£1.20

[illegible]

BEEBUG

for the BBC micro

Vol 4 No 2 JUNE 1985

BRITAIN'S LARGEST COMPUTER USER GROUP
MEMBERSHIP EXCEEDS 30,000

FEATURES

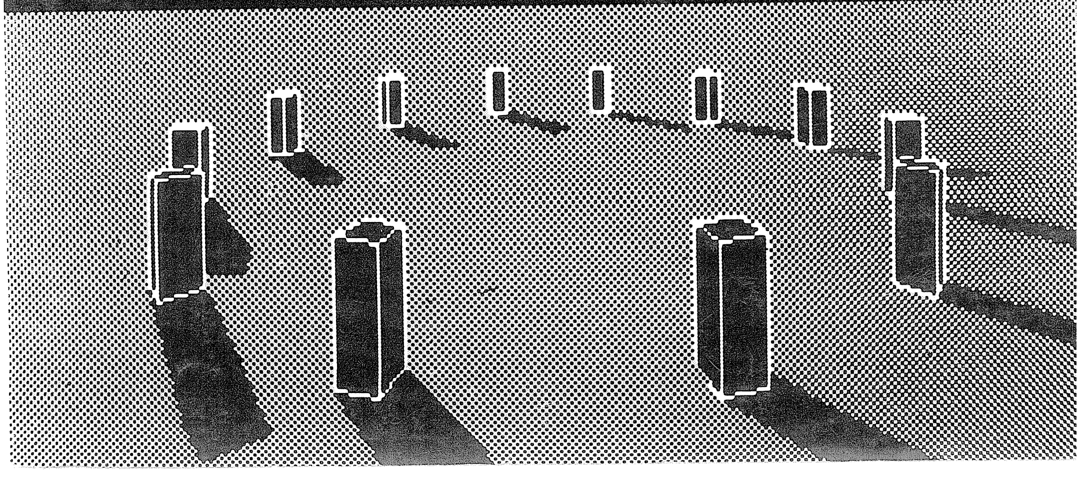
- Multiple windows
- Wordwise plus segments
- Extended disc catalogue
- Trackman

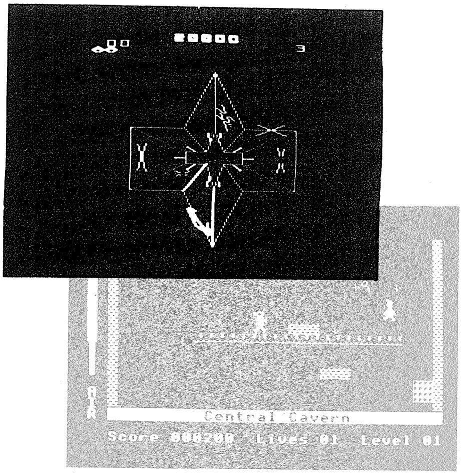
REVIEWS

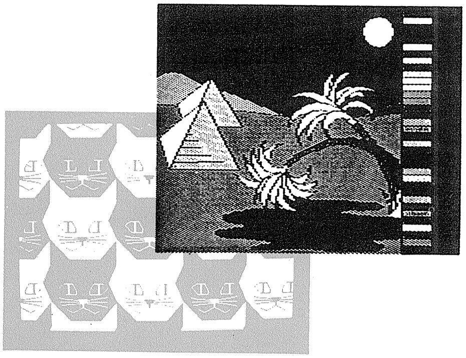
- BBC model B+
- Comal
- Games
- Home education

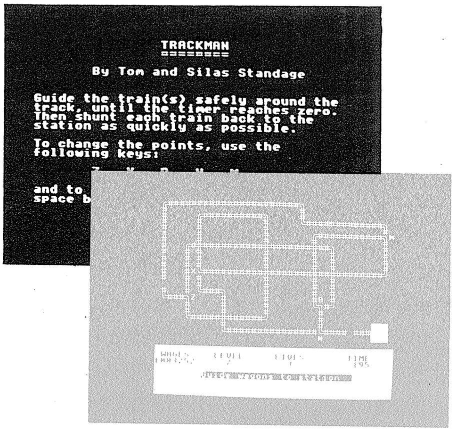
PLUS

- Adventure Games
- BEEBUG Workshop
- Beginners start here
- And much more








TRACKMAN
========

By Tom and Silas Standage

Guide the train(s) safely around the track, until the timer reaches zero. Then shunt each train back to the station as quickly as possible.

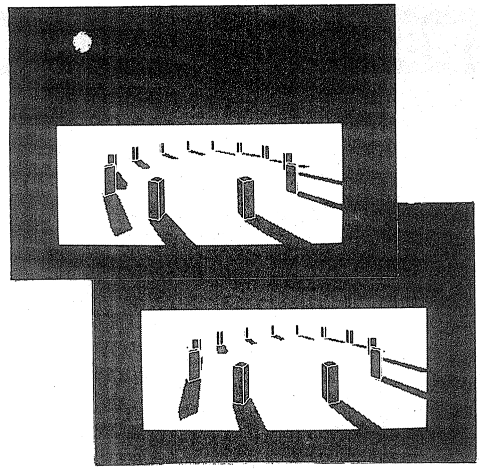
To change the points, use the following keys:

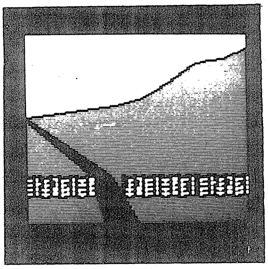
and to
space b

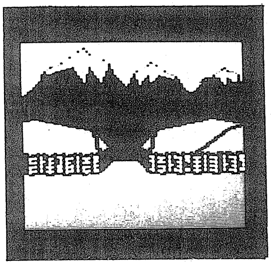
BEEBUG

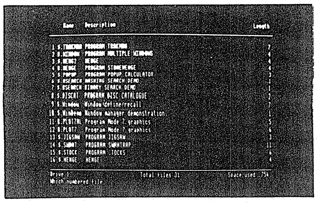
GENERAL CONTENTS

DAVID BARNES 46W









```
  Name      Description                              Length
  ---------------------------------------------------------
 1 $.TRACMAN PROGRAM TRACMAN                              7
 2 B.WINDOW  PROGRAM MULTIPLE WINDOWS                     4
 3 $.HENG2   HENGE                                        4
 4 B.HENGE   PROGRAM STONEHENGE                           4
 5 $.POPUP   PROGRAM POPUP CALCULATOR                     3
 6 $.HSEARCH HASHING SEARCH DEMO                          1
 7 $.BSEARCH BINARY SEARCH DEMO                           1
 8 $.DISCAT  PROGRAM DISC CATALOGUE                       2
 9 $.Window  Window define/recall                         7
10 $.WinDemo Window manager demonstration                 1
11 B.PLOT7AL Program Mode 7 graphics                      5
12 B.PLOT7   Program Mode 7 graphics                      6
13 $.JIGSAW  PROGRAM JIGSAW                               3
14 $.SWANT   PROGRAM SWANTRAP                            11
15 $.STOCK   PROGRAM STOCKS                               6
16 $.HENGE   HENGE                                        4
  ---------------------------------------------------------
Drive 1                  Total files 31     Space used 75k
Which numbered file
```

EDITORIAL JOTTINGS

This month we are pleased to include a first report on Acorn's new version of the BBC micro, to be known as the model B+. With its extra memory, built in disc interface and enhanced sideways ROM capability, this is a sensible, if predictable, development on Acorn's part, and we certainly wish it well. If only it had appeared twelve months ago.

We are running another competition this month, a musical one this time, to mark the conclusion of Ian Waugh's popular series on 'Making Music on the Beeb'. The first prize includes an ATPL Symphony keyboard (reviewed last month). Details on page 36.

We omitted to give due credit last month to the authors of the additional programs on the May magazine cassette/disc. The colourful and musical 'English Country Garden' was written by Brian Knott, and the first rate arcade game 'Scrumpy' was by K.J. Boorman.

This month, we have included another excellent offering for games fans, 'Swan Trap' by Peter Donn, and the winning program 'Lemming' by Mike Claxton in the Cliff Toppers Brainteaser competition, reported in full in this month's supplement.

Prize winning hints in this issue earn Bill Walker £10 and M.J. Staniforth £5. We always welcome more of your hints and tips.

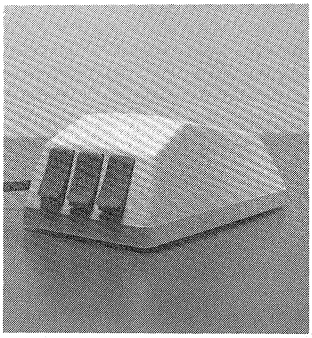
PROGRAM CLASSIFICATION

All programs in the magazine, and on the magazine cassette/disc, are marked with the symbols shown below. An unmarked symbol indicates full working, a single line through a symbol shows partial working (normally some modifications will be required), and a cross through a symbol indicates a program that will not work on that type of system.

Basic I	I	Electron	
Basic II	II	Disc	
Tube	⊖	Cassette	

MORE MICE

A rival to the AMX mouse has arisen in the form of the Magic Mouse from SMC. Like the current king, the pretender to the throne also comes with a graphics package with all the joys of the Macintosh, and a character designer. The Magic Mouse connects to the Beeb via the joystick port. Costing £59.95, this mouse is designed for both the Beeb and the Commodore 64. Further details (house training, feeding habits, etc.) from SMC on Ø1-441 1282.



MAGS

There are two new specialist mags for the letter boxes of the very rich that make you appreciate BEEBUG even more. Ed-IT is a monthly newsletter to appeal to the 'rapidly growing interface between education and technology'. Although costing a rather hefty £12Ø per year, the first two issues are free on approval. Further details from Ed-IT on Ø5Ø9-239948. Machine Intelligence News is another monthly bulletin, this time for AI freaks. A snip at £13Ø, MIN offers a three month money-back trial. Give the publishers, Oyez IBC, a ring on Ø1-236 4Ø8Ø.

PADS

The popular Pixel Pad range of graphics planning sheets has been extended with one especially for mode 7 on the Beeb. Each sheet displays a full mode 7 screen area marked up for characters and chunky pixels (unfortunately still in the near impossible to see blue ink used on the previous pads) Each pad contains 5Ø sheets and costs £3.99. Further details from Peter Bamford on Ø1-994 6477.

DOUBLE DEALINGS

Anyone clumsy with scissors will appreciate the new Disc Doubler from Associated Computer Marketing. This is a small device, costing £15 that will take a bite out of the disc sleeve exactly opposite the write permit notch so that the reverse side of the disc can be used in a single sided drive by turning the thing over. Naughty.

NEW SOFTWARE

Mosaic follow the excellent 'Saga of Eric the Viking' with 'The secret diaries of Adrian Mole'. Like Eric, the Adrian Mole adventure has been written by Level 9 computing and costs £9.95. Newcomer to software, West Wales Electronics, is offering a game that it claims is 'enjoyable, educational and cheap' and is called 'Market Gardener'. Can't say much about the first two but cheap it is at £3.5Ø. Also claimed to be educational are 'Quick thinking plus' from the ubiquitous Mirrorsoft at £7.95. and 'BBC B Mathematics' from OEP which will set you back £19.95. Icon has released a little hoard of gems: 'Drain mania', 'Caveman Capers', 'Chrysalis', and 'Contraption' - all arcade games - and 'Frankenstein 2ØØØ' - an arcade adventure. All of them cost £7.95. 'Sub Strike' is of a similar zap-em vein and costs just £6.5Ø from Tommorrow's Dream.

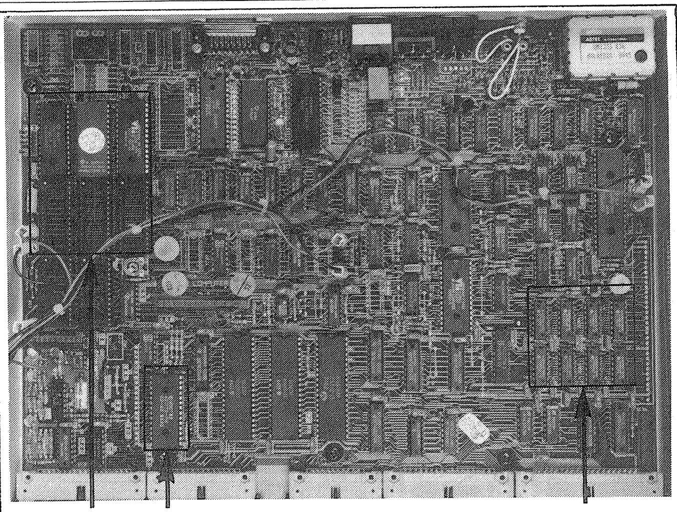
NEW BOOKS

A few more Beeb tomes have filtered through to the BEEBUG office. Artificial Intelligence enthusiasts will welcome 'Lisp - The Language of Artificial Intelligence' by A. Berk, published by Collins. From Macmillan comes 'File Handling on the BBC Micro' by Brian Townsend. Prentice Hall has been busy with two Beeb books just out. 'Mathematical Programs in BBC Basic' by Alan Whittle and 'BBC Micro Advanced Programming' by Joe Telford - a name that may be familiar to readers of another Beeb magazine.

NEW BROM SWEEPS CLEAN

Clares has released another ROM to join the throng. BROM is yet another Basic utility ROM offering the likes of a full screen editor, bad program recover, search and replace, and so forth. BROM costs £34.5Ø from Clares on Ø6Ø6-48511.

NEW BBC MICRO



Geoff Baines reports

The King is dead. Long live the King. Acorn has stopped manufacturing the model B and is concentrating on the much awaited enhanced machine, the model B+. Until stocks of the old B run out Acorn is selling the B+, only with disc interface built-in, for £499. The new machine is in an identical case to the old and looks for all the world like your own micro. The only way that you can tell a B+ is to turn it on and see the 'Acorn OS 64K Acorn 1770 DFS' start-up message. However, the circuit board has been totally redesigned (as issue 10) with new features.

PAGED MEMORY

The most important new feature is an increased RAM capacity. The model B+ incorporates a RAM paging system similar to the Aries board now marketed for the old B. It has a total RAM capacity of 64K with a little over 26K available for Basic programs in any display mode. The B+ memory map is given below.

Although only 20K is needed for paging out the screen RAM, an extra 32K is included because all the RAM is now on cheaper 64K chips. The other 12K occupies the same area of the memory map as sideways ROM - a kind of sideways RAM. However, Acorn recommends that this RAM is not used as there is no guarantee that future machines will access this area in the same way, or even at all. However, that is not to say that it won't be used if past form of software authors is anything to go by.

SIDEWAYS ROMS

Another major difference of the issue 10 B+ board is the sideways ROM sockets. There are now six sockets located at the left rear of the machine. Each socket is able to accommodate a 32K ROM as two 16K sideways ROMs addressed contiguously. One socket is filled with the Basic and operating system, both on a single ROM. This socket can, by means of links on the board, be made to operate as ROMs 15 and 14 or 0 and 1. So without repositioning ROMs you can reconfigure the machine to boot up in, say, View.

The chances are that Acornsoft will be releasing software package pairs (such as View/Viewsheet or Logo) on 32K ROMs. Other (16K or 8K) ROMs can be accommodated by altering links on the board. In any event extension ROM boards should work with the new machine just as soon as they are redesigned to fit in the new position in the case.

Both the OS (now called OS 2.0) and the Basic are identical in function to the current OS 1.2 and Basic II in all ways apart from the addition of the paging control.

DISC DRIVES

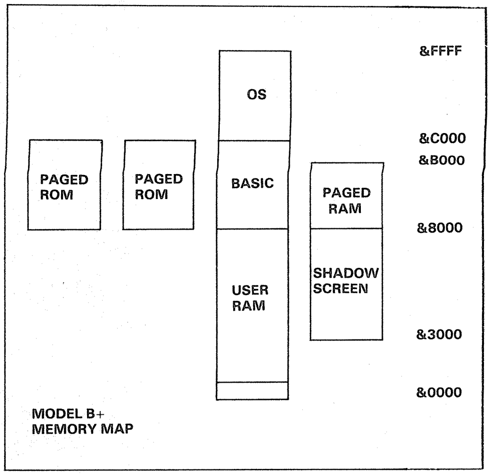
The third major difference of the B+ is the disc drive interface. This is now based on a 1770 disc controller chip, as used in the Electron Plus 3 and the ABCs. It does not, however, use the Advanced DFS (though this will operate) but the '1770 DFS'. This is largely compatible with the old DFS. Machine code programs that control the disc drive by poking to the old 8271 controller chip registers (as some protected discs do) will not operate. However, the 1770 DFS emulates most of the old 8271 commands so any program correctly using the OS calls will be okay. Elite runs on the B+.

The 1770 DFS has additional commands to the old DFS. It now includes a format command. Annoyingly this requires RAM workspace so you still need to save your program on a formatted disc before formatting another one. There is also a verify command and the *DRIVE command has been extended to allow 80 track drives to read 40 track discs. *EX performs the equivalent of *INFO *.*, *FREE displays the number of files and free sectors on a disc and *MAP shows where they occur. Somewhat out of place is *ROM which displays the names of the occupants of each sideways ROM socket.

There are other differences on the new issue 10 board as well. Seemingly most radical of these is the change from a 6502 processor to a 6512. However, the change is in fact very slight. The new processor requires different clocking hardware but as far as software is concerned it is identical to the old 6502.

The 1 MHz bus expansion no longer requires external control line 'clean-up' circuitry for reliable use and both it and the Tube interface are now properly buffered.

There are far fewer switchable links on the new board. Like the last model B board (issue 7) the trend with the B+ is to close most options with solder at the manufacturing stage. Indeed nearly every chip on the board is now soldered in position. The only socketed ones are the sideways ROMs and a few miscellaneous TTL chips. This does make for a more reliable Beeb, but it also means that repairs are now almost exclusively the province of the expert.



COMPATIBILITY

The B+ powers up as an old model B. The shadow RAM has to be enabled (use *SHADOW and then change mode, or use MODE 128 to 135) before any use can be made of the extra memory. Using the shadow screen is entirely transparent from high level languages. That is to say that in Basic you wouldn't know that the paging process was going on. You just have a Beeb with a lot of memory.

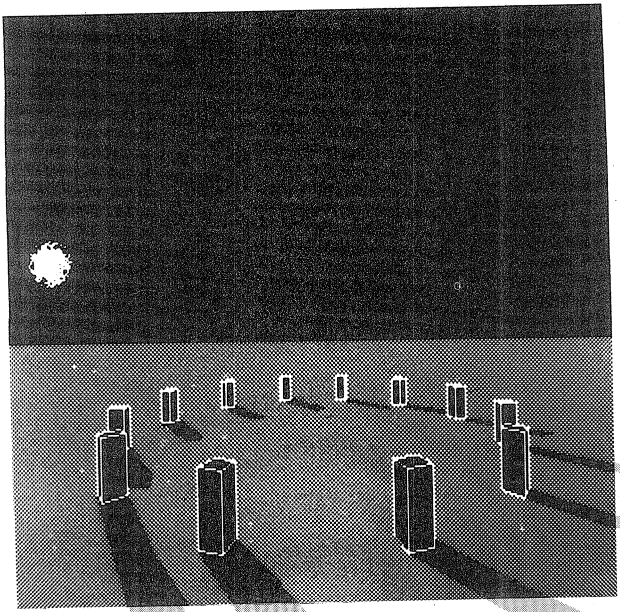
In start-up mode the B+ is claimed to be compatible with just about all the software around today for the B. However, we found that this is stretching the point a little. Many games on disc will not run because of the protection methods used on the disc. Many others just won't run. ROM software, too, is not without problems. Most of BEEBUGSOFT's repertoire will work okay in the switch-on mode but the dual screen system in both Sleuth and Exmon won't operate. View and Viewsheet work of course, and the other side will be pleased that Wordwise is alright.

Less useful, for the moment at any rate, is the B+ with the shadow RAM enabled. There is next to no software that will make use of this extra memory. View and Viewsheet will but there is still a small problem with View 2.1. Wordwise just doesn't want to know.

The B+ is certainly an improvement on the old B. The only criticism you can really level at it is: why not a year or two ago? However, for us existing Beeb owners it can only do good by prolonging the life of our machine and filling in the gap waiting for the model C...

Stonehenge

Stonehenge is the subject of Bill Walker's stunning graphics program. Watch the changing shadows as the mid-summer sun moves across the sky.



With the summer solstice nearly upon us it seems appropriate that us modern day techno-druids should celebrate the event in a more up to date manner than the usual prancing around fields in our nightshirts. We have called upon the powers of the Beeb to re-create the sun's movements over Stonehenge in the comfort of our own home. This not only saves on fares to Salisbury plain but is a lot less damaging to the monument as well.

This program will draw a black and white picture of the monoliths on the screen along with the rising sun. Then in a matter of minutes the sun's path through the sky is acted out on your TV screen. As the sun moves, the shadows of the stones move around too in this simple but remarkably effective display of Britain's oldest rock group.

The program is quite long so type it in carefully, especially the assembler section in the middle, and save it before you run it.

PROGRAM NOTES

There is nothing complicated about the principle of this program. All it does is use the tried and tested VDU19 method of animation. As the mode 1 display used can support only 4 colours and two are required for the colour switching to animate the display, this leaves only two for the visible picture. To get around this problem a third visible colour, a grey, is created by stippling white on the black background.

The grey is used for the ground around the monoliths. It is to draw this that the machine code section of the program is included.

Many of the parameters in this program can be changed to good effect. The sun's angle in the sky, the number of stones, and so on. The plethora of REM statements indicate just what is what throughout the program.

```
  10 REM PROGRAM STONEHENGE
  20 REM VERSION B0.1
  30 REM AUTHOR  BILL WALKER
  40 REM BEEBUG  JUNE 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
  70 ON ERROR GOTO 2520
  80 :
 100 MODE 1
 110 PROCsetup
```

```
  12Ø PROCinit
  13Ø FOR AZ=-4ØTO3ØSTEP1Ø
  14Ø AL=3Ø*COSRADAZ
  15Ø PROCcalc
  16Ø PROCpic
  17Ø NEXT AZ
  18Ø FORVX%=1ØØØTO2ØØØSTEP1ØØ
  19Ø PROCcalc
  2ØØ PROCpic
  21Ø NEXT
  22Ø VDU4
  23Ø END
  24Ø :
 1ØØØ DEFPROCsetup
 1Ø1Ø N%=11:REM number of stones-1
 1Ø2Ø S%=6ØØ:REM plotting scale
 1Ø3Ø VX%=1ØØØ:REM Viewpoint, screen is
plane x=VX%
 1Ø4Ø G%=-2ØØ:REM ground is at height G%
 1Ø5Ø DIM B%(N%,7,2),T%(7,1)
 1Ø6Ø REM B%() holds the 3D coords of ea
ch block's corners
 1Ø7Ø REM T%() holds the 2D coords
 1Ø8Ø AZ=3Ø:REM sun's angle round from n
orth
 1Ø9Ø AL=2Ø:REM sun's angle up from grou
nd
 11ØØ PROCcalc:REM generate shadow facto
rs from AZ and AL
 111Ø REM Def chars for sun
 112Ø VDU23,224,Ø,Ø,4,Ø,2,3,42,31
 113Ø VDU23,225,8,51,238,228,246,223,255
,255
 114Ø VDU23,226,Ø,128,32,8,192,224,224,2
5Ø
 115Ø VDU23,227,63,23,43,47,79,79,43,55
 116Ø VDU23,228,255,255,255,255,255,255,
255,255
 117Ø VDU23,229,242,252,244,251,188,238,
19Ø,228
 118Ø VDU23,23Ø,31,7,7,13,6,Ø,1,Ø
 119Ø VDU23,231,255,254,127,63,29,119,22
,44
 12ØØ VDU23,232,164,232,244,144,24Ø,192,
128,Ø
 121Ø SUN$=CHR$224+CHR$225+CHR$226+STRIN
G$(3,CHR$8)+CHR$1Ø+CHR$227+CHR$228+CHR$2
29+STRING$(3,CHR$8)+CHR$1Ø+CHR$23Ø+CHR$2
31+CHR$232
 122Ø VDU29,64Ø;512;5
 123Ø MASK=2:COL=7:REM used in palette s
witching
 124Ø REM assemble cls/ground m/c
 125Ø MC%=&AØØ
 126Ø FOR I%=ØTO2STEP2:P%=MC%
 127Ø [OPT I%
 128Ø LDA#&3Ø:STA &71;SCRN HI
 129Ø LDA#&28:STA&72;COUNT HI
 13ØØ LDA#&ØØ:STA&7Ø;SCRN LO
 131Ø TAY
 132Ø LDA&74:AND#&55:STA&75
 133Ø LDA&74:AND#&AA:STA&76
 134Ø LDA&74:EOR#&FF:STA&74
 135Ø TAX
 136Ø .LØ TXA
 137Ø AND(&7Ø),Y:STA(&7Ø),Y
 138Ø INY:BNE LØ
 139Ø INC&71:DEC&72:BNE LØ
 14ØØ LDA#&28:STA&72;
 141Ø .L1 TXA
 142Ø AND(&7Ø),Y:ORA&75:STA(&7Ø),Y
 143Ø INY:TXA
 144Ø AND(&7Ø),Y:ORA&76:STA(&7Ø),Y
 145Ø INY:BNE L1
 146Ø INC&71:DEC&72:BNE L1
 147Ø RTS
 148Ø ]:NEXT
 149Ø ?&74=&ØF
 15ØØ ENDPROC
 151Ø :
 152Ø DEFPROCinit
 153Ø REM build stones' coords
 154Ø LOCAL B%,A%,QA%,R,RC
 155Ø R=4ØØ:REM RADIUS OF RING
 156Ø RC=3ØØ:REM CENTRE OF RING ON X AXI
S
 157Ø B%=Ø
 158Ø RESTORE 16ØØ
 159Ø REM Angle of stone's visible sides
 to viewer
 16ØØ DATA-3Ø,3Ø,Ø,Ø,3Ø,-3Ø,6Ø,-6Ø,Ø,Ø,3
Ø,-3Ø
 161Ø I%=36Ø/(N%+1):FOR A%=18Ø-I%/2 TO 1
5 STEP-I%
 162Ø READ QA%
 163Ø PROCblinit(B%,A%,QA%)
 164Ø READ QA%
 165Ø PROCblinit(B%+1,-A%,QA%)
 166Ø B%=B%+2
 167Ø NEXT
 168Ø ENDPROC
 169Ø :
 17ØØ DEFPROCblinit(B%,A%,QA%)
 171Ø REM Build one stone's coords
 172Ø LOCAL C,S,X,Y,CX,CY,I%
 173Ø CX=R*COSRADA%+RC
 174Ø CY=R*SINRADA%
 175Ø S=2Ø*SINRADQA%
 176Ø C=2Ø*COSRADQA%
 177Ø X=S+CX:Y=C+CY
 178Ø B%(B%,Ø,Ø)=X:B%(B%,4,Ø)=X:B%(B%,Ø,
1)=Y:B%(B%,4,1)=Y
 179Ø X=C+CX:Y=-S+CY
 18ØØ B%(B%,1,Ø)=X:B%(B%,5,Ø)=X:B%(B%,1,
1)=Y:B%(B%,5,1)=Y
 181Ø X=-S+CX:Y=-C+CY
 182Ø B%(B%,2,Ø)=X:B%(B%,6,Ø)=X:B%(B%,2,
1)=Y:B%(B%,6,1)=Y
```

```
1830 X=-C+CX:Y=S+CY
1840 B%(B%,3,Ø)=X:B%(B%,7,Ø)=X:B%(B%,3,
1)=Y:B%(B%,7,1)=Y
1850 FORI%=ØTO3
1860 B%(B%,I%,2)=G%+8Ø
1870 B%(B%,I%+4,2)=G%
1880 NEXT I%
1890 ENDPROC
1900 :
1910 DEFPROCcalc
1920 REM shadow factors from AZ and AL
1930 SFX=COS RAD AZ/TAN RAD AL
1940 SFY=-SIN RAD AZ/TAN RAD AL
1950 ENDPROC
1960 :
1970 DEFPROCpic
1980 REM Draw pic given AZ, AL, VX%
1990 GCOL2,128+MASK:GCOL1,MASK EOR 3
2000 MASK=MASK EOR 3
2010 CALL MC%
2020 PROCsun
2030 FORZ%=Ø TO N%:PROCshadow(Z%):NEXT
2040 FORZ%=Ø TO N%:PROCbox(Z%):NEXT
2050 VDU19,1,COL;Ø;:COL=COL EOR 7:VDU19
,2,COL;Ø;
2060 ENDPROC
2070 :
2080 DEFPROCbox(B%)
2090 REM draw one stone
2100 LOCAL I%,M
2110 FORI%=ØTO7
2120 M=S%/(VX%-B%(B%,I%,Ø))
2130 T%(I%,Ø)=M*B%(B%,I%,1):T%(I%,1)=M*
B%(B%,I%,2)
2140 NEXT
2150 PLOT4,T%(4,Ø),T%(4,1):PLOT4,T%(5,Ø
),T%(5,1):PLOT87,T%(Ø,Ø),T%(Ø,1)
2160 PLOT87,T%(1,Ø),T%(1,1):PLOT87,T%(3
,Ø),T%(3,1):PLOT87,T%(2,Ø),T%(2,1)
2170 PLOT4,T%(1,Ø),T%(1,1):PLOT87,T%(6,
Ø),T%(6,1):PLOT87,T%(5,Ø),T%(5,1)
2180 PLOT5,T%(4,Ø),T%(4,1):PLOT5,T%(Ø,Ø
),T%(Ø,1):PLOT5,T%(3,Ø),T%(3,1)
2190 PLOT5,T%(2,Ø),T%(2,1):PLOT5,T%(6,Ø
),T%(6,1):PLOT5,T%(5,Ø),T%(5,1)
2200 PLOT5,T%(1,Ø),T%(1,1):PLOT5,T%(Ø,Ø
),T%(Ø,1):PLOT4,T%(2,Ø),T%(2,1)
2210 PLOT5,T%(1,Ø),T%(1,1)
2220 ENDPROC
2230 :
2240 DEFPROCshadow(B%)
2250 REM draw stone's shadow
2260 LOCAL H%,M,SØ%,S1%,S2%
2270 FORI%=ØTO7
2280 H%=B%(B%,I%,2)-G%
2290 SØ%=B%(B%,I%,Ø)+H%*SFX:S1%=B%(B%,I
%,1)+H%*SFY
2300 S2%=G%
2310 M=S%/(VX%-SØ%)
2320 T%(I%,Ø)=M*S1%:T%(I%,1)=M*S2%
2330 NEXT
2340 PLOT4,T%(Ø,Ø),T%(Ø,1):PLOT4,T%(4,Ø
),T%(4,1):PLOT87,T%(5,Ø),T%(5,1)
2350 PLOT4,T%(Ø,Ø),T%(Ø,1):PLOT87,T%(1,
Ø),T%(1,1):PLOT4,T%(5,Ø),T%(5,1)
2360 PLOT87,T%(6,Ø),T%(6,1):PLOT4,T%(1,
Ø),T%(1,1):PLOT87,T%(2,Ø),T%(2,1)
2370 PLOT4,T%(6,Ø),T%(6,1):PLOT87,T%(7,
Ø),T%(7,1):PLOT4,T%(2,Ø),T%(2,1)
2380 PLOT87,T%(3,Ø),T%(3,1):PLOT4,T%(7,
Ø),T%(7,1):PLOT87,T%(4,Ø),T%(4,1)
2390 PLOT4,T%(3,Ø),T%(3,1):PLOT87,T%(Ø,
Ø),T%(Ø,1):PLOT4,T%(1,Ø),T%(1,1)
2400 PLOT87,T%(2,Ø),T%(2,1):PLOT4,T%(3,
Ø),T%(3,1):PLOT87,T%(Ø,Ø),T%(Ø,1)
2410 PLOT4,T%(3,Ø),T%(3,1)
2420 PLOT87,T%(Ø,Ø),T%(Ø,1)
2430 ENDPROC
2440 :
2450 DEFPROCsun
2460 IF AZ>5Ø AND AZ<31Ø ENDPROC
2470 LOCAL Y%,Z%,I%
2480 Y%=8ØØ*SIN RAD AZ:Z%=9ØØ*SIN RAD A
L
2490 MOVE Y%-48,Z%+48:PRINT SUN$
2500 ENDPROC
2510 :
2520 ON ERROR OFF
2530 MODE 7
2540 IF ERR<>17 REPORT:PRINT" at line "
;ERL
2550 END
```

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

THE VALUE OF THE £ - Bill Walker

Basic variable names containing the pound sign are quite legal in BBC Basic. This can be useful to distinguish variables associated with money in financial programs, eg. total£ and pay£.

Multiple Windows

Stephen Todd describes a versatile utility that allows up to ten different windows to be created and accessed on the screen with ease.

One of the least exploited features of the BBC micro to date is its ability to split the screen into windows in any mode. The most obvious example of the recent surge to take advantage of the windowing commands is the recently released AMX mouse. The mouse ROM offers an excellent windowing function, but it is rather expensive if only the windowing function is required.

This routine allows you to select, by number, up to ten predefined text windows independently of each other, without redefining them each time you switch from one to another. Each window can be any size and place on the screen, separated from one another or overlapping. They can be called from within programs or directly from the keyboard and can be easily redefined to suit the individual. A window may be cleared on selection or left un-cleared. This option is selected when defining the window. Lastly the foreground and background colours of each window can be selected independently. The colours chosen will be used whenever that window is selected for use.

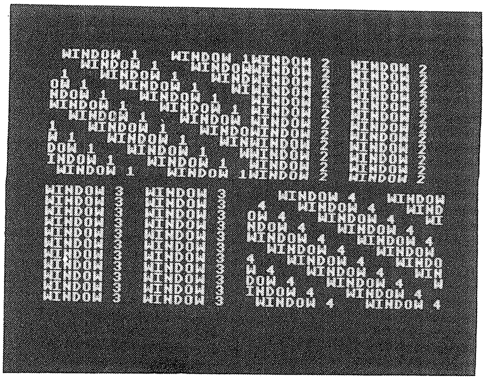
Once you have typed in, saved and run the program listed here you can save the machine code alone by typing:

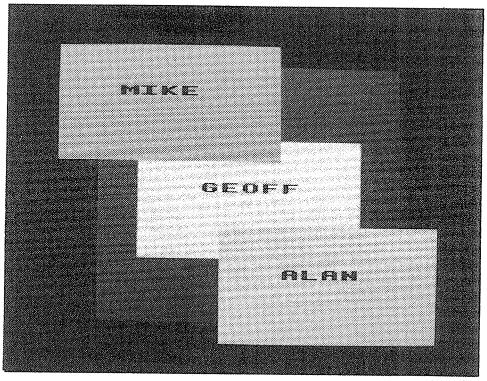
```
*SAVE WIND 0900 0AFF
```

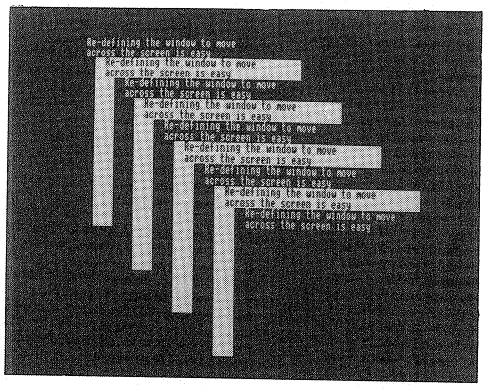
In future the routine can be loaded and run with:

```
*LOAD WIND
C%=1:CALL &0AD8
```

Once the routine has been run the windows will be available, and pressing Break or Ctrl-Break will not affect the window definitions. If it becomes necessary to remove the window facility this can be done by typing *FX247 and pressing Break once.







```
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
Re-defining the window to move
across the screen is easy
```

<u>DEFAULT WINDOWS</u>

The routine incorporates default definitions for all ten windows:

window 1	top left quarter in 40 column mode.
window 2	top right quarter in 40 column mode.
window 3	bottom left quarter in 40 column mode.
window 4	bottom right quarter in 40 column mode.
window 5	centre of screen in 40 column mode.
windows 6-10	similar to above but for 80 column modes.

YOUR OWN WINDOWS

The selection and definition of windows is done by way of a seldom used VDU code. Code six is normally used to enable the VDU drivers. It is therefore redundant in most programs and can be used to call up the window routine. Programs which make use of VDU6 will still work since this code is still sent to the VDU drivers when selecting and defining windows.

After the routine has been called a window can be selected by typing VDU6,n where n is the window number required. The range of n is one to ten. Selection of an illegal window number will give an error message 'Bad window'. Since both of the numbers in this VDU command can be typed directly at the keyboard the windows can be called with only two key presses. To call a window type Ctrl-F and then one of Ctrl-A to Ctrl-J. The letters A to J represent the window numbers one to ten.

Redefining windows is a little more involved. The command for this uses the VDU6 code again but expects more than just the window number. The command is VDU6,n+10,a,b,c,d,bc,fc. The number n is the number of the window to be defined. Note that ten must be added to the window number for this command - necessary to distinguish between calling and defining window commands.

The parameters a,b,c and d define the window size and position. These are exactly as the standard VDU28 window command (page 58 of the User Guide). 'a' is the left x value and 'd' is the top y value.

The first parameter 'a' has an additional function. It is this parameter which determines whether a window is cleared when it is selected. If the window is defined with the normal value of 'a', the window will be cleared when it is selected. If 128 is added to the normal value of 'a', that window will not be cleared when selected.

The other two parameters, bc and fc, are the background and foreground colours of the window. These are the same as for the Basic COLOUR command (page 162 in the User Guide). The effect is identical to the user typing or including in the program the statements 'COLOUR bc:COLOUR fc' after the window has been selected. The window will only be completely filled with the background colour if it is set to clear on selection.

After the windows have been defined it may at some time be necessary to recall the previous definitions. This is done by typing VDU6,0 (or CTRL-F,CTRL-@). The previous values will be printed out one per line. Each line begins with the window number and is followed by the eight parameters required for a window definition. The parameters are laid out in the same order as they would be typed when defining a window. This allows simple redefinition using the cursor, copy and delete keys.

EXAMPLE WINDOWS

The mode 1 screen can be split vertically into two, each half in different colours with the following two windows. The second window is not cleared on selection.

```
VDU6,11,0,31,19,0,128,3
VDU6,12,148,31,39,0,129,2
```

These windows can be selected with VDU6,1 and VDU6,2 respectively.

MACHINE CODE

The assembled machine code is entirely transparent to the machine user assuming the two pages allocated to storage of the machine code are not interfered with. The default location is in pages nine and ten which will suit disc users not using the RS423 interface. Cassette users will have to relocate the machine code by changing the value of mach% in line 110 if the cassette interface is to be used after the routine has been run (the CALL address given above will also have to be changed). If not, the code can be left in the original place. It is important to note that the machine is likely to hang if this code is corrupted. The user is therefore advised to find a safe place in memory for the code. This could be done by moving PAGE up by 512 or by reducing HIMEM by 512 after loading the code at the default values of PAGE or HIMEM.

The program as listed will operate with Basic I and II. However Basic II users can save on typing by ignoring lines 21ØØ onwards and replacing all appearances of FNEQUB(n), FNEQUW(n), FNEQUD(n), and FNEQUS(s$) with EQUB(n), EQUW(n), EQUD(n), and EQUS(s$) respectively.

```
   1Ø REM PROGRAM MULTIPLE WINDOWS
   2Ø REM VERSION BØ.2
   3Ø REM AUTHOR  S.A. Todd
   4Ø REM BEEBUG  JUNE 1985
   5Ø REM PROGRAM SUBJECT TO COPYRIGHT
   6Ø :
  1ØØ IF ?&287=&4C ?&287=Ø:CLS:PRINT''"P
ress break and re-run":END
  11Ø IF HIMEM>&8ØØØ mach%=&B6ØØ:HIM=&B8
ØØ ELSE mach%=&9ØØ:HIM=&8ØØØ
  12Ø PROCassem
  13Ø MODE 4
  14Ø VDU6,1:FOR I%=Ø TO 21:PRINT" WINDO
W 1  ";:NEXT
  15Ø VDU6,2:FOR I%=Ø TO 23:PRINT"WINDOW
 2  ";:NEXT
  16Ø VDU6,3:FOR I%=Ø TO 23:PRINT"WINDOW
 3  ";:NEXT
  17Ø VDU6,4:FOR I%=Ø TO 21:PRINT" WINDO
W 4  ";:NEXT
  18Ø REPEAT UNTIL GET=32
  19Ø VDU6,15,5,2Ø,15,1Ø,131,4
  2ØØ MODE2
  21Ø VDU6,5:FOR I%=Ø TO 13:PRINT"BEEBUG
 ";:NEXT:PRINT"windows";
  22Ø REPEAT UNTILGET=32:MODE 7
  23Ø END
  24Ø :
 1ØØØ DEFPROCassem
 1Ø1Ø FOR L%=Ø TO 3 STEP 3
 1Ø2Ø P%=mach%
 1Ø3Ø [OPT L%
 1Ø4Ø .pchar OPT FNEQUB(&2Ø)
 1Ø5Ø .oldvec  OPT FNEQUW(Ø):OPT FNEQUB(
&6Ø)
 1Ø6Ø .flag OPT FNEQUB(Ø)
 1Ø7Ø .screen OPT FNEQUB(Ø)
 1Ø8Ø OPT FNEQUD(&ØØ13ØCØØ):OPT FNEQUW(&
18Ø):OPT FNEQUD(&ØØ27ØC14):OPT FNEQUW(&1
8Ø)
 1Ø9Ø OPT FNEQUD(&ØD1318ØØ):OPT FNEQUW(&
18Ø):OPT FNEQUD(&ØD271814):OPT FNEQUW(&1
8Ø)
 11ØØ OPT FNEQUD(&Ø62112Ø6):OPT FNEQUW(&
18Ø)
 111Ø OPT FNEQUD(&ØØ27ØCØØ):OPT FNEQUW(&
18Ø):OPT FNEQUD(&ØØ4FØC28):OPT FNEQUW(&1
8Ø)
 112Ø OPT FNEQUD(&ØD2718ØØ):OPT FNEQUW(&
18Ø):OPT FNEQUD(&ØD4F1828):OPT FNEQUW(&1
8Ø)
 113Ø OPT FNEQUD(&Ø64314ØC):OPT FNEQUW(&
18Ø)
 114Ø .temp OPT FNEQUW(Ø)
 115Ø .templ OPT FNEQUW(Ø):OPT FNEQUB(Ø)
 116Ø :
 117Ø .printvalues
 118Ø pha:txa:pha:tya:pha:lda #Ø:sta fla
g
 119Ø lda #&1F:sta temp:lda #&99:sta tem
p+1
 12ØØ bit HIM:bmi setup:lda #&F1:sta tem
p:lda #&98:sta temp+1
 121Ø .setup
 122Ø ldx #Ø:ldy #1:sty templ+1
 123Ø .loopl
 124Ø stx templ:txa:clc:adc #1:cmp #1Ø:b
eq notab
 125Ø pha:lda #32:jsr pchar:pla
 126Ø .notab
 127Ø jsr printnum:lda #32:jsr pchar:jsr
 pchar:jsr pchar:lda #ASC"6"
 128Ø jsr pchar:lda #ASC",":jsr pchar:ld
a #6:sta templ+2
 129Ø lda #32:jsr pchar:lda templ
 13ØØ clc:adc #11:jsr printnum
 131Ø .loop2
 132Ø lda #ASC",":jsr pchar:lda #32:jsr
pchar:ldy templ+1:lda screen,Y
 133Ø cmp #11:bcs notgreater:pha:lda #AS
C"Ø":jsr pchar:pla
 134Ø .notgreater
 135Ø jsr printnum:inc templ+1:dec templ
+2:bne loop2
 136Ø jsr &FFE7:ldx templ:inx:cpx #1Ø:bn
e loopl
 137Ø pla:tay:pla:tax:pla:rts
 138Ø :
 139Ø .vec
 14ØØ pha:lda flag:cmp #1:beq number
 141Ø cmp #Ø:bne definingwindow
 142Ø pla:cmp #6:beq checkdefine
 143Ø jmp (oldvec)
 144Ø :
 145Ø .number
 146Ø pla:bne notinfo:jmp printvalues
 147Ø .notinfo
 148Ø .checkcode
 149Ø cmp #11:bcc drawwindow
```

→

```
 1500 cmp #21:bcs badwindow::pha:sec:sbc
 #10
 1510 jsr calcoffset:inc flag:pla:rts
 1520 :
 1530 .badwindow
 1540 lda #0:sta flag:brk
 1550 OPT FNEQUB(&FF):OPT FNEQUS("Bad wi
ndow"):brk
 1560 :
 1570 .definingwindow
 1580 pla:pha:sta temp:txa:pha
 1590 ldx screen:lda temp:sta screen,X
 1600 inc screen
 1610 pla:tax:inc flag:lda flag
 1620 cmp #8:beq enddefine:pla:rts
 1630 .enddefine
 1640 lda #0:sta flag:pla:rts
 1650 :
 1660 .checkdefine
 1670 pha:txa:pha:tya:pha
 1680 lda #&DA:ldy #&FF:ldx #0:jsr &FFF4
 1690 txa:bne ctrlcode:inc flag
 1700 .ctrlcode
 1710 pla:tay:pla:tax:pla:jmp (oldvec)
 1720 :
 1730 .drawwindow
 1740 pha:lda #0:sta flag:pla:pha
 1750 jsr calcoffset
 1760 txa:pha:tya:pha:ldx screen
 1770 lda #28:jsr pchar:ldy #4
 1780 lda screen,X:php
 1790 .writewindow
 1800 lda screen,X:and #&7F:jsr pchar:in
x:dey
 1810 bne writewindow
 1820 lda #17:jsr pchar:lda screen,X:jsr
 pchar:inx
 1830 lda #17:jsr pchar:lda screen,X:jsr
 pchar
 1840 plp:bmi noCLS:lda #12:jsr pchar
 1850 .noCLS pla:tay:pla:tax:pla:rts
 1860 :
 1870 .calcoffset
 1880 sta temp:txa:pha:ldx temp:lda #1
 1890 .count
 1900 dex:beq endcount:clc:adc #6:bne co
unt
 1910 .endcount
 1920 sta screen:pla:tax:rts
 1930 :
 1940 .printnum
 1950 sta &2A:lda #0:sta &2B:jmp (temp)
 1960 :
 1970 .init
 1980 bcc over
 1990 lda &20E:sta oldvec:lda &20F:sta o
ldvec+1
 2000 lda #vec MOD 256:sta &20E
 2010 lda #vec DIV 256:sta &20F
 2020 .over
 2030 rts
 2040 .e
 2050 ]:NEXT
 2060 P%=&287:[OPT 0:jmp init:]
 2070 C%=1:CALLinit
 2080 ENDPROC
 2090 :
 2100 DEFFNEQUB(Byte):LOCAL byte,I$:byte
=Byte
 2110 ?P%=byte:IFL%<>3 P%=P%+1:=L%
 2120 IFP%<&1000PRINT;"0";~P%; ELSE PRIN
T;~P%;
 2130 I$="":IF?P%<&10I$="0"
 2140 I$=" "+I$+STR$~?P%:PRINTI$:P%=P%+1
:=L%
 2150 :
 2160 DEFFNEQUW(Word):LOCAL word,I$,I1$:
word=Word
 2170 ?P%=word MOD 256:P%?1=word DIV 256
:IFL%<>3 P%=P%+2:=L%
 2180 IFP%<&1000PRINT;"0";~P%; ELSE PRIN
T;~P%;
 2190 I$="":IF?P%<&10I$="0"
 2200 I1$="":IF?(P%+1)<&10I1$="0"
 2210 I$=" "+I$+STR$~?P%:I1$=" "+I1$+STR
$~?(P%+1)
 2220 PRINTI$;I1$:P%=P%+2:=L%
 2230 :
 2240 DEFFNEQUD(Dword):LOCAL dword,I%,I$
,I1$:dword=Dword
 2250 !P%=dword:IFL%<>3 P%=P%+4:=L%
 2260 IFP%<&1000PRINT;"0";~P%; ELSE PRIN
T;~P%;
 2270 I1$="":FORI%=P%TOP%+2:I$="":IF?I%<
&10I$="0"
 2280 I1$=I1$+" "+I$+STR$~?I%:NEXT:PRINT
I1$'"     ";
 2290 I$="":IF?(P%+3)<&10I$="0"
 2300 I$=" "+I$+STR$~?(P%+3):PRINTI$:P%=
P%+4:=L%
 2310 :
 2320 DEFFNEQUS(Str$):LOCAL str$,Len,C%,
I%,I$:str$=Str$:Len=LENstr$
 2330 FORC%=1TOLen:?(P%+C%-1)=ASCMID$(st
r$,C%,1):NEXT
 2340 C%=0:IFL%<>3 P%=P%+Len:=L%
 2350 IFP%<&1000PRINT;"0";~P%; ELSE PRIN
T;~P%;
 2360 I%=P%:REPEAT:I$="":IF?I%<&10I$="0"
 2370 I$=" "+I$+STR$~?I%:PRINTI$;:I%=I%+
1
 2380 C%=C%+1:IFC%MOD3=0ANDI%<>P%+Len C%
=0:PRINT'"     ";
 2390 UNTILI%=P%+Len:PRINT:P%=P%+Len:=L%
```

COMAL

A New Language for the Beeb from Acornsoft

Comal is a structured form of Basic that is now beginning to find favour, particularly in educational circles. Richard Lambley tells you why Basic on his Beeb is now taking a back seat.

Product : Comal
Supplier: Acornsoft
Price : £49.85 including VAT.
Additional copies of the manual are available separately at £10 each.

Comal is the one of the latest of the increasing range of computer languages for the Beeb. Learning a new language can be a formidable undertaking: not only do you have to discover again how to make your computer do even the simplest things, you also have quite a lot to unlearn. But Comal is different, at least for Basic users.

Flick through the reference section of Acorn's handbook and you'll see the same keywords and expressions. Look at the example programs and you'll find most of the familiar features of Basic: there are numbered lines, the same variable types, similar syntax and so on.

But if Comal is so much like Basic, why bother with it? A major reason is Comal's powerful control structures. Many of the advanced features of BBC Basic (the REPEAT-UNTIL loop, for example) are really borrowings from Pascal. Comal takes these borrowings a stage further - and the result is a language which combines the ease of use of Basic with the power and efficiency of Pascal, without too much of that language's intimidating academic rigour. You can be pecking out Comal programs at the keyboard within minutes of first switching on.

A writer elsewhere has likened Comal to a fairground packed with free rides. Here's just one of them. With a REPEAT-UNTIL loop, the code inside the loop is always executed at least once: not always convenient. But with Comal's WHILE-DO loop, it need not be executed at all. For example:

```
10 WHILE NOT EOD DO
20   READ language$
30   PRINT language$
40 END WHILE
50 // rest of program goes here
60 END
70 DATA Basic,Forth,Pascal,Comal
```

Note, in passing, the useful EOD, which allows you to find whether you have reached the 'End-Of-Data'. When the loop has printed the final item, the computer finds that the condition in line 10 is no longer satisfied and it jumps straight to line 50 without generating an 'out of data' error. If all the data had been used up previously, this loop would have been by-passed altogether.

The indents on lines 20 and 30, incidentally, are not mine, but were put in automatically by Comal. The philosophy behind Comal is readability; and in a long listing, this special formatting really does help the structure of the program and its flow of control to stand out.

Because of the emphasis on structure multi-statement lines are in general not allowed. So statements such as IF-THEN-ELSE have to be split over several lines. But you find then that you can see much more clearly what they're doing.

Indeed, decision-making is one of Comal's strong points. For the more complicated decision, where nested IF-THEN statements would be too tangled and messy, there is CASE-WHEN, which is a bit like Basic's ON-GOTO, only more so: it's useful where a variable can have a variety of possible values, and you want a different piece of code to be executed for each one.

So although a Comal program may have more lines than its equivalent in BBC Basic, it may actually be more compact. And if Comal is a little slower than Basic in some departments (loops seem to be about 30% slower though graphics and maths are fast), it may therefore not matter much. But debugging in Comal is much easier - and up to ten times quicker, according to some estimates.

The elegance of Comal makes using it a constant pleasure. Another of its cunning devices is the sub-string specifier, which enables you to perform effortless surgery on strings in a way you never could with LEFT$, RIGHT$ and MID$:

```
  1Ø a$:="Acornsoft"; b$:="BBC Micro"
  2Ø FOR i:=1 TO 5 DO
  3Ø   PRINT a$(1:i), b$ (-i), "Comal"(1
:-i)
  4Ø NEXT i
]run
A          o          Comal
Ac         r          Coma
Aco        c          Com
Acor       i          Co
Acorn      M          C
```

In the case of a$, the loop prints from character number 1 to character number i. If you put a negative value in the brackets (as with the other examples above), it counts from the right-hand end of the string.

A further attraction of Acorn's Comal is the program editor. Like the editor in the ZX Spectrum it checks each program line as you enter it, and if you've committed a syntax error, it echoes the line with your mistake marked by an arrowhead. If you're using auto line numbering, it even offers you the same number again for a second try.

For readability the editor converts Comal keywords to capital letters, and the names of variables, procedures and functions to lower-case. It also adds certain details of Comal's formal syntax, such as the colon and the DO in line 2Ø above and the 'i' in line 4Ø. When you run the program, Comal helpfully inspects it for structural errors such as unclosed loops before starting.

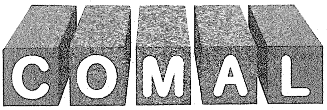
As languages go, Comal is a fairly new one. It was devised in Denmark a decade ago, but it's already in extensive use in Scandinavia and the Netherlands, especially in schools, where it's seen as a way of avoiding the muddled thinking and untidy programming often encouraged by Basic. Comal does have the GOTO statement so detested by some, though it obliges you to GOTO program labels rather than to line numbers. Even so, the manual sternly discourages its use!

Nearer home, Comal is the standard computing language in Irish secondary schools. It has also been recommended officially to schools in Scotland.

The Acorn version conforms to the Comal-8Ø standard, but includes many extensions for the BBC Micro - such as sound and graphics commands, which work just as they do in Basic. The Comal interpreter comes in a 16K ROM, accompanied by an excellent 44Ø-page manual, which is much easier going than the User Guide. A pity, though, that it isn't ring-bound.

I find I've said nothing about Comal's file-handling commands (flexible, yet far simpler to understand than their Basic counterparts), about PRINT USING, about CONT (which allows you to resume program execution after an escape) or about the profusion of other new toys.

But to sum up, my impression of Comal is that it's far too good to be shut inside the schoolroom. If I had to start from scratch again, it's the language I would go for. In my own computer, I've evicted Basic from the highest-priority ROM socket and when programming I use it only for the assembler, which Comal sadly lacks.



HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

EDITING !BOOT FILES - Paul Baron

To make a change to a !BOOT file would seem to require you to *TYPE it and then *BUILD a new one incorporating the change - very tedious with a long !BOOT file. Instead load the !BOOT file into Wordwise using the normal option 2, edit (easy in Wordwise) and save it again as !BOOT. Needless to say don't use any embedded commands!

Making the most of Wordwise Plus

Stephen Ibbs, an enthusiast of the new Wordwise Plus, has collected together all his best ideas on making the most of this word processing package, and shows how you can easily construct your own segment programs.

The arrival of Wordwise Plus opens up a whole new area of development for users happy and familiar with the old Wordwise. Wordwise Plus includes a programming language that is used to expand the range of functions already existing in the word processor and to customise it to a user's particular needs. Such programs are written, using the word processor itself, in areas of memory called 'segments'. The cassette which accompanies Wordwise Plus contains some segment procedures - continuous processing, name-sorting, and dual columns - that will start the user thinking. This article aims to further expand the user's horizons.

In the hints that follow, f1 means press function key 1 (Green), and f2 means function key 2 (white).

AUTO !BOOTING

Unlike the original Wordwise, extra commands can be placed in a disc auto-boot file to operate after Wordwise is called. Useful segment procedures do not therefore have to be loaded individually. The technique is simple: Start the !BOOT file with your usual function key definitions and a call to Wordwise. Any other command from the language should then be preceded with a colon (:). So, for example, to define three keys and then load segments Ø and 1 with procedures the !BOOT file might look like this

```
*KEYØ |!!OC 27,89|!"
*KEY1 |!!OC 14|!"
*KEY2 |!!OC 15|!"
*W.
:SELECT SEGMENT Ø
:LOAD TEXT "filename"
:SELECT SEGMENT 1
:LOAD TEXT "filename"
:SELECT TEXT
```

TELEPHONE INDEX

Using the NAMSORT and DUALCOL procedures included in the package it is very easy to produce an alphabetical phone list, with two justified columns on one sheet of paper. The way to do it is:

1. At the top of the main text area type in f1LL33f2 <return>.
2. Type in the name followed by f1FIf2 then the telephone number <Return>. Use the same format for each name, e.g. surname (in upper case) followed by Christian name or initials.
3. When the list is finished (in any order), save the file for security with filename A.
4. Spool the file, using menu option 8, with filename B.
5. Load filename B back into the main text area using menu option 2. This will have added the necessary spaces and removed the embedded commands.
6. Load the DUALCOL procedure (on the cassette) into segment Ø.
7. Load the NAMSORT procedure (on the cassette) into segment 1.
8. Go back to the main text area and press Shift-f1. If the list is long this sorting will take some time, but the end result will be an alphabetically sorted list.
9. Save this with filename C using menu option 1.
1Ø. Enter the segment Ø procedure and change the filename to filename C.
11. Press Shift-fØ

The end result will be a double column of names and numbers in alphabetical order with both the right and left edges of each column straight and justified. Afterwards the spare files can be deleted. The sorting procedure is the lengthy process, but it has probably taken you longer to read this!

PAGED PREVIEW

It is sometimes nice to preview the text a screen at a time, rather like the paged viewing of a Basic program, and this can be done in Wordwise Plus using the following simple technique:

1. Into any segment (say 8) simply type VDU14
2. At the top of the main text type f1SEG8f2

When the text is now previewed with menu option 7, it will be displayed a portion at a time, moved on with the Shift key.

LOCAL PREVIEW

It is sometimes advantageous to quickly preview text you have just added or altered, in its context. One obvious way is to move back a few lines, add a marker, move forward, add another and do a 'Marked Preview' from the menu, but this is slow. The segment procedure below will do all this automatically. It replaces the letter at the current cursor position with {, then places markers 10 lines before, and 15 lines after it.

There is a pause command embedded, and the text is previewed. A press of any key will then delete the markers, and replace the letter.

```
REM PREVIEW TEXT
DEFAULTS
SELECT TEXT
DELETE MARKERS
D$=GCT$
DELETE LEFT
TYPE "{"
CURSOR UP 10
CURSOR AT 0
FKEY 3
CURSOR DOWN 25
CURSOR AT 0
TYPE "|GPA|W "
FKEY3
PREVIEW MARKED
CURSOR TOP
DOTHIS
FIND MARKERS
DELETE AT
TIMES 2
CURSOR LEFT 5
DELETE AT 5
CURSOR TOP
REPLACE "{",D$
DISPLAY
```

CHAPTER NUMBERS

It is very useful to be able to number automatically chapters, sections, and sub-sections. This can be done in Wordwise Plus using the following procedures loaded into the segments shown:

```
(Segment 9)
REM A%=CHAPTER
REM B%=SECTION
REM C%=SUB-SECTION
A%=0
B%=0
C%=0
SELECT TEXT
DISPLAY

(Segment 0)
A%=A%+1
B%=0
C%=0
A$=GLK$
SELECT TEXT
X$=STR$(A%)
TYPE X$+"  "+A$
DISPLAY

(Segment 1)
B%=B%+1
C%=0
B$=GLK$
SELECT TEXT
X$=STR$(A%)
Y$=STR$(B%)
TYPE X$+"."+Y$+"  "+B$
DISPLAY

(Segment 2)
C%=C%+1
C$=GLK$
SELECT TEXT
X$=STR$(A%)
Y$=STR$(B%)
Z$=STR$(C%)
TYPE X$+"."+Y$+"."+Z$+"  "+C$
DISPLAY
```

The system is initialised by pressing Shift-f9 to reset the three variables. Press Shift-f0 for a new chapter, type the title, then press Return. It will appear with a figure 1 in front of it. Similarly a new section with Shift-f1 will appear as 1.1 and a new sub-section (Shift-f2) as 1.1.1. These numbers will then increment by one each time the segment routine is called, and Shift-f9 will reset the numbers. As with any chapter numbering system, it should be done once all the editing is complete.

TEXT TRANSFER 1

This is a very quick method for transfering marked portions of text from the main text area to, say, segment 6. It uses the disc, saving, then loading, before deleting a file called "ZZZZZZZ", so make sure a suitable disc is in the drive.

```
REM SELECT1
SELECT TEXT
SAVE MARKED "ZZZZZZZ"
SELECT SEGMENT 6
LOAD TEXT "ZZZZZZZ"
*DELETE ZZZZZZZ
SELECT TEXT
DISPLAY
```

TEXT TRANSFER 2

The second method of transfering text does not use the disc drive, but is very much slower. It is a useful alternative if you only have a cassette system. It transfers the text a character at a time using the GCT$ command. Unfortunately it is not possible to use the quicker GLT$ in this context. As with the LOCAL PREVIEW procedure above, a { is inserted into the text, and when the procedure reads this character it stops the transfer, then deletes the {.

```
REM SELECT2
SELECT TEXT
CURSOR TOP
DOTHIS
FIND MARKERS
CURSOR RIGHT
TIMES 2
TYPE "{"
CURSOR TOP
FIND MARKERS
REPEAT
A$=GCT$
SELECT SEGMENT 6
TYPE A$
SELECT TEXT
UNTIL A$="{"
SELECT SEGMENT 6
DELETE LEFT
SELECT TEXT
CURSOR TOP
REPLACE "{",""
DISPLAY
```

PARAGRAPH IDEAS

If the cursor is in the middle of a paragraph it is possible to use segment procedures to find the beginning and end of the paragraph for marking, transfering, deleting, etc. To delete from the present cursor position to the end of the paragraph is very simple using the FKEY6 command, duplicating the f6 'Delete to?'. Type this into any segment

```
SELECT TEXT
FKEY6,CHR$13
DELETE AT
DISPLAY
```

The CHR$13 is the ASCII number for the Return character. The segment is then called when desired and the remainder of the paragraph is deleted.

If the object is simply to place a marker at the end of the paragraph then this can be done in two ways, depending on whether the paragraphs are separated by 1 or 2 Returns. If the latter, the following very fast procedure will work:

```
SELECT TEXT
REPEAT
A$=GLT$
UNTIL A$=""
FKEY 3
DISPLAY
```

If, however, the paragraphs are separated by one Return only, then this procedure is much slower, and not really worth using. It is included in case it is needed, for example, as part of a larger procedure.

```
SELECT TEXT
REPEAT
A$=GCT$
UNTIL A$=CHR$13
FKEY 3
DISPLAY
```

Similarly, to go back to the beginning of the paragraph, again slowly:

```
SELECT TEXT
REPEAT
A$=GCT$
CURSOR LEFT 2
UNTIL A$=CHR$13
FKEY 3
DISPLAY
```

The cursor is moved left 2 positions each time because after GCT$ it moves one position to the right.

RIGHT JUSTIFY

As an alternative to placing FI embedded commands at the beginning of every line of a section of text (such as the address in a letter .head), this procedure uses a simple technique to do it for you. Simply press Shift and the segment number, then type each line followed by Return. When finished type [{]Return and the procedure will finish, placing the cursor in the correct position.

```
REM JUSTIFY
REPEAT
A$=GLK$
SELECT TEXT
TYPE "|GFI|W"+A$+"|R"
DISPLAY
UNTIL A$="{"
SELECT TEXT
DELETE LEFT 6
RECOUNT
DISPLAY
```

ALTERNATING MARGINS

When producing documents for double sided photocopying and binding, odd (front) pages require text to the right of the page and even (back) pages require the text to the left, to allow for the binding. This short procedure fulfills these requirements.

```
O%=10
REM margin for left pages
E%=5
REM margin for right pages
IF P%MOD2=1 THEN GOTO even
A%=O%
END
.even
A%=E%
```

In addition you must place
    f1SEGnf1LMA%f2
into the page heading (where n is the number of the segment in which the procedure is placed). The margin change thereby inserted in the heading has effect on the next page so an initial LM should be included at the beginning of the text.

ABREVIATIONS

It is very tempting, once the procedure language of Wordwise Plus has been mastered, to fill up all the segments with useful processing aids. The danger of this enthusiasm is that the memory will be used up very quickly. The answer is to use keyword abbreviations wherever possible, and remove the REM statements. It will be found that some procedures can be cut down to 1/4 of their former length. In this way many useful segment procedures can be present at a time, or they can be called up when needed from cassette or disc. Wordwise can take on a whole new appearance when bolstered with segments of your own making and become all the more useful.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SINGLE KEY STRING SEARCH - David Jupe

The following key definition sets up function key zero to search for any specified string in a Basic program:

```
*KEY0|N|!hLINEN$:P=PA.+1:@%=5:|!uN=256*?P+P?1:M=P+3:P=P+P?2:|!g|!'$M,N$)|!qN;:|!uA%=?M
:|!V&B50E:M=M+1:U.A%=13:|!q:U.?P=255EL.U.?P=255|M
```

Enter the search string after the prompt. All lines containing the string are then displayed although line numbers following GOTO and GOSUB statements are not displayed correctly. Entering a null string for the search string displays the whole program at a reduced rate. Basic I users should replace the address &B50E with &B53A.

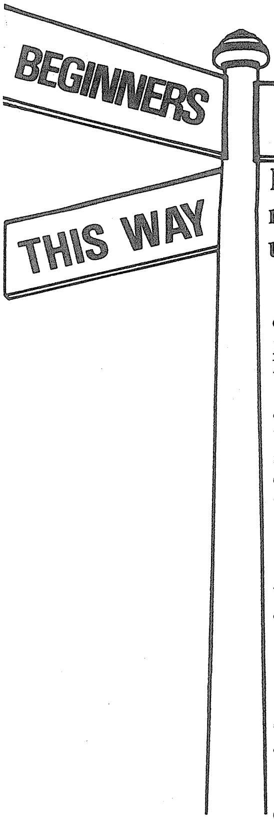
CHAINING PASCAL PROGRAMS - Mike Cope

Acornsoft Pascal lacks a CHAIN command but this can be simulated. Put the filename in the key10 buffer with

```
    oscli('key10 <filename> |M')
```

then invoke key10 with

```
    oscli('fx138,0,138')
```



INTRODUCING MACHINE CODE

Peter Lewis, who started our series for beginners, returns to describe some of the useful, and more unusual, applications of logic to our programs.

It is often claimed that a good programmer needs to be able to think logically. Be that as it may, some understanding of simple logic is essential to the writing of programs in any language including Basic. In this short article we shall be looking at a variety of situations in which logic plays an important part, and see some less obvious applications as well.

One of the most common examples of logic in programming is in the use of the IF-THEN statement. The basic format of this instruction is:

```
  IF <condition> THEN <instructions>
```

If the condition is true, then the instruction(s) after the THEN will be carried out. If not, then no action is taken. Often, in simpler programs, the THEN is followed by a GOTO and a line number. Let's take a close look first of all at how Basic handles logic.

TRUE AND FALSE

You may already be familiar with the two Basic keywords 'TRUE' and 'FALSE'. A language like Basic is not really very good at handling such values, so instead they are represented numerically as -1 and Ø. You can prove this to yourself by typing, in immediate mode:

```
    PRINT TRUE  <return>
    PRINT FALSE <return>
```

and you will see the values -1 and Ø displayed. In fact we can at any time use TRUE or -1 and FALSE or Ø. They are interchangeable, though the use of TRUE and FALSE will often result in more readable programs.

You might wonder why these two particular values are used. Well, the use of Ø for FALSE is fairly obvious, and so is the use of -1 for TRUE when you realise that in binary it becomes 11111111 (i.e. all ones).

LOGICAL EXPRESSIONS

Returning briefly to the IF-THEN statement, we can find some more useful facts about the use of logic in programs. A typical such statement might be:

```
120 IF X>=Ø AND X<=9 THEN Array(I%)=X
```

Now we know that the condition (X>=Ø AND X<=9) must be either true or false, but from our previous discussion it must equally have a value of -1 or Ø. So a logical expression has a numerical value! Just try typing some of the following in immediate mode:

```
PRINT 4=4   gives -1 (TRUE)
PRINT 4<>2*2   gives Ø (FALSE)
PRINT 2=2 OR 3=9   gives -1 (TRUE)
PRINT 7>9 OR 4<5 AND 3=4 gives Ø (FALSE)
```

You might find these results surprising at first sight, but they just serve to emphasize the connection, in BBC Basic, between the apparent logic values of TRUE and FALSE, and the numeric values of -1 and Ø. You might also have found the last example above a little confusing to work out. Logical operations such as AND and OR have rules governing the order in which they are carried out just as do the operations of add, subtract, multiply and divide. That order is:

NOT - AND - OR/EOR

The simplest solution when writing your own programs is to use brackets to make the order quite clear.

PROBLEMS WITH LOGIC

The IF-THEN or its expanded version IF-THEN-ELSE is very widely used in programming. However, it is very easy to try too much and end up thoroughly confused. Let's have a look at what is and isn't good practice here. Enter the following very short program:

```
10 PRINT"Enter value:";
20 INPUT value%
30 IF value% THEN PRINT "TRUE" ELSE PRINT
   "FALSE"
40 GOTO 10
50 END
```

Run the program and enter the following values in turn:
0,-1,127,33333,-999
You should find that 0 and -1 produce the results FALSE and TRUE as expected, but you should also find that all the other values produce TRUE as well. In fact, Basic treats 0 as FALSE and any other value as TRUE. This also has its dangers. Change line 30 above to read as follows:

```
30 IF NOT value% THEN PRINT"FALSE" ELSE
   PRINT"TRUE"
```

Now you would expect, without knowing better, that if you ran this modified program and entered the same values as before, all the results would be reversed. Try it and see. In fact, only the values of 0 and -1 produce the correct results. To see briefly why this is so we have to change the numbers into binary. The number 127 as an 8 bit binary number is 0111111. The operation of NOT reverses all the ones and zeros producing 10000000 which is 128 again in decimal, and of course, both 127 and 128 will be treated as TRUE! So be careful. Always make sure that if you use NOT it really does work as you want.

Consider another short program:

```
10 REPEAT
20 PRINT"Do you want to finish (Y or N)";
30 INPUT answer$
40 UNTIL INSTR("Yy",answer$)
50 PRINT"Finished"
60 END
```

The INSTR instruction searches to see if the character assigned to answer$ is either 'Y' or 'y' (allowing upper or lower case reply). Now this version will work quite correctly, repeatedly looping until either 'Y' or 'y' is entered. This works because INSTR returns 0 (equivalent to FALSE) if the character is not found, and 1 or 2 (both treated as TRUE) if it is (the value depending on the position).

But suppose we turned line 140 around to read:

```
40 UNTIL NOT INSTR("Nn",answer$)
```

This looks quite plausible - loop until the answer is neither 'N' nor 'n'. If you try it, you will find it will not work for just the reason explained above. If you enter 'N' or 'n' in response to the question, the INSTR function will return the value 1 or 2 (both taken as TRUE), but 'NOT 1' or 'NOT 2' will also produce TRUE and the program will terminate incorrectly.

The moral is to stick as far as possible with only the values -1 and 0 to represent TRUE and FALSE. If you do use other values be particularly wary of any use of NOT.

FLAGS

One of the most useful devices for the programmer is known as a flag. The idea is to have a variable in your program that can mark or flag the occurrence of some situation. Consider the following program (with apologies to Surac).

```
10 number%=100
20 DIM list(number%)
30 FOR I%=1 TO number%
40 list(I%)=RND(999)
50 NEXT I%
60 :
70 REPEAT
80 swapflag%=FALSE
90 FOR I%=1 TO number%-1
```

```
 1ØØ IF list(I%)>list(I%+1) THEN PROCsw
ap
 11Ø NEXT I%
 12Ø UNTIL NOT swapflag%
 13Ø :
 14Ø FOR I%=1 TO number%
 15Ø PRINT list(I%);
 16Ø NEXT I%
 17Ø END
 18Ø :
 2ØØ DEF PROCswap
 21Ø temp=list(I%)
 22Ø list(I%)=list(I%+1)
 23Ø list(I%+1)=temp
 24Ø swapflag%=TRUE
 25Ø ENDPROC
```

This program uses a bubble sort to order the elements of the array called 'list' into ascending order. The size of the list is determined by number%, set here to 1ØØ. The sort works by comparing pairs of adjacent numbers in the list and swapping them if they are in the wrong order. It is a characteristic of this sorting method that if the program makes a pass through the list of numbers without effecting any swaps, then the list is completely ordered and the program should terminate.

To do this we set up a flag, called here swapflag%, which is set to the logical value FALSE at the start of each pass through the list. Whenever two values have to be swapped, this is done by PROCswap, which also sets the flag TRUE. The main REPEAT-UNTIL continues until a pass occurs in which the flag is not set.

Flags have many uses and are used quite frequently by good programmers. They can also improve your programming style. For example, if in a REPEAT-UNTIL loop you test for some condition that will cause an early exit from the loop, don't just jump out of the loop (which can lead to problems later) but set a flag which is tested by the UNTIL. In outline:

```
REPEAT
flag%=FALSE
- - - -
IF condition THEN flag%=TRUE
UNTIL flag%
```

The same applies to a procedure. Set a flag inside the procedure, and test the flag after calling the procedure in the main program.

MORE PECULIARITIES WITH LOGIC

Because of the connection between the logical values of TRUE and FALSE, and numerical values, BBC Basic offers some intriguing and sometimes useful possibilities. For example, the following pairs of statements are equivalent:

```
IF F<>Ø THEN . . .
IF F THEN . . .

IF A=3 THEN C=C+1
C=C+(A=3)
```

one of these '+' should be '-'

```
IF Q=6 OR B=3 THEN A=B ELSE A=Ø
A=-B*(Q=6 OR B=3)

IF A=9 THEN PRINT"Ø" ELSE PRINT"1"
PRINT -NOT(A=9)
```

This idea of mixing arithmetic and logic can be put to good effect in graphics (as witnessed in several of the entries received in our 3D Surfaces competition - see supplement). Consider the following program:

```
1ØØ MODE 4
11Ø GCOLØ,1:VDU29,Ø;512;
12Ø FOR X=Ø TO 1279 STEP 5
13Ø Y=FNfunction(X)
14Ø PLOT 5+(X=Ø),X,Y
15Ø NEXT X
16Ø REPEAT UNTIL FALSE
17Ø END
18Ø :
2ØØ DEF FNfunction(X)=.......
```

The idea is that we can insert any suitable function that we like in place of the dots at line 2ØØ and this function will be plotted on the screen.

Before looking at some suitable functions, there are some other points to note. For convenience the origin has been moved (line 11Ø) to a point halfway up the lefthand side of the screen. We can thus think of the X axis as running across the centre of the screen. In line 14Ø, if X=Ø then this is true and has the value -1 thus producing a PLOT 4 instruction (same as MOVE), but otherwise is false with value Ø producing PLOT 5, equivalent to DRAW (based upon the

→32

POSTBAG

POSTBAG

PROGRAM LISTINGS

Having borrowed several copies of BEEBUG I was so impressed by it that I have now taken out a subscription for myself. One small detail bugs me however. Why are the columns of print formatted to only 39 characters when there appears to be room for 40? Not important for text but it would surely be a great help when copying a listing as this would then appear exactly the same on the screen as in the magazine. One could then spot many mistakes by simply noting the end characters in each line.

John Bridger

There is no simple reason for the 39 character line width, and its origins are lost in the mists of time. John Bridger's idea is quite useful, and with the revised format of the magazine we shall normally be listing all programs with a line width of 40 from now on.

ANOTHER LEAP

I was delighted to see the letter from Mr.D.Shaul in the April issue of the magazine, for I had tried to produce calendars for all the leap years from 1988 to 2000 inclusive, but without success.

I carried out the modification suggested by Mr.Shaul and this successfully produced the correct result for 1988. However, no luck with 1992, 1996 and 2000. Not only that, but 1984 then gave an 'incorrect' February. It will be a pity if this otherwise excellent, and amusing, program cannot have this quirk corrected.

R.Spedding

Seems we somehow slipped up on this one. However, the following amendment to the original program should do the trick for all leap years:

```
1090 IF Leap%=1 AND D%-Day
=59 D%=D%+1:Days%(D%)=29
```

We are indebted to Mr.Grilli of Loughborough for sending this mod to us. Other alternatives have also been suggested. Mr.Bradley of Cheltenham suggested a similar solution and added the further information that changing 1980 to 1904 in lines 210 and 1190, and 3 to 6 in line 1310, enables the program to be used for all years back to 1904.

WALLY OF THE MONTH

My name will not be familiar to you, but doubtless by now the episode of the wally who phoned up about the SPREADX program, and who had just missed out the '+' in line 2220 will be all round the BEEBUG coffee room. It was I!

I thought I would drop you a line to say how much I appreciated the trouble you took, even if it did not exactly tax your computing know-how. Many thanks.

<name supplied>

Well we just did our best to help. However, we are still looking for the BEEBUG coffee room.

HIGH SCORE OR HIGH SCORE?

When is a high score a high score? I have just seen in April's BEEBUG supplement a high score of 35% for the game Sabre Wulf by Ultimate. After only a couple of days of play I have mapped the full 16x16 playing area and achieved two sorts of high score: one on my last mapping run (a score of 72030 but only 68% of the adventure completed) and one when I was after all the pieces of amulet and trying to finish the game (79% complete but only 69640 points). Does either of these high scores qualify?

Bruce Goatly

We have included both of these scores in this month's new high scores table in the supplement. This is not the only game with more than one scoring feature (Elite for example). At the end of the day the best high score is probably measured by the degree of satisfaction and enjoyment that is provided by playing the game in question.

BEEBED UPDATE

The address for this ROM, reviewed in the April issue (Vol.2 No.10), is now J & O Software, 19 Lancaster Rd, Northolt, UB5 4TB.

Name	: Starmaze
Supplier	: Software Invasion
Price	: £7.95
Reviewer	: Alan Webster
Rating	: *

The packaging tempts you to "travel through the star maze to discover rare jewels and transport them back to the mothership". Not too inspiring. Already I had visions of a hundred other similar games and sure enough this one was no exception.

According to the literature, Starmaze is supposed to be a very unusual game - unusually tedious in my view. It is a mish-mash of alien shooting games, asteroids and mazes. None of these parts are implemented particularly well and the quality of the graphics is very poor indeed. No imagination seems to have gone into the design of this game.

Instructions on screen and actual documentation are virtually non-existent. The object of the game is to roam around a spacebound maze made up of a set of coloured blocks, looking for the jewels and then laboriously trying to remember your way out again to take the treasure back to the mothership. Your space ship

Title	: Manic Miner
Supplier	: Software Projects
Price	: £7.95
Reviewer	: Geoff Bains
Rating	: ****

Manic Miner is one of the few truly classic computer games. First written for the Spectrum, this Beeb version sticks closely to the original.

Your task is to guide miner Willy around the newly discovered abandoned mines of a past civilisation to find the inevitable treasure trove. Each screen displays a room fraught with obstacles and nasties to obstruct Willy. He has to collect all the keys scattered around the many levels to move on to the next room.

Manic Minor was the first of the now numerous games combining adventure elements with arcade action. It has worn very well. The graphics, based as they are on the Spectrum original, are not of the usual excellent Beeb standard. However, the displays are amply good enough to serve their purpose. Most other Beeb games would abandon graphics altogether in favour of the vast quantities of data required to describe

Name	: Tempest
Supplier	: Superior Software
Price	: £9.95
Reviewer	: Alan Webster
Rating	: ***

This is the Atari approved version of their arcade game Tempest, and the game adheres very closely to the original.

The game involves your crab like space craft moving around the outside of a 'Stargate' shooting all the aliens ascending the sectors. There are 9 different screens and an infinite number of levels. At level 49 the stargate disappears and you are left playing on a blank background - very difficult.

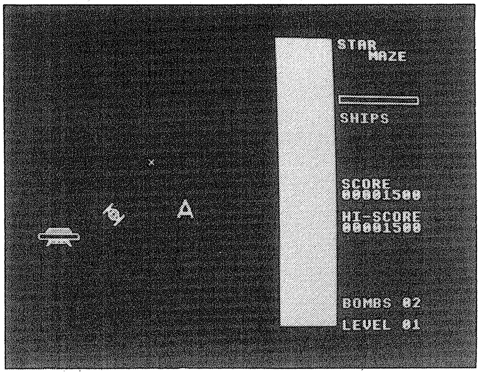
The game was launched with full page colour advertising claiming that this game was 'destined to become a 1985 top-seller...'. The packaging consists of a cassette inlay card with the instructions and frame descriptions printed on the reverse. This describes the name, shape, colour, value and frame number of the first appearance of each enemy.

The aliens are represented by

resembles the old triangular 'Asteroids' laser cannon. The so-called jewels are not as rare in the Starmaze as the blurb would have it and, somewhat ludicrously, are considerably bigger than your space ship.

The maze occupies several screens and the scrolling between them is rather jerky, surprising when you consider the simplicity of the display.

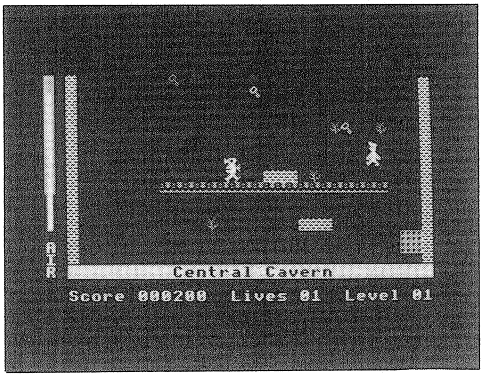
Software Invasion have produced a lot of software for the Beeb over the years and much of it very good. With Starmaze the standards have slipped considerably. I cannot recommend this game.



Willy's underground world.

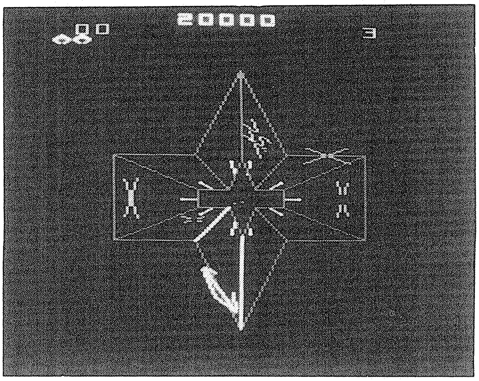
Because of all this data, the cassette takes ages to load. The piracy protection doesn't help either. With the cassette insert there is a printed table of colour codes. At the start of the game you are asked to type in a particular set of codes from the table. If you don't have the table, and so don't know the codes, the game won't start. Effective but very tedious.

However, despite this, Manic Minor is deservedly a classic. The humour of this game means that while it is very challenging to play it is captivating as well. Buy it now!



different geometric shapes, which advance outwards from the centre of the screen. The graphics are very fast, but the mode 2 resolution lets the game down with its ragged edges. Although the game itself is excellent, the keys selected to control it are less satisfactory. The original arcade game had a paddle that could be indefinitely turned through 360 degrees, which made travelling around the outside of the stargate easier.

The game is smooth, fast and furious, and provided exhilarating game playing. The graphics are excellent but the sound effects only moderate. Despite the rather poor resolution of the graphics I can still recommend this game.



Title	: Corpuscle
Supplier	: Mictograf
Price	: £7.95 cass. £10.95 disc
Reviewer	: Geoff Bains
Rating	: *

If this game, based on your body's constant fight against attack, is really the 'all action, accurate real-time simulation' that it claims to be you had better get to a doctor pretty fast.

You take the part of the body's defences and trundle along the blood stream shooting at the nasties with a kind of medical laser. Whatever happened to good old antibodies? You are provided with a map of the blood system and what can only be described as rudimentary steering. You carry on along an artery unless you wander too close to the walls near a junction. Then you get whisked away into the side passage by the rush of blood.

The game is abysmally slow and the graphics poor - consisting mainly of lots of red circles. Despite the ad's claim to full colour graphics, this mostly comprises just two hues. Presumably all this is supposed to give you the feeling that you're watching a re-make of

Title	: Skyhawk
Supplier	: Pace
Price	: £7.99 cass. £11.95 disc
Reviewer	: Geoff Bains
Rating	: ***

Skyhawk is a version of the old arcade game, 'Airlift'. You control a rescue helicopter that has to pick up the luckless survivors of a nearby tank battle (apparently only the enemy have tanks. Your side is equipped with rescue helicopters). These must then be flown back to the safety of your Red Cross base where your chopper is refuelled too.

Unfortunately, the enemy has little regard for the niceties of the Geneva Convention and will do his best to destroy your craft and passengers with his tanks, aircraft and barrage balloons (yes, barrage balloons! Nobody said that this was realistic.)

The barrage balloons can be avoided without too much effort and the tanks are easily dispatched with the usual arsenal of a Red Cross helicopter - bombs and a forward facing missile launcher - the fighter planes, however, are a different story and have an uncanny knack at being

Name	: Caveman Capers
Supplier	: Icon Software
Price	: £7.95
Reviewer	: Alan Webster
Rating	: ****

BBC software entered the realms of true cartoon action with Aardvark's FRAK!, and now Icon continue the trend for lovable characters and prehistoric settings with Caveman Capers. This is an everyday story of Ogg the caveman, who happens to be stranded late in the afternoon a long way away from his modest semi-detached cave and loving wife.

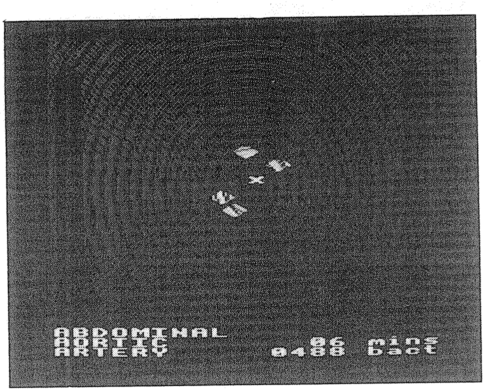
Ogg then discovers a new means of transport - Kickstart the turtle. Can you help Ogg control the turtle, so that he can dash to the phone-box (in the stone-age!) to let his wife know that he will be late home for tea.

Caveman Capers features 60 levels of action, with snakes, pterodactyls, toadstools, dinosaurs and other pre-historic hazards. The graphics are fast, smooth and well presented, especially when Ogg falls flat on his face.

'Fantastic Voyage'. The total absence of Raquel Welch on screen leaves me unconvinced.

The bacteria that you are to defend your body against are a horrifying species I have not met before. They look for all the world like flickering yellow blobs - very dangerous and liable to bring on a severe case of sleeping sickness within minutes.

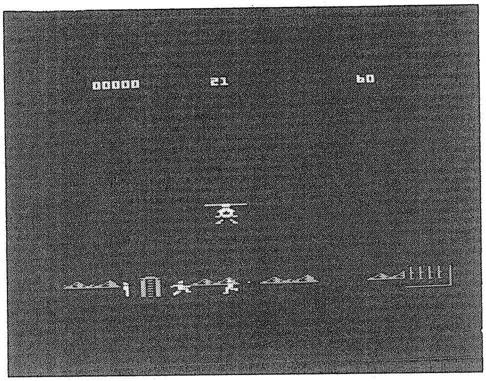
Unless you have a vampirical obsession with blood, this game is unlikely to appeal for longer than the opening screens. Not recommended.



very successful. Fortunately they are not scrambled until you have at least rescued a few of the wounded.

The graphics in Skyhawk are excellent. There are three planes of movement in this game which gives the overall picture a very realistic effect - as your chopper moves you can see through the windows of the bombed out houses, the distant hills move in parallax. The characters are well designed to the point of being cute.

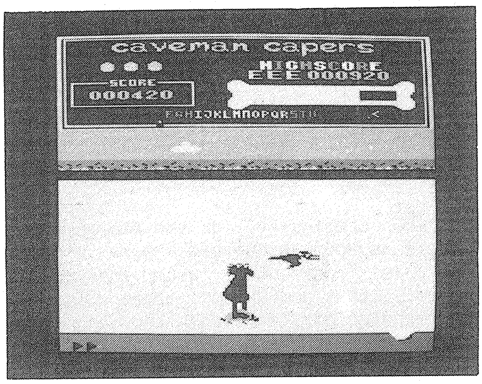
The action is smooth and fast as well and this game should provide a good challenge to any budding chopper pilot.



The cassette is packaged neatly, with the instructions printed on the back of the insert. Icon seem to have put a great deal of time and thought into this piece of software, and have been rewarded with a most likeable game.

One small reservation to be made is about the layout of the score and title screen. This seems unnecessarily cramped, as there is plenty of spare space on the screen between the score and the actual action.

This is an amusing and cute game that is fun to play, and I have no hesitation in saying that it is the most enjoyable game that I have played for a long time.



The Penman Plotter

Terry Hallard reports on the Penman Plotter, a self-propelled graph plotter of somewhat original design which can also double up as a Logo turtle when not otherwise busy.

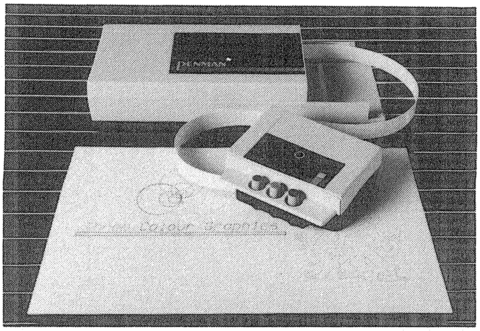
```
Product  : Penman Plotter
Supplier : Penman Products
           8 Hazelwood Close,
           Dominion Way, BN14 8NP.
           0903-209081
Price    : £286.35
```

The Penman plotter is different from any other plotter on the market. It is a self-propelled drawing device that has more in common with a turtle than the normal flat bed plotters available for home micros.

The Penman is more, however, than just a socially ambitious turtle. Granted that it wanders about a sheet of paper under computer command and it draws lines as it goes, but there the similarity ends. It is capable of very reasonable quality drawing/plotting which is the equal of any plotter in the under £500 range. It can print text of any size, down to lower case letters only 1mm high. It switches between any of three pens, be they different colours or different thicknesses. It can act as a mouse for input, albeit not up to AMX standard.

I am not completely certain that this machine can't talk! I swear that it chuckles and mutters to itself as it scuttles busily about the piece of paper it works on. It certainly has the ability to stop and unravel itself if it gets close to being caught up in its own lead.

The originality of the Penman shows itself as soon as the package is opened. One is confronted with instruction manual, leads and a long, cream coloured, rectangular box. It turns out that the turtle itself is a quarter of this box, the rest being the interface and control software. Pulling them gently apart exposes the pen sockets on the front of



the plotter and the wheels beneath. A flat ribbon cable about a metre long also unreels - this is the Penman's umbilical cord. The package design is of the highest order.

Apart from the solenoid-actuated raise/lower arms for the pens, the Penman has only two moving parts - two tiny electric motors equipped with optical encoders to read the amount of movement. A small recessed, freely swivelling tailwheel completes the triangle to give stability. Beneath the body are two slits for the positional photosensors. These are parallel to the edge which carries the pens.

Setting it all up is simplicity itself. A 25 way 'D' plug set into the side of the main case connects via a lead to the RS423 serial port at the back of the Beeb. A small transformer supplies the power. You have to make certain, if you are to get the highest quality output, that the drawing surface is completely flat and level. It must also be firm and black. This is so that, using white paper, the Penman can sense where the edges are - it bleats and whimpers when it feels lost.

When the power is switched on the Penman is ready. If the serial lead is not connected it undertakes a short self test, writing its name and drawing a set of interlinked circles. It then waits for instructions. Just before startup in computer mode the Penman has to be placed in a 'home' position, about two inches in and up from the bottom left hand side of the paper, facing the left. On startup it turns and finds the X-axis, moving forwards until a photosensor 'sees' the edge of the paper. It then gradually wriggles itself until it is satisfied that it is aligned parallel to the edge. It then turns through ninety degrees and repeats the process to align itself in the Y-axis. Then it moves to the centre of the paper and starts drawing.

You only have to see Penman draw a couple of straight lines and a circle to realise that the firmware, based on an 8-bit microprocessor, is very clever indeed. It rarely draws a straight line 'a la turtle' - driving itself straight along the line, drawing as it goes (though it can). The three main drawing pens are off-centre and it actually seems to go through a three point turn manoeuvre in order to keep the current pen moving along a straight line. The drawing action has to be seen to be appreciated.

The car drawing accompanying this article was the result of a program called 'MICAD' which has routines sending output to either a Hewlett Packard plotter or Penman. It probably shows Penman at its worst - not, I hasten to add, that this is Penman's fault. The circle/ellipse drawing routines in this program use a polygon procedure. This results in a lot of short juddery stop-swivel-start movements which have an accumulative effect of sending Penman off course. If you look closely you can see one or two points where lines do not quite meet.

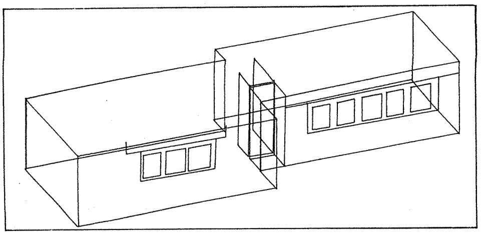
The manufacturers recommend that 'homing' checks be built into any drawing process at reasonable intervals, usually after 400mm of movement. This sends the plotter back to the corner where it goes through the alignment process described earlier. Then with its position firmly in mind it returns very accurately to the drawing and carries on. In fact the MICAD program has now been updated to incorporate automatic homing at frequent intervals. However, this procedure slows down the plotting considerably.

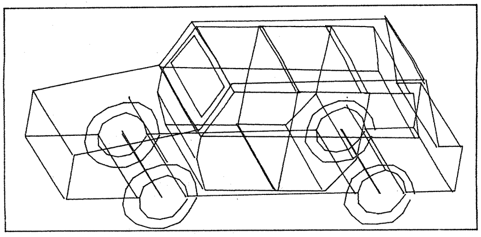
The Penman can also operate as a 'normal' turtle and take its commands from Acornsoft's Logo. It has a fourth pen socket at its centre for this purpose.

Plotter commands for the programmer to incorporate in software are very simple and are dealt with quite thoroughly in the manual. It is easy to control as a robot - the accompanying software has example programs which put the Penman into a number of trial situations. These allow a number of interesting demonstration shapes to be drawn, bar charts, an orthographic elevation of a house, two animals, some very impressive Gothic lettering (true!) and a demonstration Logo procedure.

This program also allows practice use as a turtle, robot and optical input device. In this latter mode the Penman runs back and forth across a diagram and puts the information picked up by the photosensors on to the screen - crudely, but good enough for a demonstration for schools. There are also 25 Basic routines for Penman already defined for the potential programmer to incorporate directly into a program as needed.

The makers of Penman do not claim that this is a high quality plotter, suitable for volume output and business use. They intend it to be a plotter available to the average micro enthusiast at a price that can be reasonably afforded and they complement the plotting action with several other uses. The makers have, in my opinion, succeeded admirably in their aims. The quality of manufacture, the varied functions and the highly original design approach make it a very useful device indeed.





BEEBUG Workshop

Intriguingly named 'hashing' is just one of the techniques described by Surac this month in his third, and for the time being, last workshop on searching and sorting.

In the last two Workshops I went into some detail about how to sort lists into order. Sorting is often part of a file management system and, this month, we'll see another aspect, how to search for a particular item in a whole set of data. Although these examples use arrays, the techniques can equally well be applied to files of records on disc.

When we search a list, we look for the entry with a particular value - the "Search Key". We could search by looking at every record in turn until we find the correct one, but that might take ages. A better way is a "Binary Search", which relies on the list already being sorted. It is then rather like using a dictionary. You start in the middle and narrow your search to one half. Then find the correct quarter, and so on until you hit the word you are looking for.

```
            BINARY SEARCH
10000 DEF FNbinsrch(st%,fin%,
      match)
10010 LOCAL mid%,test
10020 mid%=(st%+fin%)DIV 2
10030 test=array(mid%)
10040 REPEAT
10050   IF test>match THEN
        fin%=mid%
10060   IF test<match THEN
        st%=mid%
10070   mid%=(st%+fin%)DIV 2
10080   IF test<>match THEN
        test=array(mid%)
10090   UNTIL test=match OR
        st%>=(fin%-2)
10100 IF test<>match THEN mid%=-1
10110 IF test<>match AND
      array(fin%)=match THEN
      mid%=fin%
10120 =mid%
```

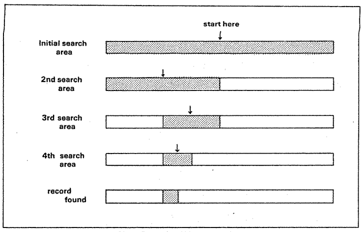


Fig. 1 Example of binary search

The job of FNbinsrch, given the start and finish indices and the search key, is to return the index of the element in 'array()' which holds 'match'. If it does not find it, it returns the value -1. The core of the function is a REPEAT loop which halves the gap between the indices until it either finds the right value, or the 2 indices are adjacent. If the latter, it checks whether the top index holds the search key.

In your programs, you should replace every reference to array() with the name of the actual array used, already sorted in ascending order. The array could contain numeric or string data sorted using any of the techniques described in previous Workshops. If you are using an array of sorted pointers as I described last month, then you will need references such as:

```
array$(ptr%(mid%))
```

To give an idea of its speed, a binary search of a 1000-element list will find any value with no more than 9 tries.

That's one way to do it, if the data is already sorted. Another way needs no previous sorting but, since it tends to waste space, is best used for disc files, a subject I hope to cover in a future Workshop. A "hashing" technique uses the value of the main key of each record to calculate the index to the record. Each record (or array element) is then stored at its calculated position. Below is a comparatively easy hashing function.

The function assumes each record or value is a string. Line 1ØØ2Ø makes sure the string to be hashed is at least 5 characters long. Lines 1ØØ3Ø-1ØØ6Ø then convert the ASCII values of the first 5 characters into a single, large, integer. Line 1ØØ7Ø uses this 'nonsense' number to force the random number system into a new sequence. Line 1ØØ8Ø skips the first number generated, while the function returns (line 1ØØ9Ø) a random number in the range 1-tsize%, where 'tsize%' is the total size of the storage (array).

HASH NUMBER GENERATOR

```
1ØØØØ DEF FNhash(strg$)
1ØØ1Ø LOCAL hash%,i%
1ØØ2Ø strg$=LEFT$(strg$+"     ",5)
1ØØ3Ø hash%=Ø
1ØØ4Ø FOR i%=1 TO 5
1ØØ5Ø   hash%=hash%*1Ø+
        ASC(MID$(strg$,i%,1))
1ØØ6Ø   NEXT
1ØØ7Ø hash%=RND(-hash%)
1ØØ8Ø hash%=RND(tsize%)
1ØØ9Ø =RND(tsize%)
```

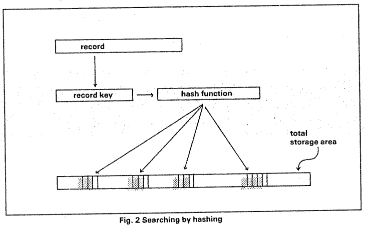


Fig. 2 Searching by hashing

Using random numbers makes use of the fact that the generator can be reset with a RND(-n) command. It then produces a new set of totally repeatable pseudo-random numbers, whose sequence is controlled only by the value of "n". In this case, the same string always gives the same index. Once a string has been stored at its calculated position, it can always be found again by carrying out the same hashing calculation.

Be warned - even the best hashing algorithm will produce duplicate codes for totally different strings. We must be ready for this when both storing and recovering data.

To store data, hash the key and look at the corresponding location. If it is empty, save the data. If it is occupied, search forward for the first empty location or until the array (or file) is all filled. The latter should never happen, but you never know... Otherwise, put the data in the first spare position after the hash index - if you do reach the end of the array (or file) in searching from a mid-point, go back to the very start.

Finding a record is similar. The hash code shows where to start looking - search until you find either the record or a blank, or you have checked everything. Finding a blank means that the key does not exist in the file.

HASH HANDLERS

```
11ØØØ DEF PROChashstore(strg$)
11Ø1Ø LOCAL iters%,ptr%
11Ø2Ø ptr%=FNhash(strg$)
11Ø3Ø iters%=Ø
11Ø4Ø REPEAT
11Ø5Ø   IF array$(ptr%)<>"" AND
        array$(ptr%)<>strg$ THEN
        ptr%=ptr%+1
11Ø6Ø   IF ptr%>tsize% THEN ptr%=1
11Ø7Ø   iters%=iters%+1
11Ø8Ø   UNTIL array$(ptr%)="" OR
        array$(ptr%)=strg$ OR
        iters%=tsize%+1
11Ø9Ø IF iters%<=tsize% THEN
      array$(ptr%)=strg$
111ØØ ENDPROC
1199Ø :
12ØØØ DEF FNhashfind(strg$)
12Ø1Ø LOCAL iters%,ptr%
12Ø2Ø ptr%=FNhash(strg$)
12Ø3Ø iters%=Ø
12Ø4Ø REPEAT
12Ø5Ø   IF array$(ptr%)<>strg$ AND
        array$(ptr%)<>"" THEN
        ptr%=ptr%+1
12Ø6Ø   IF ptr%>tsize% THEN ptr%=1
12Ø7Ø   iters%=iters%+1
12Ø8Ø   UNTIL array$(ptr%)=strg$ OR
        iters%=tsize%+1 OR
        array$(ptr%)=""
12Ø9Ø IF iters%>tsize% OR
      array$(ptr%)="" THEN ptr%=-1
121ØØ =ptr%
```

PROChashstore stores "strg$" in the previously set up "array$()", with line 11060 providing the index wrap-around during the search. If it finds the key field already in the file, it overwrites it - how would you make it do nothing?

FNhashfind looks for "strg$" in array$(), and returns its index. If the search fails, it returns "-1". In both routines, 'iters%' counts the checks, to detect a full circle, and 'tsize%' holds the maximum size of the storage.

Hashing is a very powerful random-access technique, although sorting can be quite tedious. To use it safely, the storage MUST have space for more than the maximum number of records expected. Allow at least a 25% excess; if you can spare 50%, any search or store is likely to be very quick, rarely needing more than a couple of checks to find or store a record.

The magazine cassette/disc contains complete demonstrations of both searching techniques using 4-character strings as data. The first demonstration also uses the Shell string sort from last month to presort the data for the binary search.

Next month, something completely different - high speed graphics.

22←

position of the origin). This ensures that we MOVE to the first point on the graph, and then DRAW to all subsequent points.

Line 160 is commonly used in programs merely to prevent the program terminating, and probably corrupting the display slightly. You will need to press Escape to exit from the program.

Now try some functions. A fairly traditional one is:

```
400*SIN(X/100)
```

This gives a typical sine curve. Now try the following modification:

```
400*SIN(X/100)*(1+2*(SIN(X/100)<0))
```

The expression SIN(X/100)<0 is either true (-1) or false (0). Multiplying by 2 and adding 1 gives the value 1 or -1. The way this is used turns all the downward (or negative) loops in the curve into upward (or positive) loops. Here's another one to try (I leave you to work out the logic yourself):

```
400*SIN(3*X/100)*(X>524 AND X<728)
```

There is ample scope here for further development, and why not experiment by making up some curves of your own?

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CURSOR CHANGE - A.S. Clarke

For a change of scene (or cursors at any rate) type

```
VDU23;10,0;0;0;0;0;0;
```

VDU21 ANOMALY - M. J. Staniforth

If large quantities of data are sent to the screen when this has been disabled with the VDU21 command, the computer will eventually hang up with the symptoms of a full buffer. This does not occur if the screen is instead disabled with the *FX3 command. Running this program:

```
10 VDU21
20 FOR I%=1 TO 100
30 PRINT I%
40 NEXT I%
50 VDU6
```

will hang the computer, but substitution of the following will run to completion:

```
10 *FX3,6
50 *FX3,0
```

ADVENTURE GAMES

by Mitch

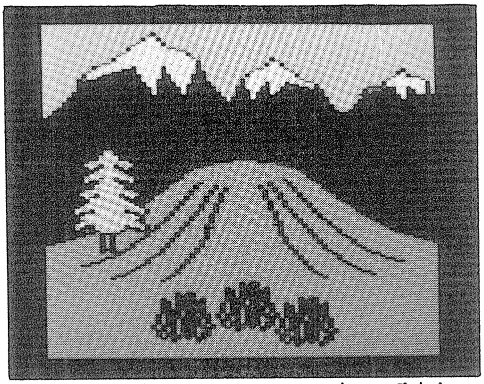
In the beginning was Hardware, followed closely by Software. Firmware took a little longer but we were kept amused by Sinclair's promises, dubbed 'Vapourware'. Finally down the dungeon chute has tumbled 'Bookware'.

Mosaic Publishing Ltd. have coined this term to describe their combined packages of books and associated adventure software. Mosaic's two earlier attempts in this field, 'The Pen and the Dark' and 'The Stainless Steel Rat' were greeted with a wave of indifference owing to the poor quality of the software. However, Mosaic appears to have learnt from its mistakes and has combined with Level 9 Software to produce the latest improved offering.

'The Saga of Erik the Viking'. £9.95 from Mosiac Publishers Ltd and Level 9 Software

This game is a vast improvement, featuring hundreds of locations each beautifully illustrated. The illustrations are being drawn on an alternate screen while you are reading the accompanying text. To view the pictures you must press the tab key, which reveals the quarter screen views. The booklet accompanying the game gives sufficient extracts from the original book by Terry Jones of the Monty Python team, to give you the clues necessary to complete the adventure. As with all Level 9 games there are heaps of text and objects to amuse and confuse.

As I have always thought I would make a really good Viking (I'm really into rape and pillage), I rushed off to find my winged helmet and sword. Finding the Boat House, and trying EXAMINE BOAT I was informed that what we have here is A KNORR. This threw me straight away as the



only Knorr I know contains Chicken Noodle! However, as I spend most of my adventure time in the soup, I soon felt quite at home.

A motley crew of warriors will accompany you in your travels and you will be informed periodically what each character is currently up to. Blind Thorkhild for instance, seems to spend his time munching buttered fish, and boring everyone with his prophecies of doom and gloom. Ragnar Forkbeard is forever singing songs about gold - shades of the Hobbit! The object of the quest is to rescue Erik's wife and family who have been kidnapped by evil dog-faced warriors. With the help of your happy band and a friendly enchanter you must overcome dragons, giants, eagles and an evil enchantress. By steering your ship across the seas you will find different lands full of caves, forests and glaciers.

This is a full sized Level 9 style game which will keep you amused for many days and should you continue to flounder on the rocks, the Hints and Answer sheet is enclosed. Remember to get a friend to

Extended disc catalogue

Automated catalogues of disc programs are popular with many micro users, but often suffer from the limited information displayed. David Andrews describes how to improve on this and make your disc catalogues really friendly.

One of the problems with standard disc catalogues is that little more than the filename is shown, and in the fullness of time it can be difficult to remember just what each program does.

This useful utility will produce an extended catalogue of Basic programs for any standard Acorn compatible disc (drives Ø to 3). The catalogue is produced in mode 3, giving ample room for displaying 63 characters of additional information for each file. This allows a much more detailed and comprehensive file description than is possible with the filename alone.

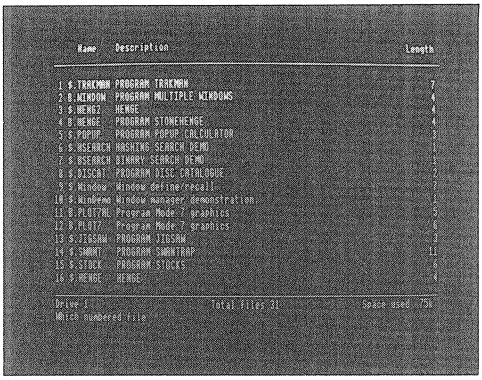
When run, the program asks for the drive to be catalogued (Return selects the current drive). The program will then search that drive and display the details of all Basic programs which conform to a certain standard. This means that any program which is to appear in the catalogue must have a first line which begins with a REM statement as, for example:

```
    1ØREM Extended Disc Catalogue
```

The catalogue will then display the entry:

```
<i> <fname> Extended Disc Catalogue  nK
```

where 'i' is the index for that file (starting from 1), 'fname' is the file-name of the program on disc (including



```
   Name    Description                                          Length

 1 $.TRAKMAN PROGRAM TRAKMAN                                         7
 2 $.WINDOW  PROGRAM MULTIPLE WINDOWS                                4
 3 $.HENG2   HENGE                                                   4
 4 $.HENGE   PROGRAM STONEHENGE                                      4
 5 $.POPUP   PROGRAM POPUP CALCULATOR                                3
 6 $.HSEARCH HASHING SEARCH DEMO                                     1
 7 $.BSEARCH BINARY SEARCH DEMO                                      1
 8 $.DISCAT  PROGRAM DISC CATALOGUE                                  2
 9 $.Window  Window define/recall                                    7
10 $.Windemo Window manager demonstration.                           1
11 $.PLOT7RL Program Mode 7 graphics                                 5
12 $.PLOT7   Program Mode 7 graphics                                 6
13 $.JIGSAW  PROGRAM JIGSAW                                          3
14 $.SWANT   PROGRAM SWANTRAP                                       11
15 $.STOCK   PROGRAM STOCKS                                          6
16 $.HENGE   HENGE                                                   4

Drive 1                  Total files 31                   Space used  73k
Which numbered file
```

directory), and nK is the length of the program in kilobytes.

In general, a program must start with a REM statement of the form:

```
    <line number>REM <text>
```

with no space between the line number and REM, and at least one space between REM and the text message to be included in the extended disc catalogue.

At the bottom of the catalogue, the total number of files on that drive and the total disc space used is shown.

If there are more than 18 files on the disc to be catalogued, the program will pause after the first 18, and continue when any key is pressed. ⟶

⟵
read it for you, or you're sure to see more than you wish.

A new crystal ball has been unveiled in Kent. A wise wizard called Ken Mathews has set up a telephone help service in the rear of his recently opened cave (software shop). Ken hopes to answer queries on all adventure games no matter which micro you possess. The service is free, although no doubt he will give you a friendly line in sales talk while he has your willing ear. The number to ring is - Gravesend (Ø474) 334ØØ8. However don't forsake us here in the BEEBUG Dungeon completely, it gets pretty lonely in here!

After the selected disc drive has been catalogued, you can choose the number of the file you wish to LOAD or CHAIN. The program then sets up function keys fØ and f9 respectively to do this for you. Pressing Return alone will take you back to the start of the program to catalogue another drive.

In practice, it would be useful to include this utility on drive Ø of all your main program discs, and to save a !BOOT file so that Shift-Break will automatically chain to it and display the extended disc catalogue on the screen. It will work in other modes as well (if you change line 11Ø) but this will affect the amount of information that can be displayed.

PROGRAM NOTES

1. PROCintro requests the drive number (Ø-3) - Return gives current drive. The machine code routine (PROCmakecode - see below) stores the names of all files (from all directories) in the array NAME$(Z%). Each file in turn is opened for reading and checked for a REM (following the first line number without a space) at line 24Ø. If this is present, the file is numbered sequentially (T%), and this number, the directory, filename and REM statement (first 63 characters or Return) are displayed, plus the size of the file to the nearest kilobyte.

2. PROCmakecode places the ASCII code of each filename letter found in the catalogue sector of track 1 in BUF%, and PROCdecode transfers the filenames to NAME$(Z%). Lines 128Ø-13ØØ add the directory letter to the front, ignoring the "Locked" prefix (if it exists) at line 129Ø.

3. If you want the catalogue to ignore a particular file, either ensure that the first line is not a REM statement, or separate the REM from the line number by a space.

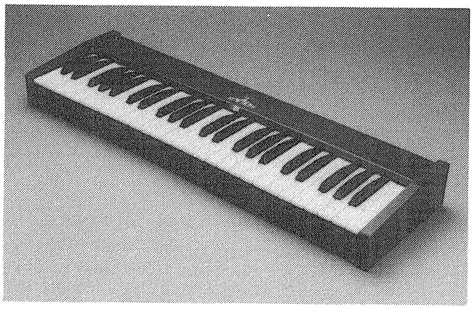
4. Line 28Ø strips off any spaces between the REM and the following statement.

5. For Basic II and Tube users, the program can be shortened and speeded up by deleting lines 134Ø to 14ØØ, and replacing PROCoscli with the keyword OSCLI at lines 19Ø, 47Ø, 48Ø and 1Ø4Ø.

```
   1Ø REM PROGRAM DISC CATALOGUE
   2Ø REM VERSION BØ.4
   3Ø REM AUTHOR  D.Andrews
   4Ø REM BEEBUG  MAY 1985
   5Ø REM PROGRAM SUBJECT TO COPYRIGHT
   6Ø :
  1ØØ ONERRORGOTO141Ø
  11Ø MODE3
  12Ø DIMbuf% 3Ø
  13Ø PROCintro
  14Ø PROCmakecode
  15Ø CALLreadcat
  16Ø PROCdecode
  17Ø CLS:PRINTTAB(5,Ø)"Name"TAB(8)"Desc
ription"TAB(68)"Length"TAB(Ø,1)STRING$(8
Ø,"_");
  18Ø VDU 28,Ø,23,79,2
  19Ø PROCoscli("DR."+STR$(D%))
  2ØØ T%=Ø:V%=Ø
  21Ø PRINT
  22Ø FORZ%=ØTOE%-1
  23Ø N%=OPENUP NAME$(Z%):V%=V%+EXT#N%
  24Ø PTR#N%=4:F%=BGET#N%:IFF%<>244THEN3
7Ø
  25Ø @%=2
  26Ø NAME$(T%)=NAME$(Z%)
  27Ø PRINTT%+1TAB(3)NAME$(T%)TAB(13);
  28Ø REPEAT:F%=BGET#N%:UNTILF%<>32:PTR#
N%=PTR#N%-1
  29Ø FORC%=1TO63
  3ØØ F%=BGET#N%
  31Ø IFF%=13THENC%=7Ø:GOTO33Ø
  32Ø PRINTCHR$(F%);
  33Ø NEXT
  34Ø PRINTTAB(77)(EXT#N%+5ØØ)DIV1ØØØ
  35Ø T%=T%+1
  36Ø IFT%=19THENVDU7:PRINT'".. Any key
 to continue ..":pause=GET:VDU11:PRINTSPC
(27):VDU11,11
  37Ø CLOSE#N%
  38Ø NEXT
  39Ø PRINTSTRING$(79,"_")
  4ØØ PRINT"Drive ";D%;
  41Ø PRINTTAB(32)"Total files ";E%TAB(6
3)"Space used  ";(V%+5ØØ)DIV1ØØØ;"k"
  42Ø INPUT"Which numbered file   "f%:IF
f%>T%THENVDU7,11:PRINTSPC(3Ø):VDU11:GOTO
42Ø
  43Ø IFf%=ØTHEN RUN
  44Ø CLS
  45Ø MODE7
  46Ø PRINTTAB(1Ø,5)MID$(NAME$(f%-1),3);
" program"TAB(1Ø,6)STRING$(15,"=")TAB(12
,9)"fØ = LOAD"TAB(12,11)"f9 = CHAIN"TAB(
9,16);
  47Ø PROCoscli("KEYØ"+"LOAD """+NAME$(f
%-1)+""""+"|M")
  48Ø PROCoscli("KEY9"+"CHAIN """+NAME$(
f%-1)+""""+"|M")
  49Ø END
 500 :
 1000 DEFPROCintro:*FX18
 1010 D%=0:oscli%=&FFF7:*FX15
 1020 PRINTTAB(27,8)"Catalogue of Progra
ms"TAB(27,9)STRING$(21,"_")
 1030 PRINTTAB(28,13)"New drive"STRING$(
7,CHR$32);:drive%=GET-48
 1040 IFdrive%>-1ANDdrive%<4THEND%=drive
%:PRINT;D%:PROCoscli("DR."+STR$(D%))ELSE
PRINT;D%
 1050 ENDPROC
 1060 :
 1070 DEFPROCmakecode
 1080 DIMBUF%&200,readcat&50
 1090 osword=&FFF1
 1100 FORZ%=0TO2STEP2
 1110 P%=readcat
 1120 [OPTZ%
 1130 LDA#&7F:LDX#blk MOD 256:LDY#blk DI
V 256
 1140 JMP osword
 1150 .blk:]
 1160 ?P%=D%:P%!1=BUF%:P%?5=3:P%?6=&53:P
%?7=0:P%?8=0:P%?9=&22:P%?10=0
 1170 NEXT
 1180 ENDPROC
 1190 :
 1200 DEFPROCdecode
 1210 E%=(BUF%?&105)/8
 1220 IF E%<1 PRINT''"No Suitable files.
"'"Press any key.":G=GET:RUN
 1230 DIMNAME$(E%-1)
 1240 FORZ%=0TOE%-1
 1250 FORY%=0TO6
 1260 NAME$(Z%)=NAME$(Z%)+CHR$(BUF%?(Y%+
8*(Z%+1)))
 1270 NEXT
 1280 G%=BUF%?(Y%+8*(Z%+1))
 1290 IFG%>127THENG%=G%AND&7F
 1300 NAME$(Z%)=CHR$(G%)+"."+NAME$(Z%)
 1310 NEXT
 1320 ENDPROC
 1330 :
 1340 DEFPROCoscli(cline$)
 1350 $buf%=cline$
 1360 X%=buf%
 1370 Y%=buf%DIV256
 1380 CALLoscli%
 1390 ENDPROC
 1400 :
 1410 ONERROROFF
 1420 CLOSE#0
 1430 MODE7
 1440 REPORT:PRINT" at line ";ERL
 1450 END
```

BEEBUG Music Competition

To celebrate the culmination of the popular series, 'Making Music on the Beeb' (see the final part opposite) BEEBUG is launching a major competition with a musical theme. The first prize winner will receive a Symphony 49-note musical keyboard, generously donated by the manufacturers ATPL, BEEBUGSOFT's music editor, MUROM, and Ian Waugh's excellent book, 'Making Music on the BBC computer', on which the series has been based. This mass of goodies - worth over £150 - will give the lucky winner all the equipment and knowledge he needs to make the Beeb sing for its supper. To prove that we are all heart we are also offering a second prize of MUROM and Ian Waugh's book.



That's the easy bit, now let's see what you have to do to collect. We are looking for the best rendition of any popular song or other piece of music performed by the unaided Beeb. The program must be entirely in Basic. The nature of the original piece doesn't matter. It is the accuracy of the rendition and the quality of sound conjured from your machine that count.

Send your program, on cassette or disc, to the editorial address clearly stating your name, address, and membership number. Tell us also the name and composer of the piece and mark the package 'MUSIC COMPETITION'. Enclose a suitable stamped, addressed, envelope if you want your cassette/disc back. All entries must be in by 31st July.

Making Music on the Beeb (Part 5)

In the last of his current articles in our music series, Ian Waugh concludes by looking at musical effects, how they can be produced, and how they can be used in your music programs.

Musical embellishments add spice and colouring to a sound. The musician has an enormous range of effects pedals and boxes at his or her disposal. We have our Beeb. It has no in-built effects but we can do a great deal to enhance the basic sound. I will assume a nodding aquaintance with the ENVELOPE command but if in doubt, keep the User Guide open at page 244 as you read on.

VIBRATO AND TREMOLO

There exists a certain amount of confusion over these two terms, even among musicians. Vibrato is a frequency modulation, tremolo is an amplitude modulation. The modulation usually follows a sine wave pattern and the most pleasant rate of modulation in both vibrato and tremolo is around seven cycles per second. The degree of modulation can vary from the subtle to the ridiculous.

Not many instruments produce tremolo. Electronic organs produce it mechanically and electronically, and singers often use vibrato or tremolo to enhance their tone. Vibrato is far more common and used by most instrumentalists.

VIBRATO

The pitch variation in vibrato is not usually as great as a semitone. Extreme examples where the pitch varies more rapidly and over larger intervals are still technically vibrato but are really only produced by electronic means for special effects such as this:

```
   10 ENVELOPE1,1,0,3,-3,0,20,20,63,-1,0,
-4,126,100
   20 SOUND1,1,101,160
```

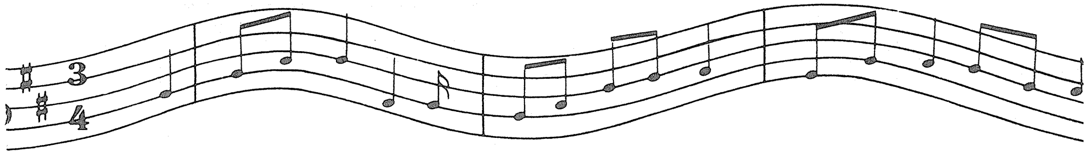
You will soon find that there are few vibrato effects suitable for use in a purely musical context; extremes are simply not musical.

```
   10 REM PROGRAM 5.1
   20 REM ENVELOPE COMPARISONS
   30 pitch=149
   40 ENVELOPE1,4,0,0,1,1,0,1,4,-1,0,-3,
126,80
   50 ENVELOPE2,2,0,0,1,2,0,2,4,-1,0,-3,
126,80
   60 ENVELOPE3,3,0,0,1,2,0,2,4,-1,0,-3,
126,80
   70 ENVELOPE4,4,-2,1,1,1,1,1,4,-1,0,-3
,126,80
   80 ENVELOPE5,6,1,-2,1,1,1,1,4,-1,0,-3
,126,80
   90 ENVELOPE6,4,1,1,1,1,1,1,4,-1,0,-3,
126,80
  100 REPEAT
  110 env=GET-48
  120 IF env=0 THEN SOUND1,-12,pitch,40
ELSE SOUND1,env,pitch,40
  130UNTIL FALSE
```

Pressing keys 1 to 6 in will play that envelope. Pressing 0 will play the sound straight. Alter the pitch of the note, too, during your experiments as this makes a tremendous difference to the note. Envelope 4 sounds lower than the pitch of the note because PI1 immediately sets the pitch down two steps.

TREMOLO

The ENVELOPE command provides a repeat option on the pitch envelope but not the amplitude envelope. To produce tremolo this is ideally what we need. In its absence we must use a loop.



```
  10 ENVELOPE1,5,0,0,0,0,0,0,0,8,-8,0,-8,1
20,16
   20 FOR Trem=1 TO 8
   30 SOUND1,1,53,28
   40 NEXT Trem
```

This causes complications, especially if we want to use tremolo during the production of a tune. While vibrato is more musically useful and easier to apply, the full potential of tremolo has not been realised and its use for sound effects can lead to something just that little bit different.

TRILLS

To produce vibrato on an instrument you need control over the pitches in between individual notes. The notes on a piano, for example, are fixed and you can not produce vibrato on a piano. The best you can do is to alternate rapidly between two adjacent notes and this is called a trill. This envelope produces a trill between C (Pitch = 149) and D (Pitch = 157):

```
ENVELOPE1,4,0,8,-8,0,1,1,16,-1,0,-3,126,8
0
```

```
   10 REM PROGRAM 5.2
   20 REM "Military Music" Introduction
   30
   40 ENVELOPE1,4,0,8,-8,0,1,1,63,0,0,-1
2,126,126
   50 ENVELOPE2,3,0,0,1,2,0,2,126,-8,0,-
8,126,30
   55 ENVELOPE3,4,0,0,1,1,0,1,32,-1,0,-8
,96,60
   60 SOUND1,1,149,54
   70 FOR Note=1 TO 9
   80 READ Pitch,Dur
   90 SOUND2,3,Pitch,Dur
  100 SOUND3,2,Pitch+48,Dur
  110 NEXT Note
  120 END
  130 DATA 53,12,49,4,45,12,41,4,33,4,25
,4,21,4,13,4,5,6
```

The pitch does not follow a sine wave pattern as with vibrato and tremolo but a square wave pattern and trills can be played by most instruments. They can be programmed as a sequence of the two notes but it is often more convenient to switch control to an envelope. A trill can be played over any interval and this plays a trill over an interval of a fifth:

```
ENVELOPE1,4,0,28,-28,0,1,1,16,-1,0,-3,126
,80
```

A flute with the occasional trill sounds very effective, especially in military or brass band music, and a trill on a sustained note above a melody line is quite common.

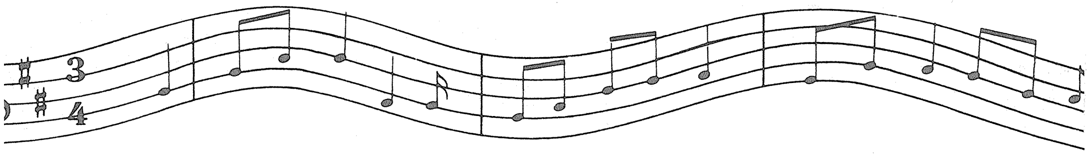
ECHO ECHo ECho Echo echo AND REVERBERATIONNNN

Echoes are produced when sound waves are reflected from a smooth hard surface such as a cliff. A sound emitted in a room will bounce around the walls, floor and ceiling resulting in a most complex series of multiple reflections called reverberation. Because of the enormous number of vibrations involved, reverberation is not possible on the BBC micro. Some programs, both commercial and in magazines, offer a 'reverb' control but this is really a slow decay or sustain, not technically reverberation.

ECHOES

Let's assume we want an echo on a single note. Ideally, we would like to create this with a single ENVELOPE command - but we can't. To produce an echo, there must be a discernible gap between notes which means the volume has to drop and rise again. We can not do this with one envelope unless we play it

```
   10 REM PROGRAM 5.3
   20 REM Echo Production
   30 REM Using Multiple Envelopes
   40
   50 Dur=3
   60 ENVELOPE1,1,0,0,0,0,0,0,126,-4,-4,
-6,126,102
   70 ENVELOPE2,1,0,0,0,0,0,0,102,-4,-4,
-6,102,78
   80 ENVELOPE3,1,0,0,0,0,0,0,78,-4,-4,-
6,78,54
   90 ENVELOPE4,1,0,0,0,0,0,0,54,-4,-4,-
6,54,30
  100 ENVELOPE5,1,0,0,0,0,0,0,30,-4,-4,-
6,30,6
  110 FOR Echo=1 TO 5
  120 SOUND1,Echo,101,Dur
  130 NEXT Echo
```



twice and that alone would not produce a drop in volume. One method is to use a series of envelopes and play the sound using each in turn.

Program 5.3 produces a rather good echo but is wasteful of envelopes. Try doubling the SOUND commands like this:

```
 120 SOUND1,Echo,101,Dur:SOUND1,Echo,10
1,Dur
```

The duration of the note determines the echo repeat time, and long notes will not produce a very good effect. The next program tries to reduce the envelope waste by using only one, and repeatedly redefining it within a procedure. In doing this, we must be careful not to redefine an envelope before it's ready to be used.

```
 10 REM PROGRAM 5.4
 20 REM Echo Using a Procedure
 30 :
 40 EchoSpeed=10
 50 RateOfDecay=3
 60 :
 70 FOR Note=1 TO 5
 80 READ Chan,Pitch,Dur
 90 PROCEcho
100 NEXT
110 END
120 :
130 DATA 1,5,16,2,33,16,3,53,28,1,69,2
,2,65,32
140 :
150 DEF PROCEcho
160 AA=126
170 FOR Count=1 TO Dur
180 ALD=AA-RateOfDecay
190 ENVELOPE1,1,0,0,0,0,0,0,AA,-4,-4,-
1,AA,ALD
200 SOUNDChan,1,Pitch,1
210 TIME=0:REPEAT UNTIL TIME>EchoSpeed
220 AA=ALD
230 NEXT Count
240 ENDPROC
```

Line 210 holds up the whole program while the echo runs its course. If we try to build the delay into the SOUND command by increasing the duration it does not work because the sound chip stores the commands and while it is waiting to execute them, the rest of the program will have already redefined the envelope. Remove line 210 and see what happens. Note also that the durations given to the SOUND command from the DATA statement at line 130 no longer determine the length of the note. This is determined by the variable, EchoSpeed. The duration values maintain the note-length relationship between notes. Alter EchoSpeed and RateOfDecay and listen to the effect.

PSEUDO ECHO

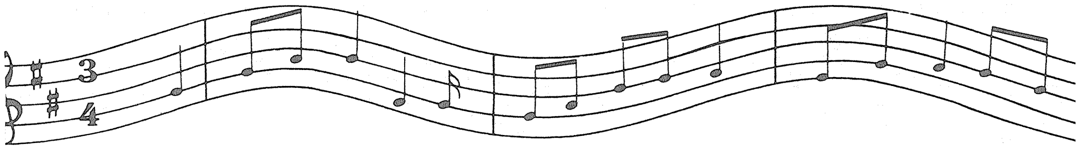
The implementation of a single note echo is not particularly easy although the results can be very good and worth the effort. One alternative is to create a pseudo echo using the pitch envelope. Try these:

```
ENVELOPE1,20,12,-12,0,1,1,0,126,-6,-6,-6,
126,0

ENVELOPE2,4,4,0,32,4,1,0,126,-1,-1,-1,126
,0

ENVELOPE3,4,4,16,12,1,1,1,126,-1,-1,-1,12
6,0

ENVELOPE4,6,8,-16,8,8,4,2,126,-1,-1,-1,12
6,0

ENVELOPE5,4,32,-64,4,1,1,16,126,-1,-1,-1,
126,0

ENVELOPE6,8,0,8,-12,1,1,1,16,-1,0,-3,126,
80
```

If you want an echo of two or more notes this is the way to do it and even if you don't, you can often get away with using two notes where you really only wanted one.

PLAYING TUNES WITH THE PITCH ENVELOPE

It is only one step away from these examples to devise envelopes that will play a tuneful sequence of notes. For example:



```
  10 ENVELOPE1,11,16,4,8,2,1,1,100,0,0,-
100,100,100
   20 SOUND1,1,101,20
```

Consider the length of the sequence carefully. If AR is altered from -100 to -1, the release phase will occur and it will sound like another echo envelope and lose its impact. As it is, the envelope will last as long as the SOUND's duration parameter. You can calculate the time, in hundredths of a second, of a single pitch envelope as:

```
   Time = (PN1+PN2+PN3)*T
```

If you want to terminate the sound in the middle of a pitch envelope, as above, calculate the amount of time required and apply it to the SOUND command by dividing by 5. In this case the envelope repeats once and has an extra note on the end. This is a total of 9 x T which is 99. Dividing by 5 gives us our duration of 19 or 20. Here are more envelopes which play a musical sequence of notes.

```
  10 ENVELOPE1,136,12,-4,4,17,60,12,100,
0,0,-100,100,100
   20 SOUND1,1,41,144
```

This plays a 'whole tone' scale.

```
  10 ENVELOPE1,10,8,-8,0,18,18,6,100,0,0
,-100,100,100
   20 SOUND1,1,45,84
```

This wraps around.

```
  10 ENVELOPE1,9,20,20,0,10,10,5,100,0,0
,-100,100,100
   20 SOUND1,1,149,135
```

CHORUS

When a group of musicians play in unison, they each play a slightly different pitch. This difference is very small but it tells the ear that there is more than one instrument. This is known as a chorus effect and is responsible for the beautiful sound of a string orchestra.

If we play the same pitch on two or more channels, the pitches will not be exactly the same and the result will be a chorus effect. The pitch difference is not as great as a pitch interval and if we add 1 to one of the pitch values, the effect will be more pronounced. This example produces a rather ear-piercing scream on my micro:

```
   10 SOUND1,-12,227,120
   20 SOUND2,-12,227,120
```

Try adding:

```
   30 SOUND3,-12,227,120
```

More pleasing and musical effects occur in the lower octaves. As an example of how you can use chorus, the following program imitates an accordion.

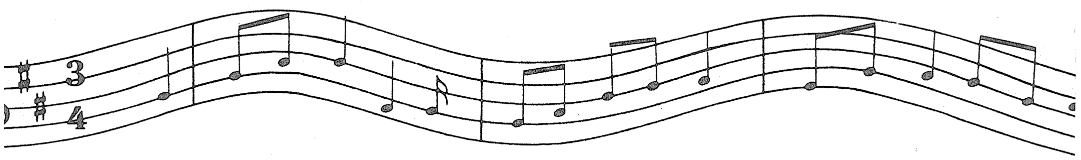
```
   10 REM PROGRAM 5.5
   20 REM French Accordion Music
   30 REM Using Chorus Effect
   40 :
   50 ENVELOPE1,3,0,0,0,0,0,0,8,-1,0,-6,
124,60
   60 :
   70 FOR Tune=1 TO 16
   80 READ Note,Dur
   90 IF Note=0 Amp=0 ELSE Amp=1
  100 SOUND1,Amp,Note,Dur
  110 SOUND2,Amp,Note+1,Dur
  120 NEXT Tune
  130 END
  140 :
  150 DATA 61,8,77,8,113,8,109,72
  160 DATA 61,8,81,8,113,8,109,32
  170 DATA 0,8,93,8,101,8,0,4,65,4,61,4,
53,4,61,4
```

This uses the natural beat frequencies of the sound chip. You can increase the effect by adding a whole pitch interval and, less subtle though it may be, I like this better:

```
   110 SOUND2,Amp,Pitch+1,Dur
```

RING MODULATION AND CHIMES

The ring modulator produces bell-like and metallic sounds and works in a way unlike any of the modules or effects we have covered so far. A standard ring modulator requires an input of two frequencies and it produces an output which is a compound of the sum of the two frequencies and the difference between



them. For example, if it was fed with a frequency of 44ØHz and 122ØHz, the output would be a compound of 166ØHz (the sum) and 76ØHz (the difference).

That's fairly straightforward but the BBC micro is not calibrated to work in hertz and so our choice of frequencies is limited. Also, we do not know what frequency is produced by each pitch command. You can use the following (approximate) formula to produce a table of pitch values and frequencies:

```
F=1Ø^((((P+92)*LOG2)/48)+LOG33)
```

or its inverse:

```
P=((LOG(F)-LOG33)*48/LOG2-92)
```

Don't worry about lack of absolute accuracy - there are likely to be small variations between different BBC micros anyway.

As the BBC micro cannot specify pitch values in hertz, our results will be less than perfect but we can still produce some convincing sounds. To calculate the frequencies:

1. Decide upon the notes, say C3 and F3, and determine their pitch.
2. Calculate their respective frequencies from the first formula. In this case they would be 533.7 (P = 1Ø1) and 715.Ø (P = 121).
3. Find the sum and the difference:

```
715.Ø + 533.7 = 1248.7
715.Ø - 533.7 = 181.3
```

4. Calculate the nearest whole values of P from the second formula. The nearest to 1248.7 is 16Ø and the nearest to 181.3 is 26.
5. Try them:

```
1Ø SOUND1,-15,16Ø,6Ø
2Ø SOUND2,-15,26,6Ø
```

It sounds quite good on my micro.

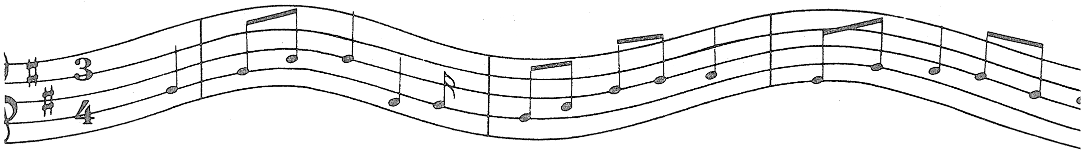
It is quite time consuming to go through this process whenever you want a chime especially if you're writing a piece of music for tubular bells. Sometimes your calculations will produce frequencies which are higher or lower than those available from the sound chip. The obvious thing to do is to write a program which allows you to input two pitches and which would output the two nearest P parameters. Bell effects are not used very often on the BBC micro so perhaps you will find the exercise worthwhile. For the more eager among us, an unscientific but workable shortcut is possible. This consists of adding a fixed pitch difference to the melody notes. This will usually produce good results.

```
   1Ø REM PROGRAM 5.6
   2Ø REM Bells and Chimes
   3Ø :
   4Ø ENVELOPE1,4,Ø,Ø,Ø,Ø,Ø,Ø,126,-1,-1,
-1,126,Ø
   5Ø :
   6Ø FOR Pitch=1 TO 8
   7Ø READ Note,Dur
   8Ø SOUND1,1,Note,Dur
   9Ø SOUND2,1,Note+12Ø,Dur
  1ØØ NEXT Pitch
  11Ø :
  12Ø SOUND1,1,117,64
  13Ø SOUND2,1,6,64
  14Ø SOUND1,1,129,16
  15Ø SOUND2,1,53,16
  16Ø END
  17Ø :
  18Ø DATA 117,16,1Ø1,16,1Ø9,16,81,32
  19Ø DATA 81,16,1Ø9,16,117,16,1Ø1,64
```

Try substituting different values for the pitch difference in line 9Ø. Add a pitch difference to other tunes to hear them played by bells or chimes. Finally, for pure cacophony, experiment along these lines:

```
1Ø FOR Bell=1 TO 2Ø
2Ø SOUND1,-15,RND(1Ø1),1Ø
3Ø SOUND2,-15,RND(153)+1Ø2,1Ø
4Ø NEXT Bell
```

The programs and table are from Making Music on the BBC Computer by Ian Waugh, published by Sunshine Books at £5.95 and used with kind permission of the publishers.



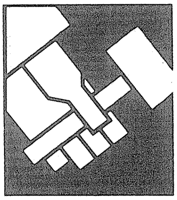
Home education from Acornsoft

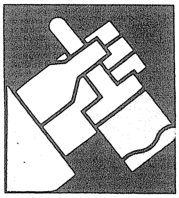
Workshop and Spooky Manor

Is home education set to take off? Acornsoft clearly think so with a new range of software. We asked Don Walton, an experienced teacher and consultant in educational computing, to take a look. Here he reports on 'Workshop' and 'Spooky Manor', the first two packages to be released.

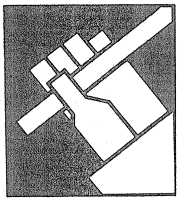
Supplier	: Acornsoft.
Title	: Workshop.
Price	: £ 9.95 Cassette
	£11.50 Disc.

'Workshop' is an attempt to produce a creative, open ended educational program which will appeal to children. The scenario is, as the name implies, a workshop, where the child can manufacture different objects of its own design. Just like a real workshop the program begins by offering the child raw materials from which to work. There is a disc shape, a triangular shape, both with the usual unavoidable ragged edges, and a square, each in a different colour. Any one of these shapes can be chosen and placed in the working area in the centre of the screen, Escape is pressed and the planning page is revealed with the chosen shape remaining in the working area. This is a very well designed page which is easy to use. With the 'tools' provided the shape can be, drilled, painted,'NOTed', moved about, rotated, squashed, cut, glued to other pieces which you may have already shaped, and scaled. You may also choose to look at the sequence of work you have carried out so far.

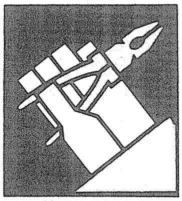




Every workshop has its problems and this is no exception. It is questionable whether children will want to struggle with the limitations and idiosyncracies of this one. An example of these limitations is the scaling function. It doesn't enlarge and reduce smoothly in small increments, but tears off, or adds on, great chunks of material. If one is even slightly over enthusiastic with the reducing facility, the nice shape you have been using is reduced to a small rectangle or disappears altogether. A quick press on the enlarging key to remedy matters reveals that the original shape has gone. There is a somewhat similar effect if the enlarging key is over used.



The most serious limitation is the 'LOOK' option which shows you what you have done. At the beginning, this displays, rather nicely I thought, a set of icons showing the sequence of operations carried out so far. When the Return key is pressed the computer works through all the operations in turn. This is a cacophony of visual noise, the computer printing every screen which has been used in full detail, as fast as it can. There is no step through, or slow down facility, so it is virtually impossible to analyse the manufacturing process. Even if it was, it is impossible to edit it. There is only one solution to an unsatisfactory 'Workshop' product, scrap the whole assembly line and begin again.



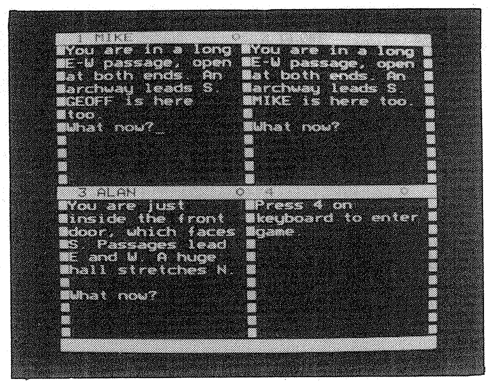
It seems to me that this is a good idea wasted. It is like a very limited design program with a few crumbs of LOGO philosophy thrown in for good measure.

I would rather buy a good design package for my children. This would be easier to use and show them that creative design, using the computer, is a joy.

Supplier : Acornsoft.
Title : Spooky Manor.
Price : £ 9.95 Cassette
£11.50 Disc.

There are lots of adventure games on the market, but Spooky Manor is an adventure with a difference. Like most Acornsoft games of this type, it is purely textual, providing a set of problems at the correct level of sophistication for children beginning to use this sort of software. The scenario, vocabulary and commands of the program, follow the usual pattern of adventure programs but the design has taken a step towards a shared adventure which could be far more interesting than the usual solitary activity.

The author has used a novel approach to involve a group of people in a common problem. The screen is divided into four sections each section dedicated to one player. The number of players can be between one and four. Each player can interact with the program by typing in their number which then opens up their section of the screen for input. There is no attempt by the computer to manage the turns i.e. you can have as many goes as you wish if the others agree, (or if they have gone to the loo). The scenario of



```
 1 MIKE                 2
You are in a long     You are in a long
E-W passage, open     E-W passage, open
at both ends. An      at both ends. An
archway leads S.      archway leads S.
GEOFF is here         MIKE is here too.
too.
What now?_            What now?

 3 ALAN                 4
You are just          Press 4 on
inside the front      keyboard to enter
door, which faces     game
S. Passages lead
E and W. A huge
hall stretches N.

What now?
```

the game is designed to encourage movement away from a simply selfish approach, to one of co-operation as the adventure proceeds.

I think a family could have fun with this program although it might soon loose its attraction once the objective has been achieved.

One further point of interest. The software has an option of using Quinkey's, i.e. Micro-writer keyboards, instead of the standard keyboard so that players can 'time share' Spooky Manor.

HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

COMMSTAR/GRAPHICS ROM CLASH - J.W. Kent

Pace's Commstar will not work properly with Computer Concept's Graphics extension ROM. Neither typing *FX240,128 (to disable the Graphics ROM) nor poking zero into the relevant location of the ROM table at &02A1 helps as the damage has already been done when the Graphics ROM grabbed its work space on Break or switch-on. One solution is to remove the Graphics ROM, however an easier method is to enter Commstar via BEEBUGSOFT's Exmon II. Type *EXMON and then *C*.

BASIC WITHOUT LINE NUMBERS - G.A. Varley

Structured programming enthusiasts who can't afford Pascal can have a little of the atmosphere of this language by writing a Basic program without the line numbers in View. The first word in the View file must be AUTO. Save the complete View file as normal and then, from Basic, *EXEC the file in. It can then be run, listed, or saved as normal. Needless to say this method will not cope with GOTO, GOSUB, ON GOTO, etc. Still, they're not exactly structured commands are they!

INPUT ERRORS - J. Rye

Unlike many versions of Basic, BBC Basic does not issue an error message if a letter is INPUT when a number is expected. It merely assigns a value of zero to the input variable. This affects hex numbers, such as &9ABC, which will be taken as zero. Instead, the program should input the hex number as a string, and convert to a decimal number using EVAL.

GRAPHITO AND TESSELATOR

Two graphics packages for the Beeb

Colin Cohen has been looking at two exciting new graphics packages from Addison-Wesley. Now that he has been able to tear himself away from his computer Colin reports on his experiences.

TESSELATOR, an Interactive Tesselation Design Program (on 40T disc £29.95, on cassette £22.95) by Hallam, published by Addison-Wesley.

Escher, whose picture of two hands drawing themselves popularized the idea of artistic tesselation, wrote '...later the designing of new motifs came with rather less struggle than in the early days, and yet this has remained a very strenuous occupation, a real mania to which I became enslaved and from which I can only with great difficulty free myself'.

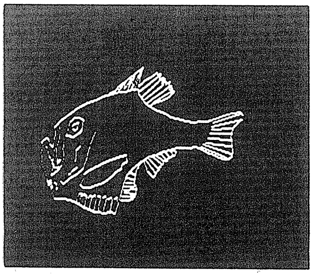
Even if you have only a passing interest in graphics (and none in programming or even computers) this program is addictive. The program more or less does for the would be tesselator what Wordwise does for the writer. Neither will improve your talents if you have none, and neither intrudes more than the absolute minimum between your ideas and their execution by the computer.

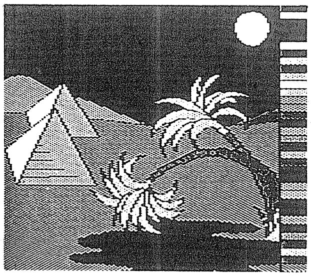
Tesselations work on the mathematical principle that a variety of shapes or tiles can be made to interlock, leaving no gaps for a background colour between them, and is widely used in the production of textiles and wall/floor coverings. The term is now accepted as covering any shapes that repeat in regular order without gaps, and fall into two classes. The first class is the regular tesselation of equilateral triangles or regular hexagons, while the second is the semi-regular combination of regular polygons - for instance a hexagon with a square on each face and the space between them infilled with an isosceles triangle.

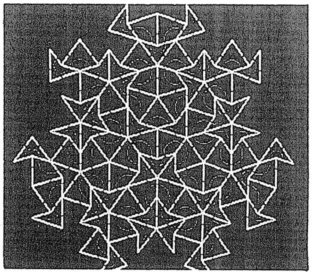
Clearly, hexagonal floor tiles tesselate (the term comes from the Latin tessala, for a small decorative tile) - but fish, birds and other irregular shapes? The answer is yes: if you are Escher it can be done only with blood, sweat and tears, while if you use this program it becomes an enjoyable process.

A series of basic shapes can be loaded, from squares to interlocking darts and kites, the latter introducing the user to the concept of using a pair of interlocking shapes which can be manipulated separately or as a pair. The shapes you want are created by loading the main program and using the cursor and other keys to distort or rotate the tile.

Having done the distortions (to get the outline one wants) pressing 'T' will cause the design to tesselate to the size chosen in a scale from 0.2 to 4.0. Then you can return to your shape to decorate and paint it (in monochrome, two or four colours, with a consequent lowering of the screen resolution) and re-tesselate







in another scale to your heart's content.

The 78 page manual is among the better I have seen, and the program is sufficiently robust in fact for one to start before a thorough reading. However, it is still possible to be inadvertently taken back into Basic with a loss of current data. Regrettably, not all the single key commands are the same in each part of the suite - for instance there is no facility to save the data in the Patchwork section. However, this section is not only not protected, but is listed in the manual so you can work out your own dump. Clearly one cannot expect that an irregular tesselation saved from one part of the program can be loaded back into the regular part, but there is an Epson printer dump for those who need hard copy. Naturally the program cries out for a colour monitor - and printer.

GRAPHITO, a Complete Graphics System (40T disc £27.95, cassette £21.95), by Hallam and published by Addison-Wesley.

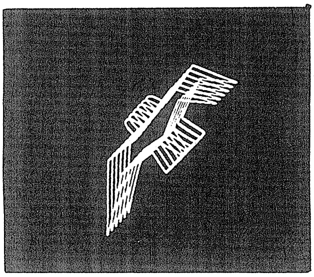
While Tesselator can be enjoyed just for itself, Graphito is quite different - more of a tool for the programmer. Written by the same author, the four main 2D and two 3D programs are unprotected and indeed require the user to enter a short Basic program to make use of the procedures in the main program. For each program there is a two-line header which is listed in the manual with variations to allow you to make more economic use of memory if certain main program resources are not required. Clearly better use of memory allows displays in higher resolution or colour at the expense of the procedures.

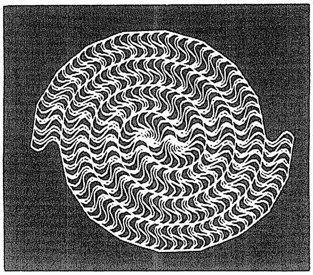
Up to 26 procedures are available within the programs and these are accessed by the short programs referred to above. Also included are a series of some three dozen pre-drawn motifs and four alphabets which are called up and manipulated via the procedures. In some of the programs it is possible to alter the motifs (those supplied or your own) from the keyboard or by joystick. Some of the basic motifs look rather dull, but even these can be enhanced enormously. Each PROCload routine in the manual is accompanied by a mini screen dump to show what you should get. The 3D programs do provide a degree of hidden surface removal.

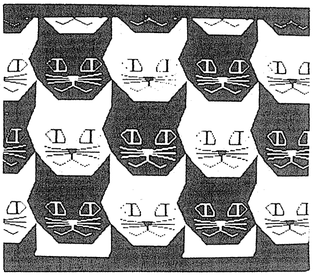
Apparently there are exactly 17 ways in which an asymmetric motif can be arranged to form a two-dimensional network pattern. These are known as wallpaper groups and Graphito will produce them all by regular stepping, mirroring, rotating and 'flopping' - this form of manipulation is not to be confused with tesselation as they do not need to interlock fully, or even at all. Interesting interference patterns can also be built by overlapping two existing designs to make a third.

As I have said, Graphito is more of a tool than an end in itself, and the programs can be incorporated in your own work. Either package could be used to introduce mathematicians to graphics (or vice versa), and indeed the Tesselator manual has a section on the program's use in maths teaching.

Certainly a much better understanding of both program structure and one's own aims are required for Graphito than Tesselator and Graphito's 112 page manual, while comprehensive, will be daunting for the beginner with too many references to points that will be covered later.

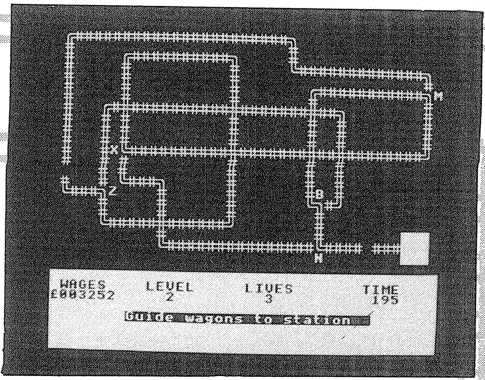






Trackman

Throw the points and control the trains in this entertaining and demanding game by Silas and Tom Standage.



WAGES £003252 LEVEL 2 LIVES 3 TIME 195

Guide wagons to station

Can you rise to the challenge and safely negotiate the railway system without causing any damage? The better you do your job, the more money you will make.

Trackman is an enjoyable change from most games. This one features no arcade zapping lasers nor mutant beings from other planets, but a tortuous and twisting railway track with trains.

You have to keep the train or trains moving safely around the network until the time-out counter reaches zero. Then your task is to return the train(s) to the station as quickly as possible to boost your earnings. You earn money in two ways. Firstly, you earn £10 for every bend taken, or set of points crossed, and you earn a bonus for directing the trains back to the station. You can lose money too, either by reversing the trains, or by taking too long on the return to the depot.

The game requires a lot of thought, and some fast responses, especially when there is more than one train on the track only seconds away from disaster. Your main task is to control the five sets of points and run a safe and efficient railway.

The keys to change the points are:
Z, X, B, N, M
Each key will change the correspondingly labelled set of points on the screen layout. All the trains will be reversed on pressing the space bar, but remember that this loses you money, and should be kept for emergencies.

If you always wanted to play trains and run a railway of your own, now's your chance with our unusual and demanding game of Trackman.

PROGRAM NOTES

For those who find this game too fast, or are running the program across the tube (tube-trains!), you can slow the action down by altering the value of 200 at line 350. Changing this value to 300 should be sufficient.

If you wish to change the running order for the number of trains and the speed, then the data at lines 2210 and 2220 can be changed. The data is arranged in pairs, the first item being the number of trains, and the second item being the speed (1 for slow, 2 for fast).

The track is decoded between lines 1030 and 1100 from the data at lines 1120 to 1180. Each piece of data contains a single digit number followed by a character. This represents the number of times that character is repeated. For example, with an asterisk representing a space and the letter 'a' a straight piece of track, the data '9*9*4a' decodes to 9 spaces followed by 9 more spaces (single digits only are allowed) followed by 4 straight pieces of track. The codes used for the track layout are:

*	space
a	straight track
A	Top right hand bend
B	Top left hand bend
C	Bottom left hand bend
D	Bottom right hand bend
£	Railway station

Bends are viewed as if they formed a square.

With this information you can redesign the track layout to suit your own inclinations (great fun for all you train buffs).

Because this program runs in mode 1, there is insufficient room on a disc based machine (with PAGE set to &19ØØ or higher) to run this game. Either set PAGE to &12ØØ before loading and running, or use a move-down routine such as the one in BEEBUG Vol.3 No.5.

```
   1Ø REM PROGRAM TRAKMAN
   2Ø REM VERSION B1.2
   3Ø REM AUTHOR  T & S STANDAGE
   4Ø REM BEEBUG  JUNE 1985
   5Ø REM PROGRAM SUBJECT TO COPYRIGHT
   6Ø :
  1ØØ ON ERROR GOTO 294Ø
  11Ø T%=4:highscore=Ø
  12Ø DIM X%(T%),Y%(T%),D%(T%),S%(T%),E%
(T%)
  13Ø DIM NT(9),SP(9),x(5),y(5)
  14Ø MODE1
  15Ø VDU19,Ø,4,Ø,Ø,Ø
  16Ø VDU23,1,Ø;Ø;Ø;Ø;
  17Ø REPEAT:PROCcb
  18Ø PROCt:MODE1
  19Ø VDU23,1,Ø;Ø;Ø;Ø;
  2ØØ PROCbegin:PROCs
  21Ø LEVEL=1:TTG=3ØØØ
  22Ø BL=Ø:crash=Ø:IF LEVEL=1Ø LEVEL=9:T
TG=TTG+5ØØ
  23Ø PROCset
  24Ø T%=NT(LEVEL):SP%=SP(LEVEL):COLOUR1
  25Ø TIME=Ø:PRINTTAB(26,1)"Hi = £";high
score
  26Ø COLOUR3:PRINTTAB(8,28);STRING$(23,
CHR$32);:REPEAT:PROCsc
  27Ø PROCpoints
  28Ø SOUNDØ,-9,5,1
  29Ø IF LEVEL=6 OR LEVEL=9 FORYYY=1TO2
  3ØØ NT%=T%:IF LEVEL=6 OR LEVEL=9 NT%=N
T%-1
  31Ø FORtrain=1TONT%:PROCm(train)
  32Ø IF crash=1 train=99
  33Ø NEXT
  34Ø IF LEVEL=6 OR LEVEL=9 NEXTYYY:PROC
m(NT%+1)
  35Ø FORF=1TO2ØØ-(SP%*1ØØ):NEXT
  36Ø UNTIL crash=1 OR TIME>TTG
  37Ø IF TIME>TTG PRINTTAB(32,26);"Ø";SP
C(3):PROCtune(2):GOTO4ØØ
  38Ø IF crash=1 PROCcrash:L=L-1:IF L>Ø
PROCset:GOTO22Ø
  39Ø PROCgameover:UNTILØ
  4ØØ TIS=Ø:PRINTTAB(8,28);"Guide wagons
 to station"
  41Ø BL=1:TIME=Ø:REPEAT:PROCsc
  42Ø FORFF=1TO3:IF X%(FF)=35 AND Y%(FF)
=21 AND E%(FF)=1 TIS=TIS+1:E%(FF)=Ø:VDU3
1,35,21,255
  43Ø NEXTFF
  44Ø FOR TRAIN=1 TO NT(LEVEL)
  45Ø PROCm(TRAIN):NEXTTRAIN
  46Ø PROCpoints:SOUNDØ,-9,5,1
  47Ø UNTILTIS=NT(LEVEL) OR crash=1
  48Ø IF crash=1 PROCcrash
  49Ø IF crash=Ø SC=SC+(3ØØØ-TIME):FORF=
1TO2ØØSTEP1Ø:SOUND1,4,F,1:NEXT
  5ØØ FORF=1TO4:E%(F)=1:NEXT
  51Ø LEVEL=LEVEL+1
  52Ø BL=Ø:PRINTTAB(8,28);SPC(25)
  53Ø GOTO22Ø
  54Ø END
  55Ø :
 1ØØØ DEFPROCs
 1Ø1Ø VDU28,Ø,31,39,24,17,131,12,26
 1Ø2Ø VDU17,3,1Ø,17,128
 1Ø3Ø RESTORE112Ø
 1Ø4Ø FORo=1TO22:READs$:Q=1
 1Ø5Ø v=ASC(MID$(s$,Q,1))-48:IF v=Ø THEN
 11ØØ
 1Ø6Ø v2=ASC(MID$(s$,Q+1,1))+159:IF v2>2
53 v2=v2-1
 1Ø7Ø IF v2=2Ø1 v2=32
 1Ø8Ø PRINT STRING$(v,CHR$v2);
 1Ø9Ø Q=Q+2:GOTO1Ø5Ø
 11ØØ PRINT:NEXT
 111Ø VDU31,6,16,9Ø,31,6,12,88,31,39,6,7
7,31,27,16,66,31,27,22,78
 112Ø DATA "1*1A9a9a4a1BØØ","1*1a9*9*4*1
aØØ","1*1a5*1A9a1a1B5*1aØØ"
 113Ø DATA "1*1a5*1a9*1*1a5*1D9a4a1BØØ",
"1*1a5*1a9*1*1a9*9*1*1aØØ","1*1a5*1a9*1*
1a7*1A9a2a1BØØ"
 114Ø DATA "1*1a5*1a9*1*1a7*1a7*4*1aØØ",
"1*1a3*1A9a9a5a1B8*1aØØ","1*1a3*1a1*1a9*
1*1a7*1a2*1a8*1aØØ"
 115Ø DATA "1*1a3*1a1*1a9*1*1a7*1a2*1a8*
1aØØ","1*1a3*1a1*1a9*1*1a7*1a2*1a8*1aØØ"
,"1*1a3*1a1*1D9a9a9a3a1CØØ"
 116Ø DATA "1*1a3*1a1*1a9*1*1a7*1a2*1aØØ
","1*1a3*1a1*1a9*1*1a7*1a2*1aØØ","1*1a3*
1a1*1D3a1B6*1a7*1a2*1aØØ"
 117Ø DATA "1*1D3a1C5*1a6*1a7*1a1*1*1aØØ
","4*1*1a5*1a6*1a7*1D1B1a1CØØ","5*1a5*1a
6*1a8*1aØØ"
 118Ø DATA "5*1D8a4a1C8*1aØØ","9*2*1a9*6
*1a8*3£ØØ","9*2*1D9a6a1C8a3£ØØ","9*9*9*1
*8*3£ØØ"
 119Ø ENDPROC
 12ØØ :
 121Ø DEFFNn(X,Y)
```

```
1220 VDU31,X,Y:A%=135
1230 !&70=USR(&FFF4)
1240 =?&71+96
1250 :
1260 DEFPROCset
1270 RESTORE1310:T%=4
1280 FORF=1TOT%:E%(F)=1
1290 READ X%(F),Y%(F),D%(F)
1300 S%(F)=255:NEXT
1310 DATA 1,9,3,31,12,4,18,18,1,7,10,1
1320 ENDPROC
1330 :
1340 DEFPROCm(T%)
1350 IFE%(T%)=Ø ENDPROC
1360 COLOUR3
1370 VDU31,X%(T%),Y%(T%),S%(T%)
1380 I%=D%(T%):S=S%(T%)
1390 X%=X%(T%):Y%=Y%(T%)
1400 X%=X%+(1 AND I%=2)-(1 AND I%=4)
1410 Y%=Y%+(1 AND I%=3)-(1 AND I%=1)
1420 X%(T%)=X%:Y%(T%)=Y%
1430 J%=FNn(X%(T%),Y%(T%))
1440 COLOURT%:IF T%=4 COLOUR1
1450 IF J%=255 VDU31,X%(T%),Y%(T%),229:
S%(T%)=255:COLOUR3:ENDPROC
1460 SC=SC+1Ø
1470 IF J%=229 PROCcr:ENDPROC ELSE IFJ%
<223 OR J%>227 PROCcr
1480 IF I%=1 AND (J%=226 OR J%=227) PRO
Ccr
1490 IF I%=2 AND (J%=224 OR J%=227) PRO
Ccr
1500 IF I%=3 AND (J%=224 OR J%=225) PRO
Ccr
1510 IF I%=4 AND (J%=226 OR J%=225) PRO
Ccr
1520 IF J%=224 AND I%=4 I%=3 ELSE IF J%
=224 AND I%=1 I%=2
1530 IF J%=225 AND I%=2 I%=3 ELSE IF J%
=225 AND I%=1 I%=4
1540 IF J%=226 AND I%=3 I%=4 ELSE IF J%
=226 AND I%=2 I%=1
1550 IF J%=227 AND I%=4 I%=1 ELSE IF J%
=227 AND I%=3 I%=2
1560 SOUNDØ,-9,5,1
1570 S%(T%)=FNn(X%,Y%)
1580 X%(T%)=X%:Y%(T%)=Y%
1590 D%(T%)=I%
1600 VDU31,X%,Y%,229
1610 COLOUR3
1620 ENDPROC
1630 :
1640 DEFPROCpoints
1650 A$=INKEY$(Ø):*FX21
1660 IF A$="Z" PROCc(1)
1670 IF A$="M" PROCc(2)
1680 IF A$="B" PROCc(3)
1690 IF A$="X" PROCc(4)
1700 IF A$="N" PROCc(5)
1710 IF A$=" " PROCswap
1720 ENDPROC
1730 :
1740 DEFPROCc(q)
1750 W=FNn(x(q),y(q))
1760 IF q<>4 W=W+1
1770 IF q<3 AND W=227 W=225
1780 IF q=3 AND W=226 W=224
1790 IF q<>4 THEN 1810
1800 IF W=227 W=224 ELSE W=227
1810 IF W=228 W=226
1820 VDU31,x(q),y(q),W
1830 SOUND 1,4,53,2
1840 ENDPROC
1850 :
1860 DEFPROCcr
1870 SOUNDØ,-15,5,2
1880 crash=1:ENDPROC
1890 :
1900 DEFPROCswap
1910 FL=Ø:FORF=1TOT%:IF S%(F)>223 AND S
%(F)<228 FL=1
1920 NEXT:IF FL=1 VDU7:ENDPROC
1930 FORF=1TOT%:D%(F)=D%(F)+2
1940 D%(F)=D%(F) MOD 4
1950 IF D%(F)=Ø D%(F)=4
1960 NEXT
1970 SC=SC-5Ø:IF SC<Ø SC=Ø
1980 ENDPROC
1990 :
2000 DEFPROCcb
2010 VDU23,224,Ø,Ø,31,48,32,39,36,36
2020 VDU23,225,Ø,Ø,248,12,4,228,36,36
2030 VDU23,226,36,36,228,4,12,248,Ø,Ø
2040 VDU23,227,36,36,39,32,48,31,Ø,Ø
2050 VDU23,255,36,36,255,36,36,255,36,3
6
2060 VDU23,254,255,255,255,255,255,255,
255,255
2070 VDU23,229,255,195,165,153,153,165,
195,255
2080 x(1)=5:x(2)=38:x(3)=27:x(4)=7
2090 x(5)=27:y(1)=16:y(2)=6:y(3)=17
2100 y(4)=12:y(5)=21
2110 ENDPROC
2120 :
2130 DEFPROCbegin
2140 VDU19,Ø,4,Ø,Ø,Ø,19,3,3,Ø,Ø,Ø
2150 VDU19,2,2,Ø,Ø,Ø
2160 VDU23,1,Ø;Ø;Ø;Ø;
2170 PROCset
2180 FORF=1TO9
2190 READ NT(F),SP(F)
2200 NEXT:SC=Ø:L=3
2210 DATA 1,1,2,1,2,2,3,1,3
2220 DATA 2,3,2,4,1,4,2,4,2
2230 ENDPROC
2240 :
2250 DEFPROCtune(i)
```

```
 2260 ENVELOPE1,1,0,0,0,1,1,1,60,-4,-1,-
1,126,90
 2270 ENVELOPE2,1,0,0,0,1,1,1,60,-4,-1,-
1,126,90
 2280 ENVELOPE3,1,0,0,0,1,1,1,60,-4,-1,-
1,126,90
 2290 ENVELOPE 4,1,-4,-3,-2,12,12,12,126
,-4,-2,-1,126,86
 2300 *FX15,0
 2310 contr=0
 2320 IF i=1 THEN RESTORE 2410 ELSE REST
ORE 2450
 2330 REPEAT
 2340 FORC=&201TO&203
 2350 READP,D:P=P+5
 2360 SOUNDC,C,P,D*2
 2370 contr=contr+1
 2380 NEXTC
 2390 UNTILcontr=102/i
 2400 ENDPROC
 2410  DATA145,2,145,2,69,2,145,2,133,2,
117,2,137,2,137,2,61,2,137,2,125,2,109,2
,133,2,133,2,57,2,133,1,117,1,97,1,125,5
,109,5,49,5
 2420 DATA117,2,101,2,41,2,117,1,101,1,4
1,1,109,2,73,2,45,2,105,2,69,2,41,2,97,3
,85,3,21,3,9,1,9,1,9,1,57,1,57,1,57,1,13
,1,13,1,13,1,61,1,61,1,61,1,17,1,17,1,17
,1,65,1,65,1,65,1
 2430 DATA145,2,145,2,69,2,145,2,133,2,1
17,2,137,2,137,2,61,2,137,2,125,2,109,2,
133,2,133,2,57,2,133,1,117,1,97,1,125,5,
109,5,49,5
 2440 DATA117,2,101,2,41,2,117,1,101,1,4
1,1,109,2,73,2,45,2,105,3,69,3,41,3,117,
2,101,2,41,2,117,1,101,1,41,1,109,2,73,2
,45,2,105,2,69,2,41,2,97,9,85,9,21,9
 2450 DATA117,2,53,2,53,2,113,2,53,2,53,
2,117,2,101,2,81,2,129,2,101,2,89,2,129,
2,33,2,33,2,113,2,33,2,33,2,117,4,97,4,8
1,4,101,1,69,1,53,1,97,1,69,1,49,1,89,1,
41,1,41,1,85,1,37,1,37,1,81,1,33,1,33,1
 2460 DATA85,1,37,1,37,1,89,1,41,1,41,1,
97,1,49,1,49,1,53,4,101,4,101,4,5,4,5,4,
5,4
 2470 :
 2480 DEFPROCcrash
 2490 FORF=185TO0STEP-6
 2500 SOUND1,-15,F,1
 2510 SOUND2,-15,F+10,1
 2520 SOUND3,-15,F+20,1
 2530 NEXT
 2540 *FX21 5
 2550 *FX21 6
 2560 *FX21 7
 2570 SOUND0,-12,5,2
 2580 SOUND0,-15,5,6
 2590 FORF=-15 TO 0
 2600 SOUND0,F,5,1:NEXT
 2610 COLOUR3:FORF=1TONT%+1
 2620 VDU31,X%(F),Y%(F),S%(F)
 2630 NEXT
 2640 ENDPROC
 2650 :
 2660 DEFPROCsc
 2670 PRINTTAB(0,25);:VDU17,1,17,131
 2680 S$=STR$(SC):S$=STRING$(6-LEN(S$),"
0")+S$
 2690 PRINT" WAGES     LEVEL      LIVES
    TIME"
 2700 PRINT"`";S$;TAB(12);LEVEL;TAB(22);
L;TAB(26);:IF BL=1 PRINT TIME;SPC(2) ELS
E PRINT TTG-TIME;CHR$32
 2710 VDU17,3,17,128:ENDPROC
 2720 :
 2730 DEFPROCt:VDU19,3,3,0,0,0
 2740 VDU19,2,2,0,0,0,17,2
 2750 S=6:PRINTTAB(16,4)"TRACKMAN"
 2760 PRINTTAB(16,5)STRING$(8,"=")
 2700 PRINT"£";S$;TAB(12);LEVEL;TAB(22);
L;TAB(26);:IF BL=1 PRINT TIME;SPC(2) ELS
E PRINT TTG-TIME;CHR$32
 around the"'" track, until the timer re
aches zero."'" Then shunt each train bac
k to the"'" station as quickly as possib
le."
 2790 PRINT'" To change the points, use
the"'" following keys:"''TAB(8)"Z   X
B   N   M"''" and to reverse the train(s
) use the"'" space bar."
 2800 COLOUR1:PRINTTAB(8,30)"Press space
 bar to start.";
 2810 REPEAT UNTIL INKEY-99
 2820 PROCtune(1)
 2830 ENDPROC
 2840 :
 2850 DEFPROCgameover
 2860 IF SC>highscore highscore=SC
 2870 VDU31,14,9:PRINTSPC(10):VDU31,14,1
0:PRINT"GAME  OVER":VDU31,14,11:PRINTSPC
(10)
 2880 FORF=1TO4000:NEXT
 2890 *FX15 0
 2900 FORF=1TO32:VDU30,11
 2910 SOUND0,-15,4,1:I$=INKEY$(3)
 2920 NEXT:*FX21 4
 2930 ENDPROC
 2940 :
 2950 ON ERROR OFF:MODE 7
 2960 IF ERR<>17 REPORT:PRINT" at line "
;ERL
 2970 END
```

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams
Assistant Editor: Geoff Bains
Production Editor: Phyllida Vanstone
Technical Assistant: Alan Webster
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft
Additional thanks are due to *Sheridan Williams, Adrian Calcraft, John Yale and Tim Powys-Lybbe.*

Editorial Address
BEEBUG
PO BOX 50
St. Albans
Herts.

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. A leaflet, 'Notes of Guidance for Contributors' is available on receipt of an A5 (or larger) SAE.

In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", or other means, but please ensure an adequate written description of your contribution is also included. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, subscription queries and orders for back issues to the subscriptions address.

MEMBERSHIP SUBSCRIPTION RATES

£ 6.40	*6 months (5 issues) UK ONLY*		
£11.90	UK - 1 year (10 issues)		
£18	Europe,	£21	Middle East
£23	Americas & Africa,	£25	Elsewhere

BACK ISSUES
(Members only)

Vol.	Single Issues	Volume sets (10 issues)
1	80p	£7
2	90p	£8
3	£1	£9
4	£1	–

Please add the cost of post and packing as shown:

DESTINATION	First issue	Each subsequent issue
UK	30p	10p
Europe	70p	20p
Elsewhere	£1.50	50p

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

Back issues are for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

Subscriptions, Back Issues &
Software Address

BEEBUG
PO BOX 109
High Wycombe
Bucks. HP10 8HQ

Hotline for queries and software orders

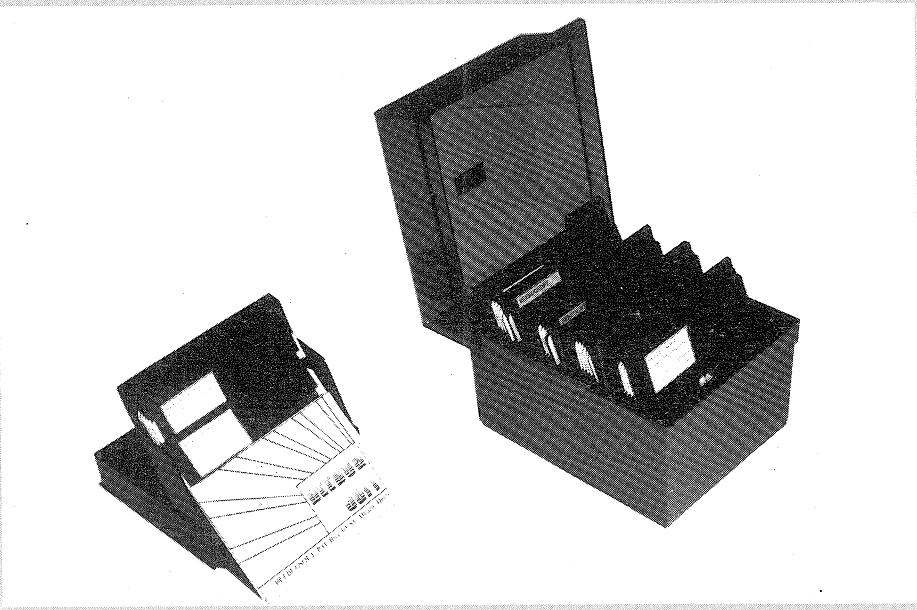
St. Albans (0727) 60263
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access and Barclaycard orders, and subscriptions
Penn (049481) 6666

If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

High Quality Low Priced Discs

Backed by The Reputation of BEEBUG



10 S/S D/D Discs – £13.90	10 D/S D/D Discs – £19.40
25 S/S D/D Discs – £33.45	25 D/S D/D Discs – £46.95
50 S/S D/D Discs – £59.30	50 D/S D/D Discs – £87.05

All Prices Include Storage Box, VAT and Delivery to Your Home (UK).

All discs are 100% individually tested, supplied with hub ring as standard, and guaranteed error free. They are ideal for use on the BBC Micro and have performed perfectly in extensive tests at BEEBUG over many months.

Orders for 25 or 50 are delivered in strong plastic storage boxes with four dividers. Orders for 10 are sent in smaller hinged plastic library cases.

We are also able to offer the empty storage container, which holds up to 50 discs for £10 including VAT and post.

Please use the order form enclosed
or order directly from:
BEEBUGSOFT, P.O. Box 109,
High Wycombe, Bucks HP10 8HQ.

BEEBUG SOFT

Magazine Cassette/Disc

JUNE 1985 CASSETTE DISC CONTENTS

STONEHENGE graphically displayed

MULTIPLE WINDOWS made easy

WORDWISE PLUS – segment programs

EXTENDED DISC CATALOGUE – full of information

WORKSHOP PROCEDURES – binary search and hash search

MAKING MUSIC – six complete programs of musical effects

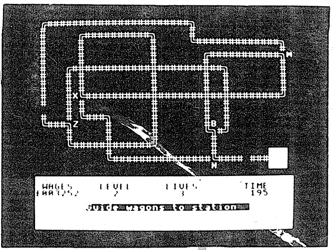
BEGINNERS EXAMPLES on using logic

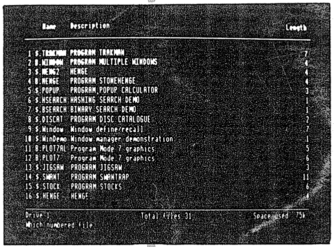
TRACKMAN – a challenging game all about trains

Extra features this month

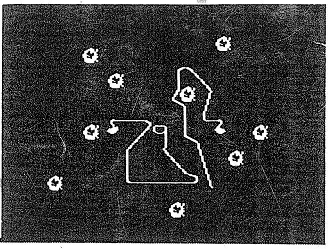
SWAN TRAP – a colourful game for two players (can also be played by one) with appropriate musical accompaniment. Very addictive.

LEMMING – This is the winning program in the Cliff Toppers Brainteaser competition. Excellent graphics and sound effects.





```
    Name    Description                              Length

 1 $.TRAKMAN PROGRAM TRAKMAN                              7
 2 B.WINDOW  PROGRAM MULTIPLE WINDOWS                     4
 3 $.HENG2   HENGE                                        4
 4 B.HENGE   PROGRAM STONEHENGE                           4
 5 $.POPUP   PROGRAM POPUP CALCULATOR                     3
 6 $.HSEARCH HASHING SEARCH DEMO                          1
 7 $.BSEARCH BINARY SEARCH DEMO                           1
 8 $.DISCAT  PROGRAM DISC CATALOGUE                       2
 9 $.Window  Window define/recall                         7
10 $.WinDemo Window manager demonstration                 1
11 B.PLOT7AL Program Mode 7 graphics                      5
12 B.PLOT7   Program Mode 7 graphics                      6
13 $.JIGSAW  PROGRAM JIGSAW                               3
14 $.SWANT   PROGRAM SWANTRAP                            11
15 $.STOCK   PROGRAM STOCKS                               6
16 $.HENGE   HENGE                                        4

Drive 1              Total files 31           Space used  75k
Which numbered file
```



All this for £3.00 (cass) £4.75 (disc) +50p p&p.
Back issues (disc since Vol.3 No. 1, cass since Vol.1 No. 10) available at the same prices.

Subscription rates	DISC UK	CASS UK	DISC O'seas	CASS O'seas
6 months (5 issues)	£25	£17	£30	£20
12 months (10 issues)	£50	£33	£56	£39

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to disc subscriptions on receipt of £1.70 per issue of the subscription left to run.

All subscriptions and individual orders to
BEEBUGSOFT, PO BOX 109, High Wycombe, Bucks, HP10 8NP.

Printed in England by Staples Printers St Albans Limited at The Priory Press.   ISSN 0263-7561