£1.00

ELBUG

FOR THE ELECTRON

MARS
LANDER

GAMES
* DIVE BOMBER
* REVERSI
PLUS
* 3D ROTATION
* NEW
  CHARACTER SET
* ELLIPTO
* BAD PROGRAM
  LISTER
PLUS
* REVIEWS
* POSTBAG
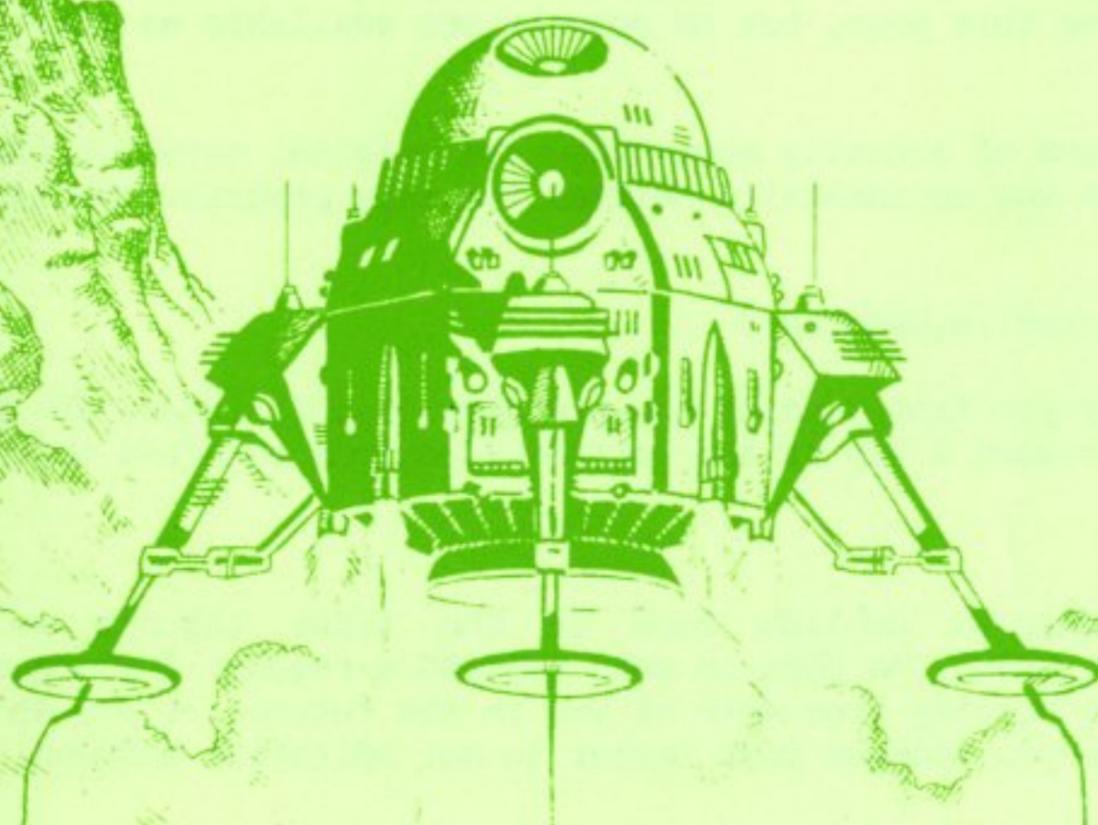* HINTS & TIPS
And much more

STEVE KYTE · R · 83

EDITORIAL

THIS MONTH'S MAGAZINE

The magazine is again packed with articles and programs. I would particularly draw your attention to the three excellent games and to the article and program on three dimensional graphics (3D Rotation), which again illustrates the quality of the computer graphics that can be achieved with the Electron. There are also further articles in the two series on Electron Graphics and Writing a Games Program and many more interesting and useful items in this issue.

ADD-ONS FOR THE ELECTRON

In the first issue of ELBUG we reported that SIR Computers of Cardiff were planning add-on units for the Electron. These are now promised for early this year, and the first product is likely to be a ROM expansion board at £40 + VAT. This will then be followed by a unit with printer and joystick connections, with further add-on units to follow.

Another company, Solidisk Technology Ltd., is also planning a General Purpose Interface (GPI), which will provide a printer port, a user port, joystick port and additional ROM sockets for the Electron. The cost is likely to be from £20 upwards depending on the options included. A prototype unit was on demonstration at the Micro User Show in December. Enquiries or comments should be sent to Solidisk Technology Ltd. at 17 Sweyne Avenue, Southend-On-Sea, Essex SS2 5JJ.

Acorn are themselves planning a sophisticated multifunction interface for the Electron to be released later this year, but no details are available as yet.

RETURN OF THE DIAMOND

During pasting (the process of actually assembling the original material for our printers) part of a line was accidentally omitted for this program. The full and correct line is as follows:

    500 DATA a mean looking gremlin,GREMLIN,4

We are sorry if this stopped you from running the program. In this issue of ELBUG, as promised, we have included a map of the various locations referred to in this adventure game.

YOUR LETTERS

This month we have been able to publish some of the first letters we have received from readers of ELBUG. We hope to make POSTBAG a regular feature of the magazine and look forward to hearing from more of you in the future. If you do write to us, then please make sure you address your letter to our editorial address.

NEXT MONTH'S ISSUE OF ELBUG

This is just to remind you that this issue of ELBUG covers both January and February, and that the next issue of the magazine will be the March issue.

Mike Williams

ELBUG MAGAZINE

GENERAL CONTENTS

PROGRAMS

HINTS & TIPS

MARS LANDER
by Alan Webster

Mars Lander is a version of the popular moon lander game in which the player attempts to navigate his space craft through a mass of meteors to land it softly on the landing pad. You control the horizontal and vertical speed of the craft. The game runs in mode 5 and makes good uses of both colour and sound.

The player can choose the level of difficulty for the game - the more difficult the game the more sensitive the controls and the rougher the terrain (ranging from quite smooth to very rough). You have 15 seconds to land the craft before your fuel runs out.
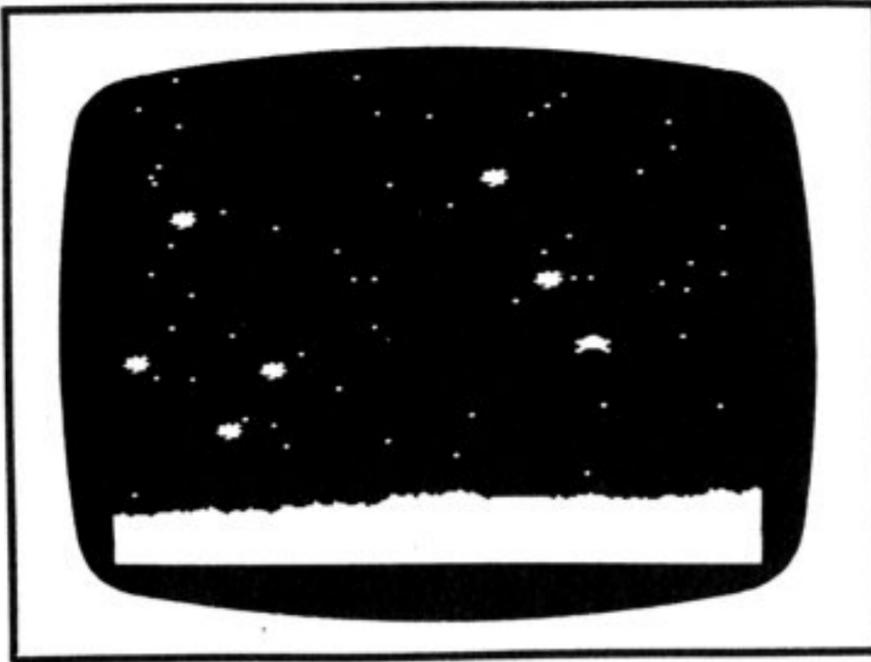
The controls are given in the program, and for those of you who find the game too easy (in excess of 2000 points) you might like to change the gravity factor by altering line 200 to:

```
200 N%=N%-(G%/1.95-(1/Y%/10
)):IF N%>80 THEN N%=80
```

This program is adapted from a version for the BBC micro first published in BEEBUG Vol.2 No.4.

```
   10 REM PROGRAM MARS LANDER
   20 REM AUTHOR ALAN WEBSTER
   30 REM VERSION E1.0
   40 REM ELBUG JAN/FEB 1984
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60:
   70 ON ERROR GOTO 750
   80 PROCchars:PROCarrays:MODE4
   90 PRINTTAB(14,2)"Mars Lander";TAB(1
2,3)"by Alan Webster"
  100 PRINTTAB(5,10)"Difficulty. 1-8 (1
=Easy) ?":G%=GET:IFG%<49 OR G%>56 GOTO
100
  110 G%=G%-47:dead=FALSE:SC%=0
  120 L%=G%*6:PRINTTAB(13,19)"Z - Left"
;TAB(13,20)"X - Right";TAB(8,21)"RETURN
 - Thrust";TAB(9,23)"Press any key":i=G
ET
  130 MODE5:A2%=100:VDU5:REPEAT:CLS:PRO
Cstars:PROCmeteors:X%=RND(900):Y%=1000:
M%=0:N%=1:landed=FALSE:PROCscene
  140 TIME=0:REPEAT:OX%=X%:OY%=Y%
  150 IF INKEY-98 IF X%>32 M%=M%-G%
  160 IF INKEY-67 IF X%<1210 M%=M%+G%
  170 X%=X%+(M%/2):IF X%<32 X%=32
  180 IF X%>1210 X%=1210
  190 M%=M%+(SGN(M%)/10):IF INKEY-74 N%
=N%+G%
  200 N%=N%-(G%/10):IF N%>80 N%=80
  210 Y%=Y%-8+N%:IF OX%=X% IF OY%=Y% GO
TO 230
  220 MOVE OX%,OY%:GCOL0,0:VDU224:MOVE
X%,Y%:GCOL0,3:VDU224
  230 R%=POINT(X%+55,Y%-31):R1%=POINT(X
%,Y%-31):SOUND0,-10,4,1:IF R%=1 dead=TR
UE
  240 IF R1%=1 dead=TRUE
```
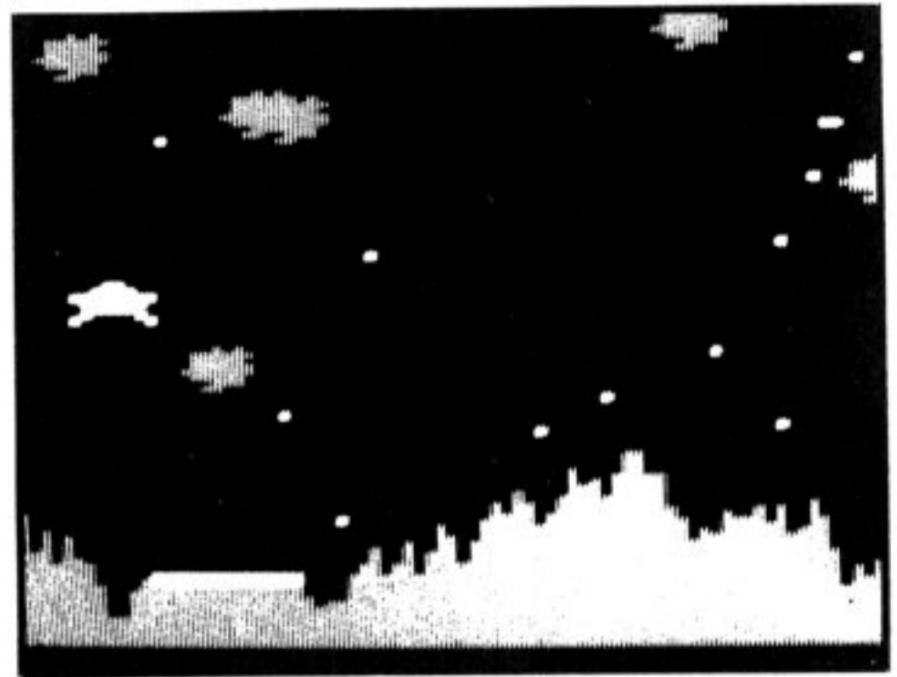
```
520 DIM Z%(8),Z2%(8),Q%(8),Q1%(8)
530 FOR Z1%=1TO8:Q%(Z1%)=RND(12):IF Q
%(Z1%)>6 Q%(Z1%)=Q%(Z1%)-13
540 Q1%(Z1%)=RND(12):IF Q1%(Z1%)>6 Q1
%(Z1%)=Q1%(Z1%)-13
550 NEXT
560 ENDPROC
570 :
580 DEFPROCexpl
590 GCOL0,0:MOVEX%,Y%:VDU224:FOR Z6%=
1TO8:Z%(Z6%)=X%:Z2%(Z6%)=Y%:NEXT
600 FOR P%=1TO14:FOR J%=1TO8:GCOL0,0:
PLOT69,Z%(J%),Z2%(J%):Z%(J%)=Z%(J%)+(Q%
(J%)*4)
610 GCOL0,1:Z2%(J%)=Z2%(J%)+(Q1%(J%)*
2):GCOL0,7:PLOT69,Z%(J%),Z2%(J%):NEXT:S
OUND 0,1,6,4:NEXTP%
```

```
250 IF R%=2 AND R1%=2 landed=TRUE
260 IF TIME>1500 dead=TRUE
270 UNTIL landed OR dead:IF N%<1 dead
=TRUE
280 :
290 G%=G%+1:IF dead GOTO310
300 SC%=SC%+(1500-TIME):FORSO%=1TO10
:SOUND1,-10,200,1:SOUND1,-10,100,1:NEXT
310 L%=L%+4:UNTIL dead:SC%=SC%-100:PR
OCexpl:TIME=0:REPEAT UNTIL TIME>150:*FX
15
320 MODE6:PRINTTAB(5,10)"Your Score w
as :";SC%:TIME=0:REPEAT UNTIL TIME>500:
RUN
330 :
340 DEFPROCscene
350 P%=RND(120)*8:FORA1%=0TO1280STEP8
:IFA1%=P% S%=A2%
360 IF A1%>P% IF A1%<(P%+120) PROCpad
:GOTO390
370 GCOL0,1:MOVE A1%,0:DRAW A1%,A2%:A
2%=A2%+(RND(L%)-(L%/2)):IF A2%<20A2%=20
380 IFA2%>400 A2%=400
390 NEXT
400 ENDPROC
410 :
420 DEFPROCpad
430 GCOL0,2:PLOT69,A1%,S%:PLOT69,A1%,
S%+4:GCOL0,1:MOVE A1%,0:DRAW A1%,S%-4:G
COL0,0:MOVE A1%,S%+8:DRAW A1%,S%+72
440 ENDPROC
450 :
460 DEFPROCchars
470 VDU23,224,0,24,60,255,102,60,66,1
29:VDU23,225,20,126,124,126,254,124,28,
48
480 ENDPROC
490 :
500 DEFPROCarrays
510 ENVELOPE 1,3,0,0,0,0,255,0,126,0,
0,-126,126,126
```
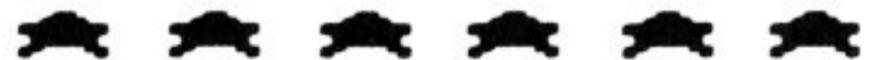


```
620 ENDPROC
630 :
640 DEFPROCmeteors
650 GCOL0,1:FOR W%=1TO(L%/2.5)+RND(5)
:MOVE RND(1000),RND(600)+200:VDU225:NEX
T
660 ENDPROC
670 :
680 DEFPROCstars
690 COLOUR2
700 FOR S%=1 TO RND(50)+50
710 PLOT69,RND(1200)+40,RND(1000)+24
720 NEXT
730 ENDPROC
740 :
750 ON ERROR OFF
760 MODE6:IF ERR=17 END
770 REPORT:PRINT " at line ";ERL
780 END
```
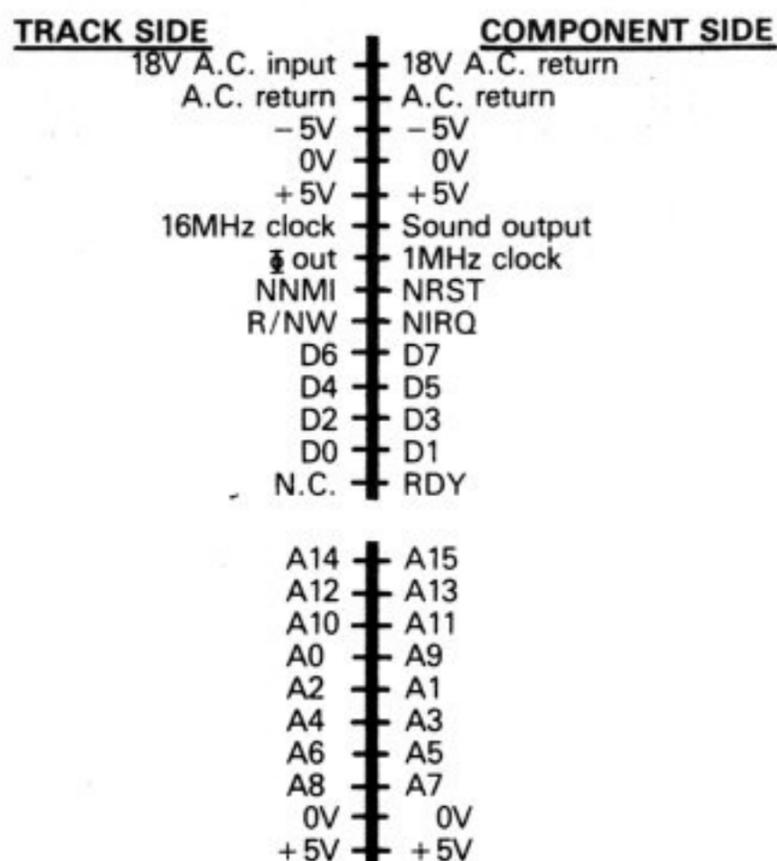
THE ELECTRON'S EDGE CONNECTOR
by Philip Le Grand

This short article is meant as a quick introduction to the connector at the back of the Electron, and is intended for the more technically minded Electron users amongst our readers.

As you can see, the edge connector at the rear of the Electron, has gold plated pins on both sides of the printed circuit board. The functions of each of these pins is given in the diagram opposite, with a brief explanation of each in the following table. Note that the numbers associated with each pin are in accordance with Acorn's numbering convention. Pin 1 is on the component side of the board, at the far left hand side, as you look at the back of the Electron, with the keyboard facing upwards. Pin 2 is directly below pin 1, pin 3 is next to pin 1, and pin 4 is below pin 3 etc. Therefore, all of the odd numbered connections are on the component side, and all of the even numbered connections are underneath the machine.

TRACK SIDE	COMPONENT SIDE
18V A.C. input	18V A.C. return
A.C. return	A.C. return
−5V	−5V
0V	0V
+5V	+5V
16MHz clock	Sound output
ɸ out	1MHz clock
NNMI	NRST
R/NW	NIRQ
D6	D7
D4	D5
D2	D3
D0	D1
N.C.	RDY
A14	A15
A12	A13
A10	A11
A0	A9
A2	A1
A4	A3
A6	A5
A8	A7
0V	0V
+5V	+5V

CONNECTOR	FUNCTION	EXPLANATION
1,2,3,4	18V A.C.INPUT and RETURN	Power supply connection for peripherals to power the Electron.
5-10,47-50	−5V,0V & +5V	Power supplies from the Electron's internal power supply.
11	SOUND OUTPUT	A direct output of the sound, unamplified.
12	16MHz CLOCK	Master clock supply for the video display.
13	1MHz CLOCK	Main CPU timing clock.
14	phi out	Connection to CPU.
15	NRST	Reset line to CPU (active low).
16	NNMI	Non-maskable interrupt (active low).
17	NIRQ	Interrupt request (active low).
18	R/NW	Read (active high), Write (active low).
19-26	D0-D7	Data bus.
27	RDY	CPU READY line (goes high when data and address lines are stable).
28	N.C.	No connection.
29 & 30	keyway	Polarising slot.
31-46	A0-A15	Address bus.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

* COMMANDS IN BASIC

When typing * commands such as *FX or *MOTOR into Basic programs the * command must be the last command on the line. This is because further colons are ignored and the whole of the rest of the line is given to the operating system once the '*' character has been recognised.

3D ROTATION
by James Hastings

This is an extremely interesting program which enables shapes to be drawn in three dimensions and then viewed from any angle. A short sequence showing output from this program was recently included in an edition of the BBC TV programme "Tomorrow's World" as an example of the use of computers in 3D graphics.

This program enables a wire frame shape to be displayed on the screen and then manipulated in various ways from the keyboard. The object to be drawn is defined in the program as a series of data statements, which are easily set up, as they are the X, Y and Z co-ordinates of the "corners" of the object. Once these are entered the program is run, and displays a front view using mode 4 graphics. The object may then be rotated about any axis, or viewed from nearer or further simply by pressing one of 12 keys. These also allow the object to be moved left, right, up and down on the screen. The program will only draw and transform objects made up from straight lines, but there is no apparent limit to the complexity of the "wire-frame" objects generated.
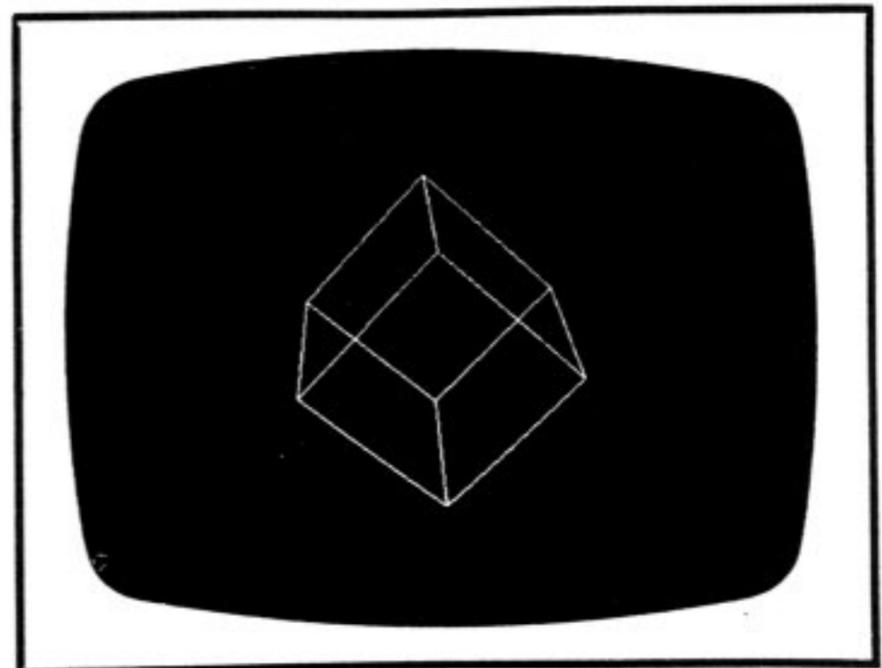
HOW TO USE THE PROGRAM.

The reason that the main program starts from line 1000 is to allow the earlier lines to be used for the data statements to define objects that you may develop later. The data statements on lines 100 to 180 define a 3D cube. Pictures of this are included with this program.

Run the program. You should see a cube, viewed from one side. The cube may now be moved using the keys described in the table below.

1. Cursor keys LEFT and RIGHT rotate around the Y axis.
2. Cursor keys UP and DOWN rotate around the X axis.
3. 'RETURN' and the ':' key (left of RETURN) rotate around the Z axis.
4. DELETE and COPY reduce and increase the size of the object.
5. Keys 'Z' and 'X' move left and right.
6. Keys 'Q' and 'A' move up and down.
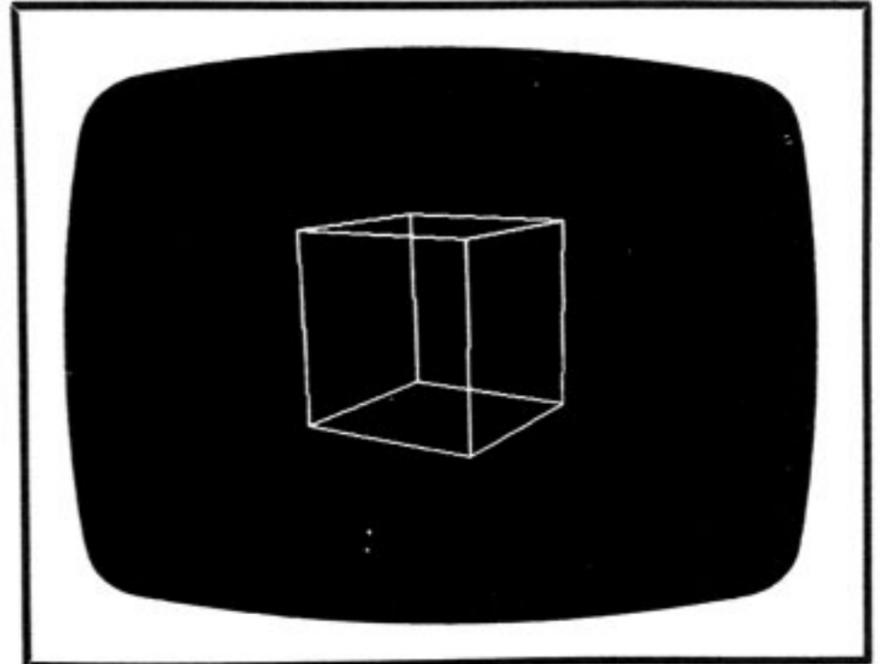
This provides the ability to view the object from any point, including actually from within the object itself, by increasing its size sufficiently. The program always draws all lines of the object concerned as if it were a wire frame.


CREATING A NEW 2D OBJECT

Designing your own shapes is very easy. To start with let us consider how to draw a simple square in 2 dimensions. (The program will equally well draw and display 2D objects in a 3D field of vision).

Take a piece of paper (graph paper is best) and draw a large cross in the middle. This will represent the X and Y axes upon which we will sketch the object, in this case a square. Now draw a large square with the cross at its centre. Number the corners of the square anti-clockwise, from 1 to 4, starting with the top right hand corner as 1.

We now need to work out the X, Y and Z co-ordinates of the square. Let's assume for convenience that the length of each side of the square is 1000 units.

2(-500,500)   1(500,500)

3(-500,-500)   4(500,-500)
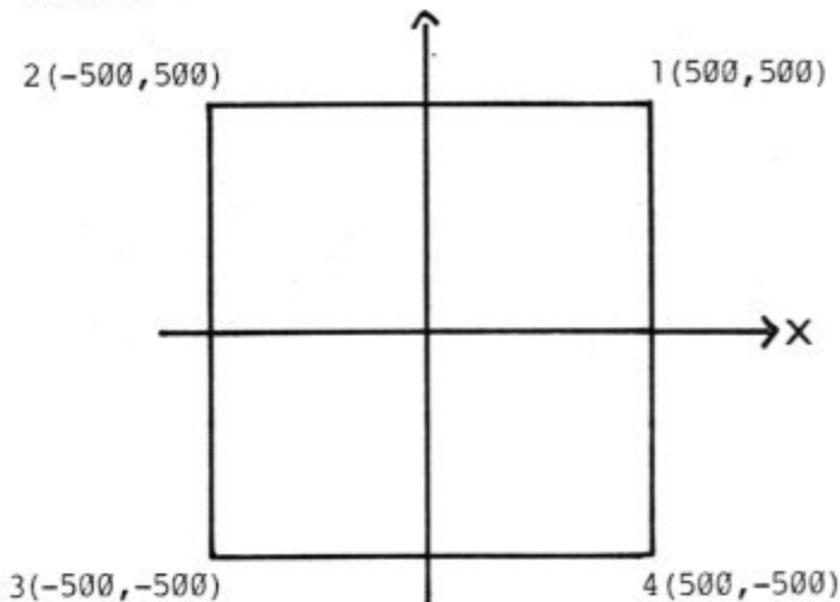
This makes the X and Y co-ordinates of point 1... 500,500. As the object is flat (we are only drawing it in 2D), the Z co-ordinate will be 0.

Thus the co-ordinates of point 1 are 500,500,0 (co-ordinates are always given in the order X, Y, Z). Similarly those of
   point 2 are -500,500,0
   Point 3's are -500,-500,0
   Point 4's are 500,-500,0

We can now compose the data statements for the program. These will be inserted into the program on any line numbers up to 990 and will replace those in the program listed below on lines up to 990. The program requires information in the following format.

1) Number of "corners", followed by the X, Y and Z co-ordinate of each "corner".

2) Number of lines to be drawn, followed by the "corner numbers" which they should join.

This may sound complicated but is in fact very straightforward. The number of corners in our square is obviously 4 and we have worked out the co-ordinates already, so the first data statement will read:
   120 DATA 4,500,500,0,-500,500,0,
       -500,-500,0,500,-500,0

The number of lines is also 4. If you look at your sketch you will see that we have numbered the corners from 1 to 4. The lines of the square go from point 1 to point 2, point 2 to point 3, point 3 to point 4, and point 4 to point 1. This is all that is required for the second data statement, which we can now write:
   130 DATA 4,1,2,2,3,3,4,4,1
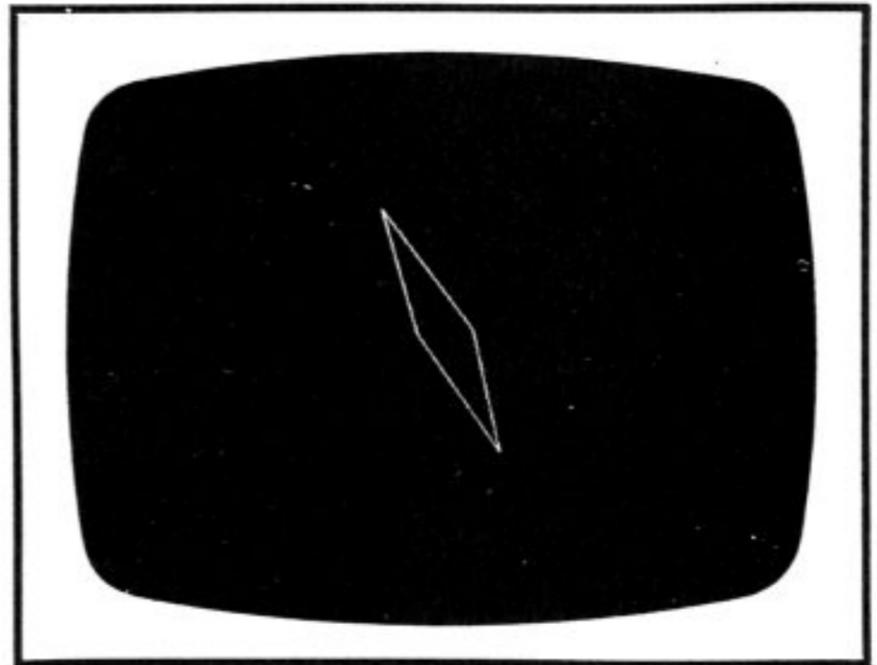
That's all there is to it. These two lines, 120 and 130, should replace the data statements in the program listed below on lines 120, 130 and 170. The actual line numbers are irrelevant as long as they are below 1000. Type them in and run the program, remembering to check that previous data statements, such as line 170 have been removed.

HOW TO DRAW YOUR OWN 3D OBJECT.
To do this we follow exactly the same process as above but now using an object with depth, for example a pyramid with a square base. As this is drawn as an extension to a square, we will be able to use some of the above calculated co-ordinates.

Take the sketch of the square made earlier and consider the Z axis. This is at right angles to the other two axes and can be thought of as extending from above the paper, through the

centre of the cross, to below the piece of paper. When calculating 3D co-ordinates you have the choice of either imagining the points not actually on the paper, or attempting to sketch them using perspective.
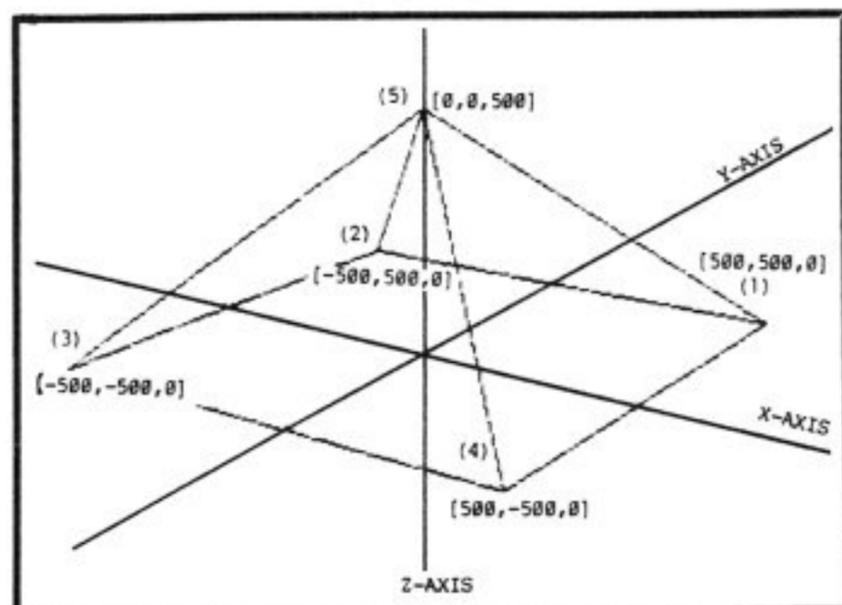
To continue with the pyramid, consider its apex, which we will define, for convenience, at a height of 500 units. This places it exactly over the intersection of the X and Y axes at a height of 500. Consequently its co-ordinates will be 0,0,500. If we number the apex as "corner" 5, you can see that it will require lines to be drawn joining it to "corners" 1,2,3 and 4. We now have an object with 5 "corners" and 8 lines. The data statements can therefore be represented as follows:

```
120 DATA 5,500,500,0,-500,500,0,
    -500,-500,0,500,-500,0,0,0,500
130 DATA 8,1,2,2,3,3,4,4,1,5,1,
    5,2,5,3,5,4
```

Type in these lines instead of all other lines up to 990 and run the program.
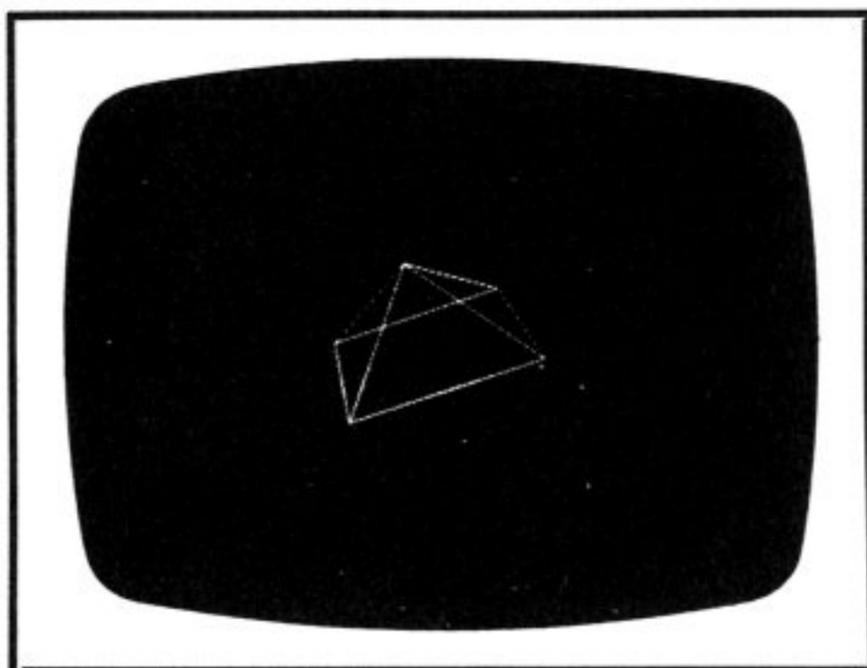
If this doesn't make sense to you, compare them with the data statements calculated above for the square. Refer also to the diagram of the pyramid above, to see how the axes would pass through it. The co-ordinates of the "corners" are also indicated.

FORMULA

For those people interested in the calculations used to perform the rotation, James has given the following formula. To calculate the new X and Y co-ordinates following a rotation about the Z axis by an angle of psi:

$$Xnew=Xold*Cos(psi)-Yold*Sin(psi)$$
$$Ynew=Yold*Cos(psi)+Xold*Sin(psi)$$

and so on for the other axes.

MODIFYING THE PROGRAM

The program runs in mode 4, but you can easily change line 1010 so that it runs in mode 1. The drawings will still be the same size, but to the higher resolution that mode 1 provides. Unfortunately, the Electron is slower in this mode, and transformations of any object will take longer. You can also amend the program so that the drawings are in white against a blue background by adding the line:

```
1015 VDU19,0,4,0,0,0
```

This gives a most attractive display.

```
  10 REM Program  3-D Rotation
  20 REM Version  E1.0
  30 REM Author   J.Hastings
  40 REM ELBUG    Jan/Feb '84
  50 REM Program  subject to copyright
  60 :
 100 REM Points data
 110 :
 120 DATA 8,-500,500,500,500,500,500,5
00,-500,500,-500,-500,500
 130 DATA -500,500,-500,500,500,-500,5
00,-500,-500,-500,-500,-500
 140 :
 150 REM Lines data
 160 :
 170 DATA 12,1,2,2,3,3,4,4,1,1,5,2,6,3
,7,4,8,5,6,6,7,7,8,8,5
 180 :
1000 ON ERROR GOTO 1220
1010 MODE 4
1020 hor%=640
1030 vert%=512
1040 VDU 29,hor%;vert%;:VDU23,1,0;0;0;
0;
1050 *FX 4,1
1060 PROCpoints
```

>>

```
1070 PROClines
1080 distance%=5000
1090 diststep%=500
1100 anglestep=PI/16
1110 movestep=32
1120 hmove%=0
1130 vmove%=0
1140 REPEAT
1150 PROC2D
1160 PROCdraw
1170 PROCupdate
1180 PROCrotate
1190 UNTIL FALSE
1200 END
1210 :
1220 REM Error trap
1230 MODE6
1240 IF ERR<>17 REPORT: PRINT " at lin
e "; ERL
1250 *FX 4,0
1260 END
1270 :
1280 DEF PROCpoints
1290 REM Dimension point arrays and re
ad in points data
1300 READ points%
1310 DIM X(points%),Y(points%),Z(point
s%),X2D(points%),Y2D(points%)
1320 FOR count%=1 TO points%
1330 READ X(count%),Y(count%),Z(count%
)
1340 NEXT count%
1350 ENDPROC
1360 :
1370 DEF PROClines
1380 REM Dimension line arrays and rea
d in lines data
1390 READ lines%
1400 DIM start%(lines%),end%(lines%)
1410 FOR count%=1 TO lines%
1420 READ start%(count%),end%(count%)
1430 NEXT count%
1440 ENDPROC
1450 :
1460 DEF PROC2D
1470 REM Convert to 2-D
1480 FOR count%=1 TO points%
1490 X2D(count%)=X(count%)*2500/(dista
nce%-Z(count%))
1500 Y2D(count%)=Y(count%)*2500/(dista
nce%-Z(count%))
1510 NEXT count%
1520 ENDPROC
1530 :
1540 DEF PROCdraw
1550 CLS
1560 FOR count%=1 TO lines%
1570 MOVE X2D(start%(count%)),Y2D(star
t%(count%))
1580 DRAW X2D(end%(count%)),Y2D(end%(c
ount%))
1590 NEXT count%
1600 ENDPROC
1610 :
1620 DEF PROCupdate
1630 phi=0: theta=0: psi=0: hmove%=0:
vmove%=0
1640 REPEAT
1650 *FX 15,0
1660 key=GET
1670 UNTIL key=13 OR key=58 OR key=88
OR key=90 OR key=65 OR key=81 OR key=13
5 OR key=127 OR key=136 OR key=137 OR k
ey=138 OR key=139
1680 REM Rotate about X axis ?
1690 IF key=139 THEN phi=anglestep
1700 IF key=138 THEN phi=-anglestep
1710 REM Rotate about Y axis ?
1720 IF key=137 THEN theta=anglestep
1730 IF key=136 THEN theta=-anglestep
1740 REM Rotate about Z axis ?
1750 IF key=13 THEN psi=-anglestep
1760 IF key=58 THEN psi=anglestep
1770 REM Change viewing distance ?
1780 IF key=127 THEN distance%=distanc
e%+diststep%
1790 IF key=135 THEN distance%=distanc
e%-diststep%
1800 IF key=90 THEN hmove%=-movestep
1810 IF key=88 THEN hmove%=movestep
1820 IF key=65 THEN vmove%=-movestep
1830 IF key=81 THEN vmove%=movestep
1840 ENDPROC
1850 :
1860 DEF PROCrotate
1870 IF phi<>0 THEN PROCXrotation
1880 IF theta<>0 THEN PROCYrotation
1890 IF psi<>0 THEN PROCZrotation
1900 IF hmove%<>0 OR vmove%<>0 THEN PR
OCshift
1910 ENDPROC
1920 :
1930 DEF PROCXrotation
1940 REM Rotate about X axis
1950 Cosphi=COS(phi): Sinphi=SIN(phi)
1960 FOR count%=1 TO points%
1970 Y=Y(count%): Z=Z(count%)
1980 Y(count%)=Y*Cosphi-Z*Sinphi
1990 Z(count%)=Z*Cosphi+Y*Sinphi
2000 NEXT count%
2010 ENDPROC
2020 :
2030 DEF PROCYrotation
2040 REM Rotate about Y axis
2050 Costheta=COS(theta): Sintheta=SIN
(theta)
2060 FOR count%=1 TO points%
2070 X=X(count%): Z=Z(count%)
2080 X(count%)=X*Costheta-Z*Sintheta
2090 Z(count%)=Z*Costheta+X*Sintheta
2100 NEXT count%
2110 ENDPROC
```

```
2120 :
2130 DEF PROCZrotation
2140 REM Rotate about Z axis
2150 Cospsi=COS(psi): Sinpsi=SIN(psi)
2160 FOR count%=1 TO points%
2170 X=X(count%): Y=Y(count%)
2180 X(count%)=X*Cospsi-Y*Sinpsi
2190 Y(count%)=Y*Cospsi+X*Sinpsi
```

```
2200 NEXT count%
2210 ENDPROC
2220 :
2230 DEF PROCshift
2240 REM MOVE CUBE UP/DOWN/LEFT/RIGHT
```

This program is adapted from a version for the BBC micro first published in BEEBUG Vol.1 No.10. ●

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

BOOTS' CASSETTE RECORDERS

Dear Sir,

My compliments upon what I consider to be an excellent magazine. I much appreciated the index for 'Start Programming' in ELBUG and only wish the 'User Guide' had a similar index.

I was also misled by the connecting instructions for the Boots CR375 cassette recorder, which has replaced the CR325 recommended in ELBUG. Much time was expended before I realised that the load plug had to be inserted in the earpiece socket and not auxilliary.

D.D.Green

Reply:

We are glad the index was appreciated. We are considering whether to produce a similar index for the User Guide.

Other readers have also written about Boots' cassette recorders. The model CR375 will work with the Electron, though it is rather more expensive than the model CR325 which it replaces.

GAMES HIGH SCORES

Dear Sir,

I got a high score of 93856 on Space City (on the ORBIT introductory cassette). I wasn't sure where I was supposed to send my high score to, so I posted it to you hoping that it will get published in the high score table.

M.Diamond

Reply:

Readers will probably have seen that we publish a games high score table for the BBC micro in the supplement that comes with your ELBUG magazine. We shall be publishing a separate table for Electron games starting next month,

provided that there is enough interest and you send us your high scores (to the Editorial address please).

WHAT IS MODE 7?

Dear Sir,

I am delighted to have received the first copy of your magazine, which I find very instructive and informative.

As a novice to computing I would appreciate your guidance regarding the 'Munchman' program (published in the first issue of ELBUG) where line 2160 states:

2160 MODE 7:*FX12

As the Electron does not have mode 7, could you tell me which mode I should use?

I have tried previously to modify a BBC program containing mode 7 by using mode 0 and mode 3, but each time was informed that I had a 'Bad program'.

G.Gallagher

Reply:

Mode 7 is an additional mode available on the BBC micro, and is the default mode for that machine in the same way that mode 6 is the default mode for the Electron. The Munchman program, which was converted from a version for the BBC micro, should have specified mode 6 at line 2160, although the program as listed will still work correctly as the Electron always defaults to mode 6 if mode 7 is selected.

Generally speaking, however, if you want to convert BBC programs using mode 7 to run on the Electron, you will also have to remove all the special control characters (they all have ASCII codes greater than 128), which control colour and other special effects. ●

POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG POSTBAG

NEW SOFTWARE FROM PROGRAM POWER
reviewed by Alan Webster

Last month we reported that Program Power had just launched a range of software for the Electron.

Program Power have already established themselves as a major supplier of quality computer games for the BBC micro and they are rapidly building up a catalogue of equally good software for the Electron.

The programs are all attractively packaged, they all load reliably and they are readily available in the shops, including chains such as W.H.Smith. The first launch for the Electron contained 11 titles of which we have selected six for review. A further 10 programs have just been announced for the Electron, enabling Program Power to offer a really wide choice of computer games for the new Electron user.

Note: These and other Program Power software are widely available through computer shops and dealers as well as many branches of Boots, John Menzies and W.H.Smith.

```
Name    : Positron
Supplier: Program Power
Price   : £6.95
Rating  : ***
```

This is a very fast Space Invaders type of game, requiring superior reflexes if you are to prevent the aliens from landing. The game is not that original, as computer games go, but worth buying if you like the adrenalin draining 'blast and zap 'em' type of game.

```
Name    : Swoop
Supplier: Program Power
Price   : £7.95
Rating  : ***
```

Swoop is one of the few games that happen to run much better on an Electron than in its original version for the BBC micro. It is a variation on Galaxians, with the birds laying explosive eggs at the bottom of the screen. The game, although nice and fast, is rather repetitive.

```
Name    : Cybertron Mission
Supplier: Program Power
Price   : £7.95
Rating  : ****
```

Cybertron Mission is an animated adventure in which you must collect various objects and return them to safes. The smooth and attractive graphics provide an enjoyable game. There are 16 rooms on each level, and up to four different levels before the game restarts. As the game progresses, you meet more and more enemies, and your task becomes increasingly harder. Beware of the ghost that makes an occasional appearance! This is one of the better games available for the Electron.

Name   : Moonraider
Supplier: Program Power
Price   : £7.95
Rating : ****

   Moonraider is a very good
implementation of the arcade game
Scramble. You have to navigate a
spacecraft through a series of caverns
and tunnels, bombing fuel dumps and
enemy craft to gain points. The game
has six stages with varying speed, any
of which can be chosen at the start.
The graphics are excellent with very
smooth movement resulting in an
interesting and challenging game.
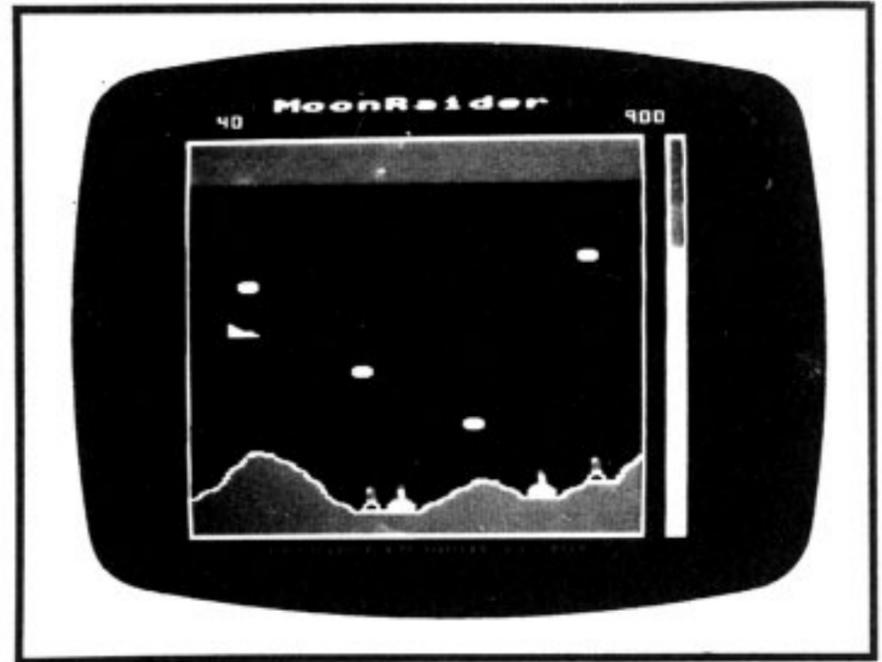
Name   : Killer Gorilla
Supplier: Program Power
Price   : £7.95
Rating : ***
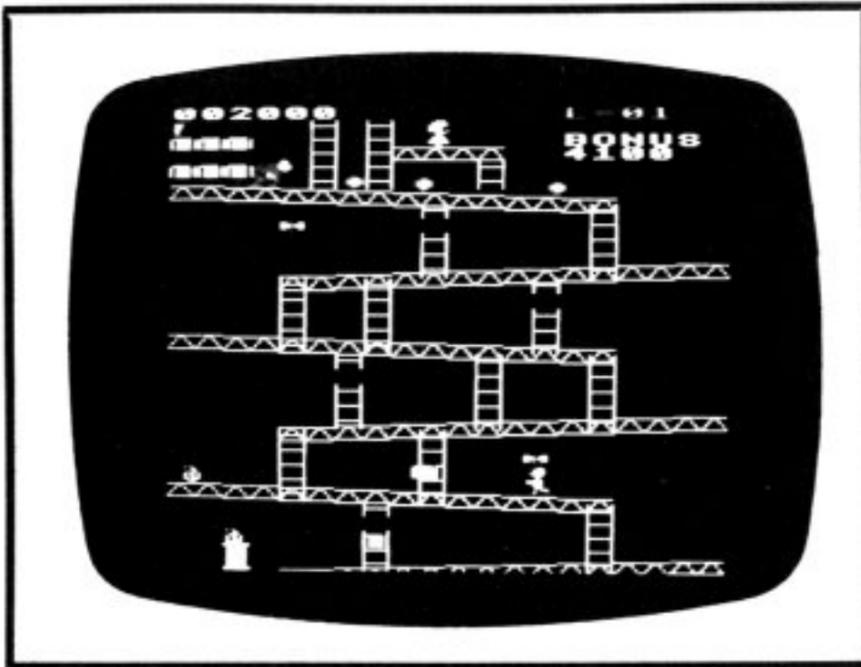
   Graphically this game compares well
with the arcade game, but the action is
slower. You take the part of the young
lad, trying to rescue the maiden from
the grasp of the gorilla. To reach the
maiden, you must climb ladders and
girders, avoiding fireballs and
barrels, until you reach the top. There
are four different screens, with lifts
and giant springs on the third screen
to cause you extra problems. If it
wasn't for the lack of speed, this game
would be thoroughly addictive.

Name   : Felix in the Factory
Supplier: Program Power
Price   : £7.95
Rating : *****

   Felix is a factory worker who is
left on his own one day. His job is to
keep the generator from running out of
oil. He is armed only with a spear to
kill the monsters and help make his
task of re-fueling the generator
easier. The generator's consumption
requires frequent journeys to get the
oil can. If the generator does run out,
then that spells trouble for Felix!
   This is probably the best game
available so far for the Electron, and
well worth the cost.

ELECTRON GRAPHICS (Part 3)
by David Graham

In the last issue we looked at the animation of simple characters. The characters involved were of a single colour only, but it is possible to apply similar techniques to move multicoloured characters around the screen. The effects obtained in this way can be quite striking, and will generally enhance the visual content of games and other programs.

GENERATING A MULTI-COLOURED CHARACTER

Broadly speaking, multicoloured characters are produced by defining a separate character for each colour invloved, and then superimposing one over the other in the required colour. Before the superimposition is attempted, the call VDU 5 must be executed. This prevents the second character from overwriting the first. In fact VDU 5 has a number of effects, as indicated in table 1.

---

* Characters are plotted at the graphics cursor position, and the text cursor is turned off. This enables text or user defined characters to be placed anywhere on the screen by the MOVE command. NOTE : TAB does not function when the text and graphics cursor are joined together, and so MOVE must be used.
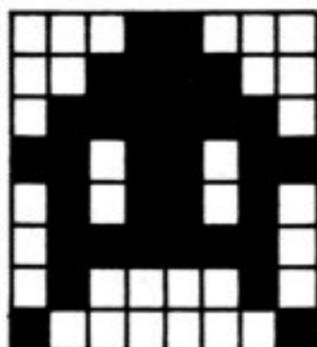* Text and graphics are only produced within the graphics window.
* The colour used is obtained from the last GCOL command, not the COLOUR command.
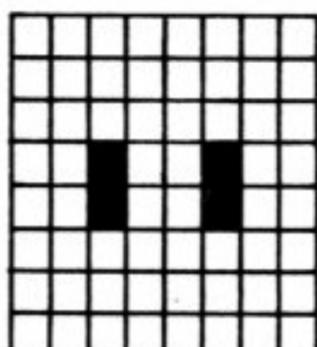* Only the foreground colour is plotted, not the background colour, allowing characters to be overprinted.
* VDU 5 is cancelled with VDU 4.

Table 1

Program 1 uses superimposition under VDU 5 to produce a multicoloured space invader. Line 100 calls PROCdef which defines two characters, one for the invader, and one for its eyes. The two characters were created using the definer program on the ELBUG

```
10 REM PROGRAM   COLOURED CHRS 1
20 REM AUTHOR    D.Graham
30 REM Version   E.10
40 REM ELBUG     Jan/Feb '84
50 REM Program   subject to copyright
60 :
100 PROCdef
110 MODE2
120 VDU5
130 GCOL0,1:MOVE 200,200:VDU224
140 GCOL0,12:MOVE 200,200:VDU225
150 END
160 :
1000 DEFPROCdef
1010 VDU23,224,24,60,126,219,90,126,66
,129
1020 VDU23,225,0,0,0,36,36,0,0,0
1030 ENDPROC
```

---

introductory cassette. Line 110 selects Mode 2, and 120 calls VDU 5 so that we can superimpose the two characters on top of each other. Note that as indicated above, TAB does not function under VDU 5, and is replaced with the MOVE statement. The parameters for MOVE are the x and y coordinates of the cursor. The graphics screen on the Electron is 1280 points across by 1024 points vertically. The GCOL statement in lines 140 and 150 is used to select the two colours - in this case red, and flashing yellow - blue.

Space Invader.

Eyes for Space Invader.

MOVING A MULTICOLOURED CHARACTER

If we wish to move a multicoloured character, we can adopt a similar technique to that used for moving a pair of monochrome characters. That is to say, assign the characters to be printed, to a string variable, and print the string in successive positions across the screen.

Program two achieves this. If you RUN this program, you will see a static multicoloured invader printed in the top half of the screen, and when the space bar is pressed, a second invader will travel across the screen.

In the animation examples given last month, each time that we printed a new image of a character, its previous image was deleted from the screen by overprinting with a space character. A similar technique could be used here, though the overprinting would have to be done with a block character defined to the background colour. But there is a much more useful way of deleting a multicoloured character. This involves the use of so called exclusive-or plotting.

If we use exclusive-or plotting to print our character, then printing it a second time in the same position will completely erase it. The great advantage of this approach is that it occurs in such a way as to leave any background completly unchanged; whereas deleting the character in the normal way would leave a trial of blanks across any scenery that the character passed across.

Exclusive-or plotting is invoked by the GCOL statement (see User Guide page 97). The statement takes the form GCOL 3,n where the 3 establishes exclusive-or plotting, and the n determines the logical colour plotted. For example, try the following statements:

```
MODE 2
GCOL 3,5
VDU 5
```

Now press Return a few times. You should see the prompt appear in magenta (colour 5). Now press say the letter 'L'. This will again appear in magenta. If you now execute CRTL-H (ie hold down the CTRL key and press H), this will

backspace the cursor, though the cursor is not visible under VDU 5. Now when you press the letter 'L', the previous one will be erased. This is because the two letters were exclusive-or plotted over the top of each other.

Program 2 uses this technique to erase earlier images of the invader as it moves across the screen. PROCdef defines the two parts of the invader as in program 1. PROCchrs (defined in lines 1050 to 1090) first creates the equivalent of two GCOL statements. RED$ is assigned the sequence 18,3,1 which has the same effect as VDU 18,3,1 which (see User Guide page 107) is equivalent of GCOL 3,1. This invokes exclusive-or plotting in red (for the invaders body). YELLOW$ is equivalent to GCOL 3,12 to exclusive-or plot flashing yellow for its eyes. Line 1080 then

```
  10 REM Program   COLOURED CHRS2
  20 REM Author    D.Graham
  30 REM Version   E1.0
  40 REM ELBUG     Jan/Feb '84
  50 REM Program   subject to copyright
  60 :
 100 PROCdef
 110 PROCchrs
 120 MODE2
 130 VDU5
 140 :
 150 MOVE640,864:PRINTINVADER$
 160 GCOL0,3:PRINT'"  Press space bar"
 170 :
 180 :
 190 REPEAT
 200 REPEAT UNTIL GET$=" "
 210 MOVE0,512
 220 PRINTINVADER$;CHR$8;
 230 FORloop%=0 TO 18
 240 PRINTINVADER$;INVADER$;
 250 VDU8
 260 NEXT
 270 PRINTINVADER$
 280 UNTIL FALSE
 290 :
1000 DEFPROCdef
1010 VDU23,224,24,60,126,219,90,126,66
,129
1020 VDU23,225,0,0,0,36,36,0,0,0
1030 ENDPROC
1040 :
1050 DEFPROCchrs
1060 RED$=CHR$18+CHR$3+CHR$1
1070 YELLOW$=CHR$18+CHR$3+CHR$12
1080 INVADER$=RED$+CHR$224+CHR$8+YELLO
W$+CHR$225
1090 ENDPROC
```

combines the GCOL codes with the actual character definitions (CHR$ 224 and 225). CHR$ 8 performs a backspace to achieve superimposition.

The vital line in the invader moving routine (lines 190 to 280) is 240. This prints INVADER$ twice in adjacent positions. The left-most printing serves to delete the previously printed image. When using this technique, the very first and very last invader positions must be catered for separately. This is achieved in lines 220 and 270. You may like to try omitting them to see the effect.

As mentioned above, the use of exclusive-or plotting has an important consequence. Backgrounds are not deleted when characters are erased. To see this in action, add the following lines to program 2.

```
 170 PROCbox
1100 DEFPROCbox
1120 GCOL0,6
1130 PLOT4,540,612:PLOT4,540,412
1140 PLOT85,740,612:PLOT85,740,412
1150 ENDPROC
```
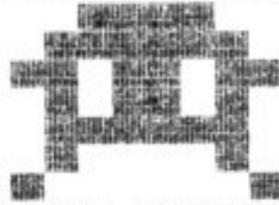
Their function is to draw a filled box in the middle of the screen in the path of the invader. If you run the program, you will see that the invader becomes faint as it passes the box, only to emerge leaving the box, and itself completely intact.

In practice the effect of over-plotting colours under GCOL 3 depends on the logical colour numbers associated with them, and it is possible to organise things in such a way that characters can appear to move in front of some objects, and behind others. We will take a closer look at this in a future issue.

In the meantime, there is just one further technique which may be used to improve the look of the moving multicoloured character. This concerns the jerkiness of its movements. It is possible to make the invader move more smoothly by using smaller increments in between each printing.

However, this slows down the printing process to a point where each character does not move quickly enough

across the screen. One way of almost doubling the speed of the Electron in such cases is to use Mode 5 rather than Mode 2; but a consequnce of this is that you may only display 4 colours simultaneously (including background); though the particular colours displayed may be chosen by the user with the VDU 19 command (see User Guide page 107).

Program 3 uses these techniques to move an invader across the screen, more smoothly than in program two. Mode 5 is chosen, and the invader is plotted at closer intervals than in program 2. The FOR loop in line 230 increments the print position by 32 graphics units (a screen is 1280x1024 units). Line 260 backsteps by the same amount to overprint, and erase, the previous image.

If you try this program you will see that the invader still sometimes takes on a 'ragged' appearance. This is due to the timing of the screen scan of the VDU. At machine code speeds it is possible to eliminate this with FX call 19 (ie *FX19). This synchronises printing operations with screen scan.

In the next issue we will take a look at the other method of producing colour graphics on an Electron, using the DRAW and PLOT statements.

```
 10 REM Program COLOURED CHRS3
 20 REM Author  D.Graham
 30 REM Version E1.0
 40 REM Elbug   Jan/Feb '84
 50 REM Program subject to copyright
 60 :
100 PROCdef
110 PROCchrs
120 MODE5
130 VDU5
140 :
150 MOVE640,864:PRINTINVADER$
160 GCOL0,3:PRINT'" Press space bar"
170 PROCbox
180 :
190 REPEAT
200 REPEAT UNTIL GET$=" "
210 MOVE0,512
220 PRINTINVADER$;CHR$8;
230 FORloop%=0 TO 1152 STEP 32
240 MOVEloop%,512
250 PRINTINVADER$;
260 PLOT0,-32,0
270 PRINTINVADER$;
```

```
 280 NEXT
 290 PLOT0,-64,0
 300 PRINTINVADER$
 310 UNTIL FALSE
 320 :
1000 DEFPROCdef
1010 VDU23,224,24,60,126,219,90,126,66
,129
1020 VDU23,225,0,0,0,36,36,0,0,0
1030 ENDPROC
1040 :
1050 DEFPROCchrs
```

```
1060 RED$=CHR$18+CHR$3+CHR$1
1070 YELLOW$=CHR$18+CHR$3+CHR$2
1080 INVADER$=RED$+CHR$224+CHR$8+YELLO
W$+CHR$225
1090 ENDPROC
1100 :
1110 DEFPROCbox
1120 GCOL0,1
1130 PLOT4,540,612:PLOT4,540,412
1140 PLOT85,740,612:PLOT85,740,412
1150 ENDPROC
```

NEW CHARACTER SETS FOR MODES 2 & 5
by Alan Baker

This program generates a completely new character set of 26 upper case letters for use in modes 2 and 5. The styling is different, but more importantly they are designed to give 26, rather than 20 characters per line, rendering much more readable and ellegant text in these two useful graphics modes.

The routine used to generate this new character font is quite straightforward. The 26 letters of the alphabet are redefined on a 6 by 5 pixel matrix using VDU 23 commands and then printed anywhere on the screen using the graphics cursor, backspacing two pixels before printing the next letter. The normal character set is not lost, and may be used concurrently.

The technique employed here could be extended to redefine further characters; though special *FX calls will need to be made to expand the memory allocated to character definitions. This is achieved using *FX20 (see page 282 of the User Guide).

If you run the program, after typing it into your Electron, you will be asked to specify a position on the screen in graphics coordinates (x must be between 0 and 1279, and y must be between 0 and 1023 with 0,0 at the bottom left hand corner of the screen), and the character string to be displayed. The text is then displayed on the screen using the newly defined character set, and the program then terminates.

PROGRAM NOTES
The procedure PROCinitialise defines the 26 new characters, any of which may
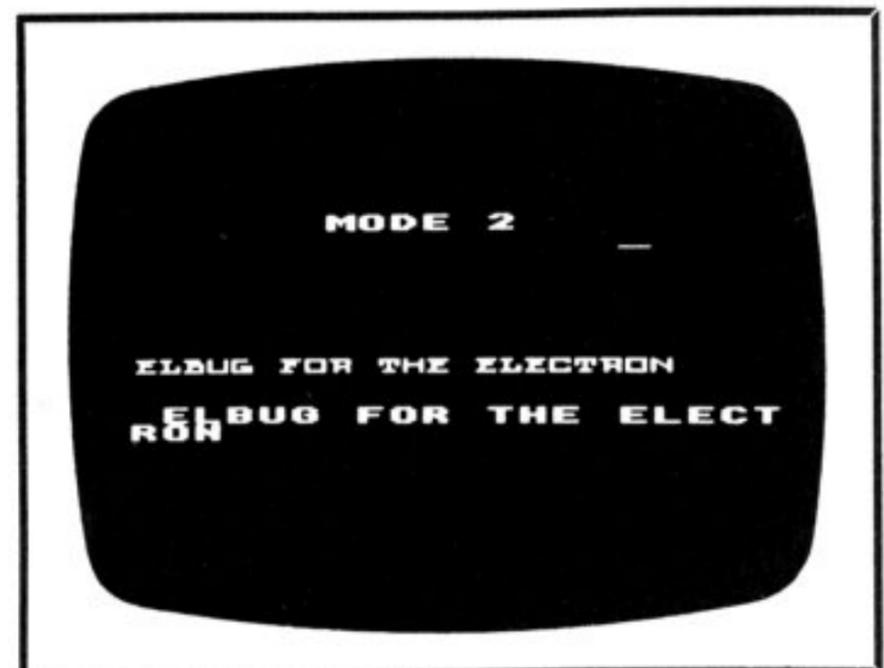
then be displayed on the screen by using the procedure PROCalpha with parameters xcoord, ycoord and text$. When this procedure is called, the parameters xcoord and ycoord point to the top left position of the first character from which the text in text$ will be displayed. These two parameters must be given in graphics coordinates, as described above. The two procedures could be readily incorporated in any of your own programs.

```
10 REM PROGRAM NEW CHARACTER SET
20 REM AUTHOR ALAN BAKER
30 REM VERSION E1.0
40 REM ELBUG JAN/FEB 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT.
60 ON ERROR GOTO 590
70 :
```

```
  80 MODE 6
  90 PROCinitialise
 100 INPUTTAB(4,4)"X-COORDINATE (0-127
9)"xcoord
 110 INPUTTAB(4,6)"Y-COORDINATE (0-102
3)"ycoord
 120 INPUTTAB(4,8)"ENTER WORDS TO BE P
RINTED"
 130 INPUTTAB(4,10),text$
 140 MODE 5
 150 PROCalpha(xcoord,ycoord,text$)
 160 END
 170 :
 180 DEF PROCinitialise
 190 VDU23,224,&70,&88,&88,&F8,&88,&88
,0;:REM A
 200 VDU23,225,&F0,&50,&78,&48,&48,&F8
,0;:REM B
 210 VDU23,226,&F8,&88,&80,&80,&88,&F8
,0;:REM C
 220 VDU23,227,&F8,&48,&48,&48,&48,&F8
,0;:REM D
 230 VDU23,228,&F8,&40,&70,&40,&40,&F8
,0;:REM E
 240 VDU23,229,&F8,&40,&70,&40,&40,&E0
,0;:REM F
 250 VDU23,230,&F0,&80,&80,&B8,&88,&F8
,0;:REM G
 260 VDU23,231,&88,&88,&F8,&88,&88,&88
,0;:REM H
 270 VDU23,232,&70,&20,&20,&20,&20,&70
,0;:REM I
 280 VDU23,233,&78,&10,&10,&10,&90,&F0
,0;:REM J
 290 VDU23,234,&B0,&A0,&A0,&F8,&88,&88
,0;:REM K
 300 VDU23,235,&E0,&40,&40,&48,&48,&F8
,0;:REM L
 310 VDU23,236,&F8,&A8,&A8,&A8,&A8,&A8
,0;:REM M
 320 VDU23,237,&88,&C8,&A8,&98,&88,&88
,0;:REM N
 330 VDU23,238,&F8,&88,&88,&88,&88,&F8
,0;:REM O
 340 VDU23,239,&F8,&48,&48,&78,&40,&40
,0;:REM P
 350 VDU23,240,&F8,&88,&88,&A8,&90,&E8
,0;:REM Q
 360 VDU23,241,&F8,&48,&48,&70,&48,&48
,0;:REM R
 370 VDU23,242,&F8,&80,&F8,&08,&88,&F8
,0;:REM S
 380 VDU23,243,&F8,&A8,&A8,&20,&20,&70
,0;:REM T
 390 VDU23,244,&88,&88,&88,&88,&88,&F8
,0;:REM U
 400 VDU23,245,&88,&88,&88,&50,&50,&20
,0;:REM V
 410 VDU23,246,&88,&88,&A8,&A8,&A8,&50
,0;:REM W
 420 VDU23,247,&88,&50,&20,&50,&88,&88
,0;:REM X
 430 VDU23,248,&88,&88,&70,&20,&20,&20
,0;:REM Y
 440 VDU23,249,&F8,&10,&20,&40,&80,&F8
,0;:REM Z
 450 ENDPROC
 460 :
 470 DEF PROCalpha(xcoord,ycoord,text$
)
 480 VDU5
 490 M%=1
 500 FOR N%=xcoord TO xcoord+(LEN(text
$)*48-48) STEP48
 510 m$=MID$(text$,M%,1)
 520 MOVE N%,ycoord
 530 IF m$<>" " THEN PRINT CHR$(159+AS
C(m$))
 540 M%=M%+1
 550 NEXT
 560 VDU4
 570 ENDPROC
 580 :
 590 ON ERROR OFF:MODE 6
 600 IF ERR<>17 REPORT:PRINT" at line
";ERL
 610 END
```

This program is adapted from a version for the BBC micro that was first published in BEEBUG Vol.1 No.7.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

UNINTERRUPTABLE PROGRAMS - G.Middleton

At last it is possible to write programs which can not be stopped with the Break or Escape keys. Use the following lines at the head of your programs:

```
10 ON ERROR RUN
20 *FX247,76
30 *FX248,20
40 *FX249,189
```

These commands alter the locations looked at whenever the Break key is pressed, so that the machine jumps straight back into Basic and re-runs your program.
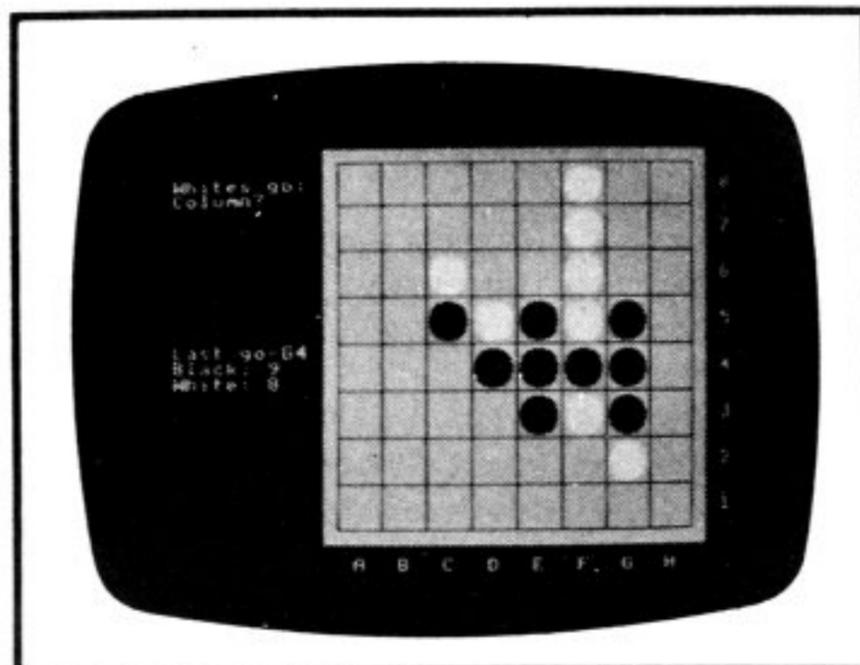
REVERSI
by J. Webb

The game of REVERSI (sometimes known as OTHELLO) is a fascinating game played on a conventional Draughts board, with two players. This version is well written and allows you to play either against the computer or against another player, using the computer to record the moves. You can also choose one of two levels of skill when playing against the computer. Even at the lower level, the program plays a pretty good game.

In this version of REVERSI the first two counters are placed on the board for you as these are fixed moves. At the start, you have the choice of colour and whether to play first or second. From then on you must always place one of your counters on the board in such a way that a line of your opponent's counters then has one of your counters at both ends. The surrounded counters are now reversed so that they change to the colour of your own pieces. Of course, you may well be able to place a single counter that captures more than a just a single row of your opponent's pieces.

If on your turn, you are not able to move to a legal position, then you must pass, by pressing P. The game ends when nobody can place any more counters on the board (usually when the board is full), and the winner is the one who has the most counters. The program maintains the current score on the screen throughout the game.

This is one of those games where it is much more difficult to explain the rules than it is to play the game itself. Once you start playing you will

quickly understand how the game works. Even so REVERSI is a game which requires considerable skill to play. In this version you are able to choose one of two playing levels at the start of each game. Beginners will find even level one quite challenging. We think you will enjoy playing this well constructed version of a popular game.


```
 10 REM Program REVERSI
 20 REM Version E1.1
 30 REM Author J.Webb
 40 REM ELBUG JAN/FEB 84
 50 REM Program subject to Copyright
 60 :
100 MODE1
110 ON ERROR GOTO 900
120 DIMposition%(100)
130 PROCcircles
140 PROCinstructions
150 REPEAT:PROCinit
160 PROCsetupboard
170 PROCfindplayers
180 REPEAT
190 REPEAT:PROCgo
200 PROCcompute
210 PROClookatgo
220 UNTILillegal%=2
230 PROCupdatescore
240 turn%=turn%+1:UNTILturn%=60
```

```
 250 PROCwin:UNTILwait$<>CHR$(32)
 260 *FX15,1
 270 MODE6:END
 280 :
 900  ON ERROR OFF:MODE 6:REPORT:PRINT
" at line ";ERL:END
 910 :
1000 DEFPROCinit
1010 FORpos%=1TO100:position%(pos%)=0:
NEXT
1020 turn%=0:value%=1:comp%=0:go%=0:ed
ge%=0:miss%=1
1030 ENDPROC
1040 DEFPROCsetupboard
1050 CLS
1060 VDU28,0,31,10,0:VDU23,1,0;0;0;0;
1070 VDU4
1080 VDU19,2,0,0,0,0:VDU5:GCOL0,2
1090 MOVE352,960:MOVE352,128:PLOT85,11
84,960:PLOT85,1184,128
1100 MOVE416,96:GCOL0,3:PRINT"A  B  C
D  E  F  G  H":GCOL0,2
1110 FORY%=160TO928STEP96
1120 GCOL0,0:MOVE384,Y%:DRAW1152,Y%:MO
VE1216,Y%+64:GCOL0,3:
1130 IFY%=928THEN1140ELSEPRINT;(Y%-160
)/96+1
1140 NEXT
1150 GCOL0,0:FORX%=384TO1152STEP96
1160 MOVEX%,160:DRAWX%,928:NEXT
1170 VDU19,2,2,0,0,0
1180 *FX15,1
1190 RESTORE2910
1200 FORposition%=1TO4:READpos%,val%,C
%:position%(pos%)=val%:GCOL0,C%:PROCfin
drowandcol(pos%):PROCcounter:NEXT
1210 ENDPROC
1220 DEFPROCfindplayers
1230 VDU4:PRINTTAB(0,10)"Do you wantto
play thecomputer?"'"(Y or N)"
1240 *FX15
1250 REPEAT:wait$=GET$:UNTILINSTR("YNy
n",wait$)>0:SOUND&11,2,40,4:PROCclear(1
0)
1260 IFwait$<>"Y"ANDwait$<>"y"THENENDP
ROC
1270 PRINTTAB(0,10)"Do you wanta  hard
 oreasy game?"'" (H or E)"
1280 *FX15
1290 REPEAT:wait$=GET$:UNTILINSTR("HhE
e",wait$)>0:SOUND&11,2,40,4:PROCclear(1
0):IFwait$="H"ORwait$="h"THENmiss%=0
1300 go%=1:PRINTTAB(0,10)"Do you wantt
o go first(Y or N)?"
1310 *FX15
1320 REPEAT:wait$=GET$:UNTILINSTR("YNy
n",wait$)>0:SOUND&11,2,40,4:PROCclear(1
0)
1330 IFwait$="Y"ORwait$="y"THENENDPROC
1340 go%=2:ENDPROC
```

```
1350 DEFPROCcircles
1360 VDU23,224,0,0,0,3,7,15,31,31
1370 VDU23,225,0,0,126,255,255,255,255
,255
1380 VDU23,226,0,0,0,192,224,240,248,2
48
1390 VDU23,227,31,63,63,63,63,63,63,31
1400 VDU23,228,255,255,255,255,255,255
,255,255
1410 VDU23,229,248,252,252,252,252,252
,252,248
1420 VDU23,230,31,31,15,7,3,0,0,0
1430 VDU23,231,255,255,255,255,255,126
,0,0
1440 VDU23,232,248,248,240,224,192,0,0
,0
1450 C1$=CHR$224+CHR$225+CHR$226:C2$=C
HR$227+CHR$228+CHR$229:C3$=CHR$230+CHR$
231+CHR$232
1460 circle$=C1$+CHR$8+CHR$8+CHR$8+CHR
$10+C2$+CHR$8+CHR$8+CHR$8+CHR$10+C3$
1470 ENVELOPE1,4,0,0,0,0,0,0,121,-10,-
5,-2,120,120:ENVELOPE2,16,4,-8,-4,16,16
,32,64,64,-64,-64,128,0
1480 ENDPROC
1490 DEFPROCclear(tab%):PRINTTAB(0,tab
%)SPC(42):ENDPROC
1500 DEFPROCgo
1510 VDU4:pass%=0:COLOUR2
1520 IFgo%<2THEN1550
1530 PROCclear(4):go%=1:PRINTTAB(0,4)"
Computing":IFvalue%=1THENvalue%=2:GCOL0
,0:ELSEGCOL0,3:value%=1
1540 ENDPROC
1550 IFvalue%=1THENPRINTTAB(0,4)"Black
's go:":value%=2:GCOL0,0:ELSEPRINTTAB(0
,4)"White's go:":GCOL0,3:value%=1
1560 IFgo%>0THENgo%=2
1570 REPEAT:REPEAT:PROCclear(5):PRINTT
AB(0,5)"Column? ";
1580 col%=GETAND223:UNTIL(col%>64ANDco
l%<73)ORcol%=80:PRINTCHR$(col%):col%=co
l%-64:IFcol%=16THEN1610
1590 PRINTTAB(0,6)"Row? ";
1600 row%=GET:row%=row%-48:PRINT;row%
1610 UNTIL(row%>0ANDrow%<9)ORcol%=16
1620 VDU5:pos%=col%*10+row%+1
1630 ENDPROC
1640 DEFPROCcounter
1650 SOUND&11,1,94,12
1660 VDU5:MOVE291+96*col%,159+96*row%:
PRINT;circle$
1670 TIME=0:REPEATUNTILTIME>50
1680 ENDPROC
1690 DEFPROCillegal
1700 VDU4:SOUND&10,2,70,25
1710 PROCclear(5)
1720 PRINTTAB(0,5)"Illegal"'" move"
1730 IFvalue%=1THENvalue%=2ELSEvalue%=1
1740 IFgo%=2THENgo%=1          ➤➤
```

```
1750 TIME=0:REPEATUNTILTIME>200
1760 ENDPROC
1770 DEFPROCweigh
1780 weight%=0:IFposition%(pos%)>0ENDP
ROC
1790 RESTORE2920
1800 FORB%=1TO8:READC%:A%=pos%
1810 IFedge%<>0THENC%=edge%:B%=8
1820 REPEAT:IFA%=pos%THEN1870
1830 IFposition%(A%)=0THENA%=199:GOTO1
870
1840 IFposition%(A%)=value%ANDposition
%(A%-C%)<>value%ANDposition%(A%-C%)>0TH
ENillegal%=1:weight%=weight%+ABS((pos%-
A%)/C%)-1:PROCcheckedge:PROCedge:A%=199
:GOTO1860
1850 IFposition%(A%)=value%ANDposition
%(A%-C%)=0THENA%=199
1860 IFweight%>bestgo%THENbestgo%=weig
ht%:compgo%=pos%
1870 A%=A%+C%:UNTILA%>100ORA%<1
1880 IFillegal%=1ANDcomp%<>2ANDcol%<>1
6THENA%=pos%:PROCchangecolour
1890 NEXT:ENDPROC
1900 DEFPROClookatgo
1910 black%=0:white%=0:IFcol%=16ANDpas
s%=0THEN1960
1920 IFpass%=1THENillegal%=2:turn%=tur
n%-1:GOTO1960
1930 illegal%=0
1940 IFposition%(pos%)>0THENPROCillega
l:ENDPROC
1950 PROCweigh
1960 FORA%=1TO100:IFposition%(A%)=3THE
Nposition%(A%)=value%
1970 IFposition%(A%)=2THENblack%=black
%+1
1980 IFposition%(A%)=1THENwhite%=white
%+1
1990 NEXT
2000 IFcol%=16ANDillegal%=0THENillegal
%=2:turn%=turn%-1
2010 IFillegal%<>2THENPROCillegal
2020 IFblack%=0ORwhite%=0THENturn%=59
2030 ENDPROC
2040 DEFPROCfindrowandcol(posit%)
2050 col%=posit%DIV10
2060 row%=(posit%-1)MOD10
2070 ENDPROC
2080 DEFPROCchangecolour
2090 REPEAT:PROCfindrowandcol(A%)
2100 IFposition%(A%)<>3THENPROCcounter
2110 position%(A%)=3:A%=A%+C%:UNTILpos
ition%(A%)=value%:illegal%=2
2120 ENDPROC
2130 DEFPROCupdatescore
2140 VDU4:PRINTTAB(0,15)"Last go-";
2150 IFcol%=16THENPRINT"P ":K%=K%+1:EL
SEPROCfindrowandcol(pos%):PRINTCHR$(col
%+64);row%:K%=0
```

```
2160 col%=0:PRINTTAB(0,16)"Black: ";bl
ack%;" "
2170 PRINTTAB(0,17)"White: ";white%;"
":IFK%=2THENturn%=59
2180 ENDPROC
2190 DEFPROCinstructions
2200 VDU19,1,2,0,0,0
2210 VDU23,1,0;0;0;0;:SOUND&11,2,80,255
2220 COLOUR1:PRINTTAB(14,3);"REVERSI"
2230 COLOUR2:PRINT''"  The object of
this game  is to have"';"the  most  dis
cs displayed  when  the"';"board is fu
ll."
2240 PRINT"  After a disc is played, a
ll of your "';"opponents discs trapped
between the"';"disc just played and ano
ther disc of"';"yours are reversed to y
our colour."
2250 PRINT"  If, on your turn, you  do
n't  have a"';"legal position to move t
o then press 'P'to pass."
2260 PRINT"  You  can  play  against
either  the"';"computer or another play
er."
2270 COLOUR3:PRINT''''TAB(6);"Press 'sp
ace bar' to play."
2280 *FX15,1
2290 REPEAT UNTIL GET=32
2300 ENDPROC
2310 DEFPROCwin
2320 VDU5:GCOL0,3:MOVE600,32:SOUND&11,
2,80,255
2330 IFblack%>white%PRINT"Black wins!"
:GOTO2350
2340 IFwhite%>black%PRINT"White wins!"
ELSEPRINT"Close game!"
2350 VDU4:COLOUR1:PRINTTAB(0,23)"Press
 spacebar to play   again"
2360 *FX15
2370 wait$=GET$
2380 ENDPROC
2390 DEFPROCedge:IFG%<>0THENENDPROC
2400 RESTORE2860:Q%=0:REPEAT:Q%=Q%+1:R
EADcheck%:UNTILcheck%=pos%
2410 RESTORE2860:FORP%=1TOQ%+1:READche
ck%
2420 IFP%=Q%-1ANDposition%(check%)>0AN
Dposition%(check%)<>value%THENweight%=0
2430 IFP%=Q%+1ANDposition%(check%)>0AN
Dposition%(check%)<>value%THENweight%=0
2440 NEXT
2450 ENDPROC
2460 DEFPROCcheckedge:IFedge%=0THENEND
PROC
2470 IFposition%(pos%-edge%)>0ANDposit
ion%(pos%-edge%)<>value%THENweight%=0
2480 IFposition%(A%+edge%)>0ANDpositio
n%(A%+edge%)<>value%THENweight%=0
2490 ENDPROC
```

»

```
2500 DEFPROCcorner:FORQ%=1TO3:READpos%
:PROCweigh:RESTORE2900:FORS%=0TOR%+Q%:R
EADT%:NEXT:NEXT:ENDPROC
2510 DEFPROCcompute
2520 G%=1:bestgo%=0:illegal%=0:IFcol%=
16THEN2550
2530 IFgo%<>1THENENDPROC
2540 comp%=2
2550 RESTORE2820:FORR%=1TO4:READpos%:P
ROCweigh:RESTORE2820:FORS%=1TOR%:READT%
:NEXT:NEXT
2560 IFbestgo%>0THEN2770
2570 IFcol%=16THEN2640
2580 IFturn%<50ORmiss%=1THEN2640
2590 FORedge%=-1TO1STEP2:RESTORE2830:F
ORR%=1TO12:READpos%:PROCweigh:RESTORE28
30:FORS%=1TOR%:READT%:NEXT:NEXT:NEXT
2600 FORedge%=-10TO10STEP20:RESTORE284
0:FORR%=1TO12:READpos%:PROCweigh:RESTOR
E2840:FORS%=1TOR%:READT%:NEXT:NEXT:NEXT
2610 edge%=0:IFbestgo%>0THEN2770
2620 G%=0:RESTORE2850:FORR%=1TO16:READ
pos%:PROCweigh:RESTORE2850:FORS%=1TOR%:
READT%:NEXT:NEXT
2630 G%=1:IFbestgo%>0THEN2770
2640 RESTORE2870:FORR%=1TO12:READpos%:
PROCweigh:RESTORE2870:FORS%=1TOR%:READT
%:NEXT:NEXT
2650 VDU19,1,2,0,0,0
2660 IFbestgo%>0ANDmiss%=0THEN2770
2670 RESTORE2880:FORR%=1TO16:READpos%:
PROCweigh:RESTORE2880:FORS%=1TOR%:READT
%:NEXT:NEXT:IFbestgo%>0THEN2770
2680 RESTORE2890:FORR%=1TO16:READpos%:
PROCweigh:RESTORE2890:FORS%=1TOR%:READT
%:NEXT:NEXT:IFbestgo%>0THEN2770
2690 IFmiss%=1THEN2760
2700 RESTORE2900:FORR%=0TO12:READpos%
2710 IFR%MOD4=0ANDposition%(pos%)=valu
e%THENPROCcorner
2720 RESTORE2900:FORS%=0TOR%:READT%:NE
XT:NEXT:IFbestgo%>0THEN2770
2730 RESTORE2900:FORR%=0TO12:READpos%
2740 IFR%MOD4=0ANDposition%(pos%)>0THE
NPROCcorner
2750 RESTORE2900:FORS%=0TOR%:READT%:NE
XT:NEXT:IFbestgo%>0THEN2770
2760 RESTORE2900:FORR%=1TO16:READpos%:
PROCweigh:RESTORE2900:FORS%=1TOR%:READT
%:NEXT:NEXT
2770 IFcol%=16THENENDPROC
2780 comp%=1:IFbestgo%>0THENpos%=compg
o%:ENDPROC
2790 SOUND&11,1,2,12:VDU4:PRINTTAB(0,5
);"Pass - no"'"legal move":TIME=0:REPEA
TUNTILTIME>200
2800 col%=16:pass%=1
2810 ENDPROC
2820 DATA12,19,89,82
2830 DATA14,15,16,17,84,85,86,87,13,18
,83,88
2840 DATA32,42,52,62,39,49,59,69,22,72
,29,79
2850 DATA14,17,39,69,87,84,62,32,15,16
,49,59,85,86,42,52
2860 DATA13,14,15,16,17,18,29,39,49,59
,69,79,88,87,86,85,84,83,72,62,52,42,32
,22
2870 DATA37,36,35,34,44,54,64,65,66,67
,57,47
2880 DATA24,25,26,27,38,48,58,68,77,76
,75,74,63,53,43,33
2890 DATA32,42,52,62,84,85,86,87,69,59
,49,39,17,16,15,14
2900 DATA19,18,28,29,89,79,78,88,82,83
,73,72,12,22,23,13
2910 DATA45,2,0,46,1,3,55,1,3,56,2,0
2920 DATA1,9,10,11,-1,-9,-10,-11
```

This program is adapted from a version
for the BBC micro first published in
BEEBUG Vol.2 No.6.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DOUBLE QUOTES - A.Baker

To get a " mark inside a piece of text delimited by "..." use two together:
PRINT "THIS IS A "" DOUBLE QUOTE"
It gives:
THIS IS A " DOUBLE QUOTE
If you are going to put a double quote in the middle of a string, then the string
must be delimited with normal quotes, and the quote marks entered inside with two
together as above.

INPUTLINE - P.G.Barnes

When using the INPUTLINE statement, TAB can be used to position the printed text
and the input itself e.g.
INPUTLINE TAB(10,5)"ENTER NAME"TAB(5,7) A$

ELLIPTO
by S. Wilkinson

This is a relatively short program which produces varied patterns by plotting a succession of filled ellipses of random size. The ellipses have the same centre, and the pattern is gradually built up using so-called 'exclusive or' plotting (achieved with the GCOL 3,1 statement in line 180 - see User Guide page 153). This program allows you to select either mode 4 or mode 0, and gives white patterns on a black background. When one pattern complete, pressing any key will produce a new display. Press Escape to exit the program.
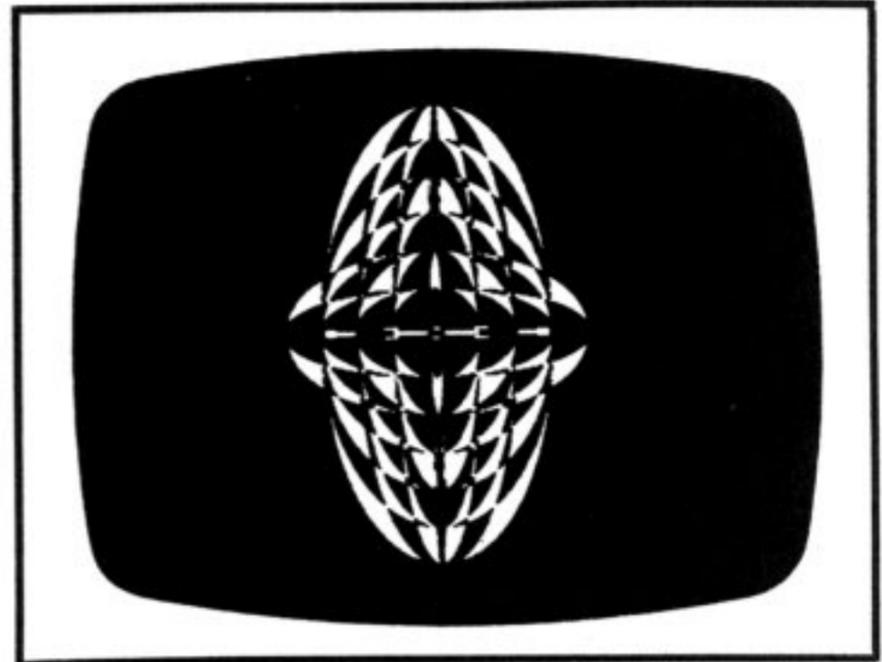
This program is adapted from a version for the BBC micro that first appeared in BEEBUG Vol.2 No.2
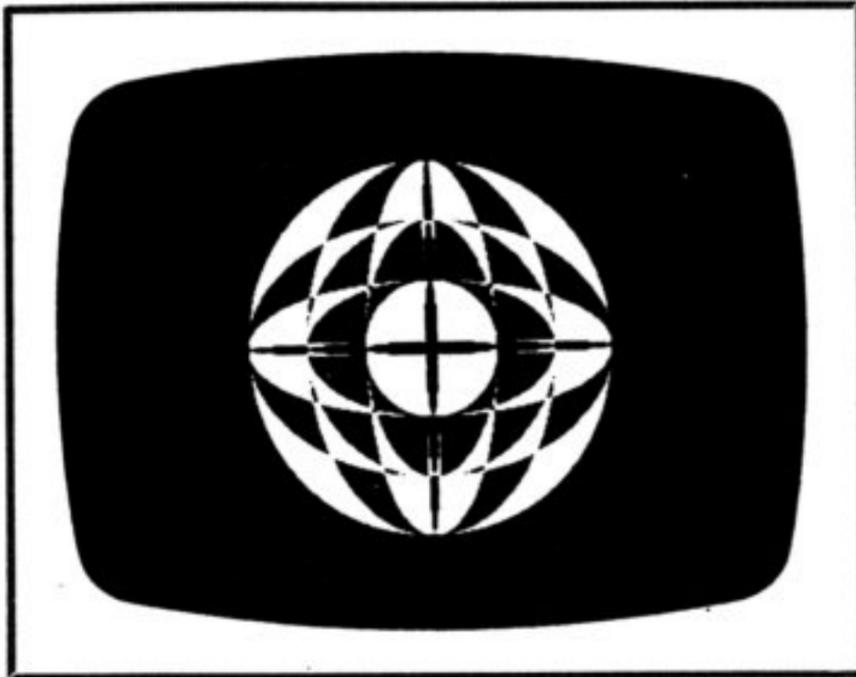

```
10 REM Program ELLIPTO
20 REM Version 1.0
30 REM Author  S.Wilkinson
40 REM ELBUG Jan/Feb 1984
50 REM Program subject to copyright
60 :
70 ON ERROR GOTO 340
80 MODE6
90 PRINT
```



```
100 REPEAT:CLS
110 PRINT'''TAB(12);"E L L I P T O"''TAB(12);"by S.Wilkinson"
120 INPUTTAB(0,15)"Mode 0 or Mode 4 : "M%
130 UNTIL M%=4 OR M%=0
140 MODE M%
150 IF M%=0 M%=2
160 VDU23,1,0;0;0;0;
170 VDU29,640;512;
180 GCOL3,1
190 REPEAT
200 C%=RND(7)*16+16
210 FORA=16 TO 512 STEP C%
220 FORB=16 TO 512 STEP C%
230 L%=-A
240 MOVEL%,0:DRAW-L%,0
250 FOR Y%=4 TO B STEP4
260 X%=A/B*SQR(B*B-Y%*Y%)/M%:X%=X%*M%
270 MOVE X%,Y%:DRAW -X%,Y%
280 MOVE X%,-Y%:DRAW -X%,-Y%
290 NEXT Y%
300 NEXT B,A
310 A=GET:CLS:UNTIL FALSE
320 END
330 :
340 ON ERROR OFF
350 MODE6:IF ERR=17 END
360 REPORT:PRINT" at line ";ERL
370 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

STEPPING THROUGH LISTINGS - R.Q.McMinn
When listing a long program you can freeze the listing by pressing Control and Shift simultaneously. Releasing either key allows the listing to continue.

HOW TO WRITE A GAME PROGRAM (Part 2)
by Mike Williams

We continue and complete the article which we started last month describing how to design and develop your own game program. In this part we cover user defined characters and their movement on the screen, and complete the game with a simple scoring system.

The first task this month is to work out how to draw the parcels. Because these are fairly small, we shall use user-defined characters rather than Draw and Plot routines. The character is defined on an 8 by 8 matrix. We have to draw parcels in three different sizes, so we will choose these to be three by three, four by four, and five by five. The reason for these particular sizes is explained shortly (and illustrated below). In order to present a more attractive game, we will arrange for each parcel to move across the sky on a toboggan. The toboggan is made up of two user defined characters as shown in the diagram.

large parcel
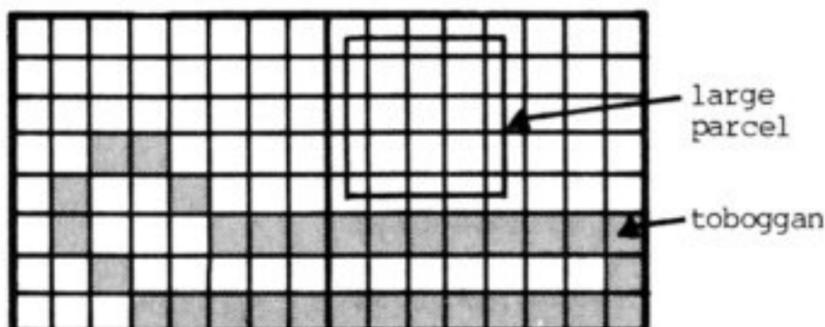
toboggan

The characters for the three sizes of parcel are designed so that each parcel appears to rest neatly on the rear of the toboggan when the two characters are superimposed. Again, with something of this kind, a certain degree of trial and error is necessary to achieve results that look best on the screen. In practice, either use paper and pencil (as in the diagrams here) or the character definer on our introductory cassette, to design the characters in the first place. The character definitions are all contained in the procedure PROCchars, the
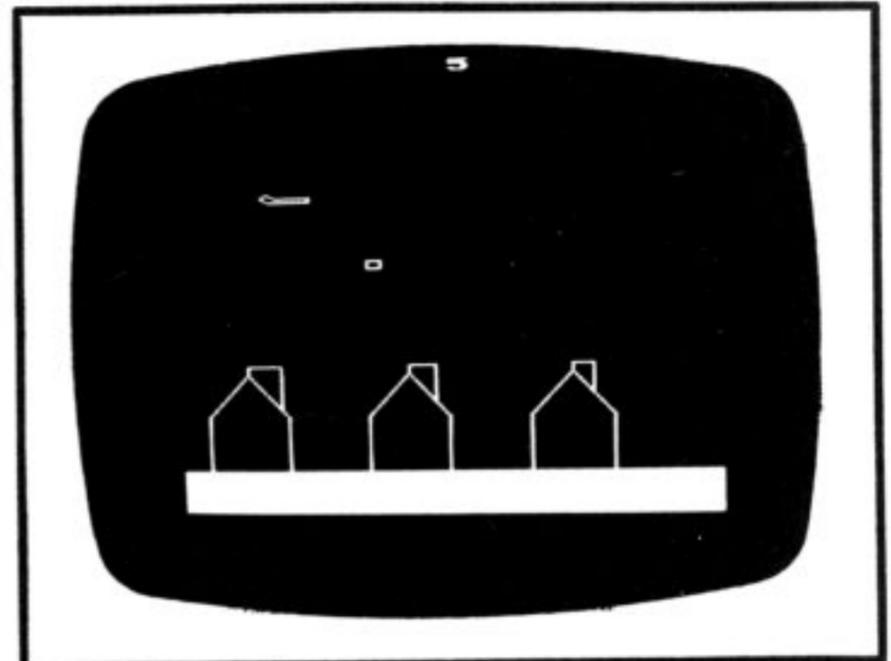
skeleton of which we wrote last month in lines 1200 and 1290. Let's start by defining the toboggan. We then have as follows

```
1200 DEF PROCchars
1210 VDU23,224,0,0,0,48,72,71,32,31
1220 VDU23,225,0,0,0,0,0,255,1,255
1290 ENDPROC
```

Refer to our series on Electron graphics, in this and the previous two issues of ELBUG, for more information on the VDU 23 command and the definition of your own characters.

If the instructions above are typed into your Electron, you can test them in immediate mode as follows. First select the correct graphics mode:
    MODE 2 <return>
Then define the two characters:
    PROCchars <return>
You can now display the toboggan by typing:
    PRINT CHR$224;CHR$225 <return>

If you find that your first attempt at defining characters is not successful, then you can redefine them and repeat the steps above. Assuming that our definitions in lines 1210 and 1220 are alright, we can now define the parcels themselves, so that they rest neatly on the toboggan:

```
   1230 VDU23,226,248,136,136,136,248,
0,0,0
   1240 VDU23,227,0,240,144,144,240,0,
0,0
   1250 VDU23,228,0,0,224,160,224,0,0,
0
```
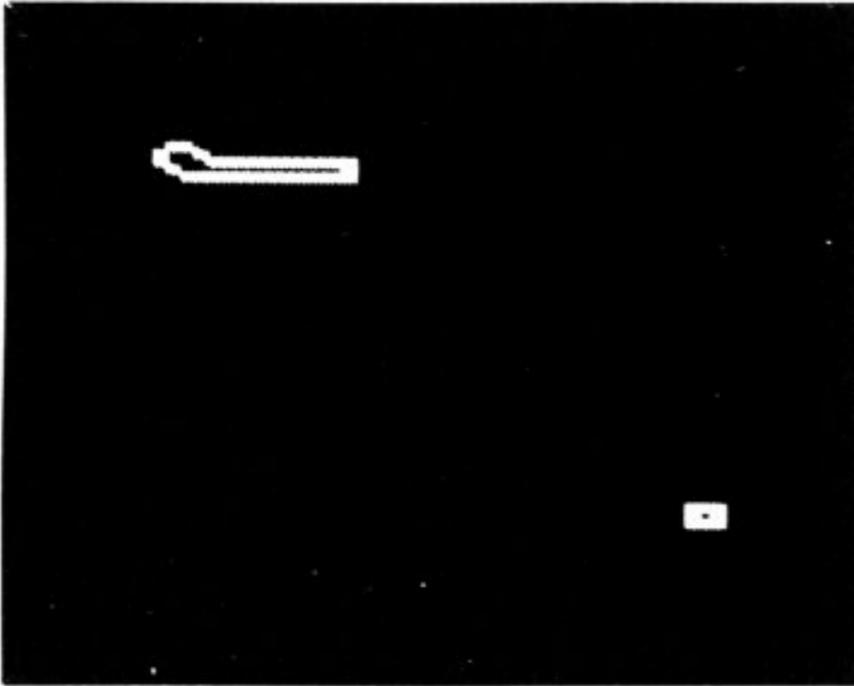

Again you can test this in immediate mode by typing:

```
   PROCchars <return>
   PRINT CHR$226 <return>
   PRINT CHR$227 <return>
   PRINT CHR$228 <return>
```

The diagram above should make the connections between the various shapes clear. The procedure PROCchars is now complete.

The next step in developing this game is to write the instructions that will make the toboggan move across the screen from right to left carrying one of the three sizes of parcel, chosen randomly. We have already defined, in outline, a procedure called PROCplay which the program calls repeatedly until the game is over. For convenience, this part of the program is repeated below:

```
   160 REPEAT
   170 PROCplay
   180 UNTIL over%
   190 END
```

The procedure PROCplay is defined at line 1400 but at the moment is empty, viz:

```
   1400 DEF PROCplay
   1490 ENDPROC
```

This procedure will eventually have to look for keyboard input, the dropping of a parcel, our success or failure in getting the parcel into a chimney, and the scoring. At the moment, we are only concerned with moving the toboggan plus parcel across the screen, and we will put this as a procedure PROCmove, and call it as the first step in PROCplay:

```
   1410 PROCmove
```

What we have to do now is to define PROCmove, and we will do this starting at line 1600, so that it follows the last procedure already defined (PROChouse). In order to display the characters which we have defined, we need to specify a position in terms of the character positions and lines on the screen. We will use the variables AX% and AY% for this purpose, and these will have to be set initially. Since the game works in mode 2, there will be 20 character positions across the screen, numbered from 0 to 19 (from left to right). Initially we shall set AX%=19 and AY%=RND(10). This will allow AY% (and hence the toboggan) to vary randomly in height within lines 1 to 10 of the screen.

this position is AX%,AY%
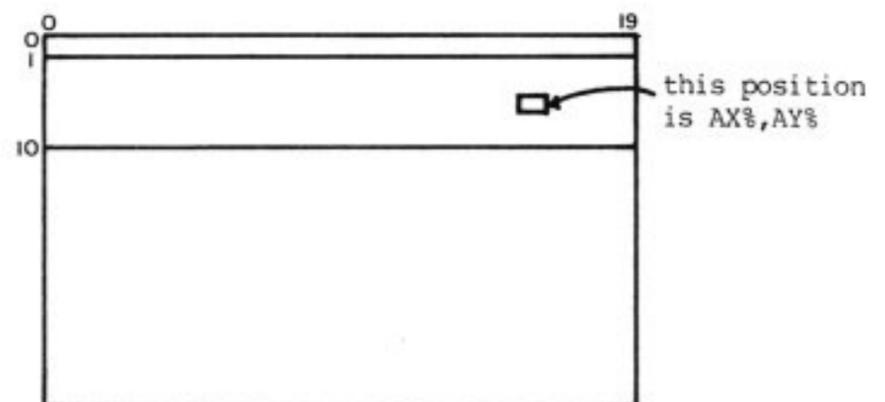
These initial values can be set up in PROCsetup as:

```
   1030 AX%=19:AY%=RND(10)
```

Every time that we call PROCmove, we will need to reduce AX% by 1 (to produce movement across the screen),

and display the toboggan at the new position. Thus the first three lines of PROCmove will be

```
1600 DEF PROCmove
1610 AX%=AX%-1
1620 PRINT TAB(AX%,AY%);CHR$224;CHR$225;CHR$32
```

Note the space character at the end, which will automatically erase the previous CHR$225 (end of toboggan) on the screen. This technique for moving objects has been described in the series on Electron graphics. We could have streamlined this section by defining the toboggan as a string variable in PROCsetup

```
1035 toboggan$=CHR$224+CHR$225+CHR$32
```
and rewriting line 1620 as

```
1620 PRINT TAB(AX%,AY%);toboggan$
```

This makes no obvious difference to the running of the program, but makes it more readable. If you run the whole program at this point, you should see the three houses displayed on the screen and the toboggan should move, once only, across the screen - we haven't yet done anything to return the toboggan to the right hand side of the screen.

Now we must make a parcel appear on the toboggan as it moves. We have defined the parcel characters so that they fit neatly on the toboggan, but unfortunately, when using PRINT, if you print one character over another, the first character is completely replaced by the second. What we require is for both characters to remain, combined together. We can achieve this by combining text and graphics together.

Normally, there are separate text and graphics cursors, although this is not always very obvious, the text cursor being moved by the PRINT and TAB instructions and the graphics cursor being moved by the MOVE, DRAW and PLOT instructions. By using the command VDU5, the two cursors are combined, and text can be printed at any graphics position on the screen (0 to 1279 horizontally, 0 to 1023 vertically). This has the added advantage, that if

two characters are plotted (or printed) in the same position, then a combination of the two characters is seen on the screen rather than just the last one printed. This is just what we need for our game.

What we must do therefore, is convert our positions AX% and AY% into corresponding graphics co-ordinates, remembering that in mode 2, a character has a width of 64 and a height of 32 in graphics units. The formulae for this are:
```
PX%=AX%*64
PY%=1024-(AY%*32)
```
where PX% and PY% are the horizontal and vertical positions of the toboggan on the screen.

In text mode, you count downwards from 0 on the screen, while in graphics mode you count upwards from 0. The parcel is actually placed over the second character of the toboggan (+32 horizontally) and moved down slightly (-4) as this looked better when viewed on the screen.

Add the following lines to the program:

```
1040 parcel = RND(3)+225
1630 PX%=AX%*64:PY%=1024-(AY%*32):VDU5:GCOL0,parcel-225:MOVE PX%+32,PY%-4:VDUparcel:VDU4
```

Line 1040 is part of PROCsetup and randomly selects one of the three parcel characters. This is used in line 1630, firstly to specify the colour of the parcel using the GCOL command, and secondly as the character to be printed using VDUparcel. VDU4 at the end of the instruction separates once again the text and graphics cursors.

We can now complete the definition of PROCmove by adding the following lines:

```
1640 IFAX%=1 THEN AX%=19:PRINTTAB(1,AY%);CHR$32;CHR$32:AY%=RND(10)
1650 ENDPROC
```

This ensures that the toboggan is completely erased when it reaches the left hand side of the screen (AX%=1) and that AX% and AY% are set to new

values so that the toboggan reappears at the right hand edge of the screen, and at a different height. This should now be visible if you run the program that we have built up so far.

Before continuing, it would be useful to complete the procedure PROCsetup. The final version of this procedure, is as follows:

```
1000 DEF PROCsetup
1010 score%=0:miss%=0
1020 drop%=FALSE:over%=FALSE
1030 AX%=19:AY%=RND(10)
1040 parcel=RND(3) +225
1050 VDU23,1,0;0;0;0;
1090 ENDPROC
```

All these steps are performed just once at the start of the program when PROCsetup is called. The variables, score% and miss%, will count the score and the number of misses. The variable drop% will be set TRUE when a parcel starts falling, and the variable over% will be set TRUE when the game is over. The VDU23 command in line 1050 removes the flashing cursor from the screen, a very useful facility in any game like ours. The other instructions have already been described.

Now back to the game. We want to use the space bar as the signal for dropping a parcel. Therefore, in our procedure PROCplay, we need to test for this, and if it happens, set drop%=TRUE and make the parcel appear to drop downwards. Since the movement of the parcel will be different when falling to that of the toboggan, two new variables, FX% and FY%, are needed to represent the position of the falling parcel. We will also create a further procedure, PROCdrop, to control the movement of the falling parcel in the same way that PROCmove controls the horizontal movement on the screen. We can now complete PROCplay and the full procedure is as follows:

```
1400 DEF PROCplay
1410 PROCmove
1420 IF INKEY-99 AND NOT drop% THEN
drop%=TRUE:FX%=AX%:FY%=AY%
1430 IF drop% THEN PROCdrop ELSE FOR
W=1 TO 10:NEXT
1440 PRINT TAB(10,0);score%;SPC(5)
1490 ENDPROC
```

INKEY-99 tests whether the space bar is pressed at that moment and that the parcel is not already dropping. If this is true, then drop% is set TRUE and we initialise the position of the falling parcel as its present position. At line 1430, the FOR-NEXT loop builds in a short delay to balance the time taken by the procedure PROCdrop so that the toboggan appears to move at the same speed whether the parcel is falling or not. Finally, the PRINT instruction places the current score on the screen, followed by five spaces so that any previous, and possibly longer, score is completely erased.

We also need to amend one line in the procedure PROCmove so that once a parcel starts falling it no longer appears on the toboggan. This is very easy to achieve by turning the existing instruction at this line into an IF-THEN statement:

```
1630 IF NOT drop% THEN PX%=AX%*64 :
PY%=1024-(AY%*32):VDU5: GCOL0,parcel-25
5:MOVE PX%+32 ,PY%-4:VDUparcel:VDU4
```

As you will appreciate by now, describing how to develop a program takes up far more space than the program itself. However, if you have managed to keep up so far, we have only two more procedures to describe in detail. First of all, here is the procedure to drop the parcel:

```
1700 DEF PROCdrop
1710 FY%=FY%+1
1720 COLOUR parcel-225
1730 PRINT TAB(FX%,FY%);CHR$parcel
1740 PRINT TAB(FX%,FY%-1);CHR$32
1750 COLOUR 7
1760 IF FY%>19THEN PROCtest:parcel =
RND(3)+225:drop%=FALSE:PRINT TAB(FX%,FY
%);CHR$32
1790 ENDROC
```

In this procedure, FY% is increased by 1 to move the parcel down and the colour is set according to the size of parcel. The parcel character is then displayed on the screen and the previous position overwritten with a space. Since the parcel character is not mixed with any other character, the simpler 'PRINT TAB' instruction is quite adequate here. Line 1750 restores the text colour to white, for displaying the score later.

The most complicated instruction in this procedure is the one at line 1760. This checks to see if the falling parcel has reached chimney height and then calls PROCtest to see if the parcel will enter the chimney or miss. It also re-initialises the program ready to send the next parcel across the screen.

Lastly, we must produce a procedure to test whether our falling parcel will fall within a chimney. This is complicated by the fact that both parcels and chimneys are of three different sizes. The smallest parcel will fit in all three chimneys, but the largest parcel will only fit in the largest chimney. In addition, the scoring is varied as well. The procedure to do this is as follows:

```
1800 DEF PROCtest
1810 ON .parcel-225 GOTO 1840,1830,18
20
1820 IF FX%=15 score%= score%+50:END
PROC
1830 IF FX%=9 score%= score%+25:ENDP
ROC
1840 IF FX%=3 score%= score%+10:ENDP
ROC
1850 score%=score%-5:miss%=miss%+1
1860 IF score%<0 THEN score%=0
1870 IF miss%>20 THEN over%=TRUE
1890 ENDPROC
```

The numbers 15,9 and 3 in lines 1820 to 1840 are the horizontal positions of the chimneys in character terms. These numbers were calculated from the graphics co-ordinates of the houses by remembering that one character is equivalent to 64 graphics units horizontally. The ON-GOTO statement selects the next instruction depending on parcel size. If the parcel is of the smallest size, we test all three chimneys, if the middle size we test the centre and left chimney, and if the largest size we test only the left most and largest chimney. The order of

these three statements is critical to the correct working of the game. If the parcel fits a chimney, the score is increased and no further action takes place.

The instructions from line 1850 onwards deal with the situation when a parcel misses or is too large to fit in the chimney aimed for. In this case the score is decreased by 5, the count of misses is increased by 1 and the routine tests to see if the maximum of 20 misses has been exceeded, in which case the game is over. We also need to check, in line 1860, that we haven't reduced the score to a negative value. If that happens the score is just returned to zero.

You should now find that the whole game, as originally described at the start of last month's article, now works. The full listing of the game that we also published in last month's issue contained two enhancements to the program we have described. A procedure called PROCnoise was defined in lines 1900 to 1990, and used in PROCtest, so that when a parcel successfully enters a chimney, a pleasant noise is heard. The envelope for this noise is defined in PROCsetup. The noise routine was developed by experimenting with the SOUND and ENVELOPE commands (see page 116 of the User Guide - see also Sound Wizard in the first issue of ELBUG).

Secondly, when the game is over, a procedure PROCend is called, which just clears the screen and displays the final score. Both of the procedures, PROCfrontpage and PROCend could be improved further, to provide more information.

This completes our two part article. There are many ideas and techniques that have been described here, and you may well find it helpful to return to this when you are writing your own programs. ●

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

REPEAT DELAYS
The delay before a key repeats, when held down, can be changed using *FX11,d. The delay, 'd', is in 100ths of a second. Likewise, the repeat rate is changed by *FX12,r. The repeat rate 'r' is also in 100ths of a second. When you switch the Electron on, 'r' and 'd' are automatically set to 8 and 32 respectively. ●

BAD PROGRAM LISTER
by Elizabeth Hanson

This program will list any Basic program (or any text file) stored in the computer, in a completely readable format, very similar to that produced by the LIST command. Its particular value is in listing programs which the command LIST will not work upon; ie programs which have become 'damaged' in the computer, resulting in the familiar response, 'Bad Program'. In this respect it forms a useful adjunct to the program 'RESCUE', which was published in last month's issue of ELBUG. In fact, LISTER will even list the remains of a program that has been partly overwritten, by loading in another shorter program on top of it.

LISTER was written for use with the RESCUE program published last month in ELBUG. RESCUE is an extremely powerful utility which will miraculously heal 'bad programs'.

LISTER allows you to examine a 'bad program' before attempting to repair it, though it has many other applications, such as reading partially overwritten programs. LISTER works by looking at the Basic tokens as they are stored in memory, and converting them back to their original keyword format. The program is quite small, just under two blocks in length, as it uses the look-up table in the Basic ROM at &8071 to make the conversions.

LISTER will not convert the line numbers following a GOTO, as these are stored in code within the computer to save space. Note that for the sake of speed (and the irregularity of the way the tokens are set up in memory), the word "AND" is printed as "ND".

USING LISTER

Type in the program and save a copy, before use, as you would any Basic program.

Assuming that you want to use LISTER to list a program already in the computer, you will need to change PAGE before you load in LISTER. This avoids overwriting the program already resident in memory. Proceed as follows:

```
MODE6              <return>
PAGE=&5B00         <return>
CHAIN"LISTER"      <return>
```

LISTER will then run and print "Start Address ?" on the screen as a prompt for you to provide a starting address for it to list from. This will normally be &E00 (in hexadecimal), but you can type in any address that you want. The listing will then start and will be printed in 1k blocks. After each block the computer will bleep, and wait for you to press any key. It will then display the current address in hex followed by the next 1k of the program, and will continue in this way until you stop the program by pressing Escape.

```
Bad Program Lister.
===================

Start Address?&E00


E00

10 REM PROGRAM   COLOURED CHRS 1
0 00AUTHOR    D.Graham
30 REM Version   E.10
40 REM ELBUG     Jan/Feb '84
50 REM Program   subject to copyright
60 :
```

```
10 REM PROGRAM BAD PROGRAM LISTER
20 REM VERSION E1.1
30 REM AUTHOR Elizabeth Hanson
40 REM ELBUG JAN/FEB 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT.
60 :
70 ON ERROR GOTO 280
80 MODE6
90 PRINTTAB(10)"Bad Program Lister."
100 PRINTTAB(10);STRING$(19,"=")
110 VDU28,0,24,39,3
120 INPUT'"Start Address",A$
130 @%=1:J%=255:A%=EVAL(A$)
140 REPEAT:PRINT''"" ";~A%
150 FORI%=0TOJ%:X%=A%?I%
```

```
   160 IFX%=&0D PRINT'A%?(I%+1)*256+A%?(
 I%+2);SPC1;:I%=I%+3:GOTO190
   170 IFX%>&7F ANDX%<&FF PROCL:GOTO190
   180 IFX%>31PRINTCHR$(X%);ELSEPRINT~X%
 ;
   190 NEXT
   200 A%=A%+J%+1:VDU7:B$=GET$:UNTILFALS
 E
   210 END
   220 :
   230 DEFPROCL:K%=-1:T%=&8071
   240 REPEAT:K%=K%+1:Y%=T%?K%:UNTILY%=X
 %ORK%>&300:L%=K%
   250 REPEAT:K%=K%-1:Z%=T%?K%:UNTILZ%>&
 7F ORK%<0
```

```
   260 IFT%?(K%+2)<40K%=K%+3ELSEK%=K%+2
   270 REPEAT:PRINTCHR$(T%?K%);:K%=K%+1:
 UNTILK%>=L%:ENDPROC
   280 ON ERROR OFF
   290 MODE 6
   300 IF ERR<>17 REPORT:PRINT" at line
 ";ERL
   310 END
```

This program is adapted from a version for the BBC micro that was first published in BEEBUG Vol.2 No.3.

RETURN OF THE DIAMOND — A MAP

Last month we published our very own adventure game called 'The Return of the Diamond'. For those of you who are still lost we now provide a map showing all the different locations. Please also read the Editorial in this issue regarding the program for 'The Return of the Diamond'.
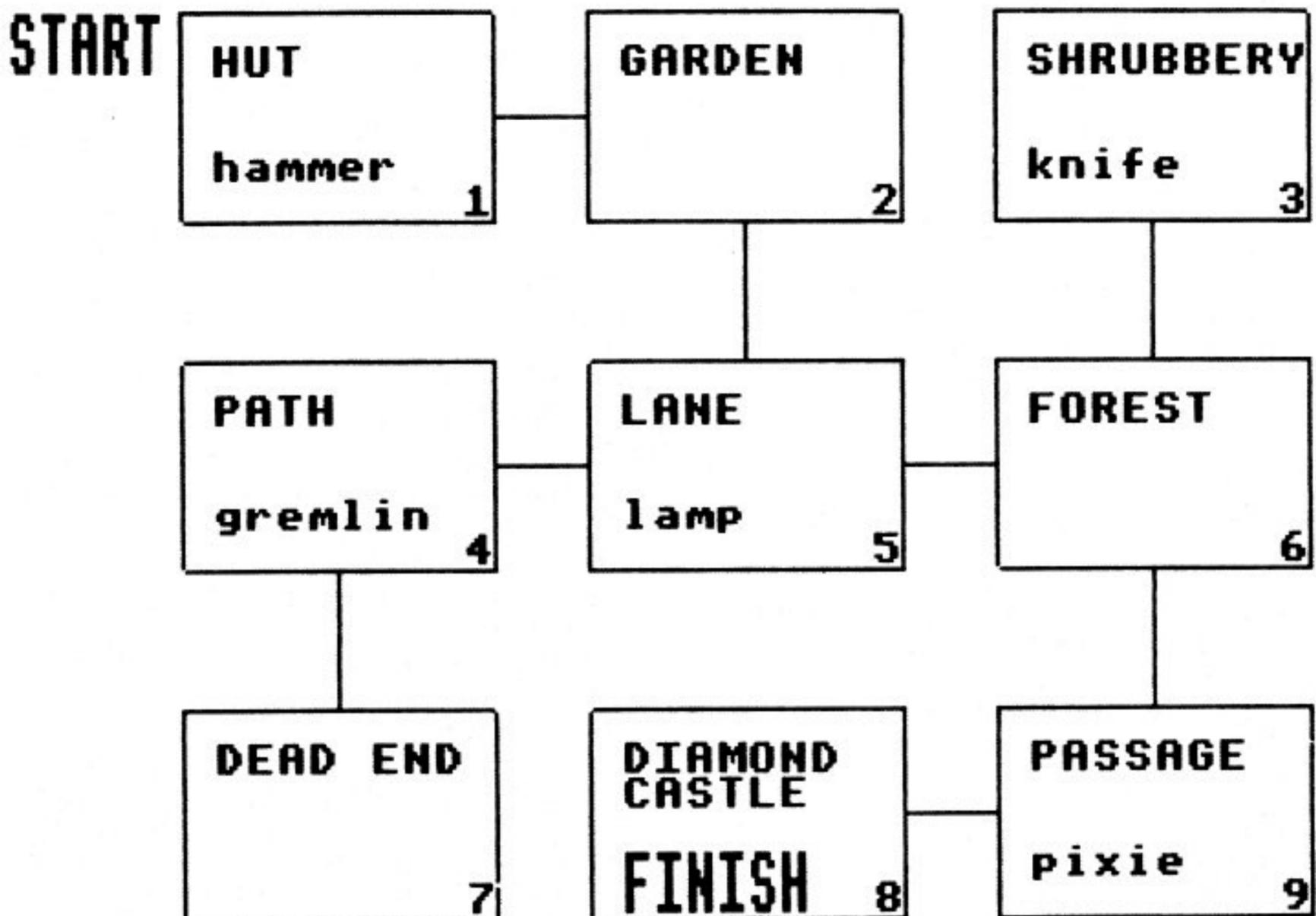
MAP OF RETURN OF THE DIAMOND

DIVE BOMBER
by N. Mallinson

Just imagine the thrill as you speed your way to your target, your scarf billowing in the wind, with the smut from the spluttering engine clouding your vision through your goggles. Your mission is simple, but dangerous. It is to destroy as many targets as possible, whilst avoiding the flak from the guns on the ground.

Well, it's not quite Biggles, but in this game you do have to destroy the various fuel and ammunition dumps, while avoiding the anti-aircraft guns. There are 100 targets on each mission, after which you may land. If you have reached the required score for that mission, then you are sent out on a new sortie, but otherwise the game ends.

You have three controls:
    :=up
    /=down
    Z=release bomb

Bombs may only be released when the plane is in a dive. You will find that bombing attacks can be made much more accurately from a low-level approach, but then you will need to be nimble-fingered to avoid crashing; and the anti-aircraft barrages are a good deal more accurate when you are flying low.

Good luck Biggles!

```
10 REM PROGRAM DIVE BOMBER
20 REM AUTHOR N.MALLINSON
30 REM VERSION E1.4
40 REM ELBUG JAN/FEB 1984
50 REM PROGRAM SUBJECT TO COPYRIGHT
60:
100 ON ERROR GOTO 2770
110 MODE 5
120 VDU 23,1,0;0;0;0;
130 PROCtitle
140 MODE 4
150 VDU 19,0,4,0,0,0
160 PROCinit
170 PROCinstructions
180 PROCnewgame
190 REPEAT
200 MODE 4
210 VDU 19,0,4,0,0,0
220 VDU 23,1,0;0;0;0;:CLS
230 PROCnewrun
240 PRINTTAB(1,7)"PRESS SPACE BAR TO
START"
250 REPEAT:VDU 7:UNTIL GET=32
260 PRINTTAB(1,7)"TAKE OFF !";SPC(14)
:VDU 7
270 REPEAT
280 IF F%=0 GOTO 320
290 IF ?(22775+320*F%)<>0 THEN PROCbo
mbhit:GOTO 320
300 F%=F%+1:IF?(22775+320*F%)<>0 THEN
 PROCbombhit:GOTO 320
310 PRINTTAB(30,F%-1);SPC1;TAB(30,F%)
;CHR$(227)
```

```
 320 IF X%=0 THEN L%=225 ELSE X%=X%-1:
SOUND 0,-15,2,5:IF X%=0 THEN PRINTTAB(1
,7);SPC(24):PRINTTAB(30,K%);CHR$(225):G
OTO 510 ELSE GOTO 510
 330 IF INKEY(-73) AND K%>6 THEN L%=22
4:N%=K%-1
 340 IF INKEY(-105) THEN L%=226:N%=K%+
1
 350 IF ?(22775+320*N%)<>0 THEN PROCpl
anehit(N%):GOTO 690
 360 PRINTTAB(30,K%);SPC1;TAB(30,N%);C
HR$(L%):K%=N%
 370 SOUND 1,-15,32+40*(L%-224),5
 380 IF INKEY(-98) AND F%=0 AND L%=226
 THEN F%=K%+1 ELSE GOTO 410
 390 IF ?(22775+320*F%)<>0 THEN PROCbo
mbhit:GOTO 410
 400 PRINTTAB(30,F%);CHR$(227)
 410 IF T%=0 AND U%>0 THEN U%=U%+1
 420 IF U%>30 OR V%<7 THEN PRINTTAB(U%
,V%);SPC1:T%=0:U%=0
 430 IF T%=1 THEN PRINTTAB(U%,V%);SPC1
:IF V%>6 AND U%<31 THEN U%=U%+1:V%=V%-1
:PRINTTAB(U%,V%);CHR$(240)
 440 IF U%>0 AND T%=0 AND 27-U%+I%<=V%
-K% THEN T%=1:PRINTTAB(U%,V%);CHR$(240)
:SOUND &11,-1,0,0:SOUND &10,-15,4,8
 450 IF G%=232 AND U%=0 AND B%>0 THEN
U%=1:V%=H%-1:I%=RND(5)
 460 IF D%=0 AND B%>0 THEN B%=B%+1
 470 IF B%>30 OR C%<7 THEN PRINTTAB(B%
,C%);SPC1:D%=0:B%=0
 480 IF D%=1 THEN PRINTTAB(B%,C%);SPC1
:IF C%>6 AND B%<31 THEN B%=B%+1:C%=C%-1
:PRINTTAB(B%,C%);CHR$(240)
 490 IF B%>0 AND D%=0 AND 28-B%+E%<=C%
-K%+(1+C%-K%)*(225-L%) THEN D%=1:PRINTT
AB(B%,C%);CHR$(240):SOUND &10,-15,4,2
 500 IF G%=232 AND B%=0 THEN B%=1:C%=H
%-1:E%=RND(3)
 510 A%=0:IF Z%<1 THEN G%=237:Z%=Z%-1:
GOTO 610
 520 IF G%=235 THEN A%=A%-1
 530 IF H%>28 THEN G%=235:GOTO 590
```



```
 540 IF H%<Y% THEN G%=234:A%=A%+1:GOTO
590
 550 R%=RND(127):G%=229
 560 REPEAT:R%=R% DIV 2:G%=G%+1:UNTIL
R%=0
 570 IF G%=235 THEN G%=G%-RND(4) DIV 4
 580 IF G%=234 THEN A%=A%+1
 590 H%=H%+A%
 600 IF G%<234 AND Z%>0 THEN Z%=Z%-1:P
RINTTAB(31,5);Z%;SPC1
 610 CALL SCRLR
 620 PRINTTAB(0,H%);CHR$(G%)
 630 IF X%<>0 GOTO 680
 640 IF ?(22775+320*K%)<>0 THEN PROCpl
anehit(K%):GOTO 690
 650 IF D%=1 THEN PRINTTAB(B%,C%);SPC1
:IF C%>6 AND B%<31 THEN B%=B%+1:C%=C%-1
:PRINTTAB(B%,C%);CHR$(240)
 660 IF T%=1 THEN PRINTTAB(U%,V%);SPC1
:IF V%>6 AND U%<31 THEN U%=U%+1:V%=V%-1
:PRINTTAB(U%,V%);CHR$(240)
 670 IF J%>0 THEN J%=J%-1:IF J%<1 THEN
ch%=&10
 680 FOR M%=1 TO DL%:NEXT
 690 UNTIL Z%<-40
 700 IF Z%>-50 THEN PROClanding ELSE I
F L%=242 THEN PRINTTAB(1,7)"SHOTDOWN !"
 ELSE PRINTTAB(1,7)"CRASHED !"
 710 TIME=0:REPEAT UNTIL TIME>100
 720 *FX 15,0
 730 UNTIL Z%=-50
 740 PROChighscores:PRINTTAB(3,M%)"ANO
THER GAME (Y/N) ?"
 750 VDU 7:I$=GET$:IF I$="Y" THEN CLS:
GOTO 180
 760 IF I$<>"N" GOTO 750
 770 *FX15,0
 780 MODE 6
 790 END
 800 :
1000 DEF PROCbombhit
1010 IF ?(22774+320*F%)=32 THEN PRINTT
AB(30,F%);SPC1:GOTO 1090
1020 IF ?(22774+320*F%)<100 THEN SOUND
&10,-15,4,10:J%=1:GOTO 1070
1030 Q%=10*?(22769+320*F%)
1040 W%=W%+Q%:S%=S%+Q%
1050 PRINTTAB(7,3);S%;TAB(25,3);W%
1060 SOUND &10,-15,4,20:J%=5
1070 PRINTTAB(30,F%);CHR$(228)
1080 ch%=&11
1090 IF F%-1<>K% THEN PRINTTAB(30,F%-1
);SPC1
1100 F%=0
1110 ENDPROC
1120 :
1130 DEF PROClanding
1140 IF F%>0 THEN PRINTTAB(30,F%);SPC1
:SOUND &11,0,0,1
1150 PRINTTAB(1,7)"LANDING !"
1160 REPEAT
```
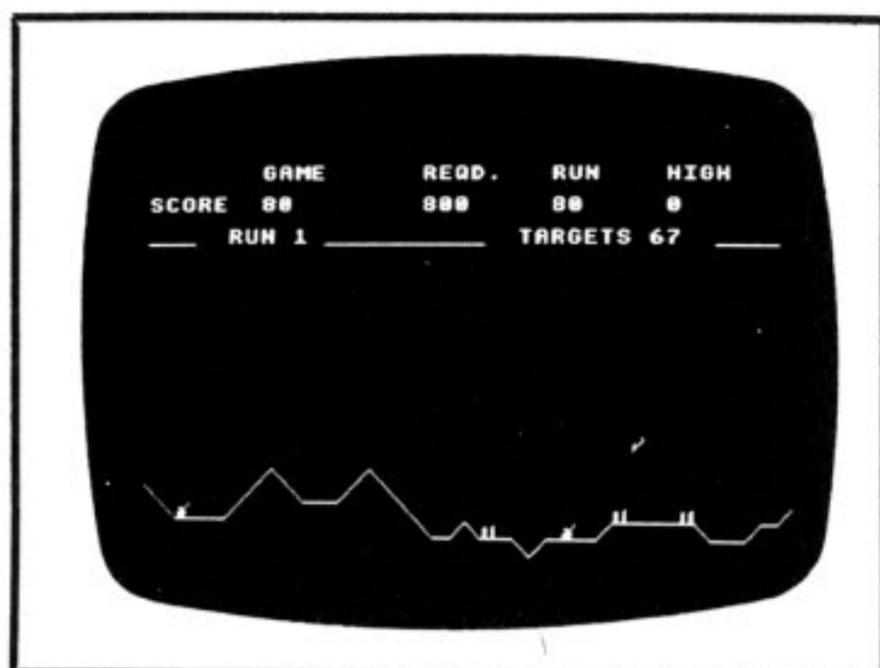
```
1170 IF K%<H%-1 THEN K%=K%+1:PRINTTAB(
30,K%-1);SPC1;TAB(30,K%);CHR$(226):SOUN
D &10,-15,3,5
1180 FOR M%=1 TO 2*DL%:NEXT
1190 IF K%=H%-1 THEN PRINTTAB(30,K%);C
HR$(243)
1200 UNTIL K%=H%-1
1210 IF W%>=RS% THEN PRINTTAB(1,7)"MIS
SION ACCOMPLISHED !" ELSE PRINTTAB(1,7)
"FAILED REQUIRED SCORE.":Z%=-50
1220 S%=S%+100:PRINTTAB(7,3);S%
1230 FOR M%=0 TO 10000:NEXT
1240 ENDPROC
1250:
1260 DEF PROCinit
1270 DIM SET 400
1280 FOR PASS=0 TO 1
1290 P%=SET
1300 [
1310 OPT PASS*2
1320 .SET    LDA #&38:STA &74:LDA #&72:
STA &75\set start
1330         LDX #39:LDA #0\clear base
move store
1340 .clear STA SET+300,X:DEX
1350        BPL clear:RTS\end when all
done
1360 .SCRLR STA SET+300\store A% move
1370        LDA &74:STA &70:LDA &75:ST
A &71\low base=start
1380        LDX #39\index for 39 squar
es
1390 .sqr   LDA #0:LDY #7\blank out sq
uare
1400 .blank STA(&70),Y:DEY:BPL blank
1410        LDA SET+300,X:STA SET+301,
X\shift move up store
1420        BEQ level:BMI up\detect mo
ve
1430        LDA &70:CLC:ADC #&38:STA &
70\move down
1440        LDA &71:ADC #1:STA &71:CLC
:BCC ind
1450 .up    LDA &70:SEC:SBC #&48:STA &
70\move up
```

```
1460        LDA &71:SBC #1:STA &71:CLC
:BCC ind
1470 .level LDA &70:SEC:SBC #8:STA &70
\back one square
1480        LDA &71:SBC #0:STA &71     •
1490 .ind   DEX:BPL sethb:RTS\end if i
ndex neg
1500 .sethb LDA &70:CLC:ADC #8:STA &72
\high base=low base+8
1510        LDA &71:ADC #0:STA &73
1520        CPX #38:BNE trans\far righ
t square?
1530        LDA &72:STA &74:LDA &73:ST
A &75:\yes-start=high base
1540 .trans LDY #7\transfer data
1550 .byte  LDA(&70),Y:STA(&72),Y:DEY:
BPL byte
1560        BMI sqr\go do next square
1570 ]
1580 NEXT
1590 CLS
1600 VDU23,224,0,80,40,16,8,5,2,0:REM
climb plane
1610 VDU23,225,0,0,0,0,97,255,0,0:REM
level plane
1620 VDU23,226,4,2,4,40,80,32,64,0:REM
dive plane
1630 VDU23,227,0,16,8,20,32,64,0,0:REM
bomb
1640 VDU23,228,0,146,84,56,254,56,84,1
46:REM expl
1650 VDU23,230,16,16,16,16,254,198,254
,255:REM cmd cen
1660 VDU23,231,8,8,8,8,8,232,232,255:R
EM cms cen
1670 VDU23,232,2,4,104,48,120,248,112,
255:REM gun
1680 VDU23,233,0,2,102,102,102,102,102
,255:REM dump
1690 VDU23,234,1,2,4,8,16,32,64,128:RE
M gnd down
1700 VDU23,235,128,64,32,16,8,4,2,1:RE
M gnd up
1710 VDU23,236,0,0,0,0,0,0,0,255:REM g
nd level
1720 VDU23,237,255,129,129,129,129,129
,65,129:REM runway
1730 VDU23,240,0,0,0,0,0,0,32,64:REM s
hot
1740 VDU23,241,8,8,8,8,8,8,0,0:REM tra
il
1750 VDU23,242,24,8,8,8,8,24,24,8:REM
hit plane
1760 VDU23,243,0,0,0,0,0,97,255,72:REM
taxi plane
1770 REM * define envelopes *
1780 ENVELOPE 1,130,5,3,5,50,60,70,0,0
,0,0,0,0
1790 ENVELOPE 2,128,10,0,0,6,0,0,0,0,0
,0,0,0
```

```
1800 DIM HSC%(6),NAME$(6)
1810 FOR M%=0 TO 6:HSC%(M%)=0:NAME$(M%
)=STRING$(11,CHR$32)
1820 ENDPROC
1830:
1840 DEF PROCnewgame
1850 DL%=392:S%=0:Y%=21
1860 RS%=700:RN%=0
1870 ENDPROC
1880:
1890 DEF PROCnewrun
1900 FOR M%=0 TO 38
1910 PRINTTAB(M%,5);CHR$(236);TAB(M%,2
0);CHR$(237)
1920 NEXT
1930 CALL SET
1940 IF RS%<2500 THEN RS%=RS%+100
1950 IF DL%>2 THEN DL%=DL%-30
1960 RN%=RN%+1
1970 IF Y%>10 THEN Y%=Y%-1
1980 G%=236:H%=20:K%=19:L%=243:N%=K%
1990 F%=0:B%=0:D%=0:U%=0:T%=0
2000 J%=0:ch%=&10:W%=0:Z%=100:X%=15
2010 PRINTTAB(7,1)"GAME";TAB(17,1)"REQ
D.";TAB(25,1)"RUN";TAB(32,1)"HIGH";TAB(
0,3)"SCORE";TAB(7,3);S%;TAB(17,3);RS%;T
AB(25,3);W%;TAB(32,3);HSC%(0)
2020 PRINTTAB(3,5)"  RUN ";RN%;SPC1;TA
B(21,5)"  TARGETS ";Z%;SPC1
2030 PRINTTAB(30,K%);CHR$(L%)
2040 R%=RND(-TIME)
2050 ENDPROC
2060:
2070 DEF PROCplanehit(k%)
2080 Z%=-50
2090 IF ?(22775+320*k%)<>64 THEN PRINT
TAB(30,k%+225-L%);SPC1;TAB(30,k%);CHR$(
228):SOUND &10,-15,6,20:ENDPROC
2100 SOUND &10,-15,6,20
2110 L%=242
2120 REPEAT
2130 PRINTTAB(30,k%);CHR$(242);TAB(30,
k%-1);CHR$(241)
2140 k%=k%+1:pk%=?(22775+320*k%)
2150 FOR M%=1 TO 1000:NEXT
2160 UNTIL pk%<>0 AND pk%<>64
2170 PRINTTAB(30,k%);CHR$(228);TAB(30,
k%-1);CHR$(241)
2180 SOUND &10,3,6,40
2190 ENDPROC
2200:
2210 DEF PROCinstructions
2220 CLS
2230 PRINT'TAB(2)"Use your aircraft to
bomb enemy";SPC(9);"installations, and
score:"
2240 PRINT'TAB(4)"SUPPLY DUMP";SPC(13)
;;CHR$(233);" 20"
2250 PRINT'TAB(4)"AA GUN";SPC(18);CHR$
(232);" 40"
2260 PRINT'TAB(4)"COMMUNICATIONS CENTR
E";SPC(3);CHR$(231);" 80"
2270 PRINT'TAB(4)"COMMAND CENTRE";SPC(
10);CHR$(230);" 160"
2280 PRINT'TAB(2)"If you reach or surp
ass the required";SPC(4);"score, you wi
ll get another run."
2290 PRINT'TAB(2)"Successive runs beco
me harder as the";SPC(4);"game speeds u
p, and the ground may";SPC(6);"become h
igher, with an increase in";SPC(6);"the
required score."
2300 PRINT'TAB(2)"If you crash or are
shot down, the";SPC(6);"game ends."
2310 PRINT'TAB(4)"AIRCRAFT CLIMB";SPC(
3);"':'"
2320 PRINT'TAB(4)"AIRCRAFT DIVE";SPC(4
);"'/'"
2330 PRINT'TAB(4)"RELEASE BOMB";SPC(5)
;"'Z' (when in dive)"
2340 PRINT TAB(2,29);"Press the Space
bar to continue :";
2350 REPEAT UNTIL GET=32
2360 ENDPROC
2370:
2380 DEF PROCtitle
2390 COLOUR 1
2400 FOR M%=6 TO 10 STEP 4
2410 PRINTTAB(5,M%);STRING$(11,CHR$(61
)):NEXT
2420 COLOUR 2
2430 PRINTTAB(5,8);"DIVE-BOMBER"
2440 COLOUR 1
2450 PRINTTAB(10,16)"by"
2460 COLOUR 3
2470 PRINTTAB(5,18)"N.Mallinson"
2480 TIME=0:REPEAT UNTIL TIME>100
2490 ENDPROC
2500:
2510 DEF PROCphstable
2520 CLS
2530 PRINTTAB(10,4)"HIGH SCORE TABLE"
2540 PRINTTAB(6,8)"No.";TAB(13,8)"NAME
";TAB(25,8)"SCORE"
```
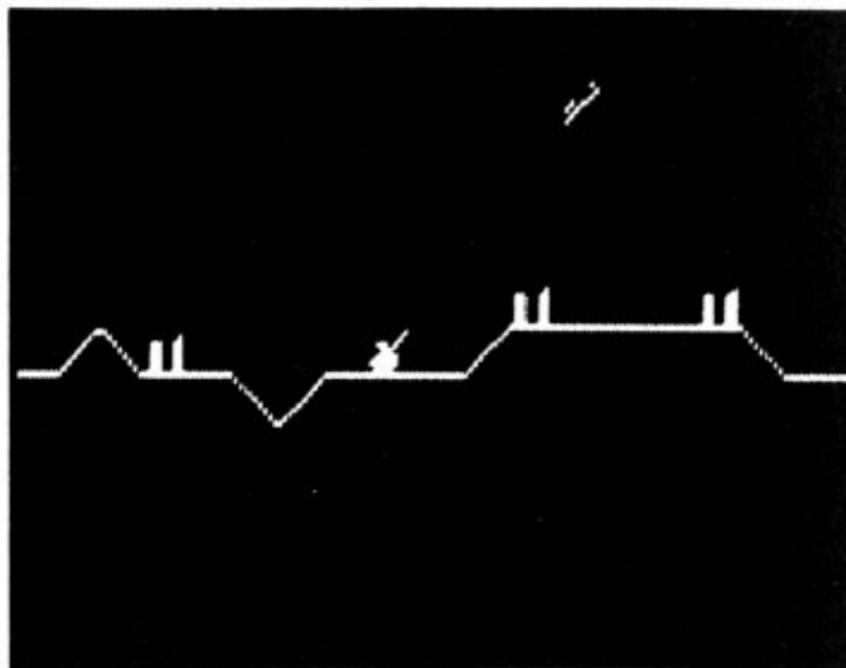
```
2550 M%=-1                              ELSE M%=M%+1:UNTIL TRUE=TRUE
2560 REPEAT                             2680 IF M%>5 THEN M%=9:ENDPROC
2570 M%=M%+1                            2690 CLS
2580 IF HSC%(M%)=0 THEN UNTIL TRUE=TRU  2700 PRINTTAB(1,7)"YOU ARE ON THE HIGH
E ELSE PRINTTAB(6,11+2*M%);M%+1;TAB(12, SCORE TABLE !"
11+2*M%);NAME$(M%);TAB(25,11+2*M%);HSC% 2710 PRINTTAB(3,12)"PLEASE INPUT YOUR
(M%):UNTIL M%=5                         NAME"''TAB(5)"(max. 10 letters)"
2590 IF HSC%(0)=0 THEN PRINTTAB(12,16)  2720 REPEAT:VDU7:PRINTTAB(10,17);SPC(1
"NO SCORE YET !"                        5):INPUT TAB(10,17)"? "N$:UNTIL LEN(N$)
2600 IF Z%=0 THEN PRINTTAB(3,27)"INSTR  <11
UCTIONS (Y/N) ?"                        2730 HSC%(M%)=S%:NAME$(M%)=N$
2610 ENDPROC                            2740 PROCphstable:M%=27
2620:                                   2750 ENDPROC
2630 DEF PROChighscores                 2760:
2640 M%=6                               2770 ON ERROR OFF:MODE 6
2650 REPEAT                             2780 IF ERR<>17 REPORT:PRINT" at line
2660 M%=M%-1                            ";ERL
2670 IF S%>HSC%(M%) THEN HSC%(M%+1)=HS  2790 END
C%(M%):NAME$(M%+1)=NAME$(M%):UNTIL M%=0
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SOUND SUPPRESSION
Sound output can suppressed by the command *FX210,n where n is any non zero value. If n equals zero, then sound output is reinstated.

LAST LINE NUMBER - Douglas Nunn
To find out the highest line number in your current program, hold down both Shift and Control and press Escape twice. Release the Shift and Control keys, which will print up 'Escape at line xxx' where xxx is the last line number.
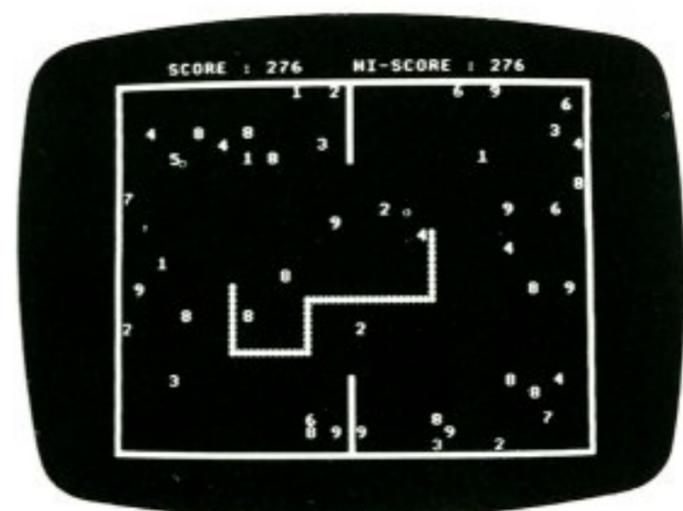
BEEBUGSOFT   Snake

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.

IDEAS, HINTS & TIPS, PROGRAMS AND LONGER ARTICLES

Substantial articles are particularly welcome and we will pay around £25 per page for these, but in this case please give us warning of anything that you intend to write.

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