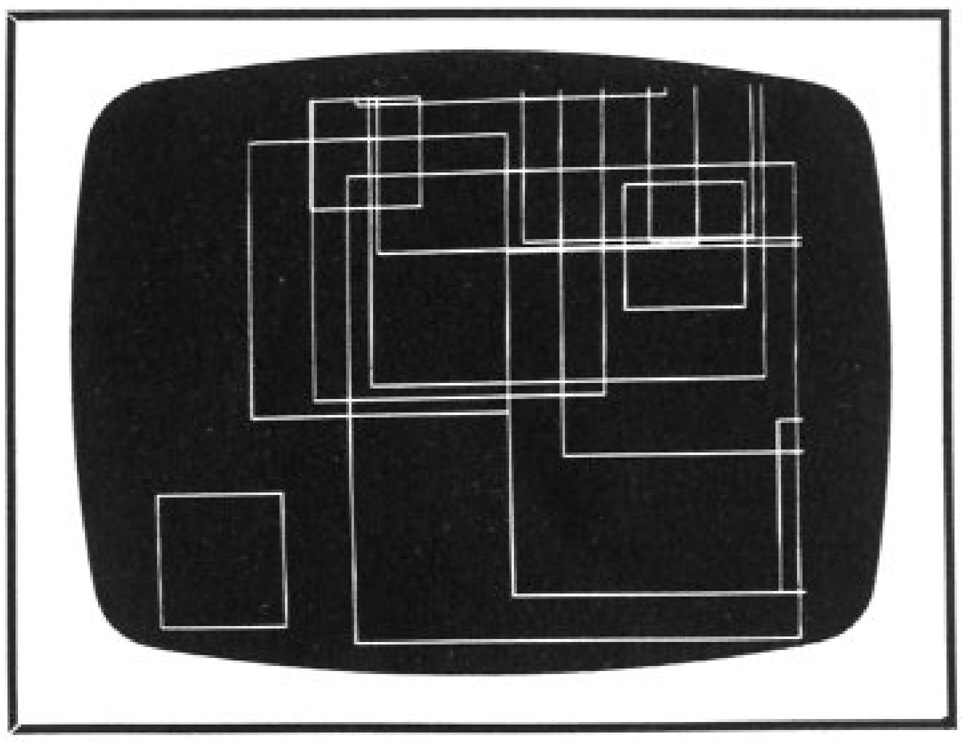
The centre of the screen is (64Ø,512) and thus the corners of the square are at points which are 1ØØ more or less than these values. Note how the program moves to one corner and then draws the four sides, finishing back at the starting point. The program is in Mode 4 (medium resolution). Whether you actually see an exact square or not on your screen is very likely to depend on your TV (or monitor) and how it is adjusted. You might like to try different modes to see the same result in high and low resolution graphics.

In creating graphics programs, it is often useful to put the instructions necessary for a particular shape as a procedure. This also enables us to introduce more flexibility. If you are not familiar with procedures then refer to page 77 in the User Guide. Here are the instructions as above, now modified as a procedure, which enables the square to be placed any size anywhere on the screen.

```
   1000 DEF PROCsquare(size,x,y)
   1010 MOVE x,y
   1020 DRAW x,y+size
   1030 DRAW x+size,y+size
   1040 DRAW x+size,y
   1050 DRAW x,y
   1060 ENDPROC
```

The procedure PROCsquare requires three parameters, the length of the side of the square, and the co-ordinates of the bottom left hand corner of the square. In order to draw a shape like this anywhere on the screen, we have to decide upon one corner as a reference point from which the other corners can be calculated. I have also written a short program to illustrate the use of the procedure. This consists of an infinite loop which randomly selects a size and position for squares which are continually drawn on the screen. Press Escape to terminate this program.

```
     10 REM Program SQUARE2
     20 REM Version E1.0
     30 REM Author Mike Williams
     40 REM ELBUG March 1984
     50 REM Program subject to Copyright
     60 :
    100 MODE4
    110 ON ERROR GOTO 190
    120 REPEAT
    130 SIZE=RND(1000)
    140 X=RND(1279):Y=RND(1023)
    150 PROCsquare(SIZE,X,Y)
    160 UNTIL FALSE
    170 END
    180 :
    190 ON ERROR OFF:MODE 6
    200 IF ERR<>17 THEN REPORT:PRINT" at
  line ";ERL
    210 END
    220 :
```



One interesting point that you should notice when you run this program is that squares which are so positioned that part of the square disappears off the screen cause no problems or errors at all. In effect, the screen area is like a window onto a much larger area on which you can still draw, but not see the results. Try typing the following short program:

```
100 MODE 4
110 MOVE40,200
120 DRAW640,1500
130 DRAW1200,200
140 DRAW40,200
150 END
```

This draws a triangle with the vertex above the top of the screen at (640,1500). This is a very nice feature of Electron graphics, as your programs need never check that you are actually drawing on the visible screen, though clearly there is not much fun to be had in deliberately drawing where you cannot see the results.

DRAWING CIRCLES

Although the DRAW command will only produce straight lines on the screen, we can still write programs that will produce acceptable circles. This is done by drawing a series of short straight lines. The length and number of lines needed will depend on how much of a purist you are, but it is initially surprising how few lines are needed to produce a realistic looking circle. The result will also depend on the size (radius) of the circle. Of course, the more lines that have to be drawn, the longer the drawing process will take.

To draw a circle does require some mathematics, though you don't really need to understand exactly what is going on. We can draw a circle by calculating the positions of a number of points equally spaced around the circumference, and then joining all the points up with straight lines. In practice, each short line is drawn as the next point is calculated. Each point on the circumference is calculated using the two formulae:

x = R * COS(A)   y = R * SIN(A)

where R is the radius of the circle, and A is the angle between the x axis and the radius to the point being calculated. The angle must be measured in radians (rather than degrees) and a complete revolution of a circle is 2*pi radians. Fortunately, BBC Basic helps us enormously here, as the two mathematical functions, cosine and sine, are provided (COS and SIN), and even the value of pi is included (as the pseudo-variable PI).

The two formulae given above assume that the origin (0,0) is at the centre of the circle, but we have already seen that this is normally in the bottom left hand corner of the screen which is not very helpful. However we can move the origin to a new position using the VDU29 command (see the User Guide page 113). The following program will draw a circle with a radius of 200 at the centre of the screen.

```
100 MODE 4
110 VDU29,640;512;
120 radius=200
130 MOVE radius,0
140 FOR angle=0 TO 2*PI STEP PI/16
150 x=radius*COS(angle)
160 y=radius*SIN(angle)
170 DRAW x,y
180 NEXT angle
190 END
```

The program first moves the origin to the centre of the screen (640,512), which now becomes the new origin and thenceforth all co-ordinates are measured from this new position. The starting point for drawing the circle is on the circumference at (200,0), with reference to the new origin. A loop is then used to repeatedly calculate a new point on the cicumference (rotating anti-clockwise) and draw a straight line from the previous point to this one, until we arrive back at the starting point. The last number on line 140 is half the number of lines used to make up the circle.

You might like to try running the program with both larger and smaller values to see the effect that this has on the shape of the 'circle'. Of course, what we are really doing is drawing a polygon with a large number of sides that then appears to us as a circle. This means in turn that by changing this number you can use the same routine for drawing any polygon that you want. For example, if you put 2, you should get a square, while 4 should give an octagon (half the number of sides in each case).

You may find that some values, because of the limitations of computer arithmetic, do not produce a closed polygon. If this happens, then replacing '2*PI' with '2*PI+0.1' in line 140 will usually overcome this problem.

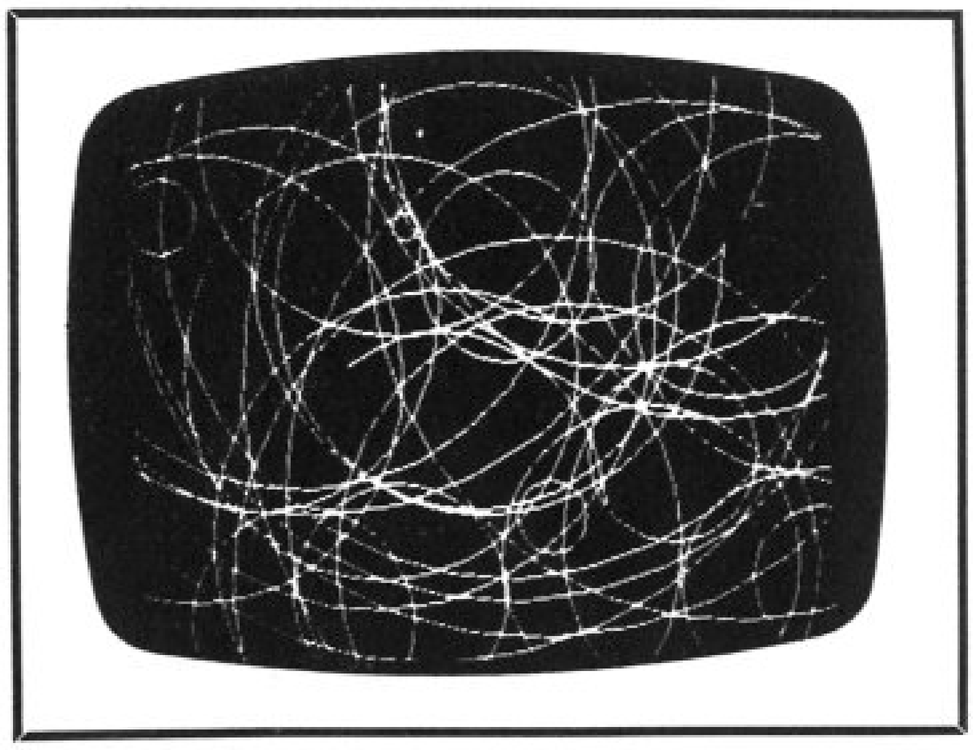
If we want to produce a program that will draw several circles, then it is sensible to write the instructions as a procedure, just as we did for the square. We can define the procedure with three parameters, allowing a circle to be drawn with any radius, and in any position on the screen. We could just package up the original circle procedure, putting a VDU29 command as the first instruction in the procedure to define a new origin at the centre of the circle to be drawn. However, because of the mathematical sines and cosines used in the formulae, circle drawing is a relatively slow process.

If we want to draw several circles, then we can speed up the process considerably by calculating all the necessary sines and cosines just once, and then saving them for use later when drawing a circle. This is done in the following program. The procedure PROCtable calculates a table of sines and cosines, which are then used by the procedure PROCcircle for actually drawing a circle.

● ● ●● ●●● ●

```
   10 REM Program CIRCLE2
   20 REM Version E1.0
   30 REM Author Mike Williams
   40 REM ELBUG March 1984
   50 REM Program subject to Copyright
   60 :
  100 MODE 4
  110 ON ERROR GOTO 210
  120 DIM C(32),S(32)
  130 PROCtable
  140 REPEAT
  150 radius=RND(800)
  160 x=RND(1279):y=RND(1023)
  170 PROCcircle(radius,x,y)
  180 UNTIL FALSE
  190 END
  200 :
  210 ON ERROR OFF:MODE6
  220 REPORT:PRINT" at line ";ERL:END
  230 :
 1000 DEF PROCtable
 1010 FOR I=1 TO 32
 1020 angle=2*PI*I/32
 1030 C(I)=COS(angle):S(I)=SIN(angle)
 1040 NEXT I
 1050 ENDPROC
 1060 :
 1100 DEF PROCcircle(radius,x,y)
 1110 LOCAL I,X,Y
 1120 VDU29,x;y;
 1130 MOVE radius,0
 1140 FOR I=1 TO 32
 1150 X=radius*C(I):Y=radius*S(I)
 1160 DRAW X,Y
 1170 NEXT I
 1180 ENDPROC
```

The circles are drawn as polygons of 32 sides and hence 32 sines and cosines are calculated and saved as the elements of the two arrays C and S. The procedure for drawing a circle first sets a new origin as the centre of the circle and then draws the circle as before except that the sines and cosines are taken from the two arrays instead of being calculated when needed. If you run the program, which draws random sized circles randomly on the screen, you will see how much quicker this is compared with the original circle drawing program. The pause that occurs at the start is the time to calculate the sines and cosines initially.

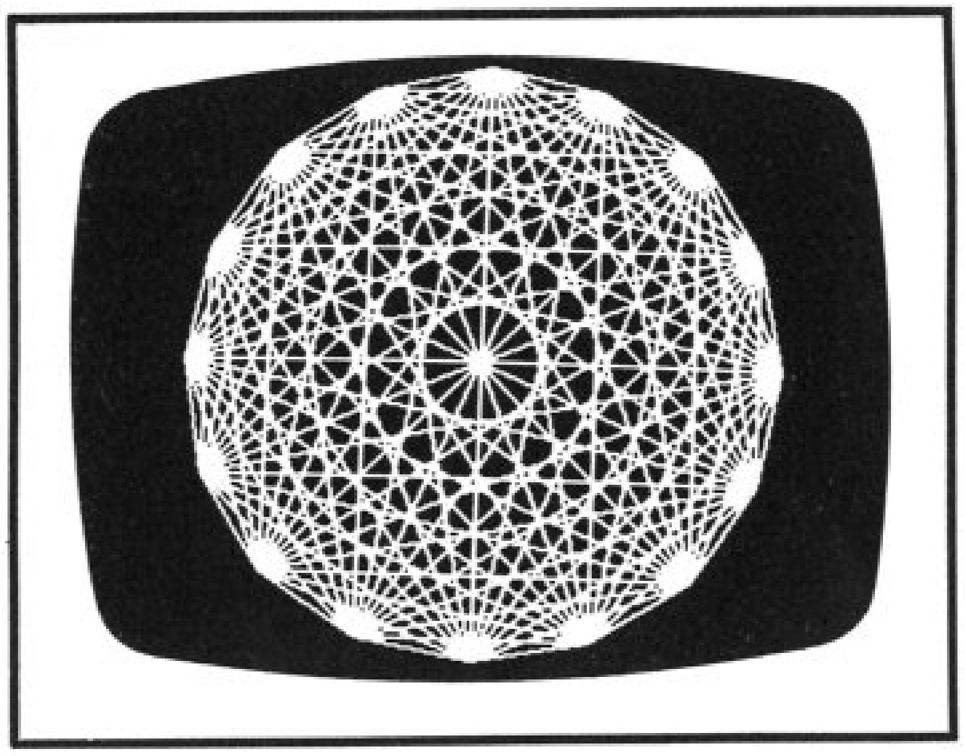


DRAWING POLYGONS

The final program to be presented this month shows how the circle drawing idea can be adapted to draw a polygon, with any number of sides, together with all the diagonals joining every vertex to every other vertex. The vertices of any regular polygon are evenly spaced out around the circumference of a circle which thus forms the basis for the method of drawing. Because every vertex will be referred to several times, the x and y co-ordinates of each vertex are calculated first and stored in the two arrays X and Y. This is just like calculating the points around the circumference of a circle without the line drawing.

Once all the points are known, two nested loops are then used to draw all the lines. The outer loop, controlled by the variable I, is repeated once for each vertex, while the inner loop, using the variable J, is repeated a reducing number of times. If you experiment with an octagon, for example, you will see the principle involved. From the first vertex 7 lines are needed, 6 from the second, 5 from the third and so on. This is exactly how the routine works.

The whole polygon is drawn by the procedure PROCpolygon with four parameters, the radius of the circle (or size), the number of sides, and the centre of the circle (or polygon). The origin is set to the centre using the VDU29 command and the co-ordinates of all the points needed are calculated and saved.



The procedure is called just once and displays a 16 sided polygon in the centre of the screen complete with all the diagonals. You could try varying the parameters to produce different displays and the program could easily be extended to draw several polygons together on the screen, in different positions and either the same or different sizes. There is plenty of scope for experimenting with all these examples.

Next month we will continue the series on graphics by looking at the PLOT command and the drawing of filled shapes on the screen.

```
  10 REM Program POLY1
  20 REM Version E1.Ø
  30 REM Author Mike Williams
  40 REM ELBUG March 1984
  50 REM Program subject to Copyright
  60 :
 100 MODE 4
 110 ON ERROR GOTO 160
 120 DIM X(64),Y(64)
 130 PROCpolygon(500,16,640,512)
 140 END
 150 :
 160 ON ERROR OFF:MODE6
 170 REPORT:PRINT" at line ";ERL:END
 180 :
1000 DEF PROCpolygon(radius,sides,x,y)
1010 VDU29,x;y;
1020 FOR I=1 TO sides
1030 angle=2*PI*I/sides
1040 X(I)=radius*COS(angle)
1050 Y(I)=radius*SIN(angle)
1060 NEXT I
1070 FOR I=1 TO sides
1080 FOR J=I+1 TO sides
1090 MOVE X(I),Y(I)
1100 DRAW X(J),Y(J)
1110 NEXTJ,I
1120 ENDPROC
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TAB

When calculating TAB positions remember that the co-ordinate of the top left hand corner of the screen are Ø,Ø not 1,1.

*SAVE EXTENSION - M.Sykes

When using *SAVE to save a file, the usual method is to type:
*SAVE <filename> <start address> +<length> <execution address>.
This can be expanded to include the load address of the file when loaded back into the computer.

The load address is normally taken to be the start address of the file. The load address is appended to the end of the normal set of numbers. i.e. *SAVE <filename> <start address> <+length of file> <execution address> <load address>. If the load address is typed in as FFFF then the computer cannot load the file back into memory unless the *LOAD <filename> <load address> command is used, offering an easy method of software protection, but do make a list of the necessary load addresses.

COLOUR VIDEO OUTPUT

The Electron is capable of producing colour pictures on the composite video output socket by joining link LK4. This link is situated behind Q7 and next to IC13 (which is labelled LM324N). The reason why Acorn did not connect this link is because the picture quality is degraded relative to the black and white picture obtained from the same socket. However, there is only a slight increase in the quality of colour from this socket and the output from the UHF output. The RGB output produces by far the best picture.

HOW TO TELL AN ELECTRON FROM A BBC MICRO

by David Fell

Inside your Electron there is a ROM (Read Only Memory) containing the Operating System (OS for short) which controls much of what goes on in your machine. As time goes by, it is possible that new versions of the operating system will be produced, either to incorporate some new features, or to correct 'bugs' or faults, which have been discovered in existing ones. To find out which version of the operating system is in your machine, Acorn have provided a special call. Try the following:

```
  *FX0 <return>
```

You should find that your Electron has replied by displaying 'OS 1.00'. The '1.00' is the version number, and it is by looking at this that you can tell which version you have. If there are any later versions, then they may well increase this number (for example 1.10 or 1.20). The BBC Micro is similar to the Electron in many ways, and has already been through a number of different versions of its operating system to incorporate features not present in the original. In addition, an American version of the BBC micro has now been produced. All of these are different versions of essentially the same operating system.

As well as the operating system, Basic is also contained within the same ROM in your Electron. To allow for the possibility that this may also need to be changed, Acorn have provided a way to tell which version of Basic is present (there is only one version at the moment, but that does not mean Acorn will not decide to produce another version). To see the way in which the Basic is identified, press the Break key on your Electron. Now type in:

```
  REPORT <return>
```

You should see a line of text displayed saying '(C)1982 Acorn'.

So far, we have seen how we can tell which versions of the operating system and Basic are in our machine. The messages displayed tell us which version, but not which machine. For example we could have OS 1.00, but is it an Electron, a BBC Micro, or an American BBC Micro operating system? Because of the inherent similarities between the machines mentioned above, it is tempting, particularly for producers of commercial software, to write one piece of software and to modify it slightly for the different machines, than to write three pieces of software to do the same thing on the three different micros mentioned. To allow a program to distinguish between the different machines, the INKEY command has been extended. If you look in your User Guide, you will notice that there is no key whose negative code is given as 256. The function INKEY-256 will return a value depending on the machine and operating system with which it is used. The current range of possible values is shown in the table.

Operating System	Value Returned
Electron 1.00	1
BBC 0.10	0
BBC 1.00/1.20	-1
BBC USA	254

Table showing values returned by INKEY-256 on different machines.

If you type:

```
          PRINT INKEY-256
```

on your Electron then the value '1' should be displayed and this value could be checked within the program itself by assigning it to a variable rather than just displaying it on the screen. For example, try running the following program:

```
10 IF INKEY-256=1 THEN PRINT "This is a
n Electron." ELSE PRINT "This is not an
 Electron, it must be a BBC Micro."
```

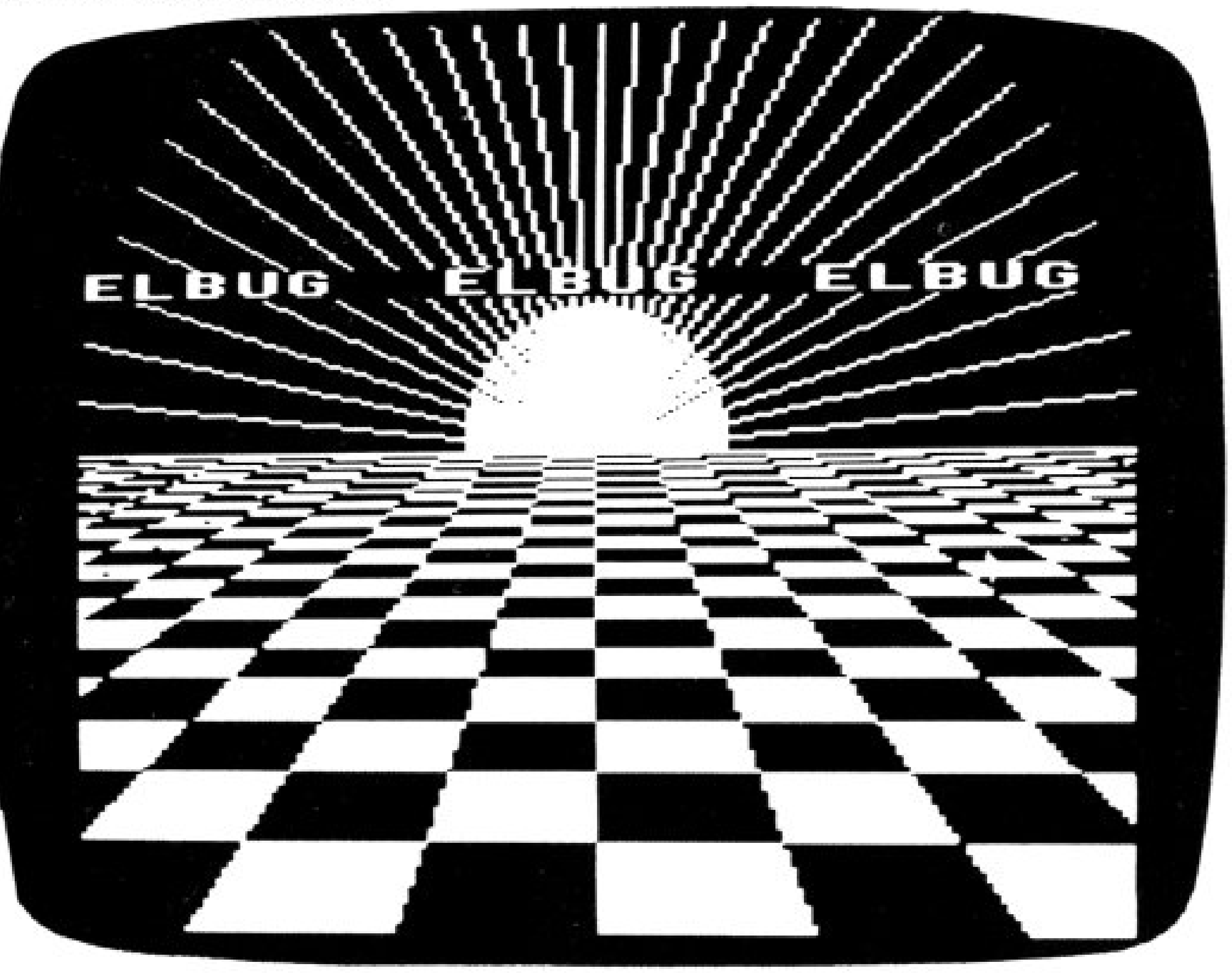
●

MOVING CHEQUER BOARD

by D. D. Harriman

This month we bring you a short and fascinating graphics program by D.D.Harriman. This is a striking example of the animated graphics that can be achieved simply by switching colours using a VDU command.

This program presents the black and white squares of a chequer board in such a way that the squares appear both to get larger and move towards you as you watch. Just type the program into your Electron and then run it to see the visual effects that result.



The program is essentially based on the use of the VDU19 command (see User Guide page 107). This is of the form:

VDU19,L,A,Ø,Ø,Ø

where L is the logical colour number (the range depends on the mode selected) and A is the actual colour (from Ø to 15). The program uses mode 2 which allows 16 logical colours. The chequer board is drawn initially using 14 of these colours which are then repeatedly changed between black and white to give the effects that you observe. The other two colours out of the 16 are used to provide white text and graphics against a black background in the top part of the screen.

Many effective animations can be produced by drawing different parts of an object in different logical colours (initially all set to black), and then highlighting each part in turn using the VDU 19 command. This program shows just how effective the results can be.

PROGRAM DESCRIPTION

After some initial assignments the program starts by drawing bands of colour, decreasing in width, from the bottom of the screen up to approximately the halfway point (lines 15Ø to 19Ø). The next routine (lines 21Ø to 27Ø) uses exclusive OR plotting to draw perspective bands (each made of two triangles) across the horizontal lines. Each such band is four times as wide at the front as it is at the rear. Having done this the moving chequer board effect is produced by repeatedly executing the procedure PROCspeed which uses VDU19 to switch the colour assignments. We have embellished the area above the chequer board with some additional text and graphics (lines 290 to 380). You could easily define this area as a text and/or graphics window where other images could be produced.

SOME USEFUL VARIABLES

J%	- determines the width and hence number of squares across the screen.
X%	- sets the x co-ordinate for the centre of the circle.
Y%	- controls the height of the chequer board on the screen.
Y%-Y	- marks the actual height of the chequer board on the screen.
A$	- the text displayed on the screen at the end.

If you change the FOR statements at lines 410 and 440 to count in the reverse direction (up instead of down), you will find the chequer board moves away from you instead of towards you.

```
 10 REM Program CHEQUER
 20 REM Version E0.2
 30 REM Author  Delos D.Harriman
 40 REM ELBUG   March 1984
 50 REM Program subject to Copyright
 60 :
100 MODE2:ON ERROR GOTO 530
110 J%=64:A$="ELBUG  "
120 VDU23;11,0;0;0;0
130 COLOUR7:COLOUR143:VDU19,15,0;0;12
140 X%=640:Y%=640:R=640:Y=Y%:C%=-1
150 FOR F=1 TO 6 STEP 0.03
160 C%=C%+1:C%=C%-(C%=7):IFC%=15:C%=0
170 GCOL0,128+C%:L=Y:Y=Y%/F
180 VDU24,0;Y%-L;1279;Y%-Y;:CLG
190 NEXT:VDU26:GCOL3,8
200 :
210 FOR F%=-640 TO 639 STEP J%
220 MOVE640+F%,Y%-Y:X1%=640+F%*4
230 MOVE X1%,0:F%=F%+J%:X2%=640+F%
240 Y2%=Y%-Y:PLOT85,X2%,Y2%
250 PLOT85,640+F%*4,0
260 MOVE X1%,0:DRAW X2%,Y2%
270 NEXT:GCOL0,7
280 :
290 FOR C=1 TO 4 STEP 3
300 FOR I=0 TO PI STEP PI/C/32
310 X1=X%+R*COS(I)/C:Y1=Y%+R*SIN(I)/C
320 MOVEX%,Y%-Y:DRAW X1,Y1-Y
330 NEXTI,C:PRINT TAB(0,9);
340 :
350 FORI%=1 TO 3:FOR F%=1 TO LEN(A$)
360 PRINT MID$(A$,F%,1);:PROCspeed
370 NEXTF%,I%
380 REPEAT:PROCspeed:UNTIL FALSE
390 :
400 DEF PROCspeed
410 FOR A%=6 TO 0 STEP -1
420 VDU19,A%,7;0;:VDU19,A%+8,0;0;
430 PROCW:NEXT
440 FOR A%=14 TO 8 STEP-1
450 VDU19,A%,7;0;:VDU19,A%-8,0;0;
460 PROCW:NEXT
470 ENDPROC
480 :
490 DEFPROCW
500 T%=TIME:REPEAT UNTIL TIME-T%>1
510 ENDPROC
520 :
530 ON ERROR OFF:MODE 6
540 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
550 END
```

●

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

EXTRA FUNCTION KEYS

The Break key can be programmed as function key 10. The most useful definition is
*KEY10 OLD|M
Then when this key is pressed, it automatically retrieves your program after the break; though if you press the key with no program in the machine you will get a "Bad Program" message.

In a similar way, the cursor and copy keys may also be programmed as function keys 11 to 15, by using *FX4,2. This disables their normal function, replacing them with the following function key values:
COPY 11, LEFT 12, RIGHT 13, DOWN 14, UP 15 ●

PLOT EQUIVALENCE

All PLOT commands can be replaced with a VDU command of the form VDU 25,K,X;Y; where K,X,Y have the values outlined in the User Guide on page 176. ●

SOUND ENVELOPE EDITOR

by Ian Soutar and Mike Williams

This month we present a program which not only makes it very easy to design and experiment with sound envelopes but shows you graphically on the screen exactly what the envelope looks like each time.

The Electron has two sound channels, one capable of producing tones (channel 1) and the other generating a variety of noises (channel Ø). Sounds do not just happen by themselves, but must be programmed like any other feature of the Electron, using in this case the SOUND and ENVELOPE commands. Now although the SOUND command is not too difficult, a quick look in the User Guide is often enough to put many people off from trying to understand and use the ENVELOPE command. The program that is included with this article is designed to help you understand sound envelopes and to allow you to experiment with sound by means of an easy to use sound and envelope editor. The User Guide also contains a whole chapter detailing the SOUND and ENVELOPE commands beginning on page 116.

SOUND COMMAND

The SOUND command on the Electron has the following format:

SOUND c , v , p , d

where c is the sound channel (Ø or 1), v is the volume or loudness, which on the Electron is either on (-15) or off (Ø), p is the pitch of the note played and d is the duration of the note. Both pitch and duration can be changed using the Sound Editor. Using the sound command alone will produce just single notes. Much more interesting effects can be produced by using a sound envelope.

On the Electron the amplitude of a note cannot be varied, and is simply switched on and off. It is the pitch of the note that is varied using the ENVELOPE command, and the resulting graph or shape of the note as time passes is called a sound envelope. Once an envelope has been defined in a program it can be used with any SOUND command by replacing the 'v' parameter by the envelope number (in the range 1 to 16).

SOUND AND ENVELOPE EDITOR

Although the ENVELOPE command used on the Electron has a total of 14 parameters, only eight of these are meaningful, the other six being set as recommended in the User Guide to maintain compatibility with the BBC micro. The Sound Editor allows you to change any of these eight parameters, immediately displaying the new sound envelope graphically on the screen. In experimenting with different envelopes, the program provides six options. These are:

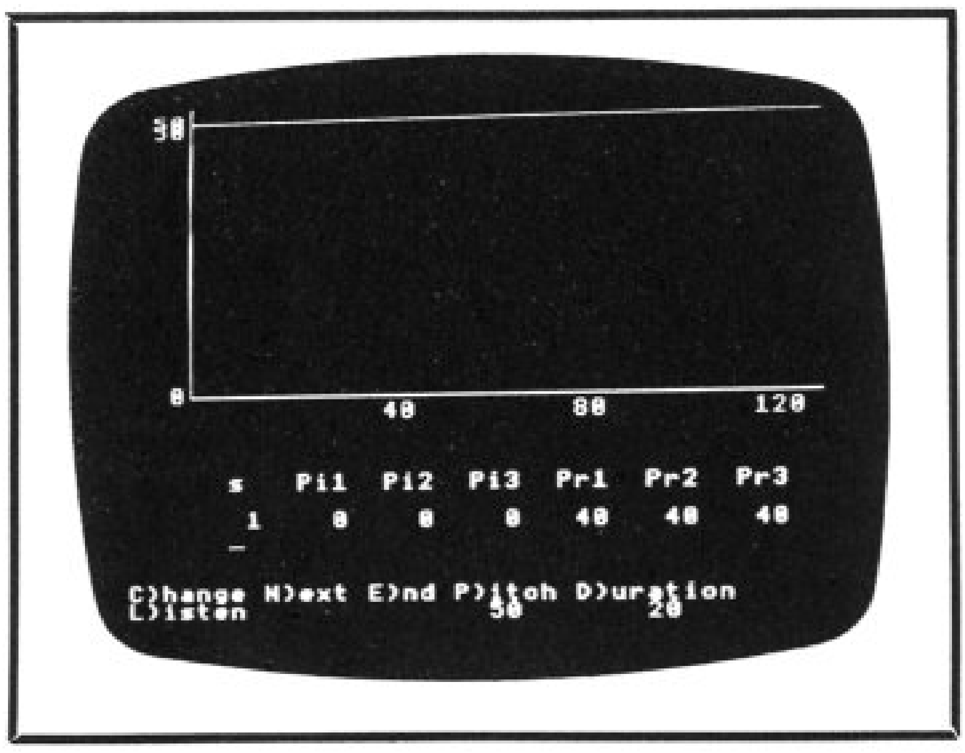
C	(Change)	Allows parameters to be changed.
N	(Next)	Moves on to the next parameter.
E	(End)	Ends the program.
P	(Pitch)	Allows the pitch of the note to be changed.
D	(Duration)	Allows the duration of the note to be changed.
L	(Listen)	Plays the current sound.

Each time any parameter is altered, a new graph of the sound envelope is immediately displayed. There is full checking on all the sound and envelope parameters displayed, and the sound may be heard at any time by pressing 'L'. Once you are satisfied with a sound that you have produced, you can copy down the sound and envelope parameters from the screen and use them in any program of your own.

CREATING ENVELOPES

The eight parameters of the envelope command are concerned with the definition of the pitch envelope (see page 121 in the User Guide). The visual display of the pitch envelope is an indispensable aid to understanding and producing really good sound effects for use in any program. When you first run the sound editor, the sound and envelope parameters are already set up as follows:

```
 s  Pi1  Pi2  Pi3  Pr1  Pr2  Pr3
 1   Ø    Ø    Ø   4Ø   4Ø   4Ø
   pitch = 5Ø   duration = 2Ø
```



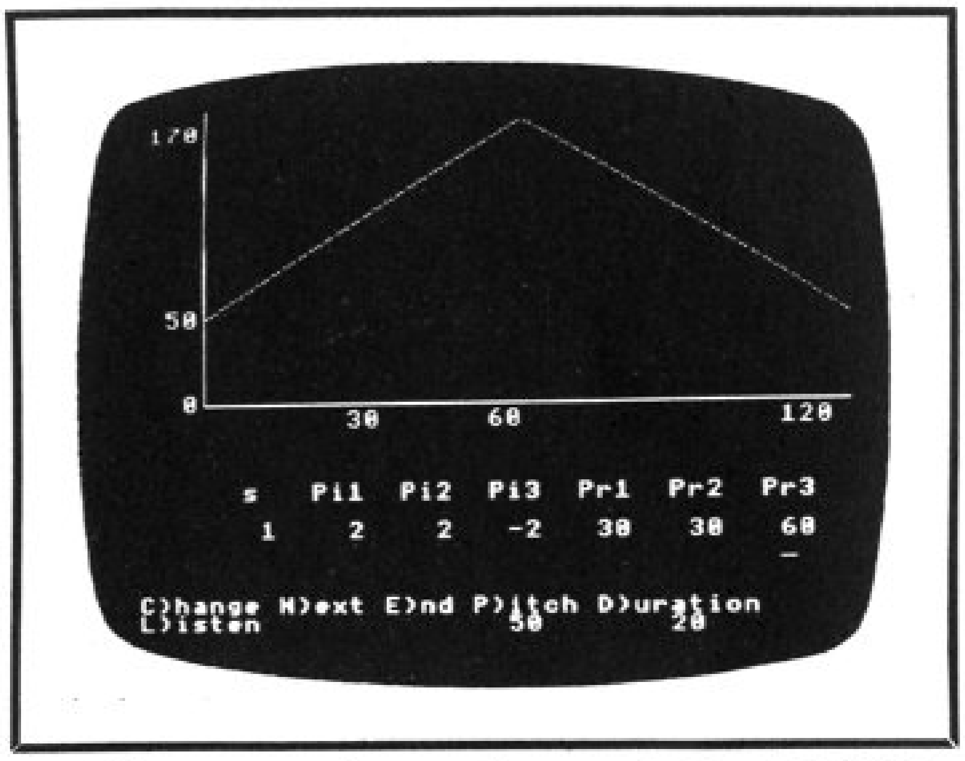
```
s   Pi1  Pi2  Pi3  Pr1  Pr2  Pr3
 1   0    0    0    40   40   40
_
C)hange N)ext E)nd P)itch D)uration
L)isten               50       20
```

You will see that the pitch envelope is a straight line, and if you press 'L' you will hear a single unchanging note that lasts for 1 second. The duration of a sound is given in twentieths of a second and hence a duration of 2Ø is equivalent to 1 second. The pitch, which is part of the SOUND command, can be altered by pressing 'P'. The range allowed is Ø to 255 while values outside this range are converted within the program by adding or subtracting 256 until the value is within range (the SOUND command does this automatically). Try entering different values of pitch and duration to see the effect.

VARYING THE PITCH

At the moment, the note produced by the sound editor is of constant pitch. To make it vary we need to alter one or more of the pitch increment parameters Pi1, Pi2, Pi3. Each controls the rate of change of pitch in one of three segments of the pitch envelope. Try changing Pi1 from Ø (no change) to 3. The graph of the pitch envelope shows an angled line taking the pitch up from 5Ø to 17Ø, where it straightens out. If you press 'L' you will hear it do this. In fact it increases at the rate of 3 units of pitch for each 4Ø units of duration (since Pr1 defines the duration of the first segment) - it thus increases by a total of 12Ø units taking it from 5Ø to 17Ø.

The other two segments can be altered accordingly. If you change Pi2 to 1, you will see that in the second segment the note now continues to increase in pitch, but at only one third of the rate of the first segment. If you change Pi3 to -4 the pitch will drop by 4 x 4Ø units returning it to the starting value of 5Ø. If you want to hear a number of full cycles of this sound, just increase the duration. Taking it from 2Ø to 6Ø should give 3 full cycles.



```
s   Pi1  Pi2  Pi3  Pr1  Pr2  Pr3
 1   2    2    -2   30   30   60
                              _
C)hange N)ext E)nd P)itch D)uration
L)isten               50       20
```

If you require only a two segment pitch envelope, you can amalgamate two adjacent segments. This is done by giving them the same Pi factor. Thus if you change Pi1 and Pi2 both to the value 2, you will get an uneven two segment inverted "V" shape envelope. If you want to make it symmetrical, just make Pr1 + Pr2 = Pr3, and change Pi3 to -2. A little experiment will show why. The values Pi1 = 2, Pi2 = 2, Pi3 = -2, Pr1 = 3Ø, Pr2 = 3Ø, Pr3 = 6Ø thus produce a symmetrical inverted "V" of duration 1.2 seconds.

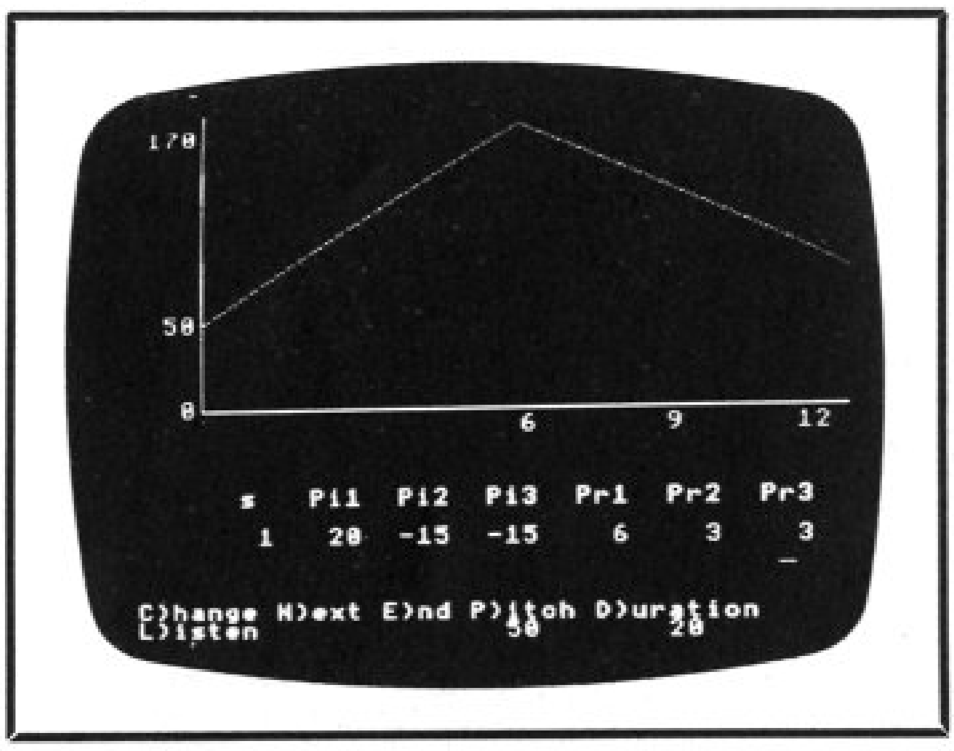
You can change the starting pitch by altering pitch on the screen - but note that if the envelope takes the pitch outside the range Ø to 255, then there will be a wrap-around effect (so that going below zero causes a jump to 255 etc). The audio effect can be quite interesting, though the screen plot does not take the wrap-around effect into account. You will also find that in some cases the values displayed beside the axes of the graph overlap and become unreadable.

RAPID PITCH CYCLES

A different kind of sound effect can be achieved by executing a whole series of much shorter pitch envelopes. To achieve this, return duration to 2Ø, and enter the following values into the pitch envelope:

s	Pi1	Pi2	Pi3	Pr1	Pr2	Pr3
1	2Ø	-15	-15	6	3	3

pitch = 5Ø   duration = 2Ø



```
      s   Pi1   Pi2   Pi3   Pr1   Pr2   Pr3
      1    20   -15   -15    6     3     3

C)hange N)ext E)nd P)itch D)uration
L)isten            50        20
```

This produces quite an interesting sound. Note that the duration of each pitch envelope is Ø.12 seconds (the 12 on the right hand side of the graph), so there are about 1Ø full cycles in the duration specified. The pitch envelope is still, as you see, an inverted "V", though not a completely symmetrical one. Changing the pitch to 22Ø to give a wrap-around makes quite a different sound. Making the envelope more oblique further changes the effect. Try the following:

s	Pi1	Pi2	Pi3	Pr1	Pr2	Pr3
1	-12	-12	-12	6	4	2

pitch = 22Ø   duration = 2Ø

There are many different effects that can be produced in this way - either by changing the overall shape of the envelope or by using the wrap-around effect, and this can be left for experiment, though here are three more, the first of which produces a tremolo or warble. Each uses a pitch setting of 22Ø (though the duration depends on how long you wish the sound effect to continue).

s	Pi1	Pi2	Pi3	Pr1	Pr2	Pr3
1	1	-1	-1	6	3	3
1	127	-1	-1	6	3	3
1	-12	-1	-1	6	3	3

```
  1Ø REM PROGRAM ENVELOPE EDITOR
  2Ø REM AUTHOR  I.SOUTAR
  3Ø REM VERSION E1.Ø
  4Ø REM ELBUG   MARCH 1984
  5Ø REM PROGRAM SUBJECT TO COPYRIGHT
  6Ø :
  7Ø DIM E%(7),P$(7),l%(7,1)
  8Ø MODE1:PROCstart
  9Ø VDU19,1,6,Ø,Ø,Ø
 1ØØ ON ERROR GOTO 75Ø
 11Ø CLS:PROCgraph:PROCedit(1,7)
 12Ø END
 13Ø :
 14Ø DEFPROCplay
 15Ø ENVELOPE 1,E%(1),E%(2),E%(3),E%(4)
,E%(5),E%(6),E%(7),126,Ø,Ø,-126,126,126
 16Ø SOUND1,1,p%,d%
 17Ø ENDPROC
 18Ø :
 19Ø DEFPROCgraph:h=1152/(E%(5)+E%(6)+
E%(7))
 2ØØ VDU24,Ø;42Ø;1279;1Ø23;:CLG
 21Ø v=p%:V=p%:FOR I%=2TO4
 22Ø V=V+E%(I%)*E%(I%+3):IF V>v v=V
 23Ø NEXT:V=v:IF v=Ø v=1
 24Ø v=48Ø/v
 25Ø GCOLØ,3:MOVE1279,512
 26Ø DRAW127,512:DRAW127,1Ø23
 27Ø MOVE127,(512+p%*v)
 28Ø P%=p%:Q%=Ø:GCOLØ,2
 29Ø FORI%=2TO 4:P%=P%+E%(I%)*E%(I%+3)
:Q%=Q%+E%(I%+3):DRAW(127+Q%*h),(512+P%*
v):NEXT
 3ØØ VDU5:MOVE-1Ø,528:PRINTSPC(3);"Ø":
MOVE-1Ø,528+p%*v:PRINTFNstr(p%):MOVE-1Ø
,512+V*v:PRINTFNstr(V)
 31Ø Q%=Ø:FORI%=5TO7:P%=Ø:IFI%=5 THEN
P%=32 ELSE IF I%=7 P%=-32
 32Ø Q%=Q%+E%(I%):MOVEQ%*h+P%,5ØØ:PRIN
TFNstr(Q%*E%(1)):NEXT
 33Ø VDU4:VDU26
 34Ø ENDPROC
 35Ø :
 36Ø DEFPROCedit(S%,F%):J%=Ø:FORI%=S%T
O F%:COLOUR3:J%=J%+1:PRINTTAB(J%*5,2Ø);
P$(I%);:COLOUR1:PRINTTAB(J%*5-1,22);FNs
tr(E%(I%)):NEXT
 37Ø COLOUR3:PRINTTAB(Ø,26)"C)hange N)
ext E)nd P)itch D)uration":J%=1
 38Ø PRINT"L)isten";:COLOUR1:PRINTTAB(
19)FNstr(p%);TAB(28)FNstr(d%)
 39Ø :
 4ØØ PRINTTAB(Ø,29);STRING$(79,CHR$32)
 41Ø PRINTTAB(J%*5,23)CHR$32;:X$=GET$:
IFX$="N"THENJ%=J%+1:GOTO51Ø
 42Ø IFX$="E" VDU22,6:END
 43Ø IFX$="L"PROCplay:GOTO4ØØ
```

```
  440 IFX$="D" PRINTTAB(0,29)"Duration
(0 to 255)";:INPUTIP$:d%=FNip(d%) MOD 2
56:PROCgraph:PRINTTAB(28,27)FNstr(d%):G
OTO400
  450 IFX$="P" PRINTTAB(0,29)"Pitch (0
to 255)";:INPUTIP$:p%=FNip(p%) MOD 256:
PROCgraph:PRINTTAB(19,27);FNstr(p%):GOT
O400
  460 IFX$<>"C"VDU7:GOTO400
  470 X%=S%+J%-1:PRINTTAB(0,29)"Range o
f ";P$(X%);" is ";l%(X%,0);" to ";l%(X%
,1):PRINT"Change to ";STRING$(20,CHR$32
):INPUTTAB(11,30),IP$:IF IP$<>"" E%(X%)
=VAL(IP$)
  480 IFE%(X%)<l%(X%,0) OR E%(X%)>l%(X%
,1)VDU7:GOTO470
  490 IFS%=1 PROCgraph
  500 PRINTTAB(J%*5-1,22)SPC(4);TAB(J%*
5-1,22);FNstr(E%(X%)):VDU28,0,31,39,29:
CLS:VDU26
  510 IFJ%=F%-S%+2 J%=1
  520 GOTO390
  530 IFS%=1 PROCgraph
  540 GOTO390
  550 ENDPROC
  560 :
  570 DEFPROCq(x,q$,y):IFq$="A"H=0:Q=0
  580 IFx<>0H=H+x*v:Q=Q+x/100:MOVE111+H
,490:PRINTq$:MOVEy,450:PRINTq$"=";INT(Q
*100)/100
  590 ENDPROC
  600 :
  610 DEFFNip(vl)
  620 IF IP$="" VL=vl ELSE VL=VAL(IP$)
  630 =VL
  640 :
  650 DEFFNstr(n)=STRING$(4-(LEN(STR$(n
))),CHR$32)+STR$(n)
  660 :
  670 DEFPROCstart
  680 p%=50:d%=20
  690 FORI%=1TO7:READP$(I%):READl%(I%,0
):READl%(I%,1):NEXT
  700 DATA " s",0,255,Pi1,-128,127,Pi2,
-128,127,Pi3,-128,127,Pr1,0,255,Pr2,0,2
55,Pr3,0,255
  710 FORI%=1TO7:READE%(I%):NEXT
  720 DATA 1,0,0,0,40,40,40
  730 ENDPROC
  740 :
  750 ON ERROR OFF
  760 IF ERR=17 GOTO 100 ELSE MODE6
  770 REPORT:PRINT " at line ";ERL
  780 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CURSOR CONTROLS

The cursor movements are all defined in the ASCII (American Standard Code for Information Interchange) code, and can be incorporated into strings for formatting a print statement. For example, if you want to print 'Electron' vertically on the screen, instead of TABbing each character individually into its position, you can use the following routine:

```
10 PRINT TAB(10,10);
20 VDU 69,10,8,108,10,8,101,10,8,99,10,8,116,10,8,114,10,8,111,10,8,110
30 END
```

Line 10 sets the cursor to the initial starting position, line 20 prints out the characters of the message, and moves the cursor down and to the left after each letter is printed. The relevant codes are:

8 move cursor to the left one position	9 move cursor to the right by one position.
10 move cursor down one line	11 move cursor up one line
13 move cursor to the start of the current line	

WIDTH

The WIDTHx command allows any text printed on the screen to be displayed in a column x characters wide. Normally there is no limit, and the computer overflows to the next line. To obtain the default setting, enter WIDTH0.

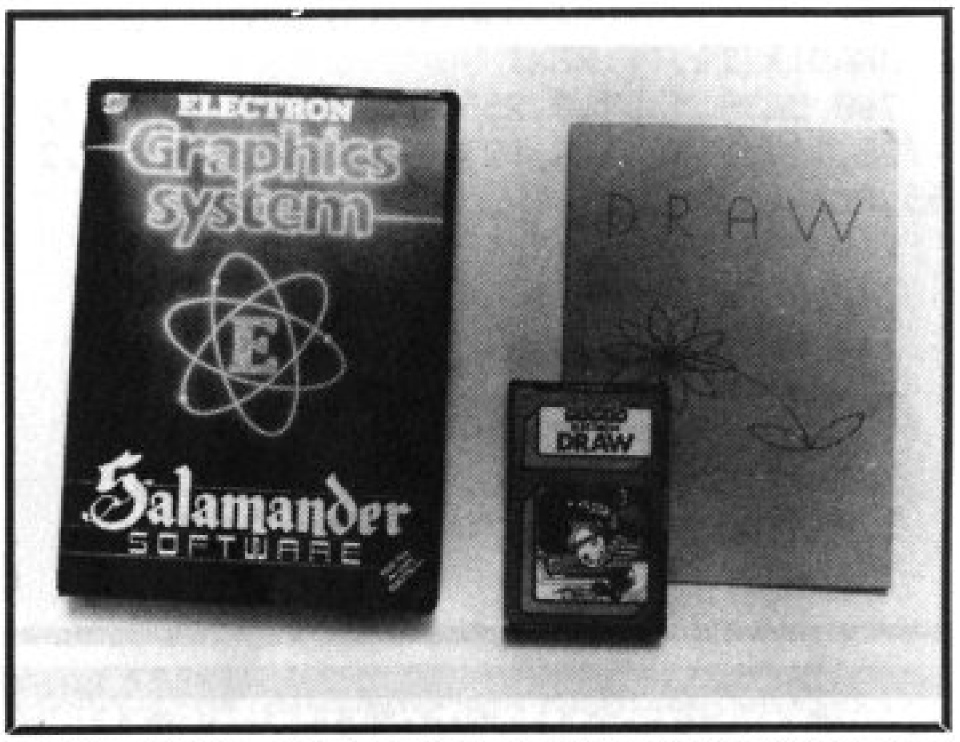
GETTING RID OF ? ON INPUT

INPUT ""A$ will not print a question mark. A message can be put between the quotes, but you must not put a ',' after the message, otherwise the question mark will be output.

GRAPHICS SOFTWARE FOR THE ELECTRON

reviewed by David A. Fell

Two companies, Program Power and Salamander Software, have released programs for the Electron which exploit the graphics capabilities of this micro. On closer examination, the two packages turn out to be quite different products, the one from Salamander Software being a quite respectable colour graphics package, while that from Program Power is more of a programming language (based on Logo) for producing black and white drawings. David Fell has tried both packages and reports in detail below.



Title   : DRAW
Supplier: Program Power
Price   : £8.95 inc. VAT
Rating   : ***

Both parts of this package are written in Basic, and loading is very easy. One fact that became obvious once the programs had loaded was that this was a BBC Micro version in an Electron case. The title page was a Mode 7 display, and included double height control codes, which the Electron could not apply to its display!

The package provides a new programming environment for the user. Some Electron owners may already have taken the time to investigate the 'Turtle' program provided on the 'B' side of the 'Welcome' cassette and described in the book "Start Programming with the Electron" by Masoud Yazdani that comes with the Electron. DRAW provides similar facilities, but also has other features. All interaction with the package takes place in mode 4, with the screen neatly split into two halves vertically. The left hand half is used for prompting, while the right hand half displays useful information.

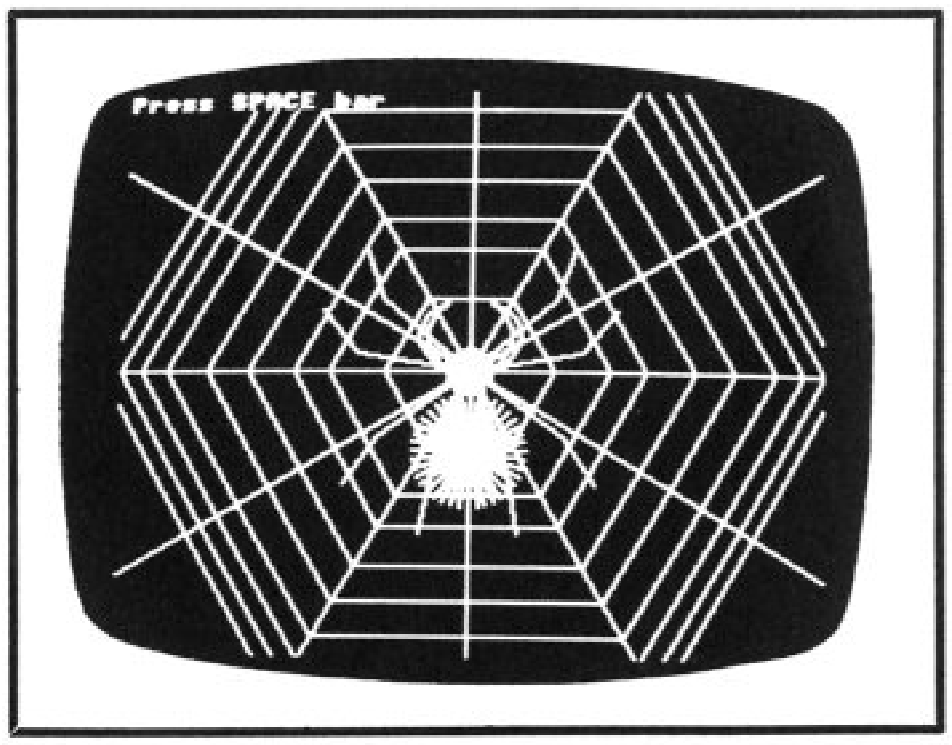
Drawing consists of a series of commands that are similar to some Basic commands. For example, the following DRAW 'program' will produce a square on the screen:

```
SQUARE SIZE
repeat 4
move SIZE
turn 90
end repeat
finish
```

The first line names a procedure as 'SQUARE', and gives it the parameter SIZE. This allows the routine to be used to produce a square of any given size, for example:

```
SQUARE 100
```

will draw a square of size 100 units.



If you follow through the commands comprising the program, you should see how logical and simple a program in DRAW, (and Logo for that matter), can appear. Using procedures like these as building blocks, you can go on to define very complex routines to draw a variety of geometrically based shapes on the screen.

Programs are stored internally with an identifying name. When, you ask to run a program (achieved merely by pressing 'R'), then all the names of the programs held within memory are listed out, along with an associated number. You now type in this number, press Return, and the program is run. As you may have gathered, DRAW allows more than one program to be resident in memory at once. In fact, you can have up to 2Ø programs in memory together, each with up to 3Ø lines.

The commands allowed within a program are:

move	program
turn	pen up
repeat	pen down
end repeat	finish

In addition to this, you may use any procedures of your own that you have already defined.

CONCLUSION

The DRAW package seems well thought out, and provides clear menus to help the inexperienced user. The manual is a photocopied one, and lacks the legibility in places that one would expect of Program Power, but the description of the program is more than adequate. The program is apparently designed to run on both the Electron and the BBC Micro. When available, the program will be able to take advantage of a printer, if attached, to produce a hard copy dump of the screen. The program could, however, have been better prepared for the Electron (Program Power have said a new version will be produced), and I would think that most users will find the Turtle program on the 'Welcome' cassette to be quite adequate.

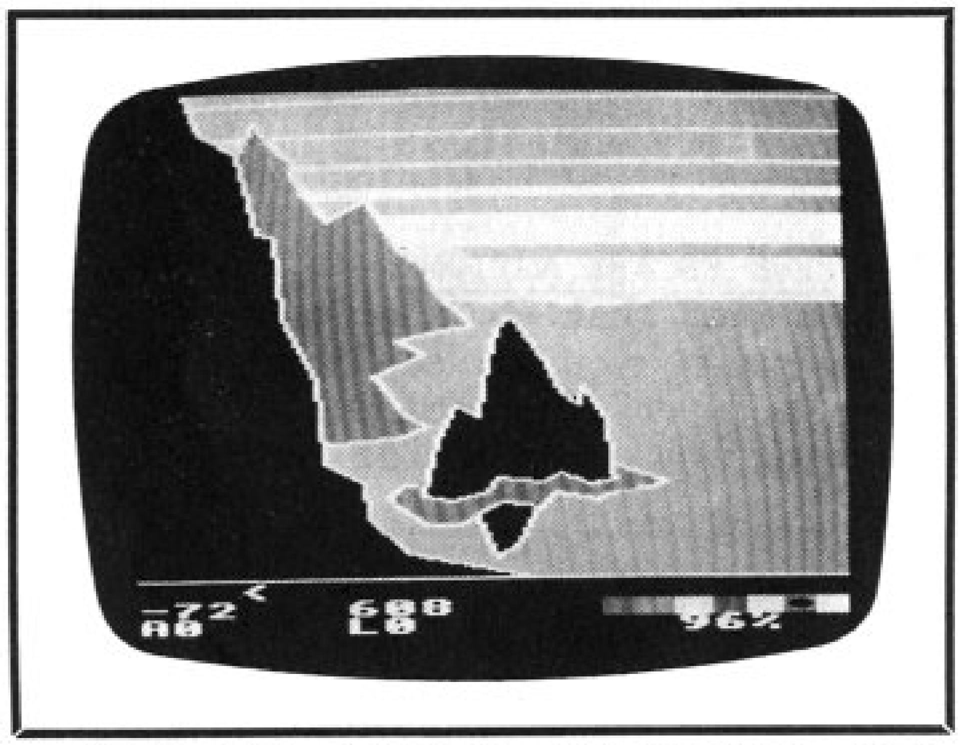
Title : Electron Graphics System
Supplier: Salamander Software
Price : £9.95 inc. VAT
Rating : ****

The Electron Graphics System from Salamander Software is a flexible and powerful piece of software that allows you to create your own screen images in any of the '2Øk' modes (i.e. modes Ø, 1, and 2), and to store and later recall them from tape. The cassette is attractively packaged, complete with a well printed manual. A similar package launched some time ago for the BBC micro, has already proved very popular, and this Electron program is based on that version.

Loading the system is very easy; being accomplished by the almost mandatory

```
CH."" <return>
```

and the two parts of the program load fairly quickly. Also provided on the tape are some example screens, which show 'mountain drawing', apparent motion by means of colour redefinition, and 'tree drawing'. These are produced in modes 2, 1 and Ø respectively (Personally, I found the tree a little strange).



When the program is running, the screen is split into two parts. The top, and larger part, contains the drawing which is being produced, with the lower section displaying various prompts and other information. The sequence of commands used to generate any picture is remembered by the program as the picture is built up, and there are facilities to erase the effects of the last command, and to redisplay the picture in the state that it was in before the last command was executed. This allows for any mistakes to be corrected.

The commands available are both

powerful and comprehensive. The cursor keys are used to position the cursor (a flashing set of cross hairs), and the Space Bar is used to 'fix' a point. For example, the command for a line is 'L', and the system then prompts for the 'start' of the line. The user moves the cursor to the desired start point, and then presses the Space Bar. The system then prompts for the 'next' point, and this is indicated in the same manner. As the cursor is moved to the next point, a flashing line is displayed to indicate where the final line would be if drawn in the current position. This 'elastic banding' facility is a very useful feature, allowing you to see the result of drawing a line before it is finally fixed.

Other facilities available include parallelogram drawing from any three points, circle and arc drawing, physical and logical colour changes, text displays from any position on the screen, filling of any shape, and pattern repetition.

The shape filling is achieved by using the built in PLOT extension, and so does not cope completely with complex shapes, small areas often being left unfilled. A more flexible routine could have been written to do this, but this would have used more memory, and hence reduced the complexity of picture that can be produced. Indeed, apart from the slightly 'sluggish' handling that results from a Basic program running in the high resolution modes, this is probably my only other criticism of the package. Whilst reading the manual, I spotted a number of errors, but a conversation with Salamander Software revealed that a new manual will be produced to correct these.

CONCLUSION

The package is well produced and allows for some quite effective pictures to be created and saved to tape. I would recommend this package to any Electron owner interested in experimenting with graphics. ●

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

ALTERNATIVE ESCAPE ACTIONS

If you find that you keep pressing the Escape key accidentally whilst running a program, then you may be interested to know that it is possible to redefine almost any key to return the escape action by using *FX220. For example *FX220,5 will only give the escape condition if you press the Control and 'E' keys together. ●

DISABLING THE ESCAPE KEY & DESTROYING PROGRAMS

*FX 200,0 Re-enables the Escape key.
*FX 200,1 Will disable the Escape key.
*FX 200,2 enables the Escape key, but pressing Break will clear the memory, preventing the program being retrieved with OLD.
*FX 200,3 disables the Escape key, and clears the memory on Break. ●

*COMMANDS - A.Pemberton

When using the structure IF...THEN *<command>, the THEN is not optional. ●

LOWER CASE BASIC - A.Armstrong

If you redefine the two characters @ and } by running the following program, you will find all upper case letters will now be printed in lower case, yet they still work as upper case.

```
10 MODE4
20 VDU 23,64,60,60,60,60,60,60,60,60
30 VDU 23,125,0,0,255,255,255,255,0,0
```
●

USING BBC MICRO PROGRAMS ON AN ELECTRON (Part 1)

by David Graham

It is early days for the Electron, and software to run on it (like the machine itself?) is in short supply. So why not convert some of the many available BBC micro programs to run on your Electron. David Graham looks into the problem.

The Electron has many things in common with its big brother the BBC micro, and the Electron was designed to be compatible with the Beeb in as many respects as possible. It should therefore be possible to convert some of the many programs for the Beeb to run on your Electron. After all they both run an identical version of BBC Basic.

In spite of the extreme similarities between the machines however, there are many things to look for when converting software, as we have discovered in our conversions of certain BBC micro programs for presentation in ELBUG. For this reason, a word of warning is in order. It is not advisable to buy commercial BBC software in the hope that it will run on the Electron. The chances are either that it will not run at all or at least that it will not run satisfactorily. This is especially true of games, which are usually in machine code, and which will generally be protected; so making them doubly difficult to modify. On the other hand, there is no harm in trying to load in a borrowed BBC micro cassette - the recording format is identical, and you might just be lucky.

In this article I shall be concerned with the modification for the Electron of programs written in Basic. But even here there are certain kinds of program which will not yield good results - and should be avoided. These are as follows:

1. Programs written largely in Mode 7 (see later) - especially those using Mode 7 graphics.

2. Programs in which multi-voice music is essential - e.g. a 3 voice rendition of a Bach Cantata.

3. High speed games - especially those for which the use of modes Ø, 1 or 2 is essential.

4. Programs for which printers, light pens or joysticks are essential.

5. Programs which use the timer in the second 6522 chip in the Beeb (extremely rare in Basic programs).

This list may give the impression that the majority of BBC micro programs cannot be used on the Electron. This is not in fact the case as we have discovered, though conversion often involves considerably more than altering a couple of program lines.

GETTING STARTED

If you have a program listing that you wish to use on the Electron, it is probably best to type it into your Electron and try to run it in the first instance. After all it may work faultlessly. It is more likely however that certain 'fixes' will be required. The remainder of the article will cover some of the problems that might arise.

Table 1 gives a list of potential problem areas. The three major difficulties arise from the Electron's lack of a Mode 7, its pruned down SOUND and ENVELOPE facilities, and its slower speed (especially noticeable in modes Ø to 3).

Of these, the latter often proves to be the most troublesome in games programs. Mode 7 can also cause a few headaches, though these are rarely insurmountable. SOUND and ENVELOPE incompatibilities are not often so serious, if only because the production of authentic sounds and sound effects is not usually vital to a program. We begin our survey with a look at Mode 7.

TABLE 1
Program Conversion Problems
BBC micro to Electron

MODE 7
The Electron has no Mode 7.

SOUND
The BBC micro has 4 sound channels (1 noise and 3 tone); the Electron has 2 only, 1 noise and one tone. On the BBC micro sound may have 16 different volume levels. On the Electron sound is either on or off.

ENVELOPE
The BBC micro may have up to 16 Envelopes simultaneously defined. The Electron allows one only. Of the 14 parameters in an Envelope on the BBC micro only the first 8 are used on the Electron because amplitude is not variable.

SPEED
The Electron runs a little slower than the BBC micro in modes 4, 5 and 6. In modes 0, 1, 2 and 3 it is considerably slower.

CURSOR DELETE
Certain versions of the cursor delete command on the BBC micro do not work on the Electron.

OTHER VDU23 CALLS
Certain other VDU 23 calls on the BBC micro - such as the ones which permit sideways scrolling and other forms of so called hardware scrolling, and changing the picture height are not implemented on the Electron.

JOYSTICKS
The ADVAL statement in BBC Basic with which the BBC micro reads the state of its analogue port, into which joysticks and other devices may be plugged, has no effect on a basic Electron (though hardware add-ons may implement this).

*FX CALLS
Certain *FX calls implemented on the BBC micro are not implemented on the Electron. These will mostly be associated with hardware not interfaced to the Electron - e.g. printer interface, analogue interface, video controller chip, 6522 etc.

MODE 7
The BBC micro has eight possible screen modes compared to the Electron's seven. Mode 7 on the BBC micro is not implemented on the Electron, and has no direct equivalent. In fact Mode 7 is a rather unusual mode in many respects. It takes only 1K of user memory so that very long programs which only use Mode 7 on the BBC micro might actually be too long to run on the Electron (since the most economical mode on the Electron is Mode 6 taking 8K).

Most programs will not fall into this category however. The other major differences with Mode 7 are its ability to display all 8 colours as well as flashing and double height text; and its different screen format (40 lines of 25 characters) - Mode 6 for example is 40 by 32.

In practice, Mode 7 is used mainly for displaying text on the BBC micro - such as program instructions, games high scores etc. Fortunately, Mode 7 text is usually readable on a Mode 6 screen, and moreover the instruction "MODE 7" is read by the Electron as "MODE 6", so that a BBC micro program that uses Mode 7 to display instructions will run on an Electron (providing it is not dumped as raw code to the screen area). But you will usually need to clean up the display somewhat.

Perhaps the best way to do this is to run the program, and take a good look at the screen. If text appears to be repeated on a pair of consecutive lines it probably means that the double height facility is in use. For example, the following lines will produce the word ELECTRON once only in double height on a BBC micro. It would print twice on an Electron.

```
10 MODE 7
20 PRINT CHR$141;"ELECTRON"
30 PRINT CHR$141;"ELECTRON"
```

To make this print the word once only on an Electron, delete line 30. The CHR$141 in line 20 should also be removed. Character 141 is the Mode 7 code for double height.

Mode 7 uses codes for a variety of purposes, and although these may have no effect in Mode 6, they can sometimes cause spurious characters to appear on the screen. You may also need to remove strings of lower case letters and other symbols from PRINT statements. For example the line

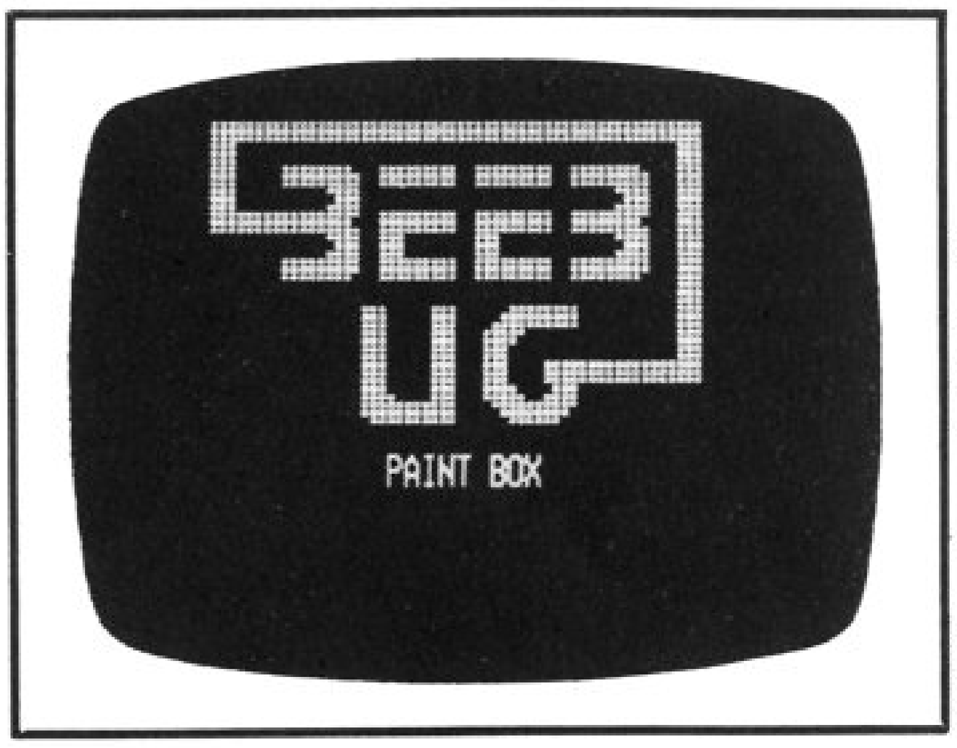
```
   10 PRINT CHR$145;"ffffffffffff"
```

will print a neat grid across the screen in red in Mode 7. The CHR$145 in Mode 7 tells the computer to print certain letters following it as red graphics characters. In other modes CHR$ is interpreted differently, and the letters are printed as they appear in the program - as a row of f's. On an Electron in Mode 6, you will just get a row of lower case 'f's in white - possibly preceded by an odd shape if character 145 happens to be defined in your machine at the time. The simple answer is to delete the whole line.

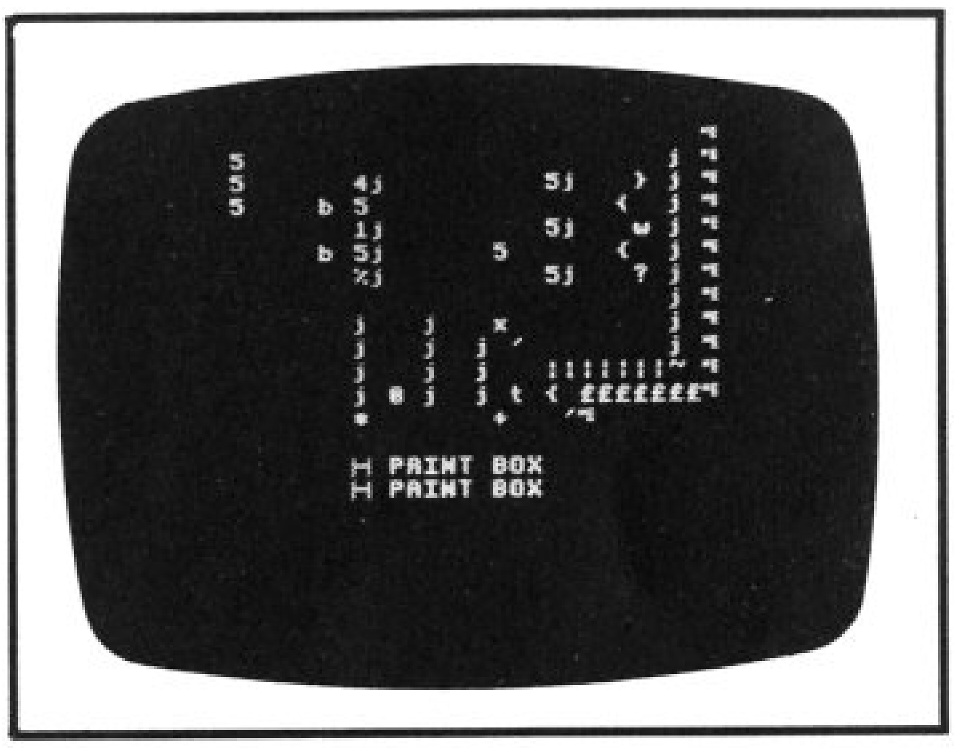
The screen shots with this article demonstrate the problem clearly. The first is of the header of a BEEBUG painting program. The logo uses Mode 7 graphics, and the words PAINT BOX are in double height. The same program run on an Electron produces a quite unrecognizable mess - note the double printing of the words PAINT BOX. Generally speaking the best approach is to redesign such pages using the facilities of other modes. Fortunately it is usually only the graphics of Mode 7 which become unreadable in other modes. The text, and this is usually the most important part, is generally legible, and will appear at more or less the same point on the screen as in Mode 7.

It is possible to recreate most of the effects available in Mode 7 in modes which will run on the Electron, but there is no simple rule of thumb. You will need to use graphics characters that you define yourself to replace the many pre-defined graphics characters of Mode 7, and to use the COLOUR statement to reproduce the Mode 7 colours.

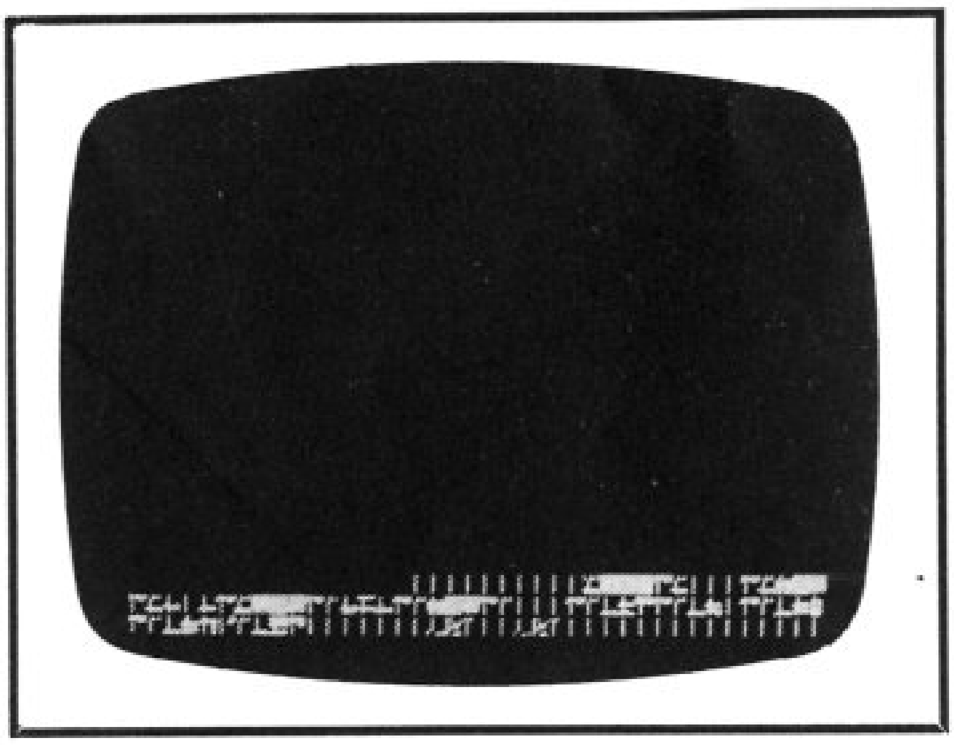
You cannot reproduce Mode 7 too closely however, since the only full colour mode on the Electron is Mode 2 and this only allows 20 characters of text per line. For this reason the best mode to replace Mode 7 will be either 1, 4 or 6. The disadvantage with 4 and 6 is that only two colours are allowed.



BEEBUG Logo in Mode 7.



BEEBUG Logo in Mode 6.


BEEBUG Logo dumped to Mode 6 Screen.

Mode 1 gives you 4 colours and 40 characters per line, but it also uses up 20K of memory compared to the miserly 1K of Mode 7. This will not cause problems as long as your program is shorter than 8.5K. Otherwise you will get a 'Bad mode' error message.

Next month we shall look at other problems of program conversion such as the use of Sound and Envelope commands.

MORE NEW BOOKS FOR ELECTRON USERS

reviewed by Mike Williams

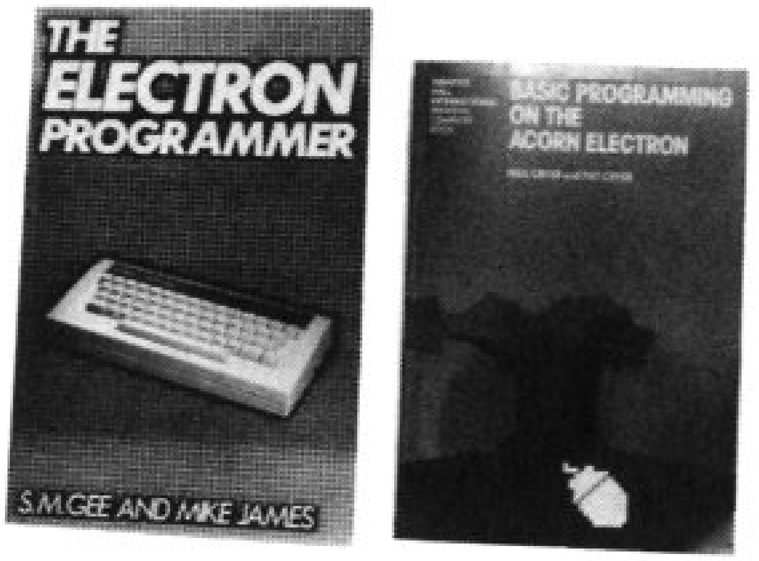
This month we look at two more books on programming for Electron users.

Book : The Electron Programmer
Authors : S.M.Gee and Mike James
Publisher: Granada
Price : £5.95
ISBN 0-246-12340-0

This book is intended for the newcomer to programming in Basic. It provides a comprehensive coverage of Basic on the Electron and there is some emphasis, particularly in the introduction, on the need to write well structured and readable programs. The book does its job in a sound if rather unexciting way. The various chapters cover progressively, the basic features of the language, control statements (GOTO, IF-THEN, REPEAT-UNTIL, FOR-NEXT), procedures and functions, text, integers, arrays, so called 'low resolution' graphics (using PRINT TAB), sound, and high resolution graphics (MOVE, DRAW, PLOT etc).

Each chapter follows the same pattern, a fairly systematic description and definition of each new language feature followed by a few, often mathematical, examples. This makes for very solid reading. I am not too happy about the preponderance of mathematical programs. Character strings and particularly graphics can be used from an early stage to provide much more interesting examples. After all, it is for their excellent graphics capabilities that many home users are attracted to buy their micro in the first place. On a slightly more academic point, it is a pity that the authors have not adopted the standard notation for thir definitions of syntax, but have felt obliged to introduce their own.

Although this book certainly does place some emphasis on programming style and structure, it is a pity that this approach is not used from the start. Likewise, with statements such as IF-THEN, less structured versions are introduced first with better structured versions and examples following later.



Overall, this book provides a sound description of Basic programming on the Electron. More diagrams and a less formal style would have made the book much more enjoyable reading. After all, programming the Electron for the home user can and should be fun!

Book :Basic Programming on the Acorn Electron
Authors :Neil Cryer and Pat Cryer
Publisher:Prentice Hall International
Price :£6.95
ISBN 0-13-066259-3

This is another book on Basic programming for the beginner and with just over 300 pages is positively packed with ideas and information, sufficient to keep the new Electron user busy for quite some time. The book contains many example programs and screen displays to illustrate the techniques described, and to provide a storehouse of routines to interest and entertain the user.

The book starts with a useful description of the Electron, its keyboard and its operation. It is very pleasing to see the idea of the procedure being introduced at a very early stage in the book (in the second chapter) and procedures form the basis of the majority of the examples throughout the book.

There is a useful chapter on saving and loading programs on cassette followed by basic programming information. The chapters progressively describe all the structured features of Electron Basic including REPEAT-UNTIL and IF-THEN-ELSE. There is no mention of GOTO or any of the unstructured forms of IF-THEN until very late in the book, which does avoid introducing the user to any bad programming habits. There is detailed coverage of all aspects of sound and graphics on the Electron, including animation techniques and user-defined characters.

One point of criticism is the quality of presentation and reproduction. The text appears to have been printed directly from computer generated material on a daisywheel printer with rather variable quality. The program listings are in at least three different type faces including dot matrix printer listings and two styles of daisywheel printer listings. This has the unfortunate result that the figure zero is sometimes printed as 'O' and sometimes as 'Ø'. Consistency of presentation is desirable, while it would have improved the general appearance of the book if the text had been typeset.

One further point is that example programs and screen displays are quite often allowed to break up the main flow of the text and on several occasions I was not sure where to continue reading.

I liked both the content and the style of presentation of this book, and with so much packed into its 3ØØ plus pages it really is good value at the price. It also has the rare merit with regard to structured programming of practicing what it preaches from beginning to end.

THE BEST OF ELBUG

Many of the best programs published in ELBUG have been collected together and published by Penguin Books under the name "Games and other programs for the Acorn Electron" at £3.95. This book is part of the Penguin Acorn Computer Library and at present there is just one other title available though others are planned.

There are 2Ø programs in all in four different categories:

Action Games

Munch-Man	Mars Lander	Invasion
Robot Attack	Hedgehog	

Thought games

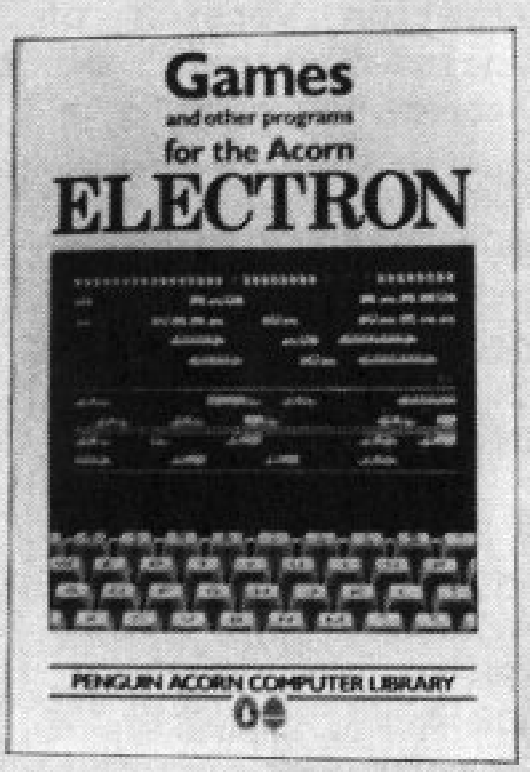
Higher/Lower	Five-Dice	Life
Anagrams	Return of the Diamond	

Visual Displays

Union Jack	Square Dance	Ellipto
Screenplay	3-D Rotation	

Utilities

Sound Wizard	Bad Program Lister
3-D Lettering	Bad Program Rescue
Double Height Text	



All 2Ø programs are now available on cassette from our software address (in High Wycombe) price £7 to members and £9 to non-members, plus 5Øp post & packing in either case.

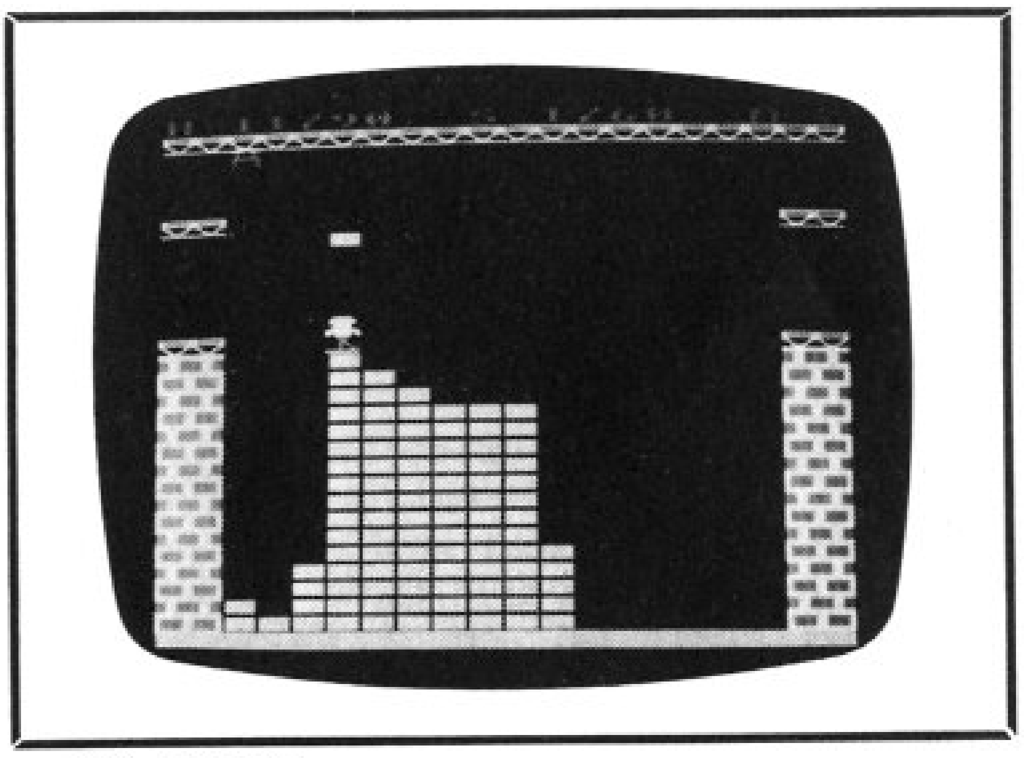
BLOCK BLITZ

by D. J. Pilling

The following program is one of the most entertaining games we have seen, and will well repay the effort spent in typing it into your micro. It uses colour, graphics and sound to produce a fast and original action game that compares very favourably with the commercial games currently available.

In this game, you are trapped in a series of caverns, from which you have to escape whilst being bombarded by blocks, dropped from above by a very mobile crane. As the blocks fall towards you, you must try to dodge them if you want to stay alive, but your only way of making good your escape is to climb onto the blocks, and up to the height of the walls. The crane is quite intelligent, and will try and drop the blocks on you, giving you the chance to practice some risky brinkmanship as you lure the blocks into the places that will best help you make your exit. If you succeed, a gantry extends from one of the walls, allowing you to escape. You are then thrown into an even deeper cavern!

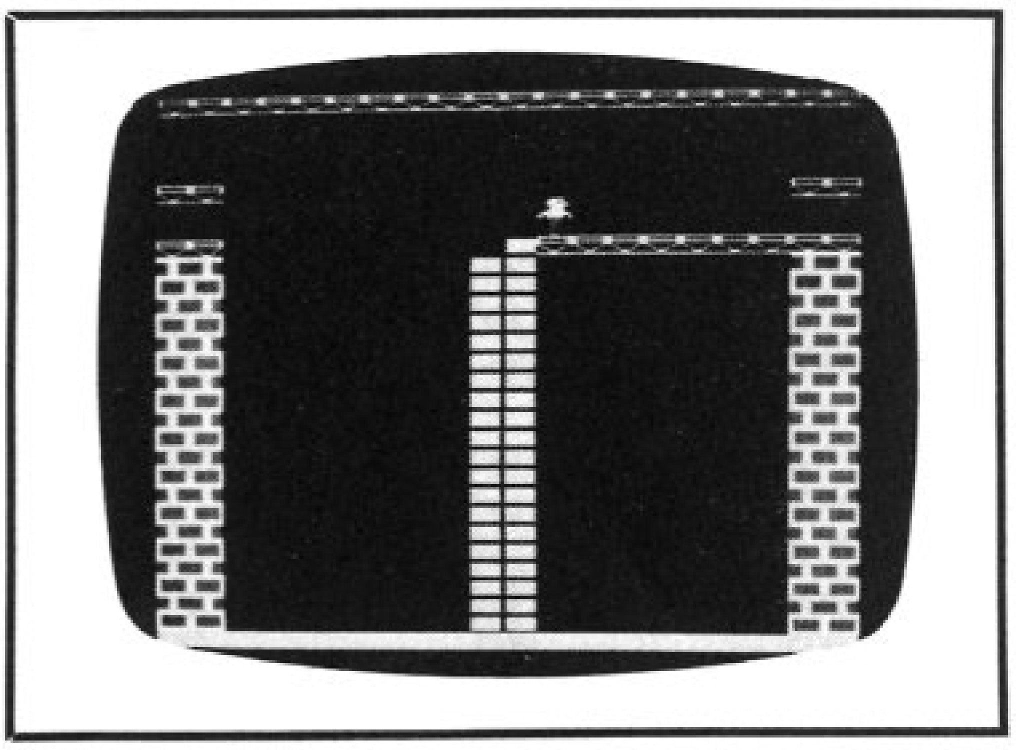
You move the man using the left and right cursor keys, and he will also climb over any blocks in the way, provided he doesn't have to step up or down more than one block at a time. You will need to be very alert to avoid being trapped on a column, or between two high walls of blocks. A demonstration version of the game is also included as part of the program which shows how easy it all is!



```
 10 REM PROGRAM BLOCK BLITZ
 20 REM VERSION E0.1
 30 REM AUTHOR  D.J.PILLING
 40 REM ELBUG   MARCH 1984
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
 70 ON ERROR GOTO 250
 80 DIM S%(19,31),MAN$(5,1)
 90 PROCchar
100 REPEAT
110 MODE 4
120 PROCsetupuser
130 MODE5
140 PROCsetup
150 REPEAT
160 PROCscreen
170 REPEAT
180 PROCbl:PROCm
190 UNTIL end%
200 PROCgame
210 UNTIL demo%
220 UNTIL FALSE
230 END
240 :
250 ON ERROR OFF:MODE 6
260 IF ERR<>17 THEN REPORT:PRINT" at
line ";ERL
270 END
280 :
290 DEF PROCchar
300 a$=CHR$8:b$=CHR$10:c$=CHR$11
310 d$=CHR$17:e$=CHR$32
320 f$=STRING$(14,CHR$32):g$=STRING$(
5,CHR$32)
330 VDU23,224,28,48,62,20,28,60,126,1
89
340 VDU23,225,189,60,24,24,56,104,200
,76
350 VDU23,226,189,60,24,24,28,23,18,24
360 VDU23,227,189,60,24,24,28,23,18,48
370 VDU23,228,189,60,24,24,56,104,200
,88
380 VDU23,229,28,6,62,20,28,60,126,189
390 VDU23,240,60,90,126,60,36,126,255
,189
```

»

```
  400 VDU23,243,255,129,255,129,66,36,2
4,255
  410 VDU23,244,0,127,127,127,127,127,1
27,0
  420 VDU23,245,255,255,255,255,255,255
,255,255
  430 VDU23,246,0,126,126,126,126,126,1
26,0
  440 VDU23,247,0,231,231,231,231,231,2
31,0
  450 VDU23,248,189,189,60,36,36,36,36,
102
  460 VDU23,249,119,34,34,127,127,65,65
,99
  470 VDU23,250,189,219,255,189,165,255
,255,60
  480 VDU23,251,60,60,60,36,36,36,36,102
  490 ENVELOPE1,3,0,2,0,0,255,0,127,0,0
,-127,80,80
  500 R$=d$+CHR$1:Y$=d$+CHR$2
  510 W$=d$+CHR$3
  520 M$=W$+CHR$240+b$+a$+R$+CHR$248
  530 MJ$=b$+e$+a$+c$+R$+CHR$251+a$+c$+
W$+CHR$250
  540 MK$=c$+e$+b$+a$+W$+CHR$240+a$+b$+
R$+CHR$248
  550 f$=e$+W$+CHR$224+b$+a$+a$+e$+R$
  560 MAN$(1,0)=f$+CHR$225
  570 MAN$(1,1)=f$+CHR$226
  580 f$=e$+a$+a$+W$+CHR$229+b$+e$+a$+a
$+R$
  590 MAN$(0,0)=f$+CHR$227
  600 MAN$(0,1)=f$+CHR$228
  610 f$=b$+e$+c$+a$+e$+R$:g$=a$+c$+W$+
CHR$224
  620 MAN$(3,0)=f$+CHR$225+g$
  630 MAN$(3,1)=f$+CHR$226+g$
  640 f$=e$+a$+b$+e$+W$+CHR$224+a$+b$+R$
  650 MAN$(5,0)=f$+CHR$225
  660 MAN$(5,1)=f$+CHR$226
  670 f$=b$+e$+c$+a$+e$+a$+a$+R$:g$=a$+
c$+W$+CHR$229
  680 MAN$(2,0)=f$+CHR$227+g$
  690 MAN$(2,1)=f$+CHR$228+g$
  700 f$=e$+a$+b$+e$+a$+a$+W$+CHR$229+a
$+b$+R$
  710 MAN$(4,0)=f$+CHR$227
  720 MAN$(4,1)=f$+CHR$228
  730 B$=R$+CHR$249+a$+b$+Y$+CHR$244
  740 TL$=e$+a$+a$+R$+CHR$249
  750 TR$=e$+R$+CHR$249
  760 BL$=TL$+a$+b$+Y$+CHR$244+e$
  770 BR$=TR$+a$+a$+b$+e$+Y$+CHR$244
  780 BD$=Y$+e$+a$+b$+CHR$244
  790 WALL$=CHR$246+CHR$246+c$+a$+a$+CH
R$247+CHR$247+c$+a$+a$
  800 HI%=0:SC%=0
  810 ENDPROC
  820 :
  830 DEF PROCsetup
```



```
  840 VDU23,1;0;0;0;0:VDU29,640;512;
  850 IF HI%<SC% THEN HI%=SC%
  860 H%=4:N%=1:M%=0:MEN%=3
  870 SC%=FNsc(H%):SCT%=0
  880 ENDPROC
  890 :
  900 DEF PROCscreen
  910 CLS:COLOUR1
  920 PRINTTAB(0,0)"H ";HI%;TAB(9);"S "
;SC%;TAB(17)"M ";MEN%
  930 COLOUR2
  940 g$=STRING$(2,CHR$243)
  950 PRINTTAB(0,1)STRING$(10,g$);
  960 PRINTTAB(0,6)g$ TAB(18,6)g$;
  970 PRINTTAB(0,29-H%)g$ TAB(18,29-H%)
g$;
  980 COLOUR2:PRINTTAB(0,30)STRING$(20,
CHR$245);
  990 PRINTTAB(0,30)STRING$(20,CHR$245);
 1000 COLOUR131:COLOUR1
 1010 g$=STRING$(H%/2,WALL$)
 1020 PRINTTAB(0,29)g$;
 1030 PRINTTAB(18,29)g$;
 1040 COLOUR128
 1050 FOR J%=0 TO 31
 1060 FOR I%=0 TO 19
 1070 IF J%=30 THEN S%(I%,J%)=2 ELSE S%
(I%,J%)=0
 1080 NEXT I%,J%
 1090 B%=28:A%=10:NG%=FALSE:end%=FALSE
 1100 PRINTTAB(A%,B%)M$
 1110 Y%=3:IF RND(2)=1 THEN  X%=0 ELSE
X%=19
 1120 ENDPROC
 1130 :
 1140 DEF PROCm
 1150 IF M%=0 THEN M%=1 ELSE M%=0
 1160 IF S%(A%,B%)<>0 THEN PROCng:ENDPR
OC
 1170 IF demo% THEN 1200
 1180 IF INKEY-26 THEN IF A%>2 THEN 1280
 1190 IF INKEY-122 THEN IF A%<17 THEN 1
230
```

➤➤

```
 1200 IF NOT demo% THEN PRINTTAB(A%,B%)
M$;:ENDPROC
 1210 IF Y%<=3 OR X%<>A% THEN PRINTTAB(
A%,B%)M$;:ENDPROC
 1220 IF A%=10 THEN 1280
 1230 IF S%(A%+1,B%)<>0 THEN ENDPROC
 1240 IF S%(A%+1,B%+1)=2 THEN PRINTTAB(
A%,B%)MAN$(3,M%):A%=A%+1:B%=B%-1:GOTO 1
320
 1250 IF S%(A%+1,B%+2)=2 THEN PRINTTAB(
A%,B%)MAN$(1,M%):A%=A%+1:GOTO 1320
 1260 IF S%(A%+1,B%+3)=2 THEN PRINTTAB(
A%,B%)MAN$(5,M%):A%=A%+1:B%=B%+1:GOTO 1
320
 1270 ENDPROC
 1280 IF S%(A%-1,B%)<>0 THEN ENDPROC
 1290 IF S%(A%-1,B%+1)=2 THEN PRINTTAB(
A%,B%)MAN$(2,M%):A%=A%-1:B%=B%-1:GOTO 1
320
 1300 IF S%(A%-1,B%+2)=2 THEN PRINTTAB(
A%,B%)MAN$(0,M%):A%=A%-1:GOTO 1320
 1310 IF S%(A%-1,B%+3)=2 THEN PRINTTAB(
A%,B%)MAN$(4,M%):A%=A%-1:B%=B%+1
 1320 IF S%(A%,B%)<>0 THEN PROCng:ENDPR
OC
 1330 IF B%=27-H% THEN PROCwin:ENDPROC
 1340 ENDPROC
 1350 :
 1360 DEF PROCng
 1370 end%=TRUE:NG%=TRUE:MEN%=MEN%-1
 1380 SOUND&0011,0,0,0:SOUND0,-15,5,12
 1390 PROCbonk
 1400 ENDPROC
 1410 :
 1420 DEF PROCd(T%)
 1430 T%=TIME+T%
 1440 REPEAT UNTIL TIME>T%
 1450 ENDPROC
 1460 :
 1470 DEF PROCwin
 1480 IF S%(X%,Y%)=1 THEN PRINTTAB(X%,Y
%)e$;:SOUND&0011,0,0,0
 1490 PRINTTAB(A%,B%)M$;
 1500 end%=TRUE:COLOUR2
 1510 IF A%=2 OR A%=17 THEN 1540
 1520 D%=A%
 1530 IF A%<9 THEN FOR C%=2 TO A%-1:PRI
NTTAB(C%,B%+2)CHR$243:SOUND2,-10,100,1:
PROCd(15):NEXT ELSE FOR C%=17 TO A%+1 S
TEP-1:PRINTTAB(C%,B%+2)CHR$243:SOUND2,-
10,100,1:PROCd(15):NEXT
 1540 REPEAT IFA%<9 PRINTTAB(A%,B%)MAN$
(0,M%):PROCd(15):A%=A%-1 ELSE PRINTTAB(
A%,B%)MAN$(1,M%):PROCd(15):A%=A%+1
 1550 IFM%=0 THEN M%=1 ELSE M%=0
 1560 UNTIL A%=1 OR A%=18
 1570 IF D%=2 OR D%=17 THEN 1630
 1580 COLOUR2:PRINTTAB(A%,B%)M$;
 1590 IF D%<9 THEN FOR C%=D%-1 TO 2 STE
P-1:PRINTTAB(C%,B%+2)e$:SOUND2,-10,100,
1:PROCd(15):NEXT ELSE FOR C%=D%+1 TO 17
:PRINTTAB(C%,B%+2)e$:SOUND2,-10,100,1:P
ROCd(15):NEXT
 1600 COLOUR0:COLOUR131
 1610 IF A%=1 PRINTTAB(3,B%-1)"YIPPEE"
ELSE PRINTTAB(11,B%-1)"YIPPEE"
 1620 COLOUR128
 1630 FOR I%=1 TO 10:PRINTTAB(A%,B%)MJ$
;:SOUND0,-15,4,1:PROCd(8):PRINTTAB(A%,B
%)MK$;:SOUND0,-15,6,1:PROCd(8):NEXT
 1640 IF A%=1 PRINTTAB(3,B%-1)SPC(6) EL
SE PRINTTAB(11,B%-1)SPC(6)
 1650 PROCd(150)
 1660 ENDPROC
 1670 :
 1680 DEF PROCbl
 1690 IF Y%=3 THEN 1800
 1700 IF T%>9 THEN 1720
 1710 IF T%>0 PRINTTAB(T%,2)TL$:T%=T%-1
:GOTO 1730 ELSE PRINTTAB(0,2)e$:GOTO 17
30
 1720 IF T%<19 PRINTTAB(T%,2)TR$:T%=T%+
1 ELSE PRINTTAB(19,2)e$
 1730 PRINTTAB(X%,Y%)BD$:S%(X%,Y%)=0:Y%
=Y%+1:S%(X%,Y%)=1:IF S%(X%,Y%+1)<>2 THE
N ENDPROC
 1740 S%(X%,Y%)=2
 1750 SOUND&0011,0,0,0:SOUND0,-15,4,1
 1760 IF T%<>0 AND T%<>19 Y%=Y%-1:ENDPR
OC
 1770 IF RND(1)>.5 THEN  X%=19 ELSE X%=0
 1780 SC%=SC%-10:PRINTTAB(11,0)SPC(5)TA
B(11,0)R$;SC%
 1790 Y%=3:PRINTTAB(X%,2)B$;:ENDPROC
 1800 SOUND2,-10,60,1
 1810 IF (A%=X% AND RND(1)>.5) THEN 1860
 1820 IF demo% THEN 1840
 1830 IF ABS(A%-X%)<2 THEN IF RND(1)>.8
 THEN 1860
 1840 IF A%<X% THEN PRINTTAB(X%,2)BL$:X
%=X%-1 ELSE PRINTTAB(X%,2)BR$:X%=X%+1
 1850 ENDPROC
 1860 IF X%<2 OR X%>17 ENDPROC
 1870 T%=X%:SOUND1,1,350,90:GOTO1730
 1880 :
 1890 DEF PROCgame
 1900 IF NG% AND MEN%=0 THEN PROClost:P
ROCsetup:ENDPROC ELSE IF NG% THEN PROCc
ng:ENDPROC
 1910 N%=N%+1:H%=H%+2
 1920 IF H%=18 THEN PROCover:PROCsetup:
ENDPROC
 1930 PROCccg
 1940 ENDPROC
 1950 :
 1960 DEF FNsc(H%)=160*(H%+1)
 1970 :
 1980 DEF PROCr:REPEAT UNTIL INKEY-74:E
NDPROC
 1990 :
```

»

```
 2000 DEF PROCsetupuser
 2010 VDU19,1,3;0;19,0,4;0;
 2020 PRINTTAB(10,3)"B L O C K  B L I T
 Z"
 2030 PRINTTAB(10,4)STRING$(20,"=")
 2040 PRINTTAB(0,7)"  You have been imp
risoned by the evil"'" Dr. X in an ever
 deeper series of"'" caverns. To add to
 your problems, they"'" are being fille
d in with blocks aimed"'" at you!"
 2050 PRINT''"  However if you can reac
h the level of"'" the brick sides by st
anding on the"'" blocks you can escape
from each cavern"'" and eventually from
 all of them."
 2060 PRINT''"  You move by using the l
eft and right"'" cursor keys but you ca
n only leap up"'" and down one block at
 a time."
 2070 *FX15,0
 2080 PRINT''"  Do you want a demo ? (Y
/N) ";
 2090 IK$=GET$:IF IK$="Y" THEN demo%=TR
UE ELSE IF IK$="N" THEN demo%=FALSE ELS
E GOTO 2090
 2100 ENDPROC
 2110 :
 2120 DEF PROCcng
 2130 SC%=SCT%:CLS:COLOUR2
 2140 PRINTTAB(4,5)"OH DEAR !"TAB(4,7)"
YOU LOST"TAB(4,9)"A MAN"
 2150 PROCsctab:SC%=SC%+FNsc(H%)
 2160 ENDPROC
 2170 :
 2180 DEF PROCccg
 2190 SCT%=SC%:CLS:COLOUR2
 2200 PRINTTAB(2,5)"NEXT -->"TAB(2,8)"C
AVERN NUMBER ";N%
 2210 PROCsctab:SC%=SC%+FNsc(H%)
 2220 ENDPROC
 2230 :
 2240 DEF PROCsctab
 2250 VDU19,1,6,0,0,0
 2260 COLOUR1
 2270 PRINTTAB(4,14)"MEN.......";MEN%
 2280 PRINTTAB(4,16)"SCORE.....";SC%
 2290 PRINTTAB(4,18)"HISCORE...";HI%
 2300 COLOUR2:PRINTTAB(1,28)"RETURN TO
CONTINUE"
 2310 PRINTTAB(1,28)"RETURN TO CONTINUE"
 2320 PROCr:VDU19,1,1,0,0,0
 2330 ENDPROC
 2340 :
 2350 DEF PROCover
 2360 CLS:COLOUR2
 2370 SC%=SC%+MEN%*FNsc(18)
 2380 PRINTTAB(2,3)"CONGRATULATIONS"TAB
(2,6)"YOU HAVE ESCAPED"TAB(2,8)"FROM AL
L THE"TAB(2,10)"THE CAVERNS"
 2390 PROCsctab
 2400 ENDPROC
 2410 :
 2420 DEF PROClost
 2430 CLS:COLOUR2:SC%=SCT%
 2440 PRINTTAB(4,4)"GAME OVER"TAB(4,6)"
YOU RAN OUT"TAB(4,8)"OF MEN"
 2450 PROCsctab
 2460 ENDPROC
 2470 :
 2480 DEF PROCbonk
 2490 CLS
 2500 FOR X%=150 TO 500 STEP 50:GCOL0,1
:MOVE X%,X%:DRAW X%,-X%:DRAW -X%,-X%:DR
AW -X%,X%:DRAW X%,X%:X%=X%+50:GCOL0,2:M
OVE X%,X%:DRAW X%,-X%:DRAW -X%,-X%:DRAW
 -X%,X%:DRAW X%,X%:NEXT
 2510 COLOUR3:PRINTTAB(6,15)"B O N K!"
 2520 FOR I%=1 TO 8:VDU19,1,3,0,0,0,19,
2,1, 0,0,0:PROCd(8):VDU19,1,1,0,0,0,19,
2,3,0 ,0,0:PROCd(8):NEXT
 2530 ENDPROC
```

BEEBUGSOFT

Snake

Snake is a very addictive, fast moving arcade game. The purpose of the game is to direct the snake around the screen, eating up the randomly placed objects. With every bite that it takes, the tail grows longer and the snake moves faster. Just to complicate things, there are also certain scattered obstacles which must be avoided.

To even complete the first screen requires skill and concentration. Further frames use different shaped playing areas, which as well as adding variety, require much greater skill to complete.

This game is much more exciting than many games of a similar nature, and once played, proves to be surprisingly compulsive.

Snake costs £4.00 inc VAT. Please add 50p post & packing.
Overseas orders: £5.50 inc post & packing - VAT not charged.
Send order with membership number to:
BEEBUGSOFT, P.O. Box 109, High Wycombe, Bucks HP11 2TD.

IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ORBIT members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG
PO BOX 109
High Wycombe
Bucks

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:
£5.90 for 6 months (5 issues)
£9.90 for 1 year (10 issues)
European Membership £16 for 1 year.
Elsewhere (Postal zones)
Zone A £19, Zone B £21, Zone C £23

SOFTWARE (Members only)

This is available from the software address.

MAGAZINE CONTRIBUTIONS AND TECHNICAL QUERIES

Please send all contributions and technical queries to the editorial address opposite. All contributions published in the magazine will be paid for at the rate of £25 per page.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG
PO Box 50
St Albans
Herts

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Technical Editor: Philip Le Grand. Production Editor: Phyllida Vanstone.
Technical Assistants Alan Webster and David Fell.
Managing Editor: Lee Calcraft.
Thanks are due to David Graham, Sheridan Williams, Adrian Calcraft, Robert Barnes and Mike Beasley for assistance with this issue.

Printed in England by Staples Printers St Albans Limited at The Priory Press.