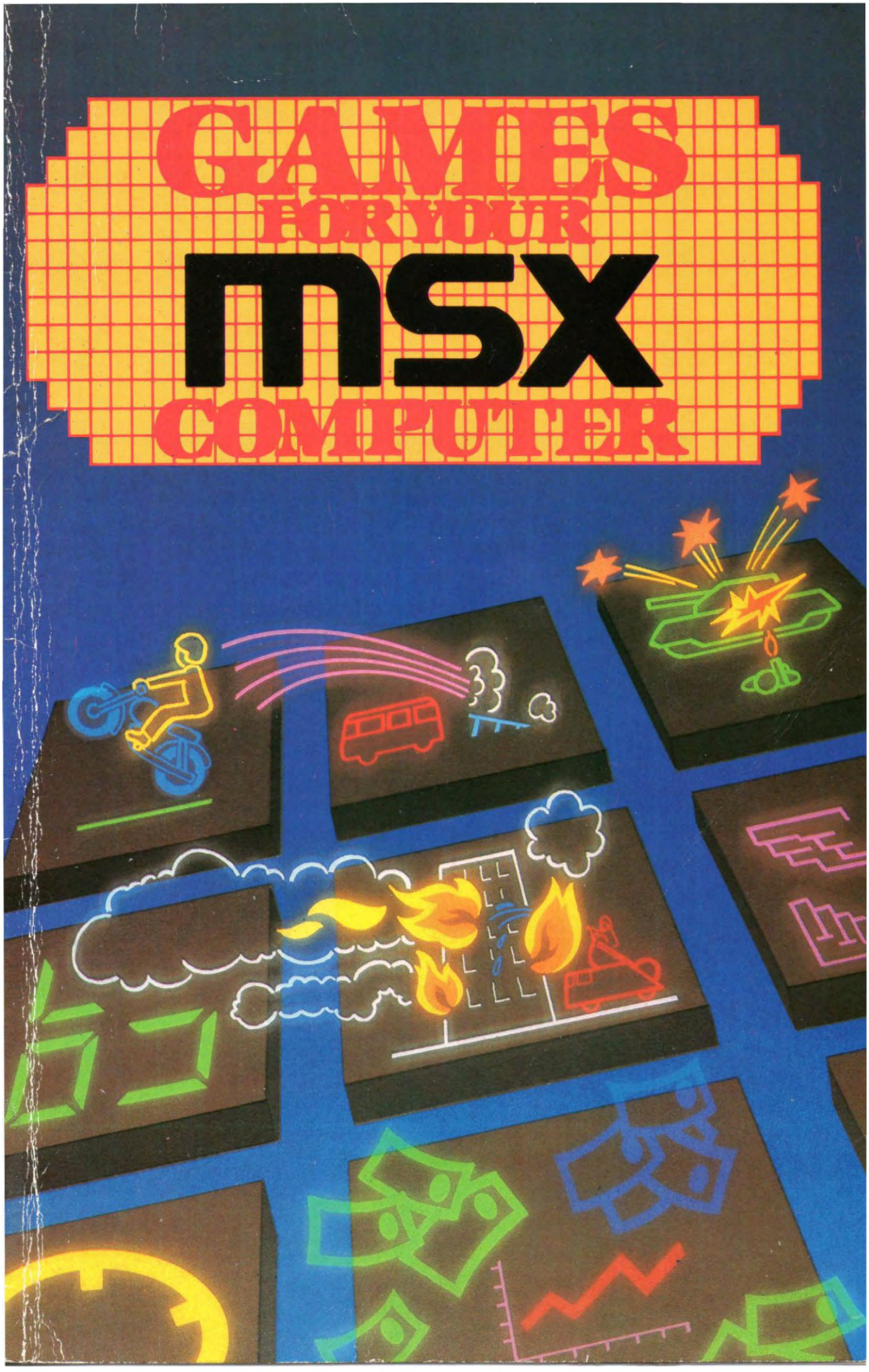
GAMES FOR YOUR MSX COMPUTER



GAMES FOR YOUR MSX

By Graham Carter

GAMES FOR YOUR MSX

By Graham Carter

Virgin Books

First published in Great Britain in 1985 by Virgin Books Ltd,
328 Kensal Road, London W10 5XJ.

ISBN 0 86369 092 0

Printed and bound in Great Britain by Richard Clay
(The Chaucer Press) Ltd, Suffolk.

Production services by Book Production Consultants,
Cambridge.

Illustrations and artwork by Sue Walliker.

Typeset by Keyline Graphics.

Distributed by Arrow Books.

TO SUE AND STEVE POWELL

TIM HARTNELL – THE SERIES EDITOR

Tim Hartnell is the most widely-published computer author in the world. Founder of the National ZX Users' Club, and founding editor of *ZX Computing* magazine, Tim has been involved over the years in a wide variety of computer activities. His published works include *The Personal Computer Guide* (Virgin Books) and *The Giant Book of Computer Games (Fontana).*

GRAHAM CARTER – THE AUTHOR

Graham Carter is a 20-year-old student in his final year of a B/TEC National Diploma course in Computer Studies. He is hoping to go to University to continue his studies in this field to degree level. He has been working with computers for over four years. His other interests include music and photography.

SUE WALLIKER – THE ILLUSTRATOR

Sue Walliker is a freelance illustrator.

ACKNOWLEDGEMENTS

I would especially like to thank Andrew Ogilvie and Carol Vincent for the original programs they supplied for use in this book, and for the help they gave me during its preparation. I would also like to thank Paul Precious, Simon Gould, Scott Vincent, Clive Gifford, Damon Pillinger and Danny Olesh for their program ideas.

Contents

Editor's Introduction

Typing in a computer program is like opening an unknown door. You do not know until you actually open the door – or, in our case, run the program – what experience is waiting for you. Of course, the sign on the door has given you some indication, but nothing can equal first-hand experience.

You do not know precisely what experiences are waiting for you in the great programs in this book. Of course, if the introduction says you're entering a space game, it's very likely the program won't play 'Guess My Number' when you get it up and running. But the listing rarely hints at the computer's game-playing strategy, or the screen display, or the fun that is waiting for you.

This book has a number of unknown doors – doors leading into outer space and into the fiendish worlds of computer intelligence, wizards and Adventure.

We've provided the doors . . . and the keys. All you have to do to turn the lock is type in the program, and run it. Whatever you find behind each door, I guarantee you won't be disappointed.

Tim Hartnell
January 1985

Author's Introduction

The programs in this book will work on any of the new range of MSX computers. Many of them use the advanced colour and graphics facilities offered by MSX BASIC to add interest and realism to the games. They are a mixture of arcade games, strategy games and gambling games.

Contained within the listings are many REM statements to explain the purpose of each block of code; by reading through the programs you may see routines which you will want to incorporate in your own programs. In particular, the 'Mastercode' and 'Hangman' games contain useful string handling routines for validating user inputs.

I have tried to make the most of the commands available on the MSX machines when writing these programs, and to demonstrate just how quickly the machine works.

I hope you enjoy typing the programs in and playing them – once you have RUN a program a few times, don't hesitate to change it if you think it can be improved upon.

Graham Carter
January 1985

Program Notes

These programs should be typed in exactly as they appear in this book. They have been specially printed so that the listings reproduced here are identical to the way they should look on your television screen.

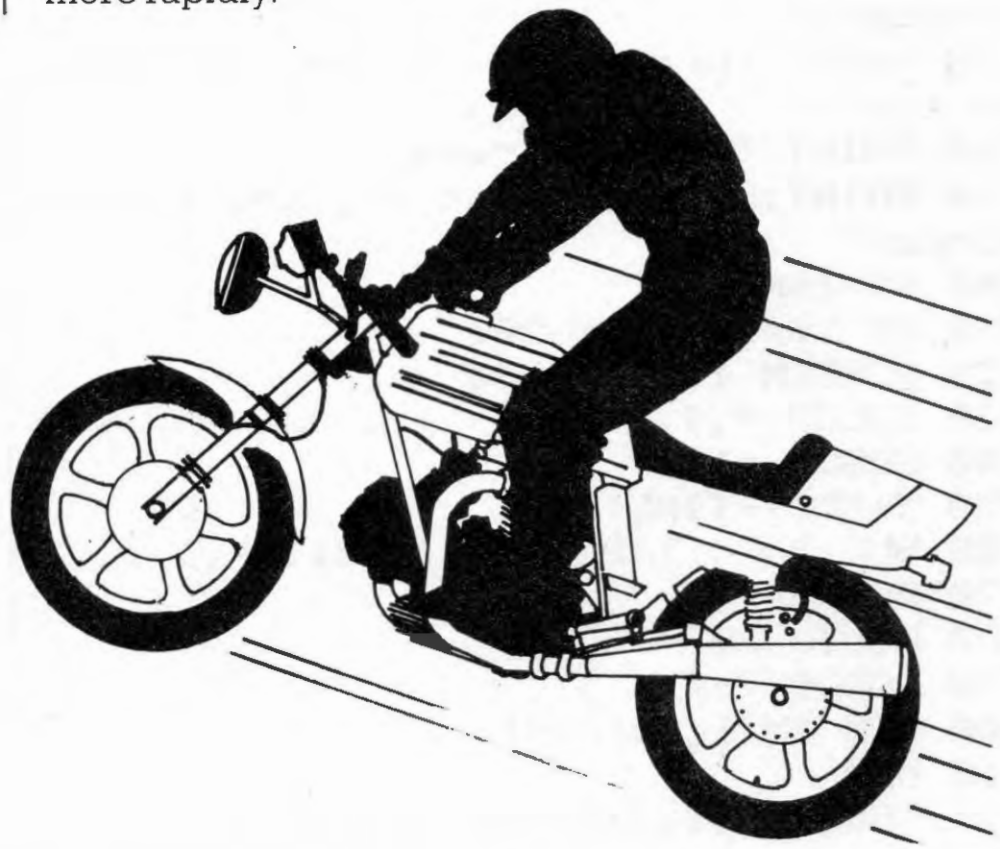
Some of the listings contain special graphics characters. There is a diagram in the manual supplied with your machine which shows where these graphics characters are on the keyboard, and which keys you must press to obtain them.

MOTORCYCLE STUNTMAN

In this game, it's on with your crash helmet and away you go. Your object is to jump over a line of buses, which change in number randomly every game.

Press any key to start, and watch your speed increase at the top of the screen. When you think that you have sufficient speed to clear the line of buses, release the bike's brakes by pressing the '1' key and see your bike fly over the ramps.

This game uses accurate formulae for thrust and velocity. If you want the speed to increase at a lower rate, just change the loop in line 380 – a number greater than 15 will make the speed increase more slowly, while a number less than 15 will make the speed increase more rapidly.



```
10 REM **********************
20 REM *Motorcycle Stuntman*
30 REM *S. Vincent and      *
40 REM *G. Carter, 1984     *
50 REM **********************
60 SCREEN 0
70 KEY OFF
80 COLOR 14,1,1
90 WIDTH 38
100 REM *Instructions*
110 REM ***************
120 PRINT TAB(10);"Motorcycle Stuntma
n"
130 PRINT TAB(10);"~~~~~~~~~~~~~~~~~~~~~~~~~~
~~"
140 LOCATE 0,4:PRINT "Press any key a
nd watch your speed "
150 PRINT "increase.  When you think
that your"
160 PRINT "bike will clear the line o
f buses,"
170 PRINT "press the '1' key to relea
se the"
180 PRINT "bike's brakes."
190 PRINT:PRINT "Press any key to con
tinue."
200 K$=INKEY$
210 IF K$="" THEN 200
220 SCREEN 1:WIDTH 30
230 COLOR 9,1,1
240 GOSUB 1110
250 X=RND(-TIME)
260 BUS=RND(1):BUS=INT(BUS*17)+2
270 SCREEN 1
280 WIDTH 30
290 GOSUB 760
300 REM *M is speed*
310 M=0
320 LOCATE 11,1:PRINT "0 M.P.H."
```

```
330 IF INKEY$="" THEN 330
340 REM *+ speed until 1 key pressed*
350 REM ****************************
360 K$=INKEY$
370 R=RND(1):M=M+5+INT(R*6)
380 FOR X=1 TO 15:NEXT X
390 IF M>170 THEN M=170
400 LOCATE 11,1:PRINT M;" M.P.H"
410 IF K$<>"1" THEN 360
420 MP=M
430 Z=RND(1):Z=INT(Z*11)
440 M=M+(Z-5)-40
450 J=9+INT(M/7.5+.5)
460 IF M<-5 THEN J=6
470 REM *Plot cycle on screen*
480 REM ***********************
490 FOR N=1 TO 4
500 PUT SPRITE 0,((8*N)+7,74),13,0
510 NEXT N
520 PUT SPRITE 0,(47,67),13,0
530 PUT SPRITE 0,(55,60),13,0
540 IF J=6 THEN 580
550 FOR N=7 TO J
560 PUT SPRITE 0,((N*8)+7,60),13,0
570 NEXT N
580 PUT SPRITE 0,(((J+1)*8)+7,67),13,
0
590 PUT SPRITE 0,(((J+2)*8)+7,74),13,
0
600 IF J<>BUS+7 THEN 890
610 REM *Plot bike's landing*
620 REM **********************
630 FOR N=J+3 TO 31
640 PUT SPRITE 0,((N*8)+7,74),13,0
650 NEXT N
660 FOR X=1 TO 1000:NEXT X
670 LOCATE 31,10:PRINT " "
680 CLS
690 LOCATE 11,4:PRINT "Great jump!"
```

```
700 LOCATE 2,6:PRINT "You cleared ";B
US;" buses at"
710 LOCATE 6,8:PRINT "a speed of ";MP
;" M.P.H."
720 FOR X=1 TO 2000:NEXT X
730 GOTO 260
740 REM *Set up screen*
750 REM ***************
760 CLS
770 LOCATE 0,11:PRINT STRING$(29,"~")
780 LOCATE 5,10:PRINT "/ ":LOCATE 6,9
:PRINT "_"
790 FOR N=7 TO BUS+6
800 LOCATE N,10:PRINT "A"
810 NEXT N
820 LOCATE BUS+8,10:PRINT "\":LOCATE
BUS+7,9:PRINT "_"
830 REM *Put bike on left of screen*
840 REM ****************************
850 PUT SPRITE 0,(7,78),13,0
860 RETURN
870 REM *Didn't make it*
880 REM ***************
890 FOR F=1 TO 10
900 LOCATE J+2,10:PRINT " ":LOCATE J+
2,18:PRINT " "
910 NEXT F
920 FOR X=1 TO 1000:NEXT X
930 CLS
940 REM *Clear Sprite screen*
950 PUT SPRITE 0,(0,208),0,0
960 IF J<BUS+7 THEN 1040
970 REM *Too fast*
980 LOCATE 3,4:PRINT "You went too fa
st and you"
990 LOCATE 4,6:PRINT "missed the land
ing ramp."
1000 LOCATE 7,8:PRINT "Try a slower s
peed."
```

```
1010 FOR X=1 TO 1500:NEXT X
1020 GOTO 260
1030 REM *Too slow*
1040 LOCATE 1,4:PRINT "You went too s
low and your"
1050 LOCATE 5,6:PRINT "bike is a writ
e-off."
1060 LOCATE 3,8:PRINT "Try again on a
 new bike."
1070 FOR X=1 TO 1500:NEXT X
1080 GOTO 260
1090 REM *Define graphics*
1100 REM ******************
1110 R$=""
1120 FOR N=1 TO 8
1130 READ D$
1140 R$=R$+CHR$(VAL(D$+"%"))
1150 NEXT N
1160 SPRITE$(0)=R$
1170 RETURN
1180 DATA 32,96,80,110,50,82,181,66
1190 END
```

MASTERCODE

This is the game where you must guess a secret code which has been generated by the computer.

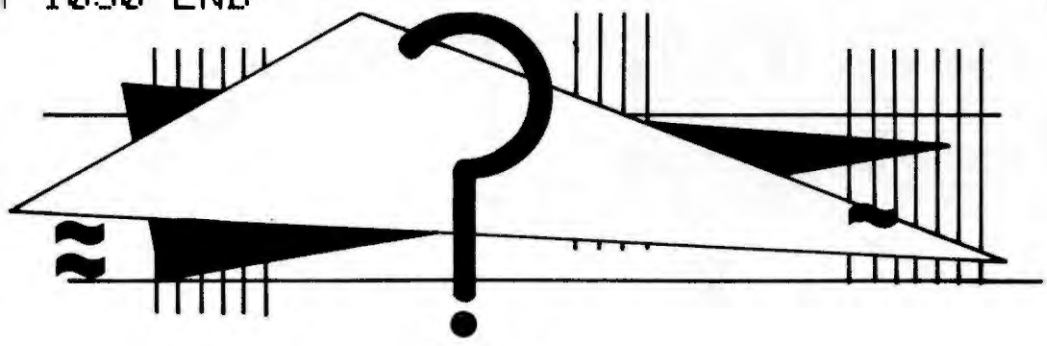
Full instructions are included in the program. There are some useful validation routines in lines 620-820 which you may wish to experiment with in the programs you write yourself.

```
10 REM **Mastercode Program**
20 REM *G. P. Carter, 1984*
30 REM *Instructions*
40 KEY OFF:SCREEN 0:WIDTH 39
50 CLS
60 PRINT TAB(15);"Mastercode"
70 PRINT TAB(15);"**********"
80 PRINT:PRINT
90 PRINT "In this game, the computer
thinks of a"
100 PRINT "four digit number which yo
u must"
110 PRINT "guess.  The code is made u
p from the"
120 PRINT "digits 0-9, and each digit
 is used"
130 PRINT "once only."
140 PRINT
150 PRINT "Enter your guess by typing
 in a four"
160 PRINT "digit number with no space
s in between"
170 PRINT "the digits; then press 'RE
TURN'."
180 PRINT"The computer will then give
 you clues"
```

```
190 PRINT "as to how close your guess
 was to the"
200 PRINT "secret code."
210 PRINT:PRINT:PRINT "Press 'RETURN'
 to continue."
220 INPUT A$
230 CLS
240 PRINT
250 PRINT "The WHITE score represents
 the number"
260 PRINT "of digits which were used
in the code"
270 PRINT "but were in the wrong posi
tions."
280 PRINT
290 PRINT "The BLACK score represents
 the number"
300 PRINT "of digits which were corre
ct and were"
310 PRINT "in the correct position."
320 PRINT
330 PRINT "At the end, the computer w
ill tell you"
340 PRINT "how many guesses you neede
d to find"
350 PRINT "the answer."
360 PRINT:PRINT:PRINT "Press 'RETURN'
 to continue."
370 INPUT A$
380 REM *Create secret code*
390 REM ********************
400 DIM A(4)
410 FOR E=1 TO 4
420 B=RND (-TIME)
430 X=RND(1):X=INT(9*X)+1
440 REM *Check digits are different*
450 REM ****************************
460 FOR J=1 TO E
470 IF X=A(J) THEN 400
```

```
480 NEXT J
490 A(E)=X
500 NEXT E
510 G=0
520 REM *G counts guesses*
530 REM *Start game*
540 REM ************
550 CLS
560 PRINT "Guess:";TAB(10);"White:";T
AB(20);"Black:"
570 PRINT "-----";TAB(10);"-----";TAB
(20);"-----"
580 LOCATE 1,21:PRINT "Enter a 4 figu
re guess."
590 LOCATE 1,22:PRINT "
                    "
600 LOCATE 1,22:INPUT B$
610 LOCATE 1,20:PRINT "
                          "
620 IF LEN(B$)=4 THEN 650
630 LOCATE 1,20: PRINT "4 digits only
.  Please enter again."
640 GOTO 590
650 REM *Check input is numeric*
660 REM ************************
670 Z=VAL(B$)
680 IF Z<>0 THEN 710
690 LOCATE 1,20:PRINT "Entry must be
0-9 only; enter again."
700 GOTO 590
710 LOCATE 1,22:PRINT "
                    "
720 REM *Are nos in guess different?*
730 REM ******************************
740 FOR S=1 TO 4
750 FOR T=S+1 TO 4
760 IF MID$(B$,S,1)=MID$(B$,T,1) THEN
 F=1
770 NEXT T
```

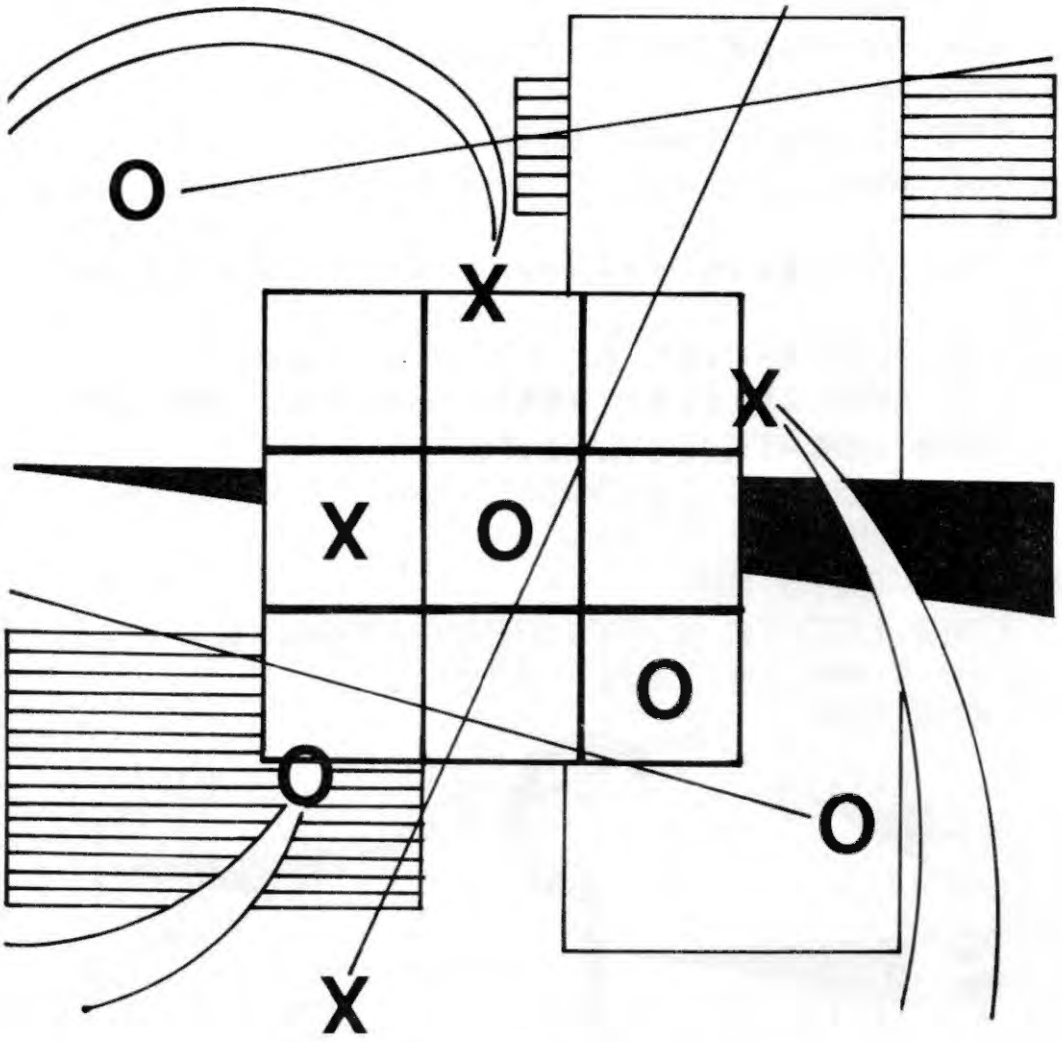
```
780 NEXT S
790 IF F=0 THEN 830
800 LOCATE 1,20:PRINT "All digits mus
t be different."
810 F=0
820 GOTO 580
830 LOCATE 1,G+2: PRINT B$
840 REM *Search for BLACK & WHITE*
850 REM ****************************
860 R=0:C=0:G=G+1
870 FOR D=1 TO 4
880 X=ASC(B$)-48
890 IF X=A(D) THEN C=C+1
900 FOR J=1 TO 4
910 IF X=A(J) THEN R=R+1
920 NEXT J
930 B$=MID$(B$,2,(LEN(B$)))
940 NEXT D
950 IF C=4 THEN 1040
960 REM *If all 4 are black go to end
*
970 REM ******************************
*
980 REM *Print black & white scores*
990 REM ****************************
1000 LOCATE 10,G+1:PRINT R-C
1010 LOCATE 20,G+1:PRINT C
1020 PRINT
1030 GOTO 580
1040 LOCATE 1,G+5:PRINT "You guessed
the code in ";G;" goes."
1050 END
```



NOUGHTS & CROSSES

This game, by Andrew Ogilvie, allows two people to play 'Noughts and Crosses' on the television screen. The computer gives a display of the board and will tell you when either player has won.

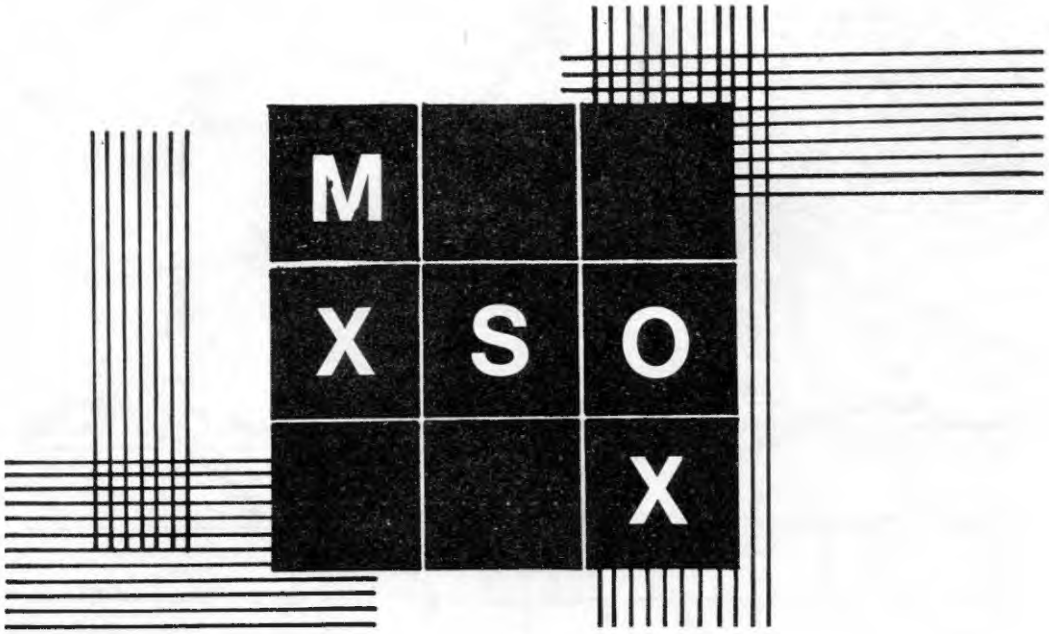
Full instructions for playing the game are included in the program.



```
10 REM **************************
20 REM *   Noughts & Crosses    *
30 REM *    Date:  07/11/84     *
40 REM *   Andrew J. Ogilvie    *
50 REM **************************
60 REM
70 REM * Dimension Arrays *
80 DIM A(9),A$(9)
90 REM * Program Start *
100 KEY OFF
110 CLS
120 COLOR 15,4,4
130 WIDTH 39
140 A=0
150 REM * Zeroise Arrays *
160 FOR I=1 TO 9
170 A(I)=0:A$(I)=" "
180 NEXT I
190 REM * Print Instructions *
200 LOCATE 16,0:PRINT "* OXO *"
210 LOCATE 2,2:PRINT "This is Noughts
 and Crosses, for two"
220 LOCATE 2,3:PRINT "players. I will
 tell you when either"
230 LOCATE 2,4:PRINT "of you has won.
 The board is numbered"
240 LOCATE 2,5:PRINT "as on the left
side of the screen."
250 LOCATE 3,7:PRINT "1¦2¦3"
260 LOCATE 3,8:PRINT "-----"
270 LOCATE 3,9:PRINT "4¦5¦6"
280 LOCATE 3,10:PRINT "-----"
290 LOCATE 3,11:PRINT "7¦8¦9"
300 LOCATE 10,11:PRINT "Just enter th
e position 1-9."
310 LOCATE 11,10:PRINT "X goes first.
"
320 CO=0
330 LOCATE 1,20:PRINT "Press <RETURN>
```

```
 to continue.";
340 INPUT A$:CLS
350 PL$="X":V1=1:W2=3
360 LOCATE 16,0:PRINT "* OXO *"
370 REM * Enter go *
380 LOCATE 2,2
390 PRINT SPC(15)
400 LOCATE 2,2
410 PRINT "Player ";PL$;" your go: ";
420 A$=INKEY$:IF A$="" THEN 420
430 A=VAL(A$)
440 IF A=0 THEN LOCATE 2,4:PRINT "Ent
er 1-9 only.":GOTO 370
450 IF A<1 OR A>9 THEN A=0:GOTO 440
460 CO=CO+1
470 REM *Check to see if place taken
*
480 IF A(A)<>0 THEN LOCATE 2,4:PRINT
"Can't go there!":GOTO 370
490 REM * Print game board *
500 A$(A)=PL$
510 A(A)=V1
520 LOCATE 28,8:PRINT A$(1);"¦";A$(2)
;"¦";A$(3)
530 LOCATE 28,9:PRINT "------"
540 LOCATE 28,10:PRINT A$(4);"¦";A$(5
);"¦";A$(6)
550 LOCATE 28,11:PRINT "------"
560 LOCATE 28,12:PRINT A$(7);"¦";A$(8
);"¦";A$(9)
570 REM * Check for win *
580 IF A(1)+A(2)+A(3)=W2 THEN 720
590 IF A(4)+A(5)+A(6)=W2 THEN 720
600 IF A(7)+A(8)+A(9)=W2 THEN 720
610 IF A(1)+A(4)+A(7)=W2 THEN 720
620 IF A(2)+A(5)+A(8)=W2 THEN 720
630 IF A(3)+A(6)+A(9)=W2 THEN 720
640 IF A(1)+A(5)+A(9)=W2 THEN 720
650 IF A(3)+A(5)+A(7)=W2 THEN 720
```

```
660 IF CO=9 THEN 800
670 REM * No winning line *
680 REM * Swap players *
690 IF V1=1 THEN V1=5:PL$="0":W2=15:G
OTO 710
700 V1=1:PL$="X":W2=3
710 GOTO 370
720 REM * Player has won *
730 LOCATE 2,14
740 PRINT "Player ";PL$;" has won!"
750 LOCATE 2,16
760 INPUT "Another go (Y) ";A$
770 IF LEFT$(A$,1)="Y" THEN 90
780 PRINT "********** Bye - Bye *
**********"
790 GOTO 840
800 REM * The game is a draw *
810 LOCATE 2,12:PRINT "The game is a
draw!"
820 PRINT
830 GOTO 750
840 END
```

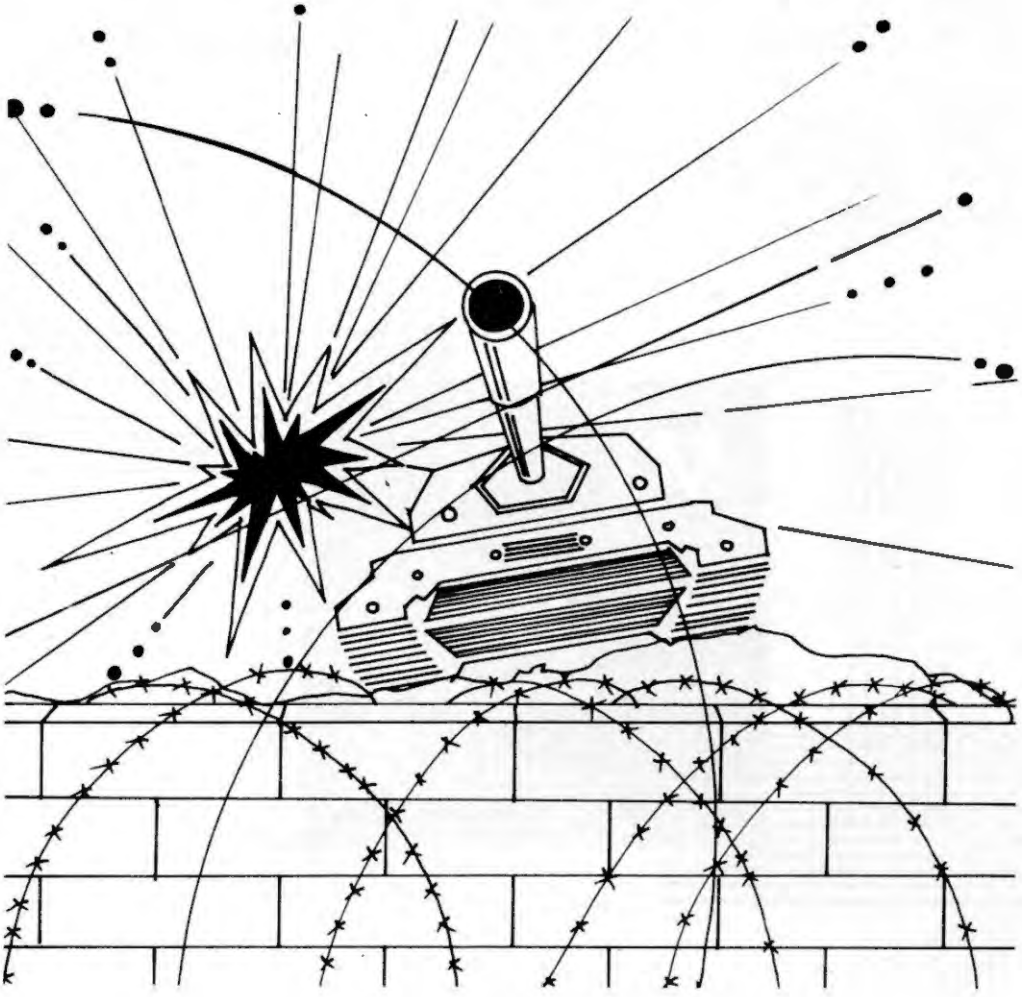


BAZOOKA!

You lie in wait behind a pile of sandbags, with your anti-tank gun loaded and ready for use. An enemy tank appears on the horizon, but there is a large wall in your line of fire. You raise the turret of your gun and fire! The shell arcs over the wall and penetrates the tank's armour, turning it into a mass of mangled metal. There is little time for praise, however, as another tank comes into view.

In this game you must judge the angle and velocity for the shell to clear the wall and hit the tank.

The shell's course is plotted according to accurate formulae as it hurtles towards the tank.



```
10 REM ************
20 REM * Bazooka! *
30 REM * ~~~~~~~~~~ *
40 REM *G. Carter,*
50 REM *S. Vincent*
60 REM *  1983/84 *
70 REM ************
80 KEY OFF
90 COLOR 2,15,15
100 CLS
110 REM *Initialise variables*
120 REM ***********************
130 B=RND (-TIME)
140 TRY=0
150 A$=STRING$(32," ")
160 WALL=RND(1):WALL=INT(18*WALL)+5
170 HGT=RND(1):HGT=INT(6*HGT)+3
180 TNK=WALL+2+INT(RND(1)*(27-WALL))
190 REM *Print tank*
200 REM ************
210 LOCATE TNK,19:PRINT "--.":LOCATE
TNK,20:PRINT "/++\":LOCATE TNK,21:PRI
NT "\OO/"
220 REM *Print wall*
230 REM ************
240 FOR A=1 TO HGT
250 LOCATE WALL,22-A:PRINT "▓"
260 NEXT A
270 REM *Print Bazooka*
280 REM ***************
290 LOCATE 1,20:PRINT "__":LOCATE 0,2
1:PRINT "■)"
300 LOCATE 3,8:PRINT "Enter angle of
projection"
310 INPUT ANG
320 IF ANG<1 OR ANG>90 THEN 310
330 LOCATE 1,20:PRINT "  "
340 LOCATE 0,8:PRINT A$
350 LOCATE 0,9:PRINT A$
360 REM *Decide on shape of barrel*
```

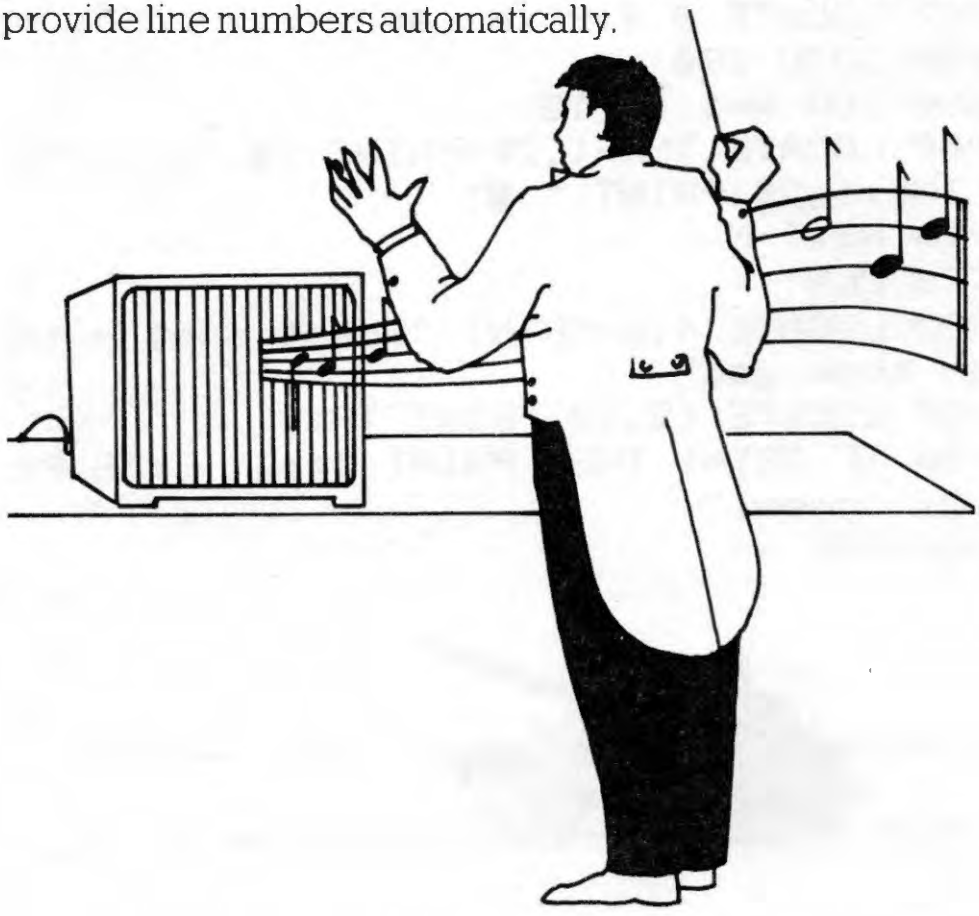
```
370 REM ****************************
380 IF ANG>10 THEN X=2
390 IF ANG>33 THEN X=3
400 IF ANG>55 THEN X=4
410 IF ANG>77 THEN X=5
420 REM *Print barrel*
430 REM ***************
440 LOCATE 1,20
450 IF X=1 THEN PRINT "▬" ELSE IF X=
2 THEN PRINT "_■" ELSE IF X=3 THEN PR
INT "▗■" ELSE PRINT "▌ "
460 LOCATE 6,8:PRINT "Enter shell vel
ocity"
470 INPUT VEL
480 IF VEL<10 THEN 460
490 TRY=TRY+1
500 VEL=VEL/4
510 LOCATE 0,8:PRINT A$
520 REM *Calculate pos. of missile*
530 REM ****************************
540 ANG=ANG*3.14159/180
550 HR=COS(ANG)*VEL
560 REV=SIN(ANG)*VEL
570 V=0:W=0:T=0
580 Y=2+(HR*T)
590 X=18-INT (REV*T-4.9*(T*T))
600 IF X<0 THEN LOCATE W,V:PRINT " "
610 IF X<0 THEN 690
620 IF Y>31 THEN 800
630 IF X>21 THEN X=21
640 REM *Print missile*
650 REM ***************
660 LOCATE Y,X:PRINT "*"
670 LOCATE W,V:PRINT " "
680 V=X:W=Y
690 T=T+.2
700 REM *Calc. where missile lands*
710 REM ****************************
720 IF X>20 THEN 750
```

```
730 IF Y>=WALL-1 AND Y<=WALL+1 AND X>
=21-HGT THEN 750
740 GOTO 580
750 LOCATE W,V:PRINT " "
760 IF X>20 THEN 790
770 LOCATE 5,8:PRINT "Oops! you hit t
he wall"
780 GOTO 820
790 IF Y>=TNK AND Y<=TNK+4 THEN 890
800 LOCATE W,V:PRINT " "
810 LOCATE 9,8:PRINT "You missed it"
820 LOCATE 4,10:PRINT "Hit any key to
 try again"
830 Z$=INKEY$
840 IF Z$="" THEN 830
850 LOCATE 1,19:PRINT "  "
860 LOCATE 0,8:PRINT A$:LOCATE 0,10:P
RINT A$
870 LOCATE 0,9:PRINT A$
880 GOTO 290
890 FOR N=1 TO 15
900 LOCATE TNK+1,20:PRINT "■ ":LOCATE
 TNK+1,20:PRINT " ■"
910 NEXT N
920 CLS
930 LOCATE 4,8:PRINT "Great shot - th
at took you"
940 LOCATE 12,10:PRINT TRY;
950 IF TRY=1 THEN PRINT "go." ELSE PR
INT "goes."
960 END
```

MUSIC PROGRAM

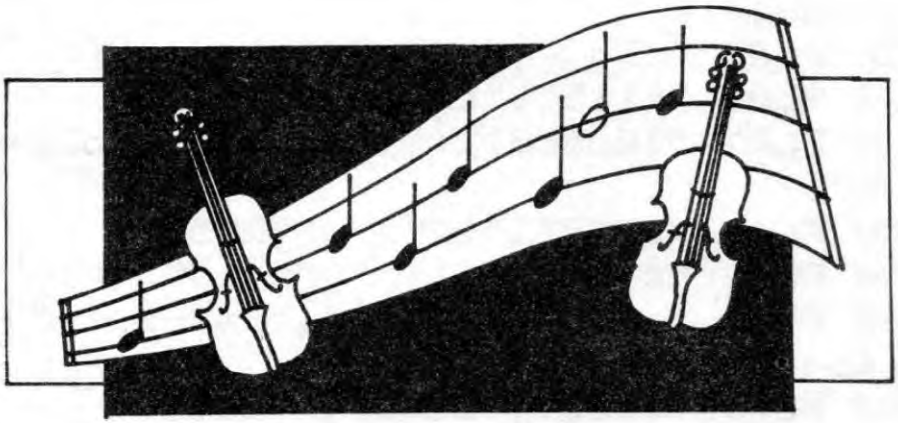
One of the best features of MSX machines is the music processor which allows you to play tunes in three-part harmony. This program demonstrates just how good the sound facilities are.

As it is very repetitive to type in this listing, it is helpful to program one of the keys at the top of the keyboard to cut down on the amount of typing when entering this program. To do this, type: KEY 1, "PLAY "+CHR$(34). Then, each time you press key F1, the start of a line of code will appear on the screen for you to complete. If you use the 'AUTO' command as well, the computer will provide line numbers automatically.



```
10 REM ♪♫♫♫♫♫♫♫♫♫♫♫♫♫♫♫♪
20 REM ♪Music Program♪
30 REM ♪~~~~~~~~~~~~~~~~~~~~♪
40 REM ♪Graham Carter♪
50 REM ♪♫♫♫♫♫♫♫♫♫♫♫♫♫♫♫♪
60 PLAY "t120","t120","t120"
70 PLAY "v10o514","v8o414","v8o214"
80 PLAY "e-o4b-g","ge-o3b-","e-12b-"
90 PLAY "f#o512d","o4do518co418b-aa-"
,"o114b-o418a-gg-f"
100 PLAY "o514","o414","o214"
110 PLAY "e-o4b-g","ge-o3b-","e-12b-"
120 PLAY "f#o512d","o4do518co418b-aa-
","o114b-o418a-gg-f"
130 PLAY "14co4b-o5d","14g12d","o214e
-b-o3g"
140 PLAY "co4b-g","14e-dd-","o2e-b-o3
e-"
150 PLAY "12e-.","o3b-ga-","o212a-."
160 PLAY "g.","o4cc#d","b-."
170 PLAY "14b-o5cd","e-12e-","14go312
b-"
180 PLAY "fe-c","14b-b-a","14o2go3go2
g-"
190 PLAY "o4a-.o518co4a-g","R4cd-","f
14o3d-e"
200 PLAY "12f.","r4cc","o2fo3de-"
210 PLAY "14b-.o518do4b-a","r4de-","o
212go314f#"
220 PLAY "12g.","gd-c","o214go3fe"
230 PLAY "t115","t115","t115"
240 PLAY "18o5cd12f","a-o4dc","o2a-o3
12f"
250 PLAY "t255","t255","t255"
260 PLAY "t115","t115","t115"
270 PLAY "o418fa-o512c","o4e-a-f#","o
214b-o3fa-"
280 PLAY "t255","t255","t255"
290 PLAY "t120","t120","t120"
```

```
300 PLAY "14","14","14"
310 PLAY "e-o4b-g","ge-o3b-","e-12b-"
320 PLAY "f#o512d","o4do518co418b-aa-
","o114b-o418a-gg-f"
330 PLAY "14e-o4b-g","14gfe-","o214e-
o3gd-"
340 PLAY "fo512c","e-18b-a-gf","o2a-o
312e-"
350 PLAY "14co4ge","14edo3b-","o214co
3ec"
360 PLAY "cb-o5c","o3go4d-e","o2b-o3g
c"
370 PLAY "o412a.","o3ao4e-d","o212f."
380 PLAY "a-.","a-e-d","o112v10b-."
390 PLAY "18ga-14b-b-","18e-d14d-d-",
"o214v8e-18b-o3e-fg"
400 PLAY "o512c.","cgf","12a-."
410 PLAY "18de-14ff","fgg","14g18o3dg
ab"
420 PLAY "12g.","o5ed-c","o414cfe"
430 PLAY "14e-co4a-","o4a-a-e-","o312
f."
440 PLAY "fo5co4d","ce-o3a-","o214b-o
3fo2f-"
450 PLAY "v1111e-","v7o418e-fgb-o514c
e-","e-b-b-e-"
460 END
```



HANGMAN

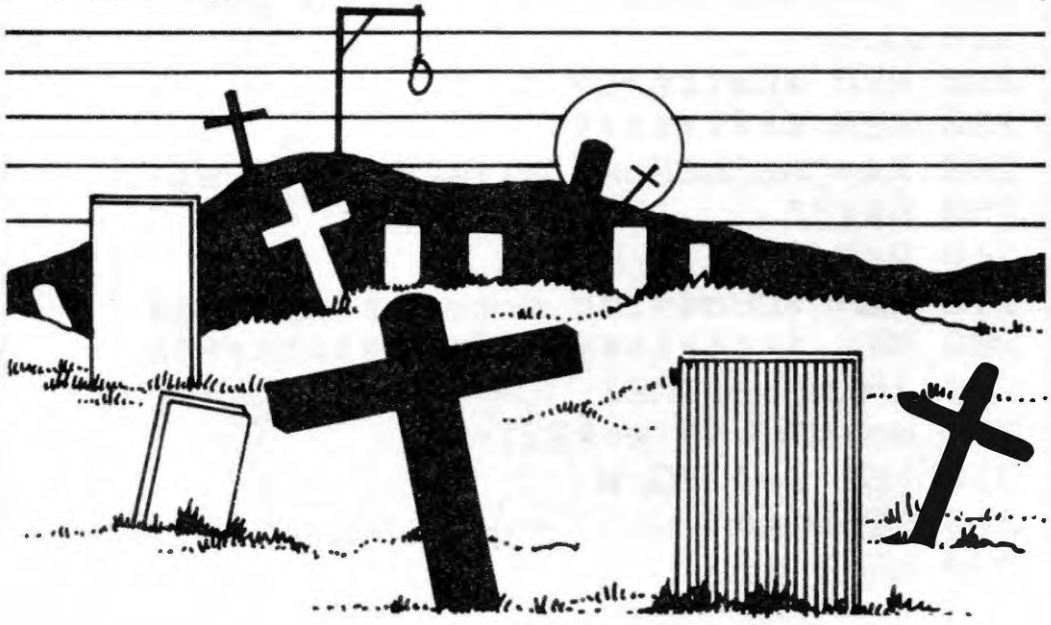
This program, written by Carol Vincent, is a version of the popular word game.

All the words are stored in DATA statements at the end of the program. You may add some words of your own by entering them as more DATA, and changing the '22' in line 300 to the number of words you intend using. Full instructions are included in the program.

This game uses string handling functions to good effect, and also demonstrates some ways of checking whether a user input is valid or not. You may like to use similar routines in your own programs.

It is very important that the words in the DATA statements consist entirely of capital letters – enter the words exactly as they appear in this listing. Also, before playing the game, please ensure that the 'CAPS LOCK' key is pressed as this program will not accept any inputs containing lower-case letters.

Please note that none of the words in this program contain the same letter twice. If you add words of your own, please ensure that they only contain a particular letter once.



```
10 REM ********************************
20 REM *           Hangman            *
30 REM *           ~~~~~~~            *
40 REM *     February 1984 by         *
50 REM Carol Vincent & Graham Carter
60 REM ********************************
70 REM *NB. Make sure CAPS LOCK is on
*
80 REM ********************************
*
90 KEY OFF
100 WIDTH 39
110 CLS
120 LOCATE 15,2:PRINT "Hangman"
130 LOCATE 15,3:PRINT "======="
140 LOCATE 0,6:PRINT "This is a game
of hangman.  All you"
150 PRINT
160 PRINT "have to do is guess the wo
rd chosen"
170 PRINT
180 PRINT "by the computer before you
 are hung!"
190 LOCATE 0,13:PRINT "Press any key
to continue."
200 L$=INKEY$:IF L$="" THEN 200
210 CLS
220 REM *Letters*
230 REM *********
240 X$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
250 R$=""
260 V=9
270 REM *Choosing word at random*
280 REM ************************
290 Z=RND(-TIME)
300 W=INT(RND(0)*22)+1
310 FOR I=1 TO W
320 READ A$
330 NEXT I
```

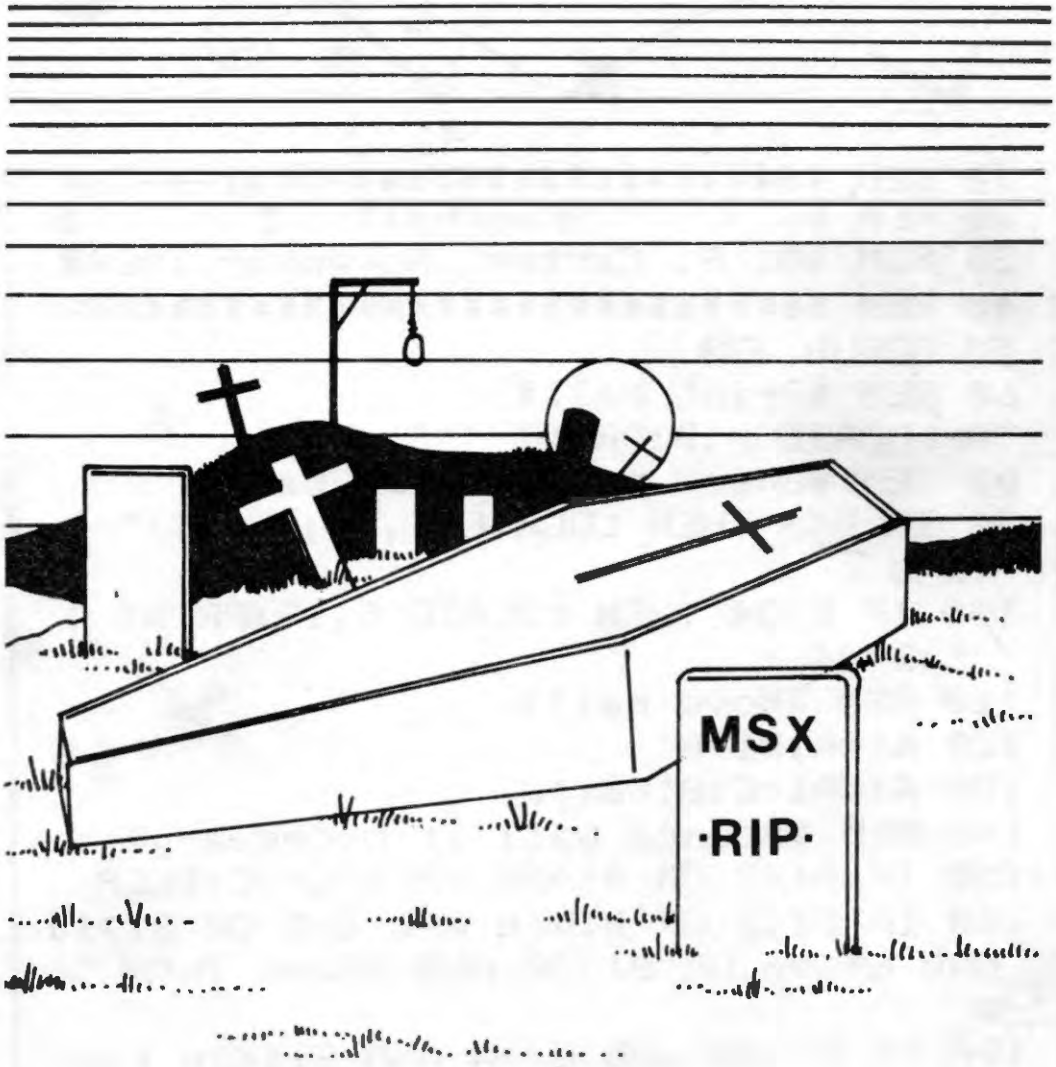
```
340 REM *Counting for hyphens*
350 REM ************************
360 FOR U=1 TO LEN(A$)
370 R$=R$+"-"
380 NEXT U
390 REM *Instructions*
400 REM ***************
410 LOCATE 7,2:PRINT "OK, you may sta
rt now!"
420 LOCATE 4,3:PRINT "Remember you ha
ve 9 guesses"
430 PRINT:PRINT TAB(2);"Here are the
let-";TAB(27);"Here is the"
440 PRINT "ters you have not";TAB(27)
;"word so far"
450 PRINT "yet used"
460 PRINT
470 PRINT X$;TAB(27);
480 REM *Printing hyphens*
490 REM ********************
500 PRINT R$
510 IF A$=R$ THEN 1540
520 REM *Entering guess*
530 REM ******************
540 PRINT:PRINT "Enter the letter ";
550 INPUT Y$
560 REM *Check for No. of letters*
570 REM ***************************
580 IF LEN(Y$)=1 THEN 650
590 PRINT
600 PRINT "One letter only"
610 PRINT "****************"
620 FOR Z=1 TO 1000:NEXT Z
630 CLS
640 GOTO 390
650 Q=ASC(Y$)
660 REM *Check input is a letter*
670 REM ***************************
680 IF Q<65 OR Q>90 THEN 1680
```

```
690 REM *Find input char. in word*
700 REM ****************************
710 F=INSTR(A$,Y$)
720 IF F=0 THEN 830
730 S=1
740 P=F
750 REM *Put char. in line of -'s*
760 REM ****************************
770 FOR K=1 TO 30
780 R$=LEFT$(R$,P-1)+Y$+MID$(R$,P+1,(
LEN(R$)))
790 P=INSTR(A$,Y$)
800 S=P-1
810 IF P=0 THEN K=31
820 NEXT K
830 F$=X$
840 X$=""
850 REM *Erase letter from alphabet*
860 REM ****************************
870 FOR I=1 TO LEN(F$)
880 IF I=INSTR(F$,Y$) THEN X$=X$+" "
ELSE X$=X$+MID$(F$,I,1)
890 NEXT I
900 IF F=0 THEN 950
910 IF A$=R$ THEN 1540
920 REM *Has player won?*
930 REM ****************
940 GOTO 390
950 V=V-1
960 PRINT "You have ";V;" chances lef
t."
970 REM *Print part of hangman*
980 REM ************************
990 R=9-V
1000 ON R GOSUB 1030,1070,1160,1220,1
260,1300,1350,1390,1440
1010 GOTO 390
1020 END
1030 REM *First part of hang*
```

```
1040 REM **********************
1050 LOCATE 10,19:PRINT "__________"
1060 RETURN
1070 REM *Second part of hangman*
1080 REM ************************
1090 LOCATE 20,19:PRINT "¦"
1100 LOCATE 20,18:PRINT "¦"
1110 LOCATE 20,17:PRINT "¦"
1120 LOCATE 20,16:PRINT "¦"
1130 LOCATE 20,15:PRINT "¦"
1140 LOCATE 20,14:PRINT "¦"
1150 RETURN
1160 REM *Third part of hangman*
1170 REM ************************
1180 LOCATE 13,13:PRINT "_______"
1190 RETURN
1200 REM *Part four*
1210 REM ***********
1220 LOCATE 13,14:PRINT "¦"
1230 RETURN
1240 REM *Part five*
1250 REM ***********
1260 LOCATE 13,15:PRINT "O"
1270 RETURN
1280 REM *Part six*
1290 REM **********
1300 LOCATE 12,15:PRINT "\"
1310 LOCATE 14,15:PRINT "/"
1320 RETURN
1330 REM *Part seven*
1340 REM ***********
1350 LOCATE 13,16:PRINT "¦"
1360 RETURN
1370 REM *Part eight*
1380 REM ***********
1390 LOCATE 12,17:PRINT "/"
1400 LOCATE 14,17:PRINT "\"
1410 RETURN
1420 REM *Last part*
```

```
1430 REM ***********
1440 CLS
1450 FOR J=1 TO 4
1460 CLS
1470 LOCATE 8,9:PRINT "**************
**********"
1480 LOCATE 8,10:PRINT "Sorry you hav
e been hung"
1490 LOCATE 8,11:PRINT "**************
**********"
1500 NEXT J
1510 PRINT:PRINT "The word was ";A$
1520 FOR Z=1 TO 1000:NEXT Z
1530 GOTO 1590
1540 CLS
1550 LOCATE 6,3:PRINT "**************
*****"
1560 LOCATE 6,4:PRINT "W E L L      D
O N E"
1570 LOCATE 6,5:PRINT "**************
*****"
1580 FOR Z=1 TO 1000:NEXT Z
1590 CLS
1600 LOCATE 6,9:PRINT "Would you like
 another go?"
1610 INPUT T$
1620 IF T$="Y" OR T$="YES" THEN 20
1630 CLS
1640 LOCATE 14,20:PRINT "OK, Thank yo
u"
1650 FOR Z=1 TO 1000:NEXT Z
1660 CLS
1670 GOTO 1020
1680 CLS
1690 LOCATE 6,11:PRINT "Please enter
letters only"
1700 LOCATE 6,12:PRINT "**************
**********"
1710 FOR P=1 TO 2000
```

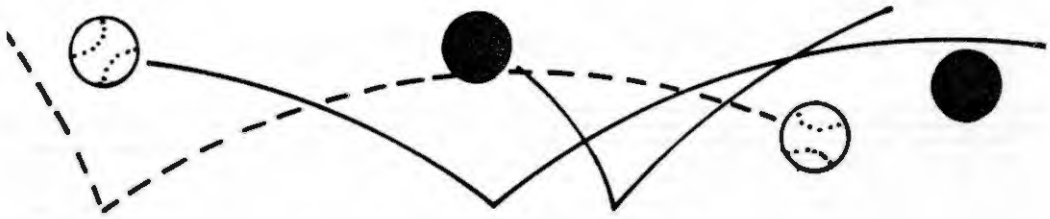
```
1720 NEXT P
1730 CLS
1740 GOTO 410
1750 DATA "PRIVATE","BREAK","HOUSE","
TABLE"
1760 DATA "TYPE","MOUTH","DIARY","COM
PUTER","PERSON","RUGBY","TODAY"
1770 DATA "PENCIL","RADIO","GARDEN","
JUMPER"
1780 DATA "HISTORY","TERMINAL","SUGAR
","NUMBER","CANOE"
1790 DATA "SOCK","MONEY","MSX"
```



BREAKBALL

This 'Bouncing Ball' game contains all the instructions in the program.

Please note that the cursor keys on the right side of the computer *must not* be held down, but should be pressed repeatedly to move the bat.



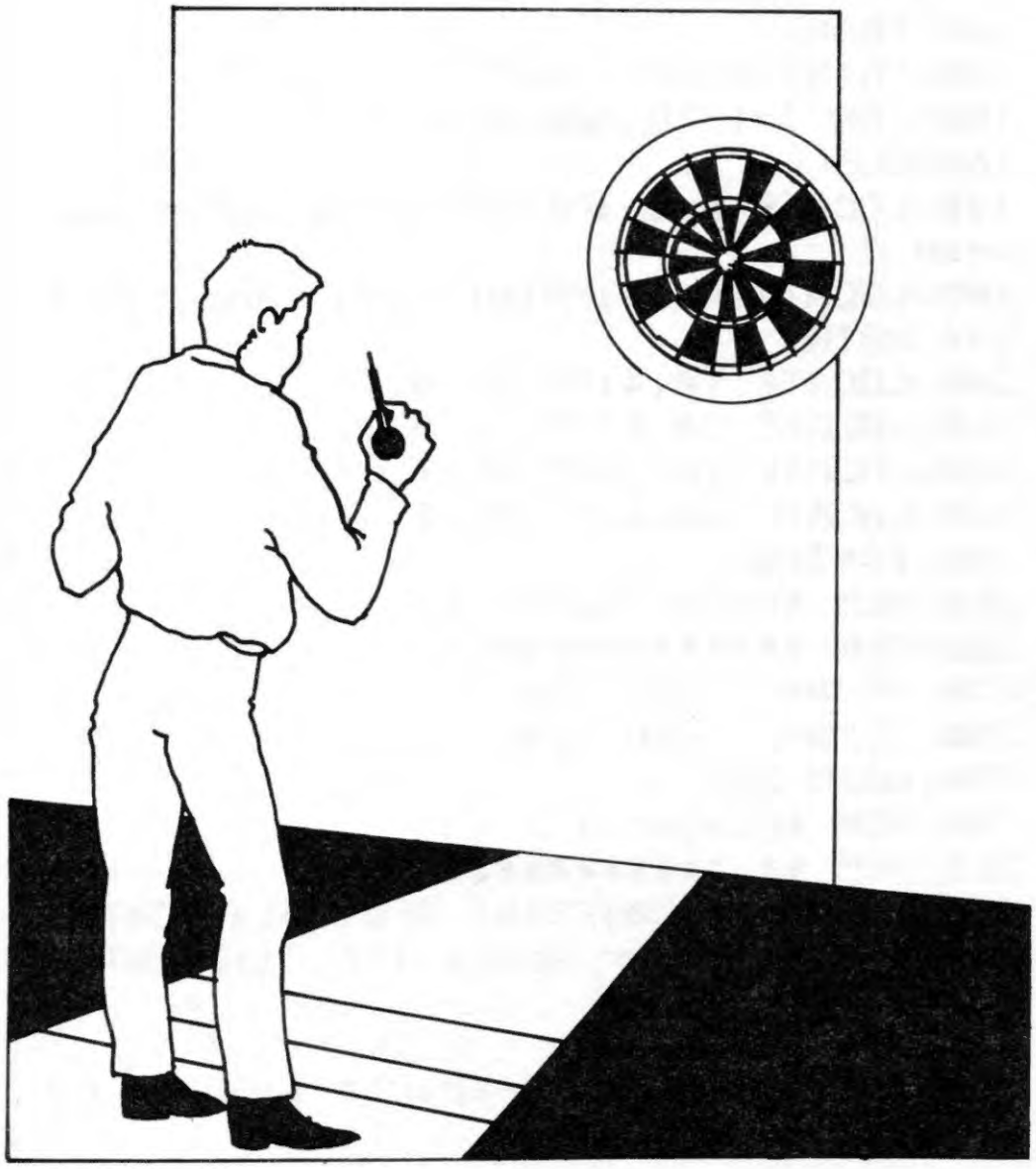
```
10 REM ********************************
20 REM *          Breakball           *
30 REM *G. P. Carter, November 1984*
40 REM ********************************
50 GOSUB 320
60 REM *Print ball*
70 LOCATE A,B:PRINT "·"
80 REM *Check position of bat*
90 IF E<3 THEN LOCATE E,19:PRINT "
":E=3
100 IF E>34 THEN LOCATE E,19:PRINT "
  ":E=34
110 REM *Move ball*
120 A1=A:B1=B
130 A1=A1+C:B1=B1+D
140 REM *Bounce ball if necessary*
150 IF A1<2 OR A1>34 THEN C=-C:BEEP
160 IF B1<2 OR B1>18 AND E=A OR B1>18
 AND E+1=A OR B1>18 AND E+2=A THEN D=
-D
170 IF B1>18 AND E<>A AND E+1<>A AND
E+2<>A THEN 270
180 F$=INKEY$
```

```
190 IF F$="" THEN 240
200 F=ASC(F$)
210 IF F=28 THEN E1=E:LOCATE E,19:PRI
NT "   ":E=E1+2:LOCATE E,19:PRINT "▬
▬":GOTO 70
220 IF F=29 THEN E1=E:LOCATE E,19:PRI
NT "   ":E=E1-2:LOCATE E,19:PRINT "▬
▬":GOTO 70
230 REM *Erase ball*
240 LOCATE A,B:PRINT " "
250 A=A1:B=B1
260 GOTO 70
270 FOR X=1 TO 1000:NEXT X
280 LOCATE A,B:PRINT " "
290 GOSUB 750
300 GOTO 70
310 STOP
320 REM *Initialisation*
330 REM ****************
340 KEY OFF:CLS:COLOR 4,14,14
350 REM *Instructions*
360 REM ***************
370 PRINT TAB(14);"Breakball"
380 PRINT TAB(14);"~~~~~~~~~~~~~~~"
390 PRINT:PRINT:PRINT
400 PRINT "In this game you must dest
roy all"
410 PRINT "the bricks at the top of t
he screen"
420 PRINT "by bouncing a ball against
 them."
430 PRINT
440 PRINT "To control the bat, use th
e left and"
450 PRINT "right cursor keys.  Do not
 hold the"
460 PRINT "keys down - you must press
 them rep-"
470 PRINT "eatedly to make the bat mo
```

```
ve."
480 PRINT:PRINT "Don't go over the ed
ge of the screen"
490 PRINT "or your bat will disappear
!"
500 PRINT:PRINT "Press any key to con
tinue."
510 F$=INKEY$
520 IF F$="" THEN 510
530 COLOR 11,1,1:CLS
540 REM *Print border around screen*
550 FOR A=0 TO 36
560 LOCATE A,0:PRINT "-"
570 LOCATE A,20:PRINT "-"
580 NEXT A
590 FOR A=0 TO 20
600 LOCATE 0,A:PRINT "|"
610 LOCATE 36,A:PRINT "|"
620 NEXT A
630 REM *Print "bricks"*
640 LOCATE 1,1
650 PRINT "++++++++++++++++++++++++++++
++++++++"
660 LOCATE 1,2
670 PRINT "++++++++++++++++++++++++++++
++++++++"
680 LOCATE 1,3
690 PRINT "++++++++++++++++++++++++++++
++++++++"
700 REM *Initialise variables*
710 REM **********************
720 C=1:D=1:E=15
730 LOCATE E,19:PRINT "▬▬▬"
740 A=RND(-TIME)
750 REM *Start position for ball*
760 REM **************************
770 A=RND(1):A=INT(A*10)+5
780 B=RND(1):B=INT(B*5)+5
790 RETURN
```

DARTS SCORER

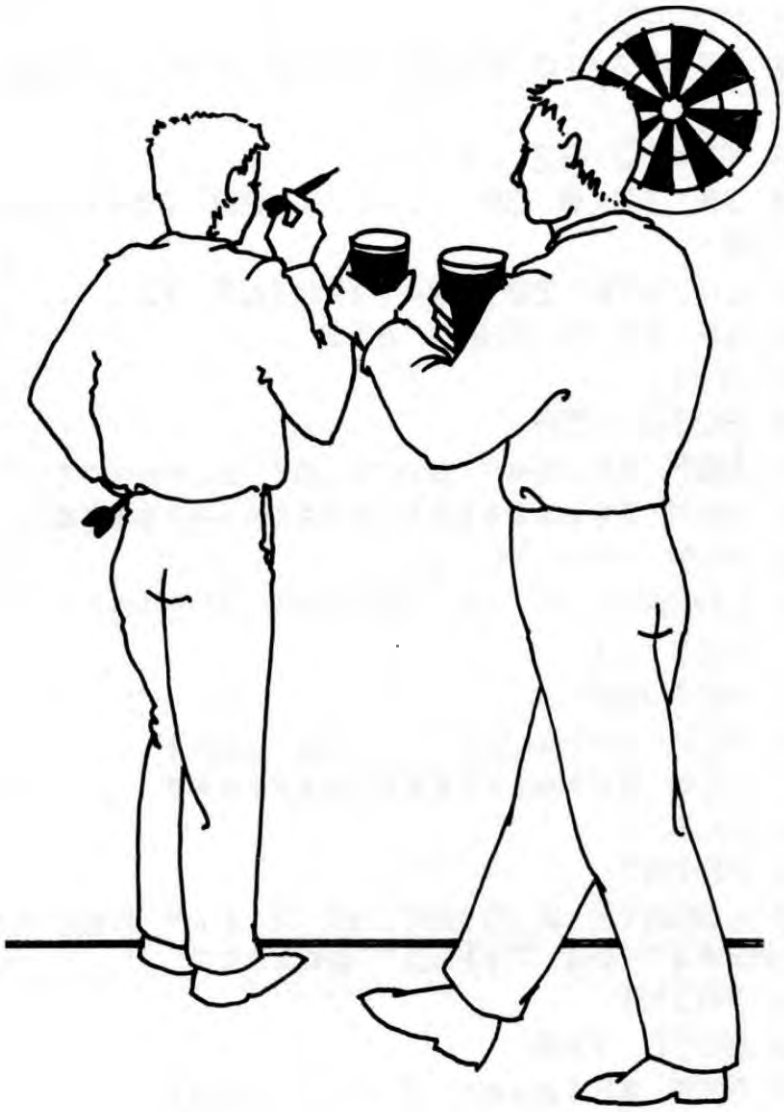
This program by Andrew Ogilvie is for darts players. It will take your mind off the scores and help you concentrate on your game. All you have to do is type in the number you are starting from, and enter your score after each go. The computer will perform the necessary calculations and tell you which player has won the game.



```
10 REM ****************
20 REM * Darts Scorer *
30 REM * ~~~~~~~~~~~~~~~~~~ *
40 REM *Andrew Ogilvie*
50 REM ****************
60 KEY OFF
70 COLOR 5,1,1
80 CLS
90 INPUT"Enter player 1's name ";N1$
100 INPUT"Enter player 2's name ";N2$
110 PRINT
120 INPUT"Enter the score to play fro
m ";T
130 PRINT
140 PRINT"Thank-you"
150 FOR I=1 TO 500:NEXT I
160 CLS
170 LOCATE 6,0 :PRINT"Darts match bet
ween "
180 LOCATE14,1 :PRINT N1$;" and ";N2$
190 PRINT
200 LOCATE 10,4:PRINT N1$
210 LOCATE 20,4:PRINT N2$
220 LOCATE 10,6:PRINT T
230 LOCATE 20,6:PRINT T
240 T2=T:G=1
250 REM *Enter Scores*
260 REM **************
270 IF G=1 THEN 300
280 IF G=2 THEN 410
290 GOTO 250
300 REM *Player 1's go*
310 REM ****************
320 LOCATE 0,20:PRINT SPC(30):LOCATE
0,20:PRINT"Enter score (1) ";:INPUT G
1
330 P1=P1+1
340 IF P1=10 THEN S=10:P1=-1:GOSUB 53
0
```

```
350 T=T-G1
360 IF T<0 OR T=1 THEN T=T+G1:GOTO 39
0
370 LOCATE 10,P1+7:PRINT T
380 IF T=0 THEN 590
390 G=2
400 GOTO 250
410 REM *Player 2's go*
420 REM ***************
430 LOCATE 0,20:PRINT SPC(30):LOCATE
0,20:PRINT"Enter score (2) ";
440 INPUT G2
450 P2=P2+1
460 IF P2=10 THEN S=20:P2=-1:GOSUB 53
0
470 T2=T2-G2
480 IF T2<0 OR T2=1 THEN T2=T2+G2:GOT
O 510
490 LOCATE 20,P2+7:PRINT T2
500 IF T2=0 THEN 660
510 G=1
520 GOTO 250
530 REM *Clear part of screen*
540 REM ***********************
550 FOR I=1 TO 12
560 LOCATE S,I+6:PRINT SPC(6)
570 NEXT I
580 RETURN
590 REM *Player 1 has won*
600 REM ******************
610 CLS
620 PRINT
630 LOCATE 0,5:PRINT N1$;" has beaten
 ";N2$;" by ";T2;" points"
640 PRINT
650 GOTO 710
660 REM *Player 2 has won*
670 REM ******************
680 CLS
```

```
690 LOCATE 0,5:PRINT N2$;" has beaten
 ";N1$;" by ";T;" points."
700 GOTO 710
710 REM *Another game*
720 REM **************
730 LOCATE 5,8:PRINT"Another game (y/
n)";:INPUT D$
740 IF D$="N" THEN 760
750 GOTO 20
760 KEY ON
770 END
```

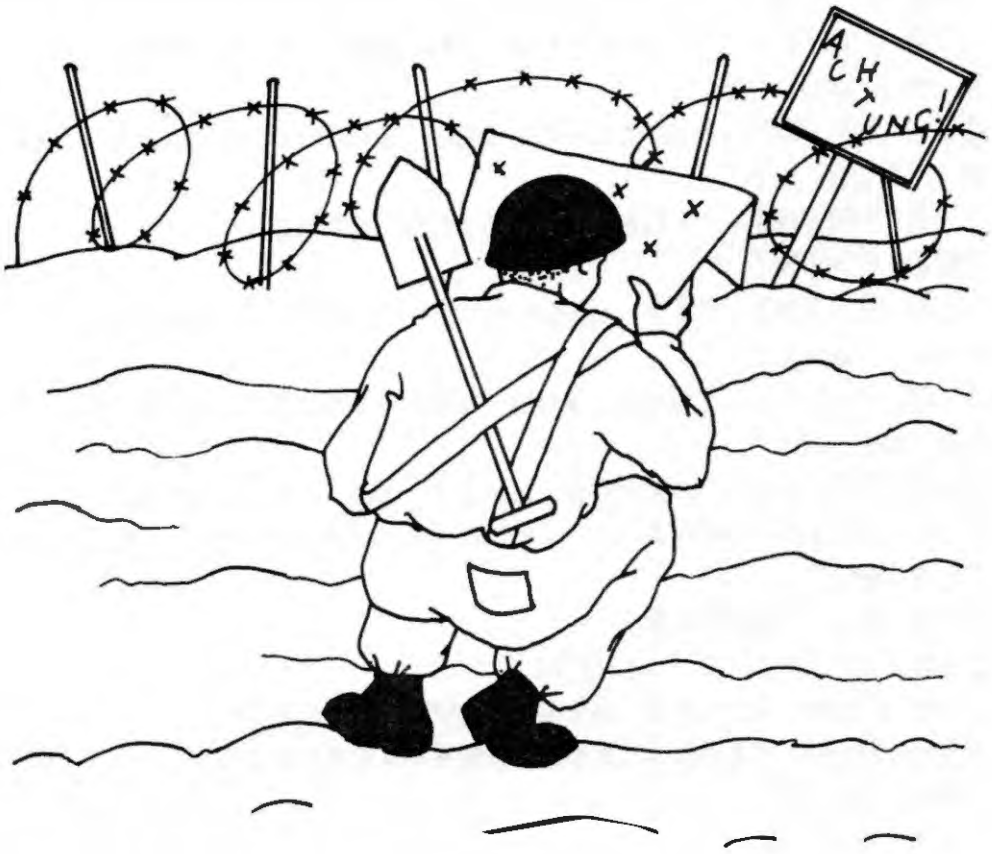


MINEFIELD

You must try to escape from the minefield in this game, written with the help of Simon Gould. Use the cursor keys to move carefully around the screen, but take note of the warning signs.

At the beginning of the game you must enter the number of mines that will be planted in the minefield. Any number less than 35 can be used.

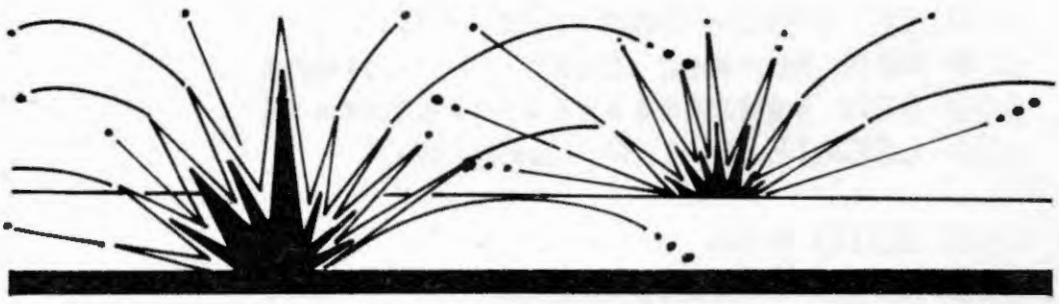
If you tread on a mine you will be blown up, and the locations of the other mines will be displayed. This game features a timer and an on-screen score.



```
10 REM ******************************
20 REM *          Minefield         *
30 REM *Simon Gould, Graham Carter*
40 REM *             1984           *
50 REM ******************************
60 REM *Instructions*
70 REM ***************
80 KEY OFF
90 CLS
100 A$=INKEY$
110 LOCATE 14,0:PRINT "Minefield"
120 LOCATE 14,1:PRINT "~~~~~~~~~~"
130 LOCATE 0,4:PRINT "You must try to
 escape from a mine-"
140 PRINT "field within a time limit
of 100"
150 PRINT "units."
160 PRINT
170 PRINT "Use the cursor control key
s to move"
180 PRINT "around the screen, but tak
e note of"
190 PRINT "the warning sign."
200 PRINT
210 PRINT "If you tread on a mine it
will expl-"
220 PRINT "ode and the locations of t
he other"
230 PRINT "mines will be revealed."
240 PRINT:PRINT "Press any key to con
tinue."
250 A$=INKEY$
260 IF A$="" THEN 250
270 REM *Initialise Variables*
280 REM ************************
290 SC=0
300 A$="o
       "
310 CLS
```

```
320 PRINT "Number of mines (2-35)"
330 INPUT MNS
340 IF MNS<2 OR MNS>35 THEN 310
350 FOR P=0 TO 500
360 NEXT P
370 DIM M(MNS,2)
380 FOR N=1 TO MNS
390 RD=RND(1):M(N,1)=INT(RD*20)+1
400 RD=RND(1):M(N,2)=INT(RD*32)
410 NEXT N
420 REM *Print Display*
430 REM ***************
440 CLS
450 LOCATE 0,21:PRINT "Timer=
          Score=";SC
460 T=0:X=19
470 Y=RND(1):Y=INT(Y*28)
480 LOCATE Y-1,X-2:PRINT A$
490 T=T+1
500 LOCATE 6,21:PRINT T
510 IF T=100 THEN 780
520 GOSUB 670
530 A=X:B=Y
540 REM *Accept input from keyboard*
550 REM ****************************
560 B$=INKEY$
570 IF B$="" THEN 480
580 IP=ASC(B$)
590 IF IP=31 AND X<23 THEN X=X+1
600 IF IP=30 AND X>3 THEN X=X-1
610 IF IP=28 AND Y<30 THEN Y=Y+1
620 IF IP=29 AND Y>0 THEN Y=Y-1
630 REM *Erase last position*
640 REM ********************
650 LOCATE B-1,A-2:PRINT"
                        "
660 GOTO 480
670 LOCATE 11,0:PRINT "    EXIT"
680 IF X=3 AND Y>11 AND Y<20 THEN 870
```

```
690 N=1
700 IF X>=20 THEN X=X-1:LOCATE Y-1,X+
1:PRINT "   "
710 YC=M(N,1)-Y
720 XC=M(N,2)-X
730 IF ABS (XC)<3 AND ABS (YC)<3 THEN
 LOCATE 11,0:PRINT "WARNING"
740 IF XC=0 AND YC=0 THEN 780
750 N=N+1
760 IF N=MNS THEN RETURN
770 GOTO 710
780 LOCATE Y-1,X-2:PRINT "   ":LOCATE
 Y,X-2:PRINT "   ":LOCATE Y+1,X:PRINT
 "  "
790 REM *Display positions of mines*
800 REM ****************************
810 FOR N=1 TO MNS
820 LOCATE M(N,1),M(N,2):PRINT "*";
830 NEXT N
840 FOR P=1 TO 1000
850 NEXT P
860 RUN
870 SC=SC+100
880 LOCATE 5,10:PRINT "Mission Succes
sful"
890 LOCATE 5,12:PRINT "Stand by for t
he next field"
900 FOR P=1 TO 1000:NEXT P
910 GOTO 440
```



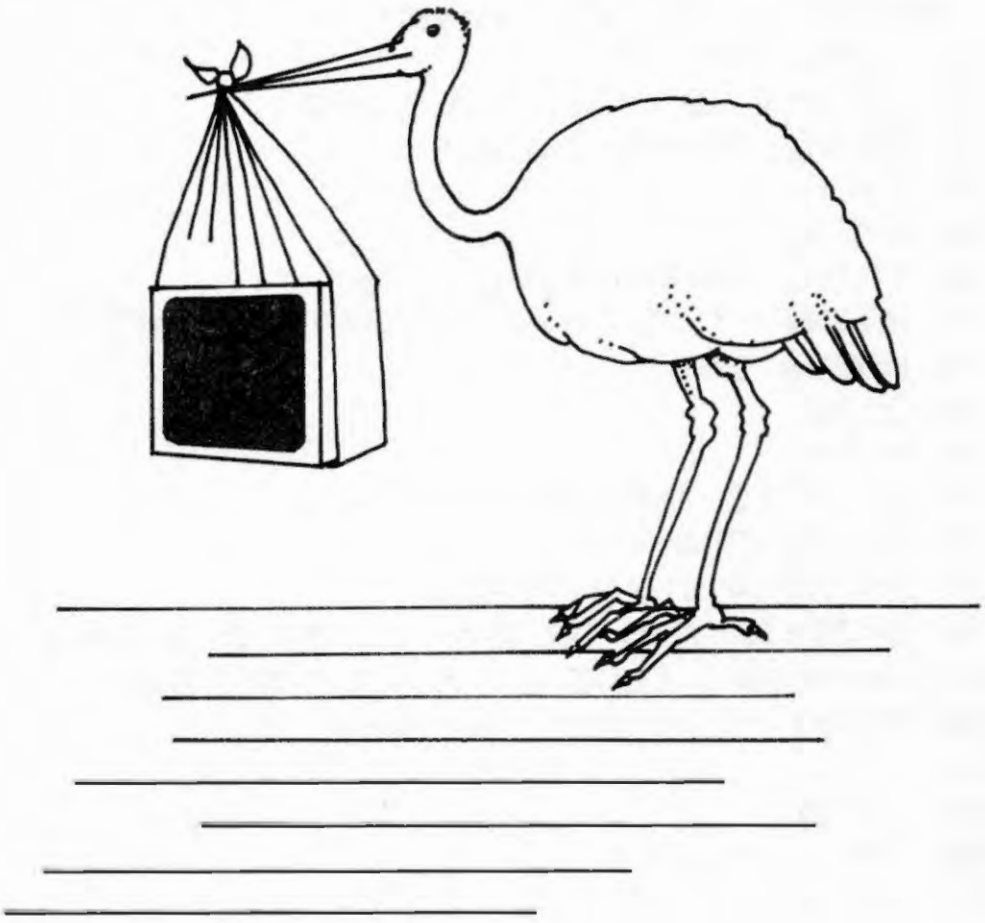
CALENDAR

Do you know the actual day of the week when you were born? If not, this program will tell you.

It prints a calendar for any month in any year, and will tell you when your birthday, or any other anniversary, will fall in the years to come.

```
10 REM ***************
20 REM *  Calendar   *
30 REM *~~~~~~~~~~~~~~~~~*
40 REM *D. Pillinger*
50 REM *D. Olesh     *
60 REM ***************
70 CLS
80 PRINT "Calendar",,"--------"
90 PRINT ,,"Enter month"
100 INPUT M
110 IF M<1 OR M>12 THEN 100
120 PRINT "Enter year"
130 INPUT Y
140 PRINT
150 PRINT "Month";M;". Year";Y
160 PRINT
170 GOSUB 360
180 D=DD
190 M=M+1
200 IF M>12 THEN M=1:Y=Y+1
210 GOSUB 350
220 IF DD<D THEN DD=DD+7
230 N=29:IF M=5 OR M=7 OR M=10 OR M=1
2 THEN N=N+1 ELSE IF M<>3 THEN N=N+2
240 PRINT "Sun Mon Tue Wed Thu Fri Sa
t"
250 P=4*D
260 FOR X=1 TO N
270 PRINT TAB(P);X;
```

```
280 P=(P+4) AND P<21
290 IF POS(3)>26 THEN PRINT
300 NEXT X
310 PRINT ,,, "Press ENTER to go on"
320 INPUT A
330 GOTO 70
340 MM=M-2:YY=Y
350 IF MM>0 THEN 400
360 MM=MM+12
370 YY=YY-1
380 C=INT(Y/100)
390 YY=YY-100*C
400 DD=1+INT(2.6*MM-.19)+YY+INT(YY/4)
+INT(C/4)-2*C
410 DD=DD-7*INT(DD/7)
420 RETURN
```



21

This is an MSX version of the classic card game. You and the computer take turns in trying to get a score as close to 21 as possible without exceeding that figure. This version of the game will recognise a five-card trick.

The computer is a difficult, but not unbeatable, opponent who weighs up the situation carefully before making a decision. This game is very friendly to play, with all the instructions contained in the program.

```
10 REM ********
20 REM *  21   *
30 REM * ♠♥♦♣ *
40 REM ********
50 REM *Scott Vincent Graham Carter*
60 REM ******************************
70 KEY OFF
80 CLS
90 COLOR 9,15,15
100 REM *Start game*
110 REM *************
120 P$="♠♥♣♥♠♥♣♥♠♥♣♥♠♥♣♥♠♥♣♥♠♥♣♥♠♥♣♥♠♥♣♥♠
♥♣♥♠♥♣♥"
130 PRINT P$
140 B=0
150 C=0
160 PRINT "Please enter your name"
170 INPUT N$
180 COLOR 7,1,1
190 CLS
200 PRINT:PRINT:PRINT N$
210 REM *Decide who goes first*
220 REM ************************
230 PRINT:PRINT "  I will toss a coin
 to see who goes first."
```

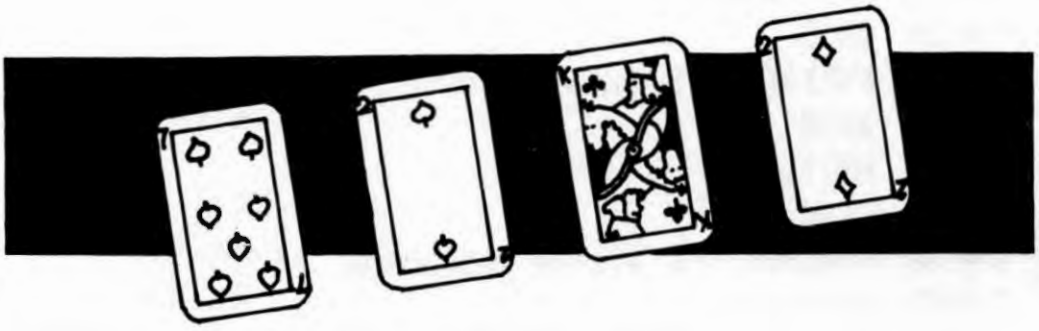
```
240 PRINT " Do you want heads or tail
s?   (H/T)"
250 A$=INKEY$
260 IF NOT (A$="h" OR A$="t" OR A$="H
" OR A$="T") THEN 250
270 PRINT TAB(10);A$
280 D=RND(-TIME)
290 X=RND(1):X=INT(X*2)
300 IF X=0 THEN B$="Heads" ELSE B$="T
ails"
310 PRINT B$;", ";
320 A=1
330 IF X=1 AND (A$="h" OR A$="H") OR
X=0 AND (A$="t" OR A$="T") THEN A=0
340 IF A=0 THEN PRINT "I win" ELSE PR
INT "You win ";N$;"."
350 FOR P=1 TO 1000:NEXT P
360 X=0
370 Y=0
380 REM *10 games*
390 REM **********
400 FOR F=1 TO 10
410 PRINT:PRINT "Press any key for ga
me ";F
420 I$=INKEY$
430 IF I$="" THEN 420
440 IF A=0 THEN GOSUB 1070
450 IF A=1 OR A=0 AND X>0 THEN GOSUB
650
460 IF A=1 AND Y<>0 THEN GOSUB 1070
470 IF Y=0 THEN X=1
480 IF X=0 THEN Y=1
490 IF X>Y THEN B=B+1
500 IF X<Y THEN C=C+1
510 PRINT:PRINT "Computer: ";N$;":"
520 PRINT TAB(5);B;TAB(11);C
530 A=A+(A=1)-(A=0)
540 NEXT F
550 FOR P=1 TO 500:NEXT P
```

```
560 REM *End of game*
570 REM ************
580 PRINT:PRINT "I enjoyed that."
590 IF B>C THEN PRINT "Maybe you will
 beat me next time." ELSE PRINT "I wi
ll beat you next time."
600 PRINT:PRINT "Another game?   (Y/N
)"
610 INPUT A$
620 IF A$="y" OR A$="Y" THEN RUN
630 PRINT:PRINT "Bye then."
640 STOP
650 PRINT:PRINT "Your cards have been
 dealt.  Press   any key to turn them
 over."
660 I$=INKEY$
670 IF I$="" THEN 660
680 GOSUB 1540
690 FOR P=1 TO 100:NEXT P:CLS
700 IF Q=1 THEN GOSUB 1590
710 PRINT:PRINT "    ";Q:PRINT
720 Y=Q
730 L=1
740 FOR P=1 TO 100:NEXT P:GOSUB 1540
750 IF Q=1 THEN GOSUB 1590
760 IF L=3 THEN CLS
770 PRINT:PRINT:PRINT "    ";Q
780 LET Y=Y+Q
790 L=L+1
800 PRINT TAB(10);L;" cards.  Total="
;Y
810 REM *5 card trick, bust or ok?*
820 REM ***************************
830 IF L=5 AND Y<=21 THEN 940
840 IF Y>21 THEN 980
850 IF Y=21 THEN 1020
860 PRINT TAB(10);"Twist or Stick?  (
T/S)"
870 A$=INKEY$
```

```
880 IF NOT (A$="t" OR A$="T" OR A$="s
" OR A$="S") THEN 870
890 PRINT TAB(10);A$
900 FOR P=1 TO 100:NEXT P
910 IF A$="t" OR A$="T" THEN 740
920 CLS
930 RETURN
940 FOR P=1 TO 100:NEXT P:CLS
950 PRINT "You have a five-card trick
.    This  can only be beaten by Pont
oon."
960 Y=22
970 RETURN
980 FOR P=1 TO 1000:NEXT P:CLS
990 PRINT "Sorry - you have bust.  I
win."
1000 Y=0
1010 RETURN
1020 CLS
1030 PRINT "Well done - you have scor
ed 21."
1040 Y=21
1050 IF L=2 THEN Y=23
1060 RETURN
1070 PRINT:PRINT "Press any key to de
al me two cards."
1080 I$=INKEY$:IF I$="" THEN 1080
1090 GOSUB 1540
1100 CLS
1110 IF Q=1 THEN GOSUB 1670
1120 PRINT TAB(10)"    ";Q
1130 X=Q
1140 M=1
1150 GOSUB 1540
1160 IF Q=1 THEN GOSUB 1670:FOR P=1 T
O 100:NEXT P
1170 IF M=3 THEN CLS
1180 PRINT TAB(10);"    ";Q
1190 X=X+Q
```

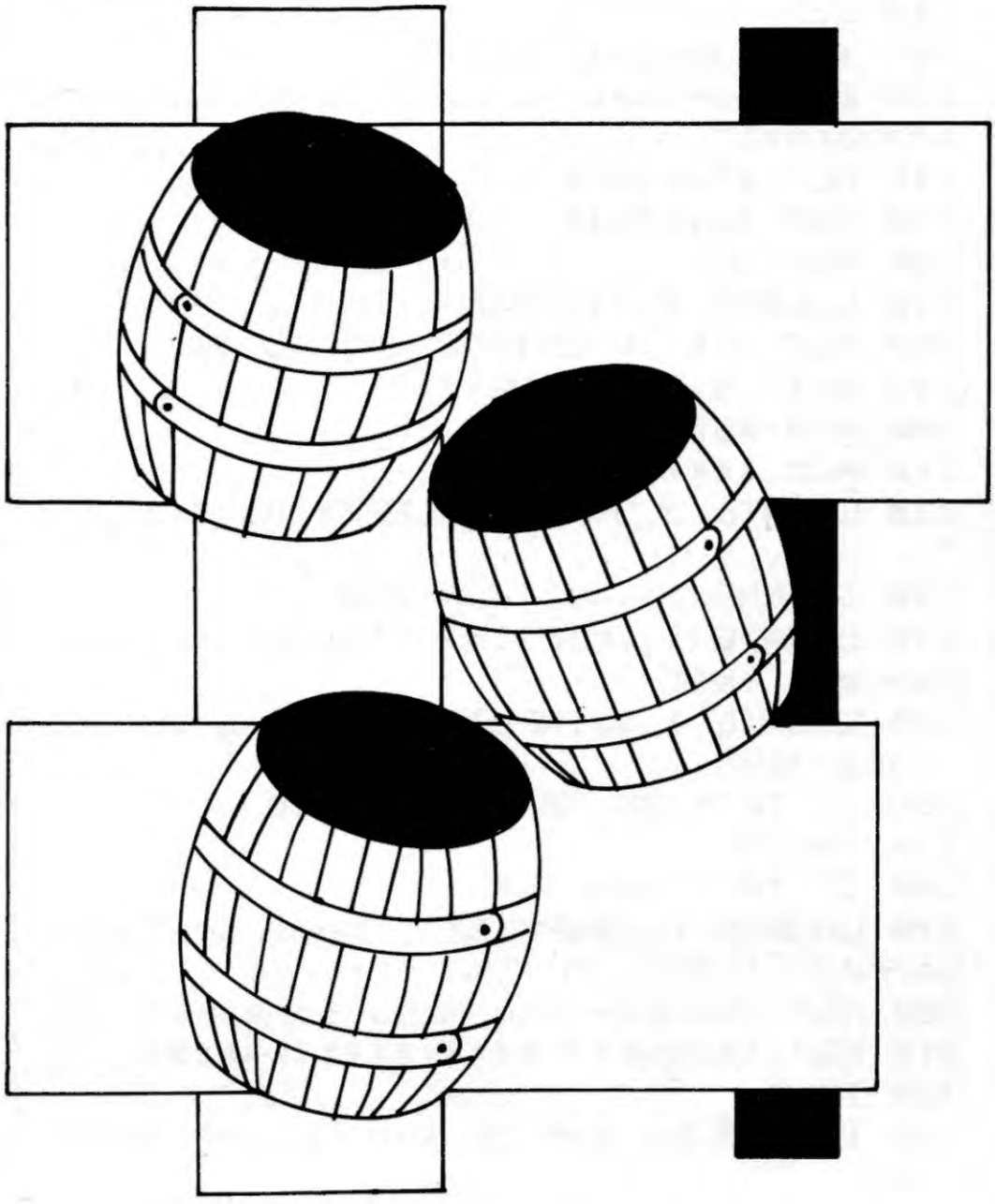
```
1200 M=M+1
1210 PRINT TAB(10);M;" cards.  Total=
";X
1220 PRINT "Press any key."
1230 I$=INKEY$:IF I$="" THEN 1230
1240 REM *5 card trick, bust or ok?*
1250 REM ****************************
1260 IF M=5 AND X<=21 THEN 1350
1270 IF X>21 THEN 1420
1280 IF X=21 THEN 1460
1290 D=RND(-TIME)
1300 IF X>14+INT(RND(0)*3) AND A=0 TH
EN 1510
1310 D=RND(-TIME)
1320 IF X>14+INT(RND(0)*3) AND RND(0)
>.3 THEN 1510
1330 FOR P=1 TO 100:NEXT P:PRINT:PRIN
T "I think I will twist."
1340 GOTO 1150
1350 CLS
1360 PRINT "I have a five-card trick.
"
1370 IF A=0 THEN PRINT "You must get
pontoon to beat me."
1380 IF A=1 AND Y=22 THEN PRINT "You
still beat me though, as you got Pont
oon."
1390 IF A=1 AND Y<22 THEN PRINT "That
 beats you.  I win."
1400 X=22
1410 RETURN
1420 CLS
1430 PRINT "I have bust.  You win."
1440 X=0
1450 RETURN
1460 CLS
1470 PRINT "I have scored 21"
1480 X=21
1490 IF M=2 THEN X=23
```

```
1500 RETURN
1510 CLS
1520 PRINT "I will stick."
1530 RETURN
1540 D=RND(-TIME)
1550 Q=INT(RND(0)*13)+1
1560 IF Q>10 THEN Q=10
1570 FOR P=1 TO 1000:NEXT P
1580 RETURN
1590 IF Y>10 THEN 1720
1600 PRINT:PRINT "An ace.  Do you wan
t it to be high or low?  (H/L)"
1610 A$=INKEY$
1620 IF NOT (A$="h" OR A$="H" OR A$="
l" OR A$="L") THEN 1610
1630 PRINT:PRINT A$
1640 IF A$="h" OR A$="H" THEN Q=11
1650 FOR P=1 TO 1000:NEXT P
1660 RETURN
1670 IF X>=7 OR X<=10 THEN Q=11
1680 IF X<7 OR X>10 THEN Q=1
1690 PRINT "I have an ace."
1700 IF Q=1 THEN PRINT "I will make i
t low." ELSE PRINT "I will make it hi
gh."
1710 RETURN
1720 PRINT "An ace.  It had better be
 low or you will bust."
1730 RETURN
```



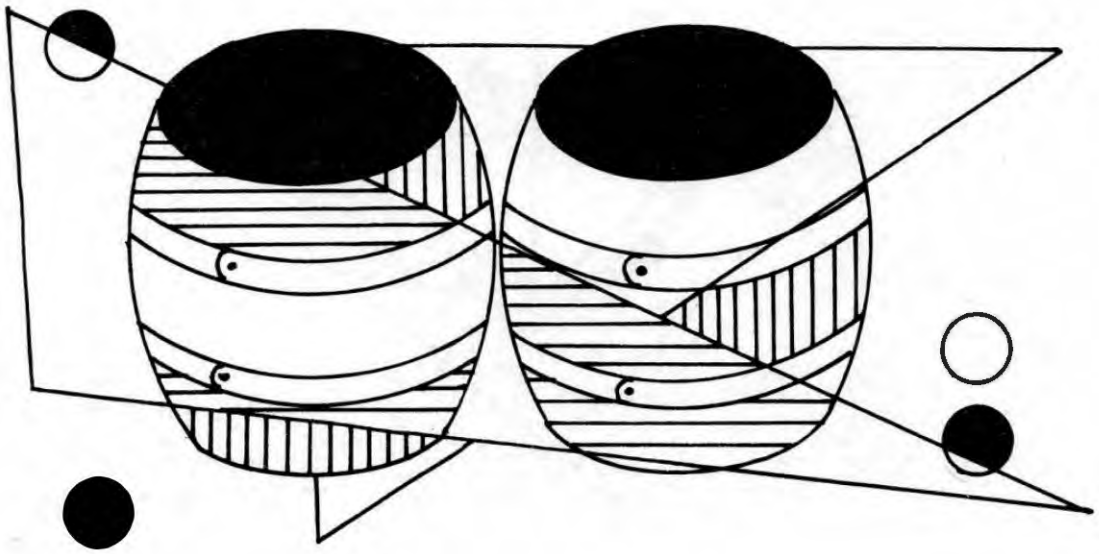
NIM

This is the classic game in which you try to force the computer to take the last of a number of barrels. The number of barrels you start off with is printed at the top of the screen. The number of barrels will change from one game to the next.



```
10 REM ****************
20 REM *     Nim       *
30 REM *     ~~~       *
40 REM *D. Pillinger,*
50 REM * D. Olesh &   *
60 REM * G. Carter    *
70 REM ****************
80 REM *Initialise Variables*
90 REM ************************
100 B=RND(-TIME)
110 CLS
120 X=INT(RND(1)*25)+5
130 B$="nnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
nnnnnnnnn"
140 REM *Titles*
150 REM ********
160 KEY OFF
170 LOCATE 0,21:PRINT "Nim",,"---"
180 FOR T=1 TO 20:FOR W=1 TO 10
190 NEXT W:PRINT:NEXT T
200 REM *Start game*
210 REM ************
220 LOCATE 3,3:PRINT LEFT$(B$,X);"
"
230 IF RND(1)<.5 THEN 320
240 LOCATE 1,10:PRINT "There are";X;"
barrels left       "
250 LOCATE 1,5:INPUT "How many do you
 take";TK
260 IF TK<1 OR TK>3 THEN 250
270 X=X-TK
280 IF X=1 THEN 440
290 LOCATE 1,10:PRINT "There are";X;"
barrels left    "
300 REM *Decide how many to take*
310 REM ************************
320 TK=3
330 IF X=2 OR X=5 OR X=6 OR X=9 THEN
TK=1
```

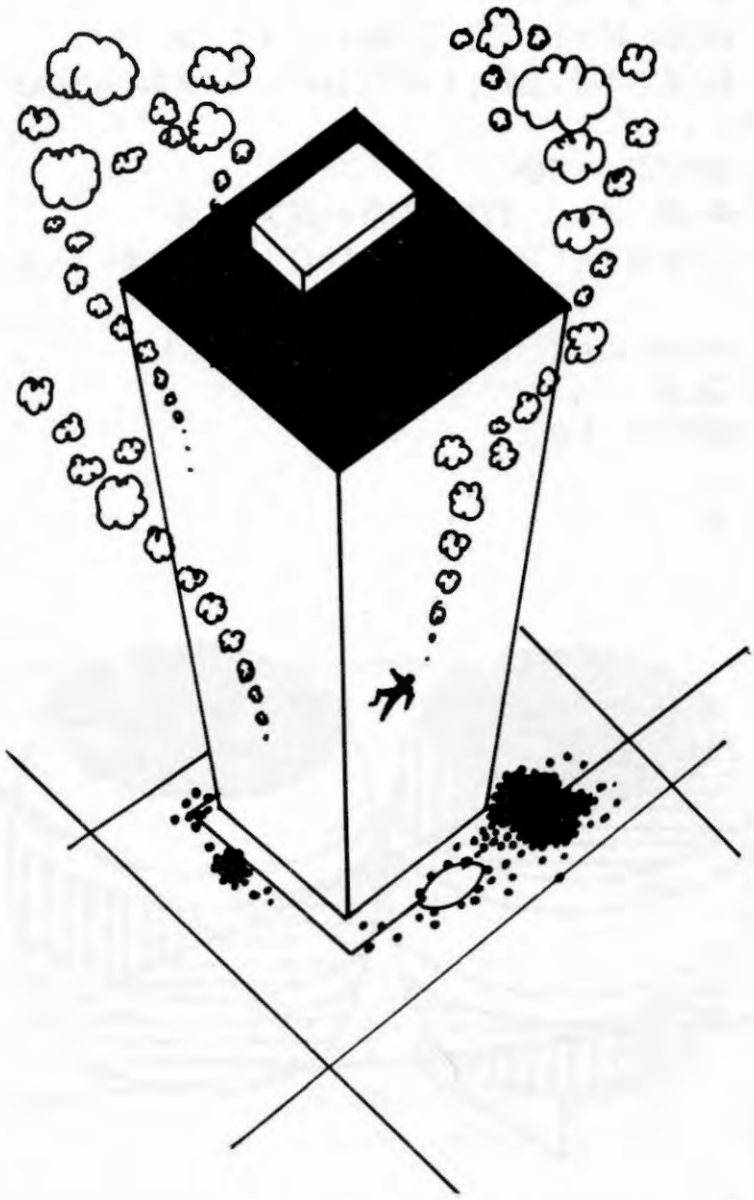
```
340 IF X=3 OR X=7 THEN TK=2
350 IF X=4 THEN TK=3
360 LOCATE 1,8:PRINT "I will take";TK
370 X=X-TK
380 LOCATE 1,10:PRINT "There";
390 IF X=1 THEN PRINT " is";X;ELSE PR
INT " are";X;
400 IF X=1 THEN PRINT " barrel";ELSE
PRINT " barrels";
410 PRINT " left "
420 IF X=1 THEN 470
430 GOTO 250
440 FOR W=1 TO 700:NEXT W
450 LOCATE 20,1:PRINT "**** You Win *
***"
460 GOTO 490
470 FOR W=1 TO 700:NEXT W
480 LOCATE 20,1:PRINT "**** I Win ***
*"
490 FOR L=1 TO 1000:NEXT L
500 CLS
510 GOTO 120
```



INFERNO

A 30-storey skyscraper is on fire and you, as Fire Chief, must save the 10 people who are still trapped in the burning building.

As they jump from the windows, you must catch them in your blanket. To move the blanket, press the cursor controls on the right side of the computer.



```
10 REM ************************
20 REM *        Inferno       *
30 REM *S. Vincent & G. Carter*
40 REM *          1984        *
50 REM ************************
60 REM *Initialise variables*
70 REM ********************
80 P=0:S=P
90 F=1:G=9
100 E=10:CLS
110 REM *Print screen*
120 REM ***************
130 KEY OFF
140 COLOR 10,1,1
150 LOCATE 0,21:PRINT STRING$(36,"~")
160 FOR X=3 TO 20
170 LOCATE 1,X:PRINT "┌┬┬┬┬┐"
180 NEXT X
190 LOCATE E-1,20:PRINT "\_/"
200 FOR T=1 TO 20:NEXT T
210 LOCATE G,F:PRINT "♀"
220 LOCATE E-1,20:PRINT "   "
230 REM *Check for key pressed*
240 REM ************************
250 I$=INKEY$:IF I$="" THEN 260 ELSE
IF ASC(I$)=28 THEN E=E+1.5
260 IF E>31 THEN E=31
270 I$=INKEY$:IF I$="" THEN 280 ELSE
IF ASC(I$)=29 THEN E=E-1.5
280 IF E<10 THEN E=10
290 INTERVAL ON
300 ON INTERVAL=1 GOSUB 850
310 LOCATE G,F:PRINT " "
320 F=F+1
330 IF F>18 THEN 370
340 H=RND(-TIME)
350 G=G+INT(RND(0)*4)-1
360 IF G<10 THEN G=10
370 IF F=21 THEN 390
380 GOTO 190
```

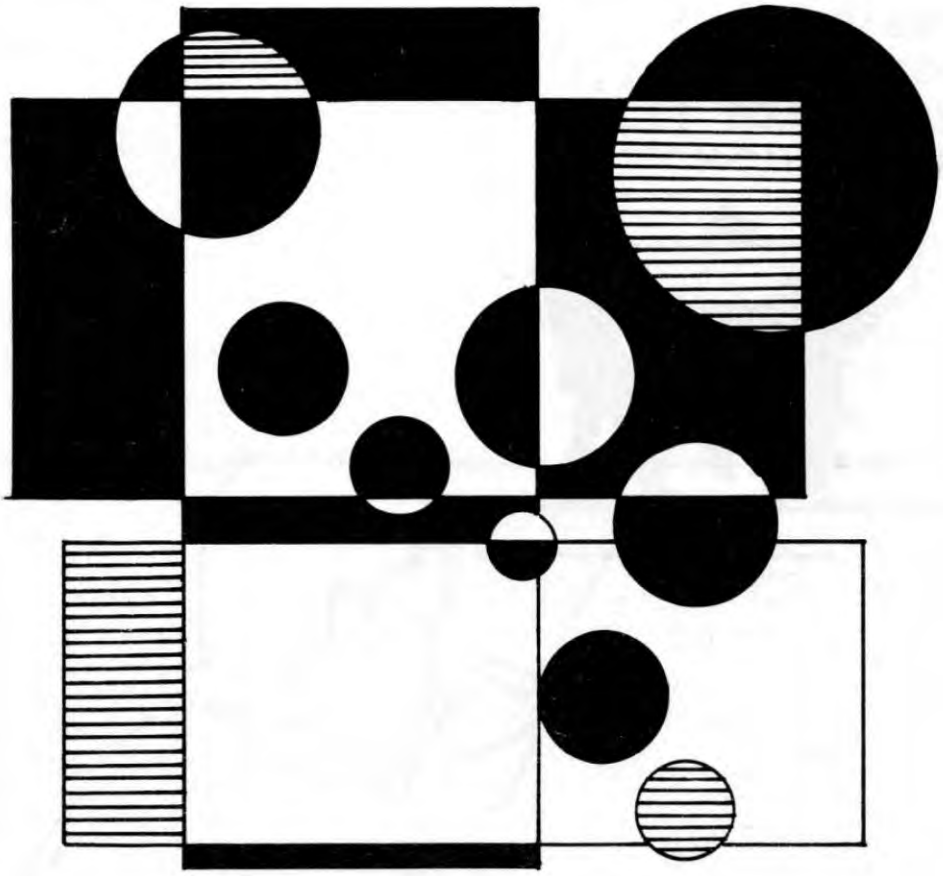
```
390 P=P+1
400 IF G=E OR G=E-1 THEN 530
410 REM *Dead!*
420 REM *******
430 LOCATE G,21:PRINT "☻"
440 FOR R=1 TO 90:LOCATE 16,17
450 PRINT "Splat!"
460 LOCATE 16,17:PRINT SPC(6):NEXT R
470 LOCATE 12,9:PRINT "Score =";S;"ou
t of";P
480 FOR T=1 TO 1000:NEXT T
490 IF P=10 THEN 600
500 CLS:GOTO 90
510 REM *Caught*
520 REM ********
530 S=S+1
540 FOR Q=1 TO 50
550 LOCATE 14,7:PRINT "Well caught!"
560 NEXT Q
570 GOTO 470
580 REM *End of game*
590 REM ************
600 CLS
610 FOR X=1 TO 21
620 PRINT STRING$(36,"*")
630 NEXT X
640 FOR K=7 TO 13
650 LOCATE 4,K:PRINT SPC(28)
660 NEXT K
670 LOCATE 5,8:PRINT "Your final scor
e is ";S
680 LOCATE 5,10:PRINT "You will be de
moted unless"
690 LOCATE 5,11:PRINT "you do better
next time."
700 GOTO 750
710 FOR K=1 TO 60
720 LOCATE 0,10:PRINT "    Well done!
":LOCATE 0,10:PRINT SPC(15)
```

```
730 GOSUB 790
740 NEXT K
750 FOR T=1 TO 2500:NEXT T
760 RUN
770 REM *On top of old Smokey*
780 REM **********************
790 PLAY "t160"
800 PLAY "o414ct255164rt16014cego512c
.o4a."
810 PLAY "12r14ft255164rt16014"
820 PLAY "fga12g.g.g"
830 PLAY "14ct255164rt16014ceg12g.d.1
4ddefed12c.c.c"
840 RETURN
850 COLOR 9,1,1
860 COLOR 11,1,1
870 RETURN
```

ALLSORTS

This program is an example of a computer sort known as the Bubble Sort, because the lighter elements of the array bubble up to the top.

This version was written by Andrew Ogilvie, and it can be amended in a number of ways. To sort a list in descending order, change the '<' in line 260 to a '>'. If you want to sort a list of numbers instead of strings, just delete all the dollar signs ($) in the program listing.



```
10 REM ******************
20 REM *Sorting Program*
30 REM *                *
40 REM *Andrew Ogilvie  *
50 REM ******************
```

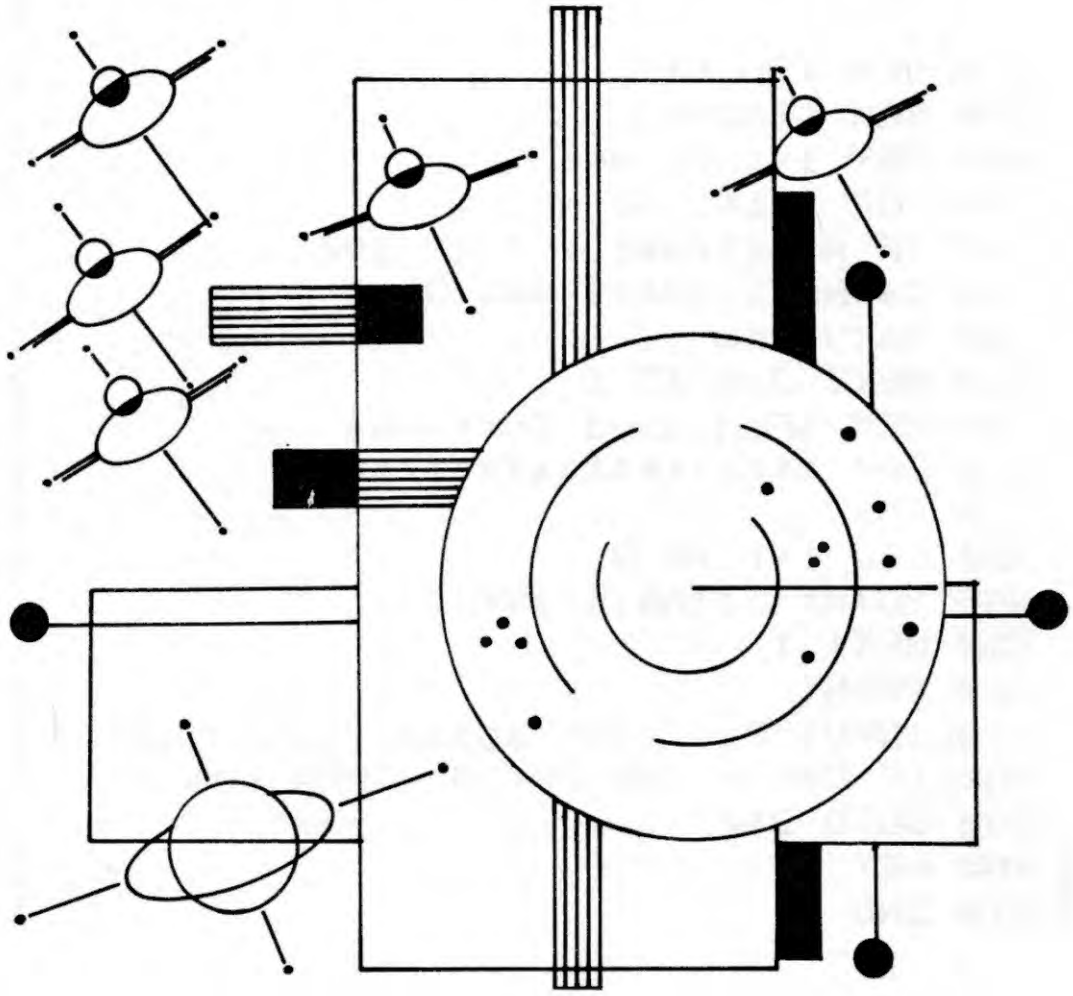
```
60 DIM N$(100)
70 WIDTH 40
80 KEY OFF
90 CLS
100 PRINT TAB(7);"Sorting Program"
110 PRINT TAB(7);"~~~~~~~~~~~~~~~~~~~~~~"
120 PRINT:PRINT:INPUT"Enter the numbe
r of items to sort";N
130 IF INT(N)<>N THEN 120
140 IF N<0 OR N>100 THEN 120
150 PRINT
160 FOR I=1 TO N
170 PRINT"Item No. ";I;" ";
180 INPUT N$(I)
190 NEXT I
200 CLS
210 PRINT"Please wait, I'm thinking !
"
220 REM *Sort*
230 REM ******
240 FOR I=1 TO N-1
250 FOR J=I+1 TO N
260 IF N$(I)<N$(J) THEN 290
270 K$=N$(I):N$(I)=N$(J)
280 N$(J)=K$
290 NEXT J:NEXT I
300 REM *Finished Sorting*
310 REM ******************
320 CLS
330 FOR I=1 TO N
340 PRINT I;TAB(5);N$(I)
350 NEXT I
360 PRINT
370 INPUT"See list again (y/n) ";D$
380 IF D$="n" OR D$="N" THEN 400
390 GOTO 310
400 KEY ON
410 END
```

NIGHT FIGHTER

Warning . . . warning . . . Incoming aliens have threatened to destroy the earth. How long can you delay their almost inevitable victory?

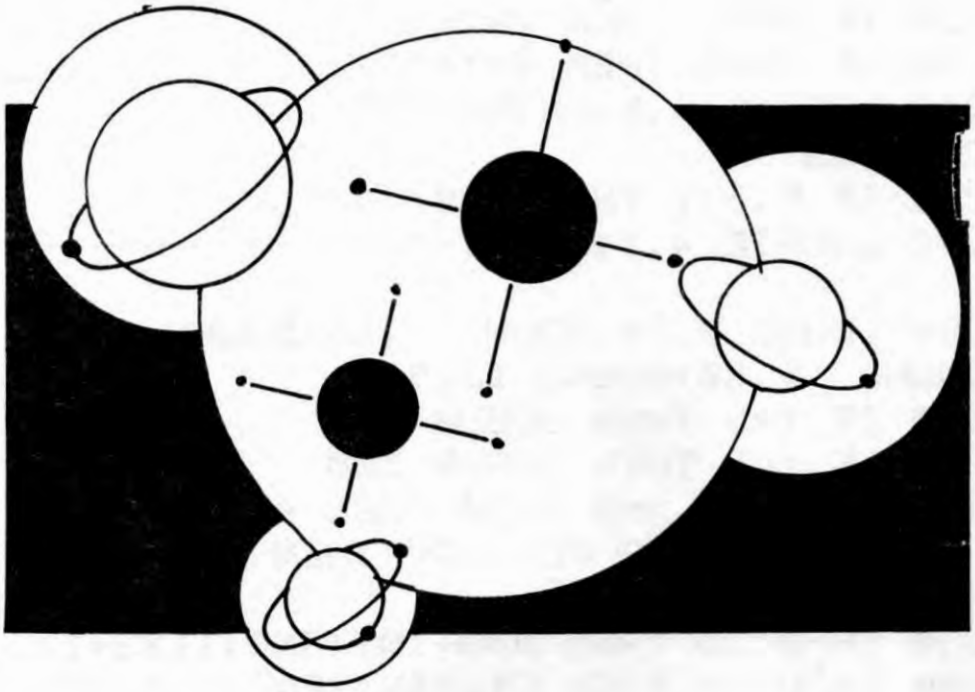
You have only a flickering (and somewhat faulty) radar screen to warn you of their approach. To make matters worse, each time you manage to destroy an attack wave, the next wave increases its speed.

Use the joystick to control your weapon. You are able to adjust the enemy attack speed; a level of 200 is incredibly easy, while one of 15 (or less) will take all your skill to control.



```
10 REM ****************
20 REM *Night Fighter*
30 REM *~~~~~~~~~~~~~~~~~~*
40 REM *D.  Pillinger*
50 REM *  D. Olesh    *
60 REM *  G. Carter   *
70 REM ****************
80 KEY OFF
90 CLS
100 REM *Initialise Variables*
110 REM ************************
120 Z=RND(-TIME)
130 INPUT "Level ";LEV
140 A=0
150 B=INT (RND(1)*10)+5
160 D=INT (RND(1)*15)+8
170 C=32
180 Y=10
190 REM *Read input*
200 REM ************
210 RD=STICK(1):F=STRIG(1)
220 IF RD=1 THEN Y=Y-1
230 IF RD=5 THEN Y=Y+1
240 LOCATE 3,Y:PRINT "?"
250 CLS
260 IF F<>-1 THEN 320
270 LOCATE 4,Y:PRINT "---------------
-"
280 SOUND 0,20:SOUND 7,63:SOUND 8,16:
SOUND 12,30:SOUND 13,9
290 IF Y=B THEN GOSUB 490
300 IF Y=D THEN GOSUB 500
310 IF B=30 AND D=30 THEN 450
320 PC=PC+1:IF PC<=LEV THEN 190 ELSE
PC=0
330 IF B<30 THEN B=B+INT(RND(1)*3+1)
340 IF D<30 THEN D=D+INT(RND(1)*3-1)
350 C=C-1
360 CLS
```

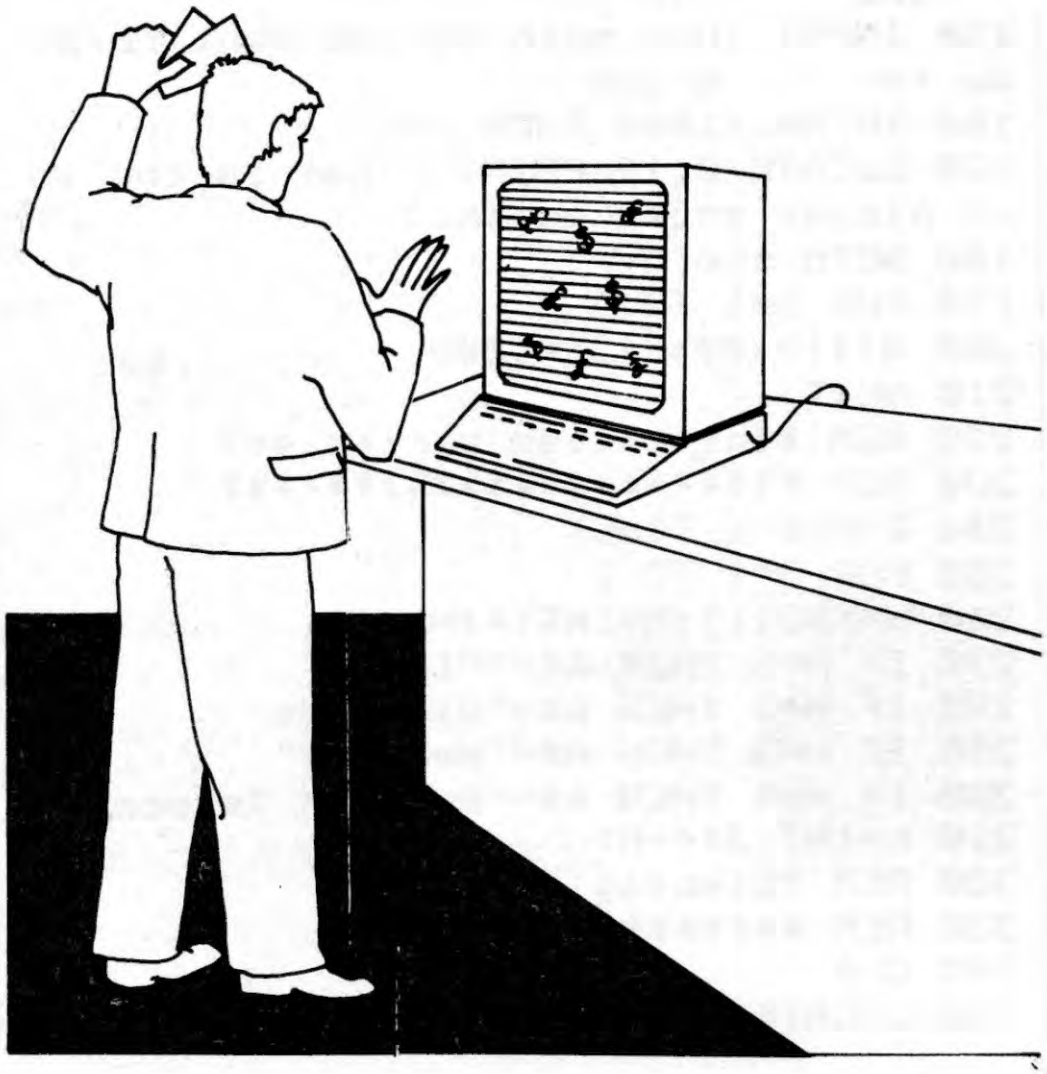
```
370 IF B<30 THEN LOCATE C,B:PRINT "?:
 "
380 IF D<30 THEN LOCATE C,D:PRINT "?:
="
390 IF C>2 THEN 190
400 REM *End of Game*
410 REM ************
420 PRINT "The World has been Demolis
hed"
430 PRINT "The Night Fighter had ";A;
" attempts."
440 STOP
450 A=A+1
460 FOR GL=1 TO 5:NEXT GL
470 LEV=LEV-LEV/3
480 GOTO 150
490 SOUND 6,31:SOUND 7,54:SOUND 12,10
:SOUND 13,9:B=30:RETURN
500 SOUND 6,31:SOUND 7,54:SOUND 12,10
:SOUND 13,9:D=30:RETURN
```



STOCK MARKET

In this program written by Paul Precious you can buy and sell shares in four concerns.

The game is for two people, with the computer calculating the value of the shares and also telling you which of the players has won.



```
10 REM *Stock Market*
20 REM *By P. Precious & G. Carter*
30 REM ****************************
40 DIM A(12),P(4),N$(2)
50 REM *Set up title screen*
60 REM ***********************
70 COLOR 12,15,12
80 KEY OFF
90 CLS
100 LOCATE 9,6:PRINT STRING$(14,"£")
110 LOCATE 9,7:PRINT "£Stock Market£"
120 LOCATE 9,8:PRINT STRING$(14,"£")
130 LOCATE 0,15:INPUT "What is the fi
rst player's name       ";N$(1)
140 INPUT "What is the second player'
s name      ";N$(2)
150 INPUT "How much do you want to pl
ay for      £";NN
160 IF NN>=1000 THEN 190
170 LOCATE 0,19:PRINT "That is too lo
w; please enter again."
180 GOTO 150
190 FOR I=1 TO 4
200 A(I)=100:P(I)=1000
210 NEXT I
220 REM *Initialise variables*
230 REM ***********************
240 B=RND (-TIME)
250 FOR J=1 TO 2
260 H=RND(1):H=INT(4*H)+1
270 IF H=1 THEN A$="Planes"
280 IF H=2 THEN A$="Diamonds"
290 IF H=3 THEN A$="Weapons"
300 IF H=4 THEN A$="British Telecom"
310 K=INT(J*4+H)
320 REM *Display screen*
330 REM ****************
340 CLS
350 LOCATE 0,4:PRINT "Stock Market",J
```

```
;")";" ";N$(J)
360 PRINT:PRINT "Planes  Diamonds  We
apons  B.Telecom"
370 PRINT "£";A(1);TAB(7);" £";A(2);T
AB(17);" £";A(3);TAB(26);" £";A(4)
380 PRINT:PRINT "1 ";A(5);TAB(7);A(6)
;TAB(17);A(7);TAB(26);A(8)
390 PRINT:PRINT "2 ";A(9);TAB(7);A(10
);TAB(17);A(11);TAB(26);A(12)
400 Y=A(1)*A(9)+A(2)*A(10)+A(3)*A(11)
+A(4)*A(12)
410 X=A(1)*A(5)+A(2)*A(6)+A(3)*A(7)+A
(4)*A(8)
420 LOCATE 0,12:PRINT "Value of Share
s: £";
430 IF J=1 THEN PRINT X ELSE PRINT Y
440 PRINT:PRINT "Cash on hand  £";P(J
)
450 IF P(J)>NN THEN PRINT:PRINT N$(J)
;" has won.":GOTO 740
460 PRINT:PRINT "Holdings...";A$;" Bu
y/Sell?"
470 LOCATE 0,17:INPUT B$
480 IF B$="N" THEN 680
490 IF B$="Z" THEN 740
500 IF NOT (B$="BUY" OR B$="SELL") TH
EN 460
510 PRINT B$;" how many ";
520 INPUT N
530 IF B$="S" THEN 590
540 IF P(J)<A(H)*N THEN 500
550 GOSUB 700
560 GOTO 680
570 REM *Can you afford it?*
580 REM **********************
590 IF N<=A(K) THEN 650
600 IF A(K)=0 THEN PRINT "You don't h
ave any to sell.":LOCATE 0,15:GOTO 46
0
```

```
610 PRINT "That is too many; enter ag
ain."
620 LOCATE 0,15:GOTO 460
630 REM *Cash, no., value of shares*
640 REM ****************************
650 N=-N
660 GOSUB 700
670 IF A(H)<10 THEN A(H)=10
680 NEXT J
690 GOTO 250
700 P(J)=INT(P(J)-A(H)*N)
710 A(K)=INT(A(K)+N)
720 F=RND(1):A(H)=INT((A(H)+INT(F*50)
-10)*N/4)
730 RETURN
740 PRINT:INPUT "Another game (Y/N)";
Z$
750 IF Z$="Y" THEN RUN ELSE END
```

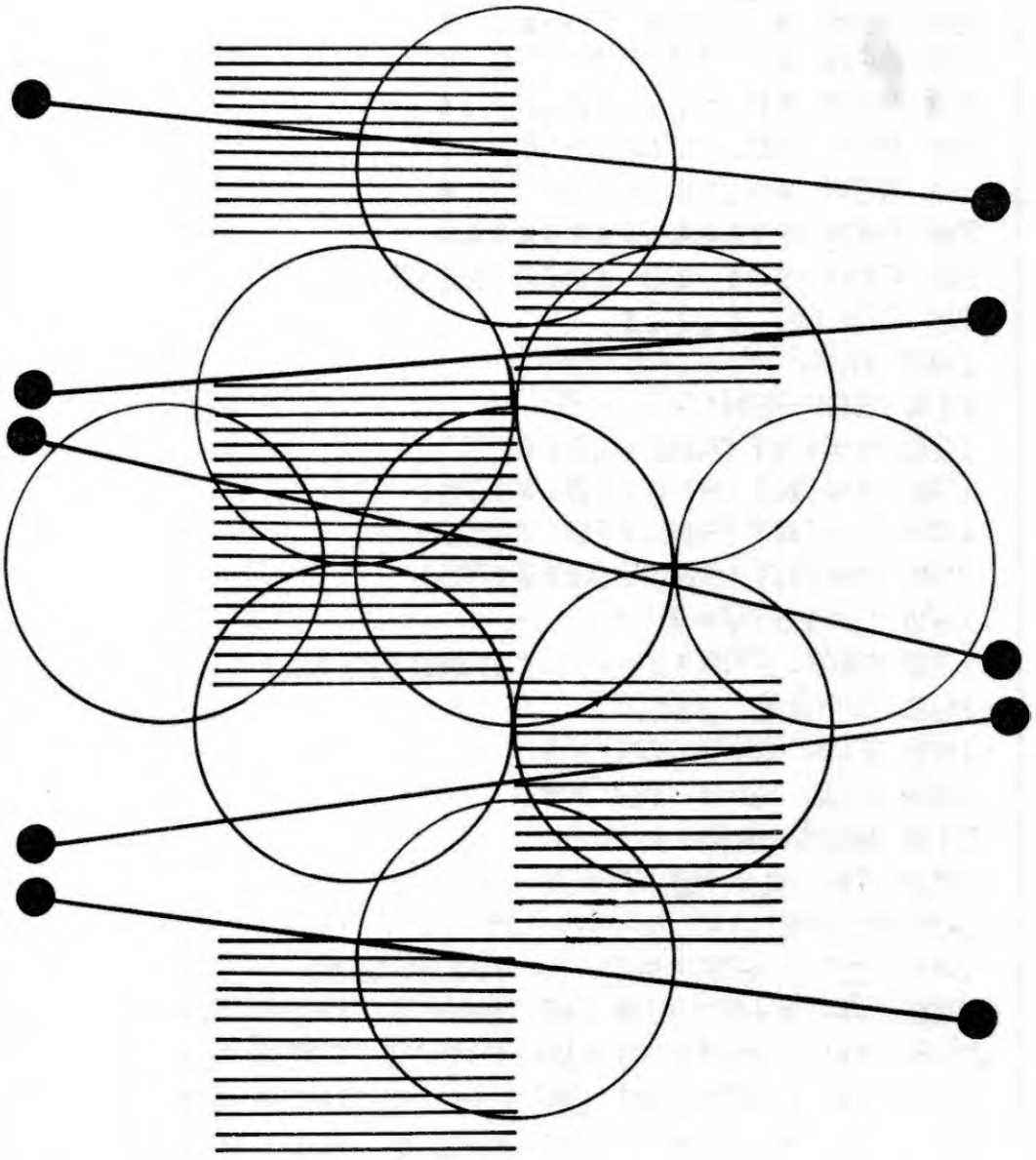
TELEPHONE BOOK

One of the most powerful features of the BASIC language is its ability to read stored data using READ, DATA and RESTORE commands. To demonstrate how this works, Andrew Ogilvie has written a program to store a personal telephone directory. You must type in your data at the end of the program, using line numbers starting at 310. When the program is run, you simply type in the surname of the person you wish to call, and their name and number are displayed on the screen.

```
10 REM ****************
20 REM *Telephone Book*
30 REM *~~~~~~~~~~~~~~~~~~~~*
40 REM *Andrew Ogilvie*
50 REM ****************
60 REM
70 CLS
80 LOCATE 7,0:PRINT"Telephone Book"
90 PRINT:PRINT
100 INPUT"Enter surname you wish to k
now the   tel. number for : ";S$
110 RESTORE
120 REM *Search for data*
130 REM ****************
140 READ N$,I$,T$
150 IF N$="end" THEN PRINT:PRINT"Sorr
y, not listed.":GOTO 240
160 IF N$=S$ THEN 180
170 GOTO 140
180 CLS
190 REM *Print data*
200 REM ************
210 PRINT"Name : ";I$;". ";N$
220 PRINT
230 PRINT"Tel. No. :";T$
240 PRINT:PRINT:PRINT
250 PRINT"Another go (y/n) ";
260 INPUT D$
270 IF D$="n" OR D$="N" THEN 300
280 IF S$="end" THEN PRINT:PRINT"Sorr
y not listed.":GOTO 230
290 GOTO 20
300 END
310 DATA Ogilvie,A,415-92656
320 DATA Carter,G,873-9321
330 DATA police,the,999
340 DATA hospital,the,897-4654
350 DATA end,of,data
```

GRAPHICS DEMONSTRATION

The following programs demonstrate some of the most spectacular features of MSX machines – their colour graphics.



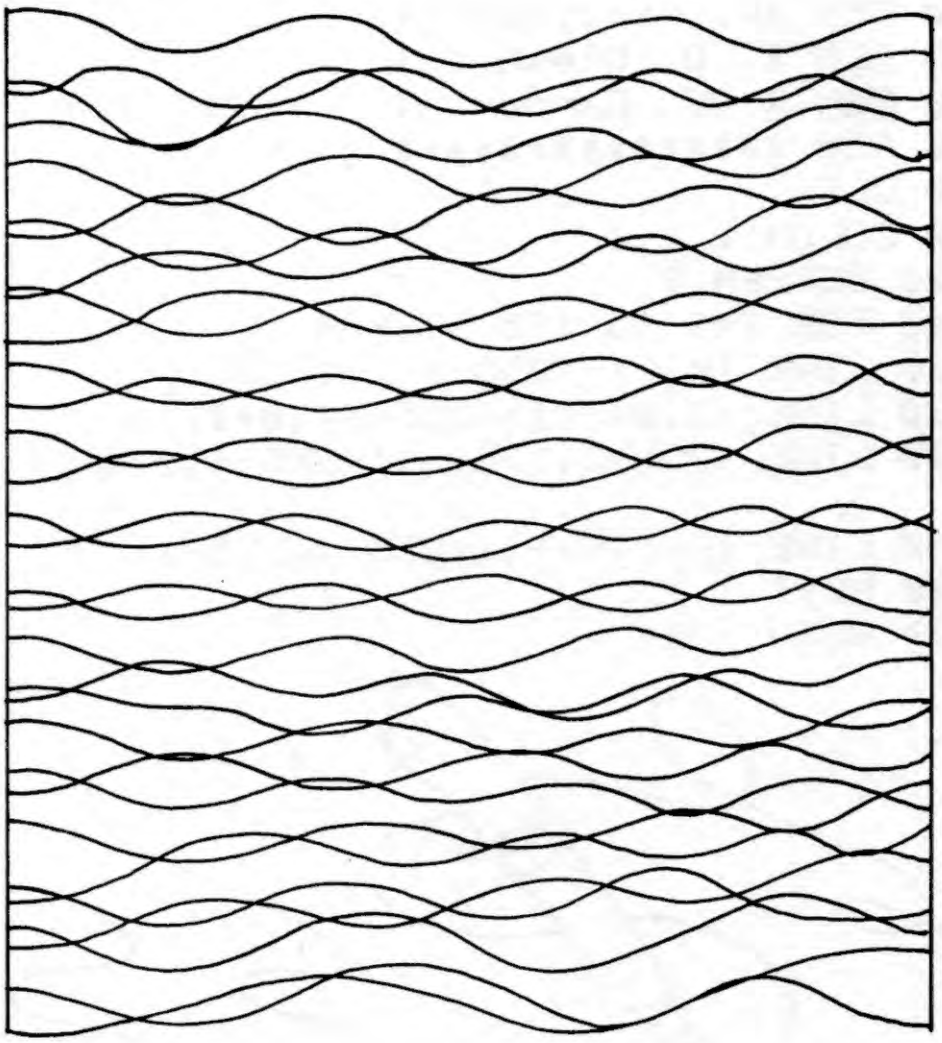
WAVE FORM

This program illustrates the power of the LINE command in the high resolution graphics mode and is possibly the best one in this section. The program is very fast, and the effect is quite hypnotic.

Wave form's shape is determined randomly, so it will differ from run to run.

```
10 REM ***************
20 REM * Wave Form   *
30 REM * ~~~~~~~~~~~~~~~ *
40 REM *D.Pillinger,*
50 REM *D. Olesh &   *
60 REM *G. Carter    *
70 REM ***************
80 FOR X=1 TO 1000:NEXT X
90 COLOR 1,1,1
100 CLS
110 SCREEN 2
120 X=INT(RND(10)*255)
130 Y=INT(RND(10)*175)
140 L=INT(RND(10)*255)
150 M=INT(RND(10)*175)
160 U=15:V=7
170 DEF FNR(X)=INT(RND(1)*X)
180 GOSUB 360
190 FOR Q=2 TO 16
200 FOR G=1 TO 150
210 NUM=NUM-1
220 IF NUM=0 THEN GOSUB 360
230 LINE (X,Y)-(L,M),Q
240 IF INKEY$=" " THEN 370
250 IF X+A>255 OR X+A<0 THEN A=-A
260 IF Y+B>175 OR Y+B<0 THEN B=-B
270 IF L+C>255 OR L+C<0 THEN C=-C
280 IF M+D>175 OR M+C<0 THEN D=-D
```

```
290 X=X+A:Y=Y+B
300 L=L+C:M=M+D
310 NEXT G
320 FOR VV=1 TO 300:NEXT VV
330 CLS
340 NEXT Q
350 IF INKEY$=" " THEN 370
360 A=FNR(U)-V
370 B=FNR(U)-V
380 C=FNR(U)-V
390 D=FNR(U)-V
400 NUM=FNR(20)+10
410 RETURN
```

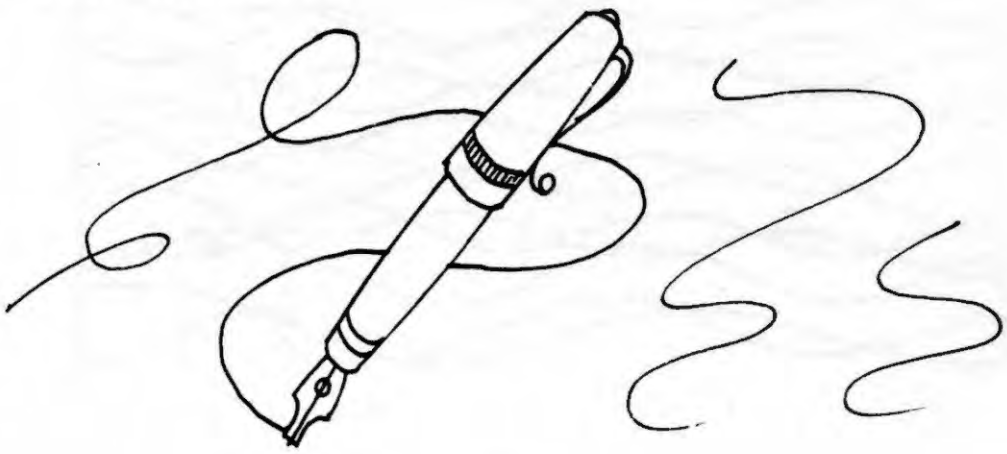


GRAPHICS

This graphics program illustrates the computer's speed and accuracy, and makes good use of the high resolution graphics in Mode 2.

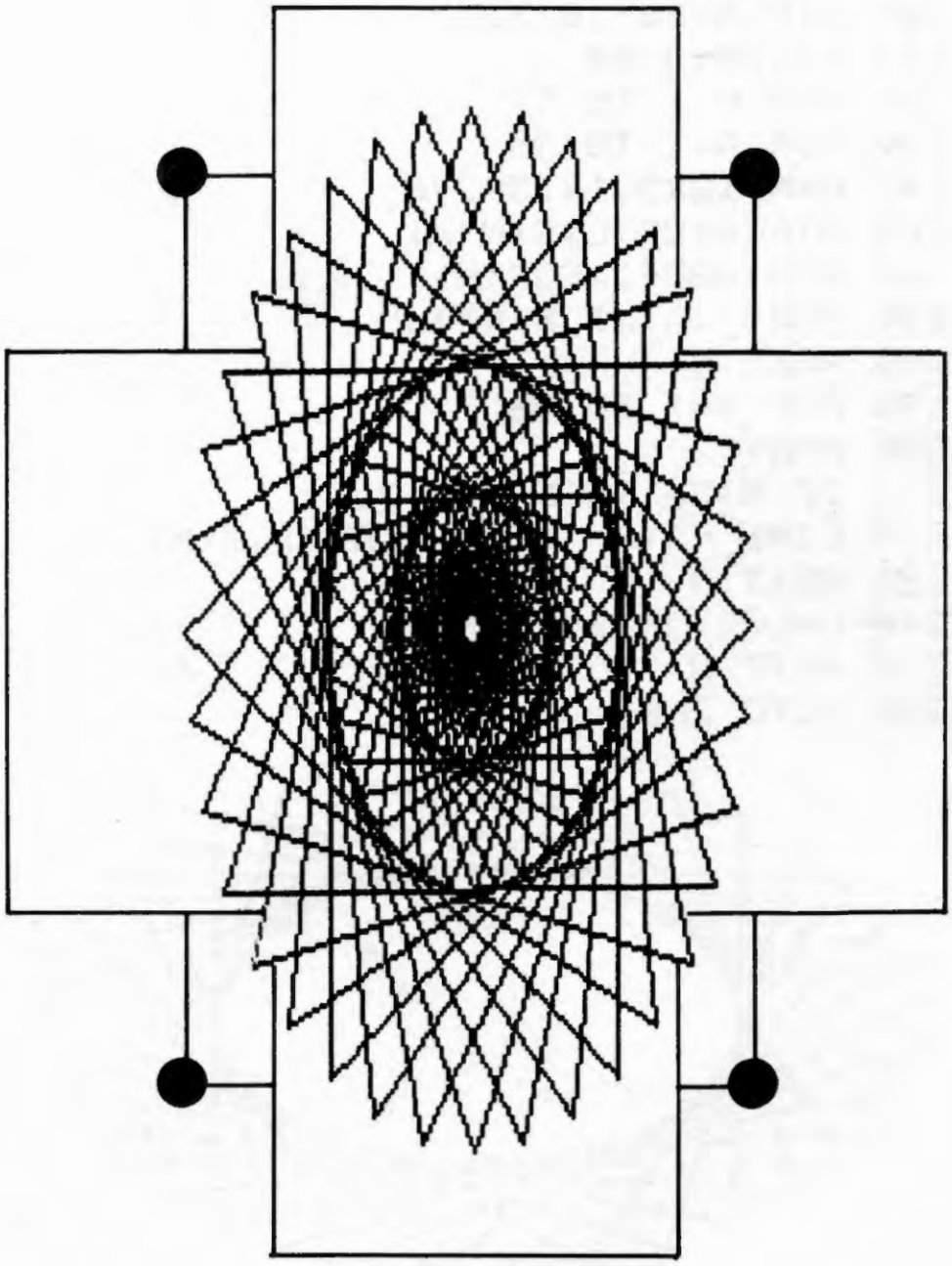
You might like to add a variable for colour, and make up a loop to make the pattern appear in different colours.

```
10 REM ****************
20 REM *  Graphics     *
30 REM *  ~~~~~~~~~~~  *
40 REM *D. Pillinger,*
50 REM *  D. Olesh &  *
60 REM *  G. Carter.  *
70 REM ****************
80 CLS
90 COLOR 1,1,1
100 SCREEN 2
110 FOR I=1 TO 175 STEP 4
120 LINE (0,I)-(255-I,I-I),3
130 LINE (I,0)-(I+(255-I),0+I),3
140 LINE (255-I,175)-(I-255-I,175-I),
3
150 LINE (I,175)-(I+255-I,175-I),3
160 NEXT I
170 GOTO 170
```

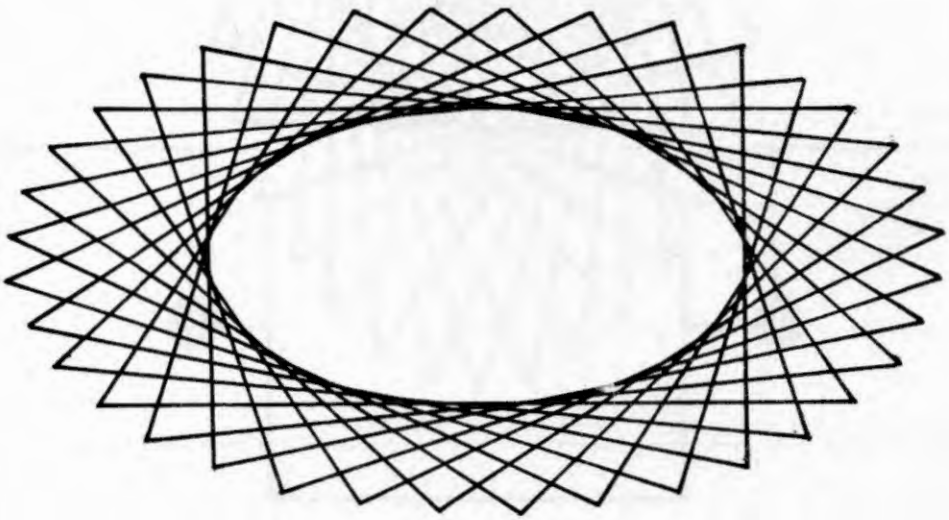


TIME'S EYE

Before drawing the picture, the program calculates the point positions. It puts them in an array and then plots the graph.

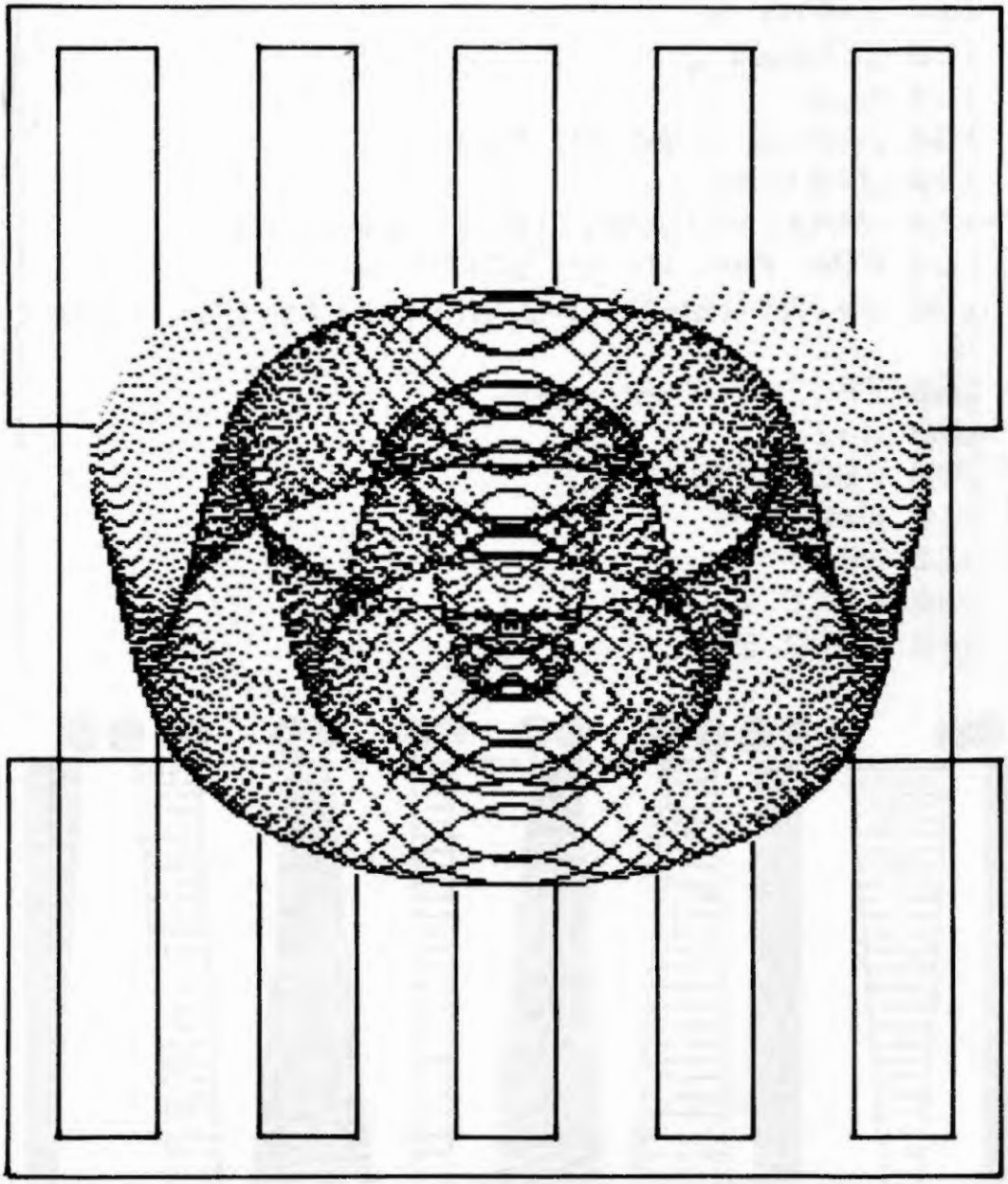


```
10 REM ****************
20 REM *  Time's Eye  *
30 REM *  ~~~~~~~~~~~~~~  *
40 REM *D. Pillinger,*
50 REM * D. Olesh &  *
60 REM * G. Carter.  *
70 REM ****************
80 COLOR 1,1,1
90 SCREEN 2
100 DIM A(36),B(36)
110 L=120:J=80
120 FOR H=1 TO 5
130 FOR N=1 TO 36
140 K=N/18*3.1415927#
150 A(N)=128+L*SIN(K)
160 B(N)=88+J*COS(K)
170 PSET (A(N),B(N)),3
180 NEXT N
190 FOR N=1 TO 36
200 M=N+12
210 IF M>36 THEN M=M-36
220 LINE (A(N),B(N))-(A(M),B(M)),6
230 NEXT N
240 L=L/2:J=J/2
250 NEXT H
260 GOTO 260
```

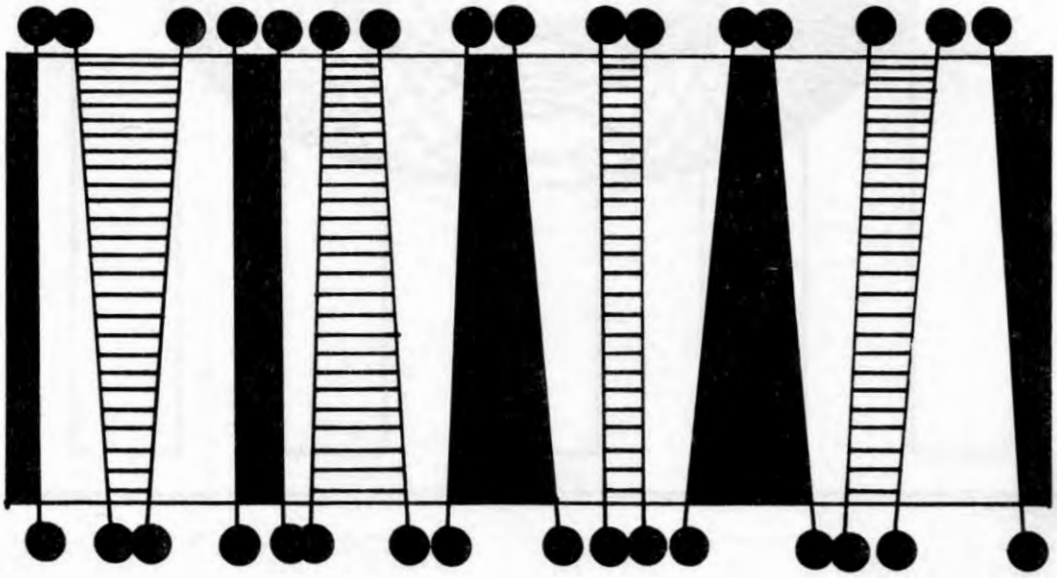


3D HAT

This program draws a three-dimensional Mexican hat on the screen. It takes a while to complete, but the wait is worth it.



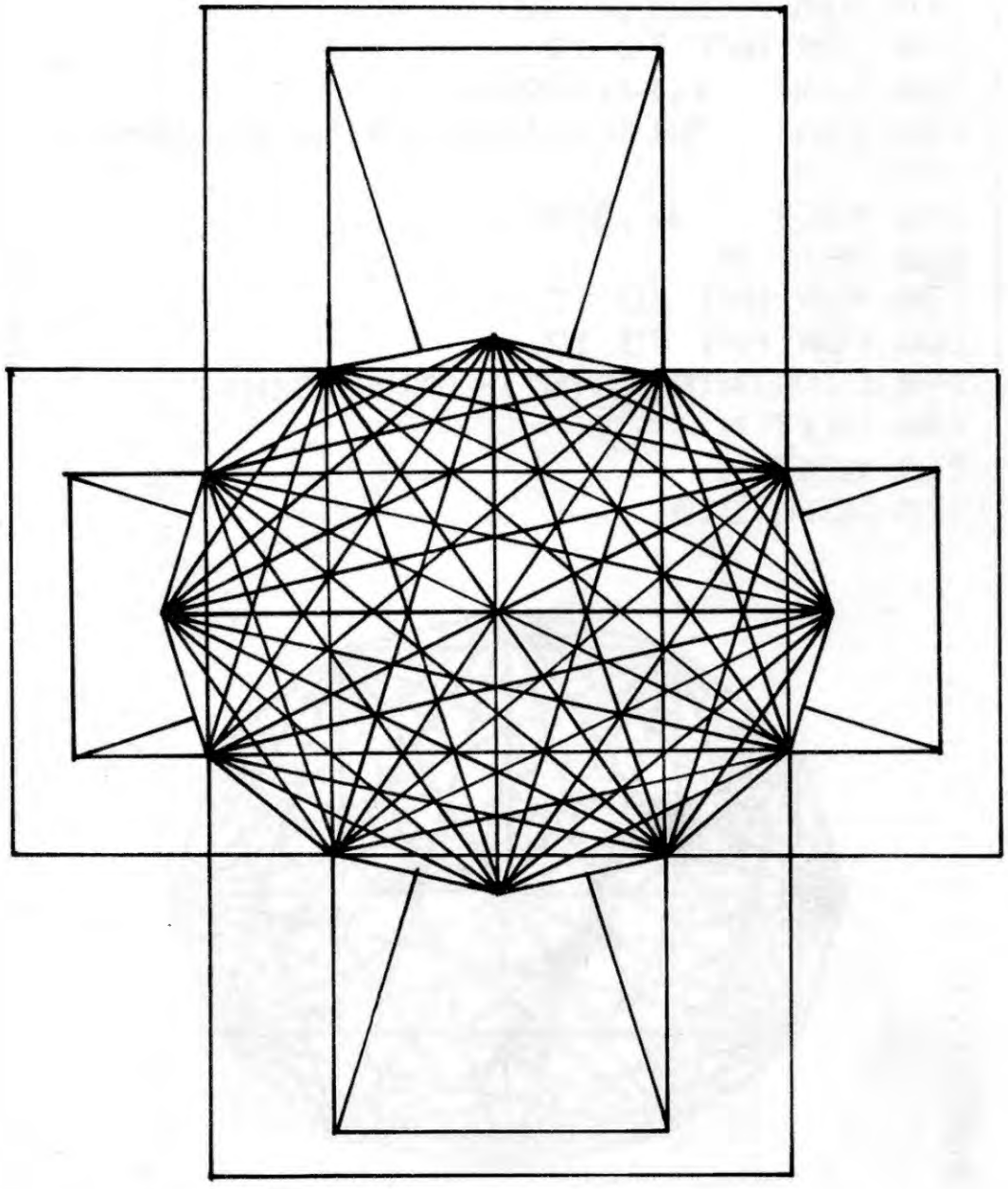
```
10 REM ***************
20 REM *    3D HAT    *
30 REM *    ~~~~~~~   *
40 REM *D.Pillinger,*
50 REM *D. Olesh      *
60 REM ***************
70 CLS
80 DEF FNA(Z)=32*SIN(Z/6)
90 PRINT "Resolution (1-10)"
100 INPUT R
110 SCREEN 2
120 CLS
130 FOR X=-100 TO 100
140 J=0:K=1
150 V=R*INT(SQR((10^4)-X*X)/R)
160 FOR Y=V TO -V STEP -R
170 Z=INT(80+FN A(SQR(X*X+Y*Y))-.707*
Y)
180 IF Z<J THEN 190
190 J=Z
200 PSET(X+110,Z)
210 K=0
220 NEXT Y
230 NEXT X
240 GOTO 240
```



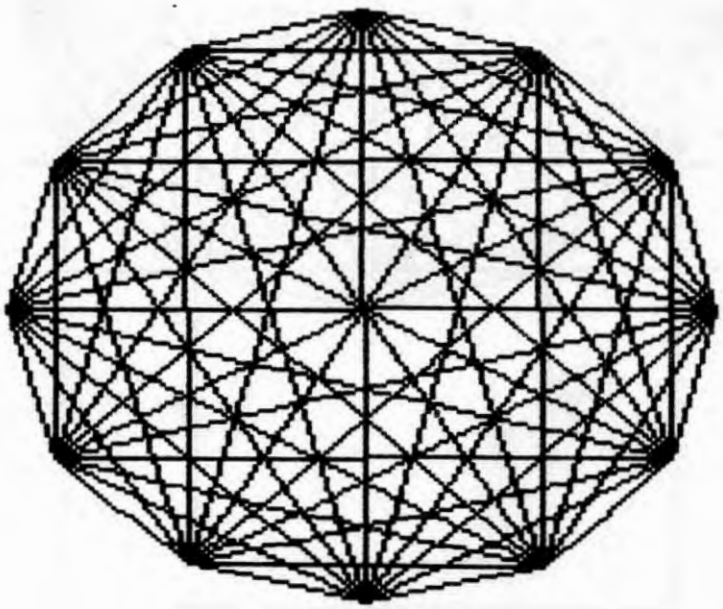
EYE

This program shows how arrays can be used to store data which will be used to create a graphic display.

The pattern produced is like a devil's eye. As the hole shrinks, the speed increases and the wheel develops.



```
10 REM ***************
20 REM *     Eye       *
30 REM *     ~~~~      *
40 REM *D.Pillinger,*
50 REM * D. Olesh & *
60 REM * G. Carter. *
70 REM ***************
80 SCREEN 2
90 COLOR 1,7,7
100 CLS
110 DIM A(12),B(12)
120 FOR N=1 TO 12
130 K=N/6*3.1415927#
140 A(N)=128+80*SIN(K):B(N)=88+80*COS
(K)
150 PSET(A(N),B(N))
160 NEXT N
170 FOR N=1 TO 12
180 FOR M=1 TO 12
190 LINE(A(N),B(N))-(A(M),B(M))
200 NEXT M
210 NEXT N
220 GOTO 220
```



How To Write Better Programs

Compiled by Tim Hartnell

It's all very well spending your time typing in programs like those in this book, but there is sure to come a time when you decide you'd like to develop some games programs of your own. In this section of the book, I'd like to discuss a few ideas which may help you write games which you'll both enjoy developing and — more importantly — you and your friends will enjoy playing.

HAVE A CLEAR GOAL IN MIND

Although in many (perhaps most) cases, your computer program will take on a life of its own as you write it, developing away from the concept you had in mind when you started programming, it is important at the outset to have a pretty good idea of what your game will involve.

This is not as obvious a suggestion as you might think. Of course, you'll know if you're developing a 'chase the ghosts around the maze as you eat the power pills' program that you are going to need a different sort of program layout to one which places you inside a Haunted Oak, peopled with gremlins and halflings. But you have to go beyond the basic "I'm going to write me an Adventure" stage to work out such things as (a) what the object of the game will be; (b) what the screen display will look like; (c) what variables, and variable names, you'll need;

(d) the nature of the player input; (e) how 'winning' or 'losing' will be determined; and so on.

Let's look at these one by one.

THE OBJECT OF THE GAME

This can usually be stated very succinctly: "To find the lost treasure of the Aztecs"; "To destroy as many asteroids as possible before running out of ships"; or "To play a game of chess". But even though this stage of the game production can be accomplished very quickly, it should not be overlooked. Get this statement — which might be just a sentence, or may run to a paragraph length or more, if there is more than one 'screen' to be worked through, with a different scenario for each screen — down in writing.

You may well discard the original aim as the program develops, and it looks like the direction it is taking is better than the one you first thought of. Despite this, it is important to have something concrete to aim at, to stop you wasting hour after hour doodling aimlessly.

THE SCREEN DISPLAY

I've found that making a sketch, or sketches, of what the display will look like once the program is up and running, is of tremendous benefit. Once you have your drawing, and it doesn't matter how rough it is so long as it shows all the important things you want on the screen, and their relative positions and size, you'll discover the program concept is likely to crystalize.

As well as seeing immediately how you will write parts of the code to achieve the game's aim, you'll get an idea of whether or not the game is even worth writing in the form you had considered. Perhaps the game will be too complex if you leave everything on the screen you were intending to; or maybe most of the screen will be wasted space, so that a more complicated game scenario should be devised.

I've discovered that sketching the proposed screen display before starting to program is particularly useful, especially when creating arcade and simulation games. You get an indication of the variables you'll need, the user-defined graphics, the kind of player inputs which will be most conducive to good player interaction, and so on.

Simulation games, as you probably know, are those in which the computer models an external reality — such as running a cake shop, a war, or an airport — and allows you to experience (after a fashion) what it would be like to take part in such an activity in real life. Simulation games are not particularly difficult to write — in terms of needing clever coding — but instead demand a methodical, painstaking approach to the program.

In my book *The ZX Spectrum Explored* (Sinclair Browne, 1982), there is a program with the unlikely name of 'Workin' for the Man', in which you are running a factory, staffed with a highly-erratic workforce, involved in the manufacture of some mythical product called 'The Zibby'. The player gets a factory report two or three times a week, and from this report has to decide how many staff he or she will hire or (attempt to) fire, how many Zibbies will be the production target for the week, and so on.

This report is the key to the program, and when I wrote the game, I started by making a sketch of how the screen would look. It was a bit like this:

FACTORY REPORT: WEEK 5

Capital in hand is $2,657.92

Your stores hold 12 Zibbies worth $169.68
They sell for $14.14 each and cost $7.41
each to make

Workforce is 7 people
Their wages are $41 each and the wage bill
this week is $287

Each person can make 10 Zibbies a week,
a total output of 70

Once I had this sketch drawn up, I was ready to go. As you can see, it gives a very good indication of the variables which will be needed. For a start, I know I'll have to control the number of the week, the capital, the contents of the stores (and their value) and so on.

I found that once I'd completed the screen display sketch, the rest of the program was relatively easy to write. Doing a sketch in this way gives you an instant guide to the main variables you'll need.

USE HELPFUL VARIABLE NAMES

I also tend to use variable names which relate in some way to that which they are representing, as it saves having to keep a list of the variables which have been assigned, and what they've been assigned to. For example, I could use WK for week, CH for capital in hand, MZ for the cost of making each Zibby and SZ for the selling price. If Z was the number of Zibbies, I would know that the total value of Zibbies I had was Z (the number of them) multiplied by SZ (their selling price) and it cost me Z multiplied by MZ (their price of manufacture) to make them. My profit, if I sold them all, would then be Z*SZ minus Z*MZ.

If you follow a similar idea, you'll find it is much easier to keep track of what is happening in your program than might otherwise be the case.

THE NATURE OF THE PLAYER INPUT

It's important to make games *easy* and fun to play. It's not good having the best Asteroids-derivative program in the world if players have trouble hitting the fire button because you've placed it right next door to the 'rotate' control.

Many programs which provide 'up', 'down', 'right' and

'left' controls, automatically use arrow or cursor keys, even though these might be most inconvenient for the player to use. Have a look at your keyboard, and see if you can find better ones. I often use "Z" and "M" for programs which need just left and right movement, with the space bar for fire. These keys seem logical to me, and no player time is wasted in learning them, or trying to remember them when the game is underway. In a similar way, I tend to use "A" (for up) and "Z" (for down) for the left hand, and the "greater than" and "less than" keys for left and right (pointing out to the player that the < and > symbols point in the relevant directions).

Use INKEY$ or GET$ whenever you can, to prevent the player from having to use the RETURN or ENTER keys to get the program underway.

HOW THE GAME WILL END

The way the game will be won and lost needs to be defined, and clear to the player. Do you need to blast all the aliens to win, and will you lose automatically if one alien lands, and you've still got ships left, or only if you have no ships left. In a two-player game, is the loser the first player to lose three lives, or seven pieces, or does the game only end when the *difference* between the two scores is three or seven or whatever.

Work this out, and make it very clear to the player. Whether the goal of the game is to clear the left-hand side of the screen of the Screaming Widgies, or to clock up a fortune of $7.3 billion, it must be both clear to the player, and *possible to achieve*. A 'win condition' which can never be achieved on the higher levels of play is most unsatisfactory. No matter how difficult it is to do, you are only defrauding players if you set goals whose achievement is not possible within the constrictions you've put into the game.

I hope these five points may give you a few ideas on how you can go ahead and write programs which will be relatively easy to write, and which will be satisfying for you and your friends to play.

GLOSSARY

A

Accumulator — the place within the computer in which arithmetic computations are performed and where the results of these computations are stored.

Algorithm — the series of steps the computer follows to solve a particular problem.

Alphanumeric — this term is usually used in relation to a keyboard, as in 'it is an alphanumeric keyboard', which means that the keyboard has letters as well as numbers. It is also used to refer to the 'character set' of the computer. The character set comprises the numbers and letters the computer can print on the screen.

ALU (Arithmetic/Logic Unit) — the part of the computer which does arithmetic (such as addition, subtraction) and where decisions are made.

AND — a Boolean logic operation that the computer uses in its decision-making process. It is based on Boolean algebra, a system developed by mathematician George Boole (1815-64). In Boolean algebra the variables of an expression represent a logical operation such as OR and NOR.

ASCII — stands for American Standard Code for Information Exchange, the most widely used encoding system for English language alphanumerics. There are 128 upper and lower case letters, digits and some special characters. ASCII converts the symbols and control instructions into seven-bit binary combinations.

Assembler — a program which converts other programs written in assembly language into machine code (which the computer can understand directly).

Assembly language is a low level programming language which uses easily memorised combinations of two or three letters to represent a particular instruction which the assembler then converts so the machine can understand it. Examples of these are ADD (add), and SUB (subtract). A computer programmed in assembly language tends to work more quickly than one programmed in a higher level language such as BASIC.

B

BASIC — an acronym for Beginners All-Purpose Symbolic Instruction Code. It is the most widely used computer language in the microcomputer field. Although it has been criticised by many people, it has the virtue of being very easy to learn. A great number of BASIC statements resemble ordinary English.

Baud — named after Baudot, a pioneer of telegraphic communications. Baud measures the rate of transfer of information and is approximately equal to one bit per second.

BCD — an abbreviation for Binary Coded Decimal.

Benchmark — a test against which certain functions of the computer can be measured. There are a number of so-called 'standard Benchmark tests', but generally these only test speed. This is rarely the aspect of a microcomputer that is most of interest to the potential buyer.

Binary — a numbering system that uses only zeros and ones.

Bit — an abbreviation for Binary Digit. This is the smallest unit of information a computer circuit can recognise.

Boolean Algebra — the system of algebra developed by mathematician George Boole which uses algebraic notation to express logical relationships (see AND).

Bootstrap — a short program or routine which is read into

the computer when it is first turned on. It orients the computer to accept the longer, following program.

Bug — an error in a computer program which stops the program from running properly. Although it is generally used to mean only a fault or an error in a program, the term bug can also be used for a fault in the computer hardware.

Bus — a number of conductors used for transmitting signals such as data instructions, or power in and out of a computer.

Byte — a group of binary digits which make up a computer word. Eight is the most usual number of bits in a byte.

C

CAI — Computer Assisted Instruction.

CAL — Computer Assisted Learning. The term is generally used to describe programs which involve the learner with the learning process.

Chip — the general term for the entire circuit which is etched onto a small piece of silicon. The chip is, of course, at the heart of the microcomputer.

Clock — the timing device within the computer that synchronises its operations.

COBOL — a high level language derived from the words Common Business Orientated Language. COBOL is designed primarily for filing and record-keeping.

Comparator — a device which compares two things and produces a signal related to the difference between the two.

Compiler — a computer program that converts high level programming language into binary machine code so the computer can handle it.

Complement — a number which is derived from another according to specified rules.

Computer — a device with three main abilities or functions:
1) to accept data
2) to solve problems
3) to supply results

CPU — stands for Central Processing Unit. This is the heart of the computer's intelligence, where data is handled and instructions are carried out.

Cursor — a character which appears on the TV screen when the computer is operating. It shows where the next character will be printed. On a computer there are usually 'cursor control keys' to allow the user to move the cursor around the screen.

D

Data — information in a form which the computer can process.

Debug — the general term for going through a program and correcting any errors in it, that is, chasing down and removing bugs (see Bug).

Digital Computer —a computer which operates on information which is in a discrete form.

Disk/Disc — this is a magnetically sensitised plastic disk, a little smaller than a single play record. This is used for storing programs and for obtaining data. Disks are considerably faster to load than a cassette of the same length program. The disk can be searched very quickly while a program is running for additional data.

Display — the visual output of the computer, generally on a TV or monitor screen.

Dot Matrix Printer — a printer which prints either the listing of a program or that which is displayed on the TV screen. Each letter and character is made up of a number of dots. The higher the number of dots per character the finer the resolution of the printer.

Dynamic Memory — a memory unit within the computer which 'forgets' its contents when the power is turned off.

E

Editor — this term is generally used for the routine within the computer which allows you to change lines of a program while you are writing it.

EPROM — stands for Erasable Programmable Read-Only Memory. This is like the ROM in the computer, except that it is fairly easy to load material into an EPROM and it doesn't disappear when you turn the power off. EPROMs must be placed in a strong ultra violet light to erase them.

Error Messages — the information given by a computer where there is a fault in the coding during a part of a program, usually shown by the computer stopping, and printing a word, or a word and numbers, or a combination of numbers only, at the bottom of the screen. This tells you what mistake has been made. Common mistakes include using the letter O instead of zero in a line, or leaving out a pair of brackets, or one of the brackets, in an expression, or failing to define a variable.

F

File — a collection of related items of information organised in a systematic way.

Floppy Disk — a relatively cheap form of magnetic disk used for storing computer information, and so named because it is quite flexible (see Disk/Disc).

Flow Chart — a diagram drawn up before writing a program, in which the main operations are enclosed within rectangles or other shapes and connected by

lines, with arrows to represent loops, and decisions written at the branches. It makes writing a program much easier because traps such as infinite loops, or non-defined variables can be caught at an early stage. It may not be worth writing a flow chart for very short programs, but generally a flow chart aids in creating programs.

Firmware — there are three kinds of 'ware' in computers: software 'temporary' programs; hardware like the ROM which contains permanent information; and firmware in which the information is relatively permanent, as in an EPROM (see EPROM).

Flip-Flop — a circuit which maintains one electrical condition until changed to the opposite condition by an input signal.

FORTRAN — an acronym for FORmula TRANslation, this is a high level, problem orientated computer language for scientific and mathematical use.

G

Gate — an electrical circuit which, although it may accept one or more incoming signals, only sends out a single signal.

Graphics — pictorial information as opposed to letters and numbers.

H

Hard Copy — computer output which is in permanent form.

Hardware — the physical parts of the computer (also see software and firmware).

Hexadecimal (Hex) — a numbering system to the base sixteen. The digits zero to nine are used, as well as the letters A, B, C, D, E and F to represent numbers. A

equals 10, B equals 11, C equals 12, and so on. Hex is often used by microprocessor users.

Hex Pad — a keyboard designed specifically for entering hexadecimal notation.

High Level Language — a programming language which allows the user to talk to the computer more or less in English. In general, the higher the level of the language (that is, the closer it is to English), the longer it takes for the computer to translate it into a language it can use. Lower level languages are far more difficult for human operators but are generally executed far more quickly.

I

Input — the information fed into the computer via a keyboard, a microphone, a cassette or a disk.

Input/Output (I/O Device) — a device which accepts information or instructions from the outside world, relays it to the computer, and then, after processing, sends the information out in a form suitable for storing, or in a form which could be understood by a human being.

Instruction — data which directs a single step in the processing of information by the computer (also known as a command).

Integrated Circuit — a complete electronic circuit imprinted on a semiconductor surface.

Interface — the boundary between the computer and a peripheral such as a printer.

Interpreter — a program which translates the high level language fed in by the human operator, into a language which the machine can understand.

Inverter — a logic gate that changes the signal being fed in, to the opposite one.

Interactive Routine — part of a program which is

repeated over and over again until a specified condition is reached.

J

Jump Instruction — an instruction which tells the computer to go to another part of the program, when the destination of this move depends on the result of a calculation just performed.

K

K — this relates to the size of the memory. Memory is usually measured in 4K blocks. 1K contains 1,024 bytes.

Keyword — the trigger word in a line of programming, usually the first word after the line number. Keywords include STOP, PRINT and GOTO.

L

Language — computer languages are divided into three sections: high level languages, such as BASIC, which are reasonably close to English and fairly easy for humans to use; low level languages, such as Assembler, that use short phrases which have some connection with English (ADD for add and RET for return, for instance); and machine code which communicates more or less directly with the machine.

LCD — this stands for Liquid Crystal Diode. Some computers such as the TRS-80 Pocket Computer use an LCD display.

LED — this stands for Light Emitting Diode. The bright red numbers which are often used on watch or clock displays are made up of LEDs.

Logic — the mathematical form of a study of relationships between events.

Loop — a sequence of instructions within a program which is performed over and over again until a particular condition is satisfied.

M

Machine Language or Machine Code — an operation code which can be understood and acted upon directly by the computer.

Magnetic Disk — see Disk and Floppy Disk.

Mainframe — computers are generally divided into three groups, and the group a computer falls into depends more or less on its size. The computer you are thinking of buying is a microcomputer; medium sized computers are known as minicomputers; and the giant computers that you sometimes see in science fiction movies are mainframe computers. Until 15 years ago mainframe computers were, in practical terms, the only ones available.

Memory — there are two types of memory within a computer. The first is called ROM (read-only memory); this is the memory that comes already programmed on the computer, which tells the computer how to make decisions and how to carry out arithmetic operations. This memory is unaffected when you turn the computer off. The second type is RAM (random access memory). This memory holds the program you type in at the keyboard or send in via a cassette or disk. In most computers the computer 'forgets' what is in RAM when you turn the power off.

Microprocessor — the heart of any computer. It requires peripheral unit interfaces, such as a power supply and input and output devices, to act as a microcomputer.

MODEM — stands for Modulator Demodulator. This is a device which allows two computers to talk to each

other over the telephone. The computers usually use a cradle in which a telephone receiver is placed.

Monitor — this has two meanings in computer terms. One meaning is a television-like display. A monitor has no facility for tuning television programs, and usually the picture produced on a monitor is superior to that produced by an ordinary television. The second meaning of a monitor relates to ROM. The monitor of a computer is described as the information it has built in when you buy it. This information allows it to make decisions and carry out arithmetic computations.

Motherboard — a framework to which extra circuits can be added. These extra circuits often give the computer facilities which are not built-in, such as that of producing sound or of controlling a light pen.

MPU — an abbreviation for Microprocessor Unit.

N

Nano-second — a nano-second is one thousand billionth of a second, the unit of speed in which a computer or a memory chip is often rated.

Non-Volatile Memory — memory which is not lost when the computer is turned off. Some of the smaller computers such as the TRS-80 Pocket Computer have non-volatile memory. The batteries hold the program you enter for several hundred hours.

Not — a Boolean logic operation that changes a binary digit into its opposite.

Null String — a string which contains no characters. It is shown in the program as two double quote marks, without anything between them.

Numeric — pertaining to numbers as opposed to letters (that is, alphabetic). Many keyboards are described as being alphanumeric which means both numbers and letters are provided.

O

Octal — a numbering system which uses eight as the base, and the digits 0, 1, 2, 3, 4, 5, 6 and 7. The Octal system is not used very much nowadays in microcomputer fields. The Hexadecimal system is more common (see Hexadecimal).

Operating System — the software or firmware generally provided with the machine that allows you to run other programs.

OR — an arithmetic operation that returns a 1, if one or more inputs are 1.

Oracle — a method of sending text messages with a broadcast television signal. A teletext set is required to decode the messages. Oracle is run by Independent Television Service in the UK, and a similar service — Ceefax — is provided by the BBC.

Output — information or data fed out by the computer to such devices as a TV-like screen, a printer or a cassette tape. The output usually consists of the information which the computer has produced as a result of running a program.

Overflow — a number too large or too small for the computer to handle.

P

Pad — see Keypad.

Page — often used to refer to the amount of information needed to fill one TV screen, so you can talk about seeing a page of a program, the amount of the listing that will appear on the screen at one time.

PASCAL — a high level language.

Peripheral — anything which is hooked onto a computer, for control by the computer, such as a disk unit, a printer or a voice synthesiser.

Port — a socket through which information can be fed out of or in to a computer.

Prestel — the British telecom name for a system of calling up pages of information from a central computer via the telephone and displaying them on a television screen. A similar commercial version in the United States is known as The Source.

Program — in computer terms program has two meanings. One is the list of instructions that you feed into a computer, and the second is used as a verb, as in 'to program a computer'.

PROM — stands for Programmable Read Only Memory. This is a device which can be programmed, and once it is then the program is permanent (also see EPROM and ROM).

R

Random Access Memory (RAM) — the memory within a computer which can be changed at will by the person using the computer. The contents of RAM are usually lost when a computer is turned off. RAM is the memory device that stores the program that you type in and also stores the results of calculations in progress.

Read-Only Memory (ROM) — in contrast to RAM, information in ROM cannot be changed by the user of the computer, and the information is not lost when the computer is turned off. The data in ROM is put there by the manufacturers and tells the computer how to make decisions and how to carry out arithmetic computations. The size of ROM and RAM is given in the unit K (see K).

Recursion — the continuous repetition of a part of the program.

Register — a specific place in the memory where one or more computer words are stored during operations.

Reserved Word — a word that you cannot use for a variable in a program because the computer will read it as something else. An example is the word TO. Because TO has a specific computer meaning, most computers will reject it as a name for a variable. The same goes for words like FOR, GOTO and STOP.

Routine — this word can be used as a synonym for program, or can refer to a specific section within a program (also see Subroutine).

S

Second Generation — this has two meanings. The first applies to computers using transistors, as opposed to first generation computers which used valves. Second generation can also mean the second copy of a particular program; subsequent generations are degraded by more and more noise.

Semiconductor — a material that is usually an electrical insulator but under specific conditions can become a conductor.

Serial — information which is stored or sent in a sequence, one bit at a time.

Signal — an electrical pulse which is a conveyor of data.

Silicon Valley — the popular name given to an area in California where many semiconductor manufacturers are located.

SNOBOL — a high level language.

Software — the program which is entered into the computer by a user which tells the computer what to do.

Software Compatible — this refers to two different computers which can accept programs written for the other.

Static Memory — a non-volatile memory device which retains information so long as the power is turned on,

but does not require additional boosts of power to keep the memory in place.

Subroutine — part of a program which is often accessed many times during the execution of the main program. A subroutine ends with an instruction to go back to the line after the one which sent it to the subroutine.

T

Teletext — information transmitted in the top section of a broadcast television picture. It requires a special set to decode it to fill the screen with text information. The BBC service is known as Ceefax, the ITV service as Oracle. Teletext messages can also be transmitted by cable, for example the Prestel service in Britain or The Source in the United States.

Teletype — a device like a typewriter which can send information and also receive and print it.

Terminal — a unit independent of the central processing unit. It generally consists of a keyboard and a cathode ray display.

Time Sharing — a process by which a number of users may have access to a large computer which switches rapidly from one user to another in sequence, so each user is under the impression that he or she is the sole user of the computer at that time.

Truth Table — a mathematical table which lists all the possible results of a Boolean logic operation, showing the results you get from various combinations of inputs.

U

UHF — Ultra High Frequency (300-3000 megaHertz).

Ultra Violet Erasing — Ultra violet light must be used to erase EPROMs (see EPROM).

V

Variable — a letter or combination of letters and symbols which the computer can assign to a value or a word during the run of a program.

VDU — an abbreviation for Visual Display Unit.

Volatile — refers to memory which 'forgets' its contents when the power is turned off.

W

Word — a group of characters, or a series of binary digits, which represent a unit of information and occupy a single storage location. The computer processes a word as a single instruction.

Word-Processor — a highly intelligent typewriter which allows the typist to manipulate text, to move it around, to justify margins and to shift whole paragraphs if necessary on a screen before outputting the information onto a printer. Word-processors usually have memories, so that standard letters and the text of letters, written earlier, can be stored.

BIBLIOGRAPHY

By Series Editor, Tim Hartnell

Usborne have released a number of very attractive books in their Usborne Computer Books series. Drawing on their vast experience in the field of producing low-priced, highly-coloured, attractive books for young readers, they've produced some books which will enlighten both young and not-so-young readers.

I'll look at three of their titles, three which cover just about the whole field of computer interests:

Information Revolution
(Lynn Myring and Ian Graham, Rigby).
Presenting an eminently readable introduction to the 'revolution' which covers such fields as computers (of course), text information services via the television screen, word processing, 'future phones' and satellite communications, *Information Revolution* is an ideal guide for the person who wants an easy-to-read introduction to the field.

Computer Jargon
(Corinne Stockley and Lisa Watts).
The tone of this book is set by the frontispiece, which has a number of odd little coloured robots sitting around a table laden with computer junk, pointing at each piece saying "This is a disk drive", "This is a digital tracer" (!) and "This is a printer".

Robotics — What Robots Can Do and How They Work
(Tony Potter and Ivor Guild).
This is definitely a candidate for the award of 'the longest

title of the year'. But it is very accurate. Don't be put off by the pretty pictures, as you'll soon discover this book has a lot of solid information. Topics covered include "What robots can and cannot do", "How arm robots work", "How to teach a robot" and "Build your own micro-robot"; this last section actually includes nine pages of circuit diagrams and all to build a little two-motor robot which, following a program typed into your micro, will run about the floor. Robotics is a field of the near future (with personal robots certain to be a bigger craze — when 'real robots' finally arrive — than computers will ever be).

Practise Your BASIC
(Gaby Waters and Nick Cutler).
You'll find this book — which predictably contains a number of exercises, puzzles and problems to solve by writing programs — should be useful in giving you a number of 'core problems' which will run on your computer and which can then be modified to take advantage of your system's special features. Program listings include 'Pattern Puzzles', 'Jumping Man', 'Horse Race', 'Word Editor' and 'Treasure Hunt', a mini-Adventure.

Help With Computer Literacy
(June St Clair Atkinson, Houghton Mifflin).
This is a large format book with an attractive cover, fairly priced for its 122 pages. It appears to be aimed at the early to middle years of secondary education, but contains a lot of material which those teaching younger children could easily adapt. Although it avoids the 'Gee Whiz' approach of the Usborne texts, it uses cartoons and diagrams to get its message across in an inviting manner.

The Interface Computer Encyclopedia
(Ken Ozanne, Interface Publications).
Compiled by a lecturer in mathematics at the NSW Institute of Technology, this work could perhaps be more accurately called 'The Computer Book of Lists', rather

than an encyclopedia. It contains annotated references to 'all' microprocessors, 'all' microcomputers, and 'most' microcomputing magazines. The inverted commas are there because — as the author admits candidly in his introduction — any such work is likely to be out of date even before it is published. Fat (445 pages) with minimalist presentation (the whole book is dumped directly from a word processor onto a dot-matrix printer) you'll find this a useful work if you want a ready reference to chips, computers and the ever-growing field of specialist magazines.

Computer Resource Book — Algebra
(Thomas Dwyer and Margot Critchfield, Houghton Mifflin).
Dwyer and Critchfield have clocked up an enviable string of successful computer books, and this one, part of a series, shows why. With simple, but valuable programs, the authors lead the reader (who can be a secondary student, or an instructor) through most of the phrases of the BASIC programming language which are common to all low-priced computers, and most educational time-sharing systems.

Apple II BASIC
(David Goodfellow, Tab Books Inc.).
Attractively packaged, this book is clearly laid out, with an abundance of example programs; it takes a commendable approach to the business of teaching programming, with the qualities of 'programming style' introduced without fanfare. In the crowded field of 'how to program your Apple' books, this one stands out. Much of the material presented is applicable to any microcomputer.

Pre-Computer Activities
(Dorothy Diamond, Hulton Educational).
This practical guide for teachers and parents can help make children familiar with essential computer processes and language before they have hands-on ex-

perience. The book contains a number of interesting activities, including investigating binary numbers using little lights, and working with cardboard 'calculators' before getting to the real thing. The discussion on computer graphics is enlivened by reference to the solid blocks which make up a 'Pacman' figure.

Word Processing Experience
(Janet Pigott and Roger Atkins-Green, Stanley Thornes Publishers Ltd.).
Designed for schools, but ideal for adapting if you'd like to increase your skill with a word processor (or simply because you'd like to see what word processors can do so you can write one for your own microcomputer), this book looks at the mechanics of word-processing, while passing on a great deal of useful information about word-processing techniques.

An Introduction to Micro-electronics and Microprocessor Systems
(G H Curtis and P G Wilks, Stanley Thornes Publishers Ltd.).
This work was written for junior college students and older school pupils, as well as for non-specialists who wanted a comprehensive — if dry — technical introduction to the subject. The going is not easy, but it's worth the effort. Topics covered include 'Logic', 'Programming the Microcomputer' and 'Analogue, Binary and Digital Systems'.

Computer Images — State of the Art
(Joseph Deken, Thames and Hudson).
This is a beautiful book, large and glossy, and packed with quality full-colour computer-generated (or, in some cases, computer-modified) images. The whole fascinating field of modern computer graphics is discussed — from television programme introductions using photographs which are colour-modified, twisted and tweeked, to the use of incredible high-resolution images in

simulators for flight training and tank manoeuvring. You'll read (and see) how computers are used to produce images, how these are used for education and communication, why 'art for art's sake' is a goal worth pursuing, and how computer images can evolve using processes uncannily akin to the processes by which groups of cells multiply and divide. If you want to see what can be done with high resolution graphics and when time, money and skill abound, you should get this book.

Computer Bluff
(Stephen Castell, Quartermaine House Ltd.).
A much more valuable book than its title indicates, it contains a lot of information on the what and how of computers, along with a generous dollop of computer jargon (or 'How to Cheat in Computer-Speak'). The style is gentle and amusing, with no appalling puns or excessive asides (such as 'didja get that joke, buster?'). A pleasant, painless book which you can digest, then give to a parent.

Penguin Books has moved into the computer field with enthusiasm. As well as a 'Getting the Most Out of Your...' series, they have a number of games books. Two which stand out are **The Penguin Book of VIC 20 Games** (Paul Copeland) and **The Penguin Book of Commodore 64 Games** (Robert Young and Paul Copeland). Priced at £4.95 each, these large format books include such programs as 'Space Venture', 'Oil Rig' and 'Red Alert'. Worth buying, even if you do not have a VIC or a Commodore 64, simply as a source of ideas for new programs to create on your own microcomputer.

Arcade Games for Your VIC 20 and **Arcade Games for Your Commodore 64** (Brett Hale, Corgi/Addison-Wesley) by contrast, are definitely only for those who have the machine specified. The programs are locked irrevocably to the computer named. Taking advantage of a number of machine-specific features (such as sprite

graphics on the 64), Brett has produced a selection of around 20 programs for each machine. Each one is listed twice, the first time for the joystick and the second time for the keyboard. Titles include 'Galaxy Robbers', 'Bullet Heads' and 'Yackman'.

CREATING ADVENTURE PROGRAMS

There are a number of books, some of which are aimed at computer owners, which will help you if you are one of the many, many computer games players who are interested in developing 'Adventure' and 'Dungeons' type programs. The place to start is with TRS Hobbies' **Dungeons and Dragons** (TM) Basic Set, which comes with the introductory rule book, Dungeon Dice (tm) and an instruction module, along with a sample scenario 'The Keep on the Borderlands'. If you're new to the field, you should start with this set to give you an idea how 'real life' Adventure programs are built up.

Additional information is provided by **Fantasy Role-Playing Games** (J. Eric Holmes, Hippocrene Books Inc.) which looks at the whole field and, despite some disparaging things to say on computer versions of such games, is worth looking for. Another overview of the field — with more sympathetic comments on the use of computers — is provided by **Dicing With Dragons — An Introduction to Role-Playing Games** (Ian Livingstone, Routledge and Kegan Paul), which includes a full 'solo Adventure', a review of the major games on the market, and a fascinating chapter on the pleasures and perils of being Dungeon Master in 'Playing God'.

Fantasy Wargaming (compiled Bruce Galloway, published Patrick Stephens) provides a complete unified system for 'historically accurate' (or at least in tune with the beliefs and circumstances of individuals in the peasant, feudal-economy times in which many Adventures are set) games. The fight, weapon and

monster tables alone are worth the book, as many of their ideas can easily be incorporated into your Adventures.

There are two computer Adventure books which you could get to help you in the fascinating area of producing Adventure games on your machine.

Creating Adventure Programs on Your Computer (Andrew Nelson, Interface Publications).
Written by the author of *More Games for Your VIC 20* and *Games for Your TI 99/4A*, in the Virgin Books games series, this book takes you through the task of developing an Adventure program of your own, concentrating more on the 'Loot and Pillage' school of gaming than the Scott Adams' 'solve this puzzle to advance' field. Three complete Adventure programs are included.

Write Your Own Adventure Programs for Your Microcomputer (Jenny Tyler and Les Howarth, Usborne) is a much quicker introduction to the field than Nelson's, but nevertheless packs a lot of valuable information into its 48 pages. Step-by-step instructions are provided for creating an Adventure from scratch. A complete program — 'Haunted House' — is included in the book.

The Age of Computers is the general title of four fine books produced by Wayland Publisher Limited. Each priced at £4.95, the books present a careful, but inviting, view of four aspects of the computer field, one on the history of computers and the others looking at specific areas of modern computer application. Each book is by Ian Litterick and Chris Smithers. The four titles are **The Story of Computers,** with Charles Babbage and Uncle Sir Clive Sinclair just inside the cover (and these two pictures accurately sum up the historical period covered by the book); **How Computers Work** (with chapter headings including 'Bits, Bytes and Binary', 'Decision-making by Transistor', and 'Talking With Computers'); **Computers in Everyday Life** (such things as 'Robots in the Home', 'Magnetic Money' and 'Medicine and the Disabled');

and **Computers and You** ('Computopia', 'Big Brother', 'War and Peace' and — a fascinating final chapter — 'Will Computers Need Us?').

Inside BASIC Games
(Richard Mateosian, Sybex).
This book is a slightly overwritten guide to understanding computer games. You'll learn how to write interactive programs in BASIC and how the principles of system development are applied to small computers. The book also looks at how the features of specific small computer systems have been supported in BASIC. If you can contend with the verbiage, you'll find this book well worthwhile.

1001 Things to Do With Your Personal Computer
(Mark Sawush, Tab Books).
Big and fat, and full of ideas, you'll find much here of interest to enlarge your computer horizons. The book tells you about writing music and stories with your computer, aiding a mechanic or a carpenter, solving simultaneous equations, astrology and much, much more.

Stimulating Simulations
(C.W. Engel, Hayden Book Company).
Here are 12 unique programs written in a good, general version of BASIC. The fascinating programs include 'Forest Fire', 'Rare Birds' and 'The Devil's Dungeon'. You're sure to enjoy playing those three, along with 'Diamond Thief', in which the computer decides who has committed the crime, then challenges you to discover which of the suspects is guilty. The material in this book is generally tightly programmed, and can be a helpful source of ideas to improve your own computer work.

The BASIC Handbook
(David A. Lien, Compusoft Publishing).
This is an encyclopedia of the BASIC language. It comes into its own when you find a program in a magazine or

book which you'd love to try, but are frustrated because it is written for another version of BASIC. Every BASIC word you've ever heard of (and many you may not have, such as NE, GOTO-OF and LE) is in here, along with a number of variations, one of which will almost certainly be on your machine.

BASIC Computer Games
(David Ahl, Creative Computing Press).
This is a classic work, still selling well despite the fact it was one of the first such books — if not *the* first — on the market. David Ahl has been in personal computers even before there were such things. Although several of the games are overly-dependent on the random number generator, you'll find there are many, many games you'll want to adapt and improve for your own computer.

How to Buy (and Survive) Your First Computer
(Carolee Nance Kolve, McGraw-Hill Book Company).
When is a business ready for a computer? How do you make an intelligent, informed choice among the hundreds of computers available? Will a computer improve a company's operations? Answers to these and a score of similar questions are in this book, which explains in detail what to consider before buying, how to select the right computer, and what to do after ordering the computer to ensure a successful installation. Ms Kolve has over 15 years computer experience (including a stint with IBM) and brings her experience to bear in a relatively easily-digestible guide.

Your First BASIC Program
(Rodnay Zaks, Sybex).
This book, liberally illustrated with large red dinosaurs in a variety of situations vaguely related to the text (one, for instance, as a cowboy getting tangled up in his ropes with the caption 'Be careful when looping'), is a gentle and worthwhile introduction to the not-so-secret secrets of programming in BASIC. When you want to move

beyond just typing in other people's programs from books and magazines, this may be a good place to start.

This bibliography was compiled by the series editor, Tim Hartnell, who has felt constrained not to recommend any of his own books. However, he asked us to mention two which could be of use and interest to you.

The first is **The Personal Computer Guide** (Virgin Books) which explains what a personal computer is, and answers questions like "Will it help my kids?", "What sort of games can we play on it?" and "What can I use it for in the home?". The book describes many of the most popular computers available today, with illustrations, technical specifications and other information to help you to choose the equipment best suited to your requirements. Also included is an introduction to BASIC programming, with details of programs suitable for use in the home, a list of suppliers and user clubs, and a guide to further reading. There are also chapters covering the personal computer's history and its future. When you're ready to upgrade, you'll find this book a good, unbiased, reference work which looks at the choices facing you.

Tim Hartnell's Giant Book of Computer Games.
Described by *Personal Computer News* as 'a good source of ideas', this 386-page book, published by Fontana, for £3.95, contains over 40 programs which will run with minimum modifications on most popular microcomputers. The games include chess (of a sort!), a 17K Adventure and 'Hyperwar'.

The Virgin Computer Games Series

GAMES FOR YOUR MSX COMPUTER

More than 20 challenging programs, each one especially written for the series and guaranteed to provide hours of entertainment.

The games include MOTORCYCLE STUNTMAN (fast-moving graphics as you take control of a motorbike and jump over a line of buses); INFERNO (the fate of people trapped inside a blazing skyscraper depends on you); MASTERCODE (can you solve the computer's code?); BAZOOKA (take aim and destroy the enemy's armoured tank); and STOCK MARKET (two players trade the stocks of four companies – in an attempt to make a million!)

GAMES FOR YOUR MSX COMPUTER will improve your programming skills as you follow the instructions to put each of the programs into your machine, and comes complete with a brief dictionary of computer terms, a selective bibliography and some hints on how to improve and extend the programs in the book.

Programs of originality and quality for all the family

VIRGIN BOOKS

ISBN 0 86369 092 0

United Kingdom £2.99