The Virgin Computer Games Series

Series Editor: Tim Hartnell

GAMES FOR YOUR SPECTRAVIDEO

£££££'s of entertaining games for only £2.95

Damon Pillinger
and Danny Olesh

Virgin

GAMES FOR YOUR SPECTRAVIDEO

By
Damon Pillinger
and
Danny Olesh

GAMES FOR YOUR SPECTRAVIDEO

By
Damon Pillinger
and
Danny Olesh

Virgin Books

First published in Great Britain in 1984 by Virgin Books Ltd, 61-63 Portobello Road, London W11 3DD.

ISBN 0 86369 047 5

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

Production services by Book Production Consultants, Cambridge.

Designed by Ray Hyden.

Illustrated by Sue Walliker.

Typeset by QV Typesetting.

Distributed by Arrow Books.

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).

DAMON PILLINGER AND DANNY OLESH — THE AUTHORS

Damon Pillinger is a 17-year-old windsurfer from Melbourne, Australia. He is studying maths and science and hopes to be a systems analyst in due course. He says his hobbies are tennis, rowing and girls.

Danny Olesh is a 19-year-old musician, also from Melbourne. During the day he works in a computer store, but at night creates pop music on an elaborate collection of electronic instruments. He is currently building interfaces to join his synthesiser to a computer. Danny's hobbies are music (naturally enough), rap dancing and travel.

SUE WALLIKER — THE ILLUSTRATOR

Sue Walliker is a freelance illustrator.

ACKNOWLEGEMENTS

Special thanks to Robert Polak of Gametronics for providing access to the computers used in the production of this book.

CONTENTS

Editor's Introduction 11
Author's Introduction 13
A Note on Graphics 15
Minefield 17
Road Race 20
Spectra-Hang 22
Lunar Engaging Module 27
Star Strike 29
Astrosmash 32
Towers of Doom 34
Night Fighter 44
Shoot 'em 46
Ness of Lochness 50
Edupack 53
Touch Typing 58
Calendar 61
Bomb Run 63
Nim 66
Graphics Demonstration – the Spectracular 69
Wizard's Hat 71
Stringy 72
Circle Draw 73
Time Gate 74
Pattern Wheel 75
Wave Form 76
Three-Dee 78
Star of Kazzbad 79
The Eye of Time 80
3D Graph 81
Creating Sound Effects on Your Spectravideo 82
How To Write Better Programs 87
Glossary 95
Bibliography 113

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
Series editor
London
March 1984

Authors' Introduction

The Spectravideo SV-328 and SV-318 are the first personal computers to supply MSX BASIC as standard on board. However, many computer manufacturers have endorsed the standard, so the programming knowledge you are developing can be transferred easily to new machines.

And the same goes for software. You can save the games in this book on disk or tape and if you ever decide to change computers, you should find they will run on your new MSX machine.

But MSX compatability is not the only, or even the main, reason for getting a Spectravideo. It is a great computer, easy to use, with flexible graphics and sound. You'll find a great deal of information in this book on both the graphics and sound which you can use to enhance your own programs.

We have tried to make the most of the commands and functions available, and to demonstrate just how quickly the machine works. We hope you enjoy playing the games as much as we enjoyed writing and play-testing them.

Damon Pillinger and Danny Olesh,
Melbourne,
March 1984

A NOTE ON GRAPHICS

Graphics have been used extensively in this book. We've used a coding system in the programs which should ensure you have little difficulty in entering the programs correctly.

If a string appears not to make normal grammatical sense — 10 PRINT "BBBPddPBB" for instance — then it is certain to be a string containing graphics. To enter such a string, do the following in each case:

UPPER CASE LETTERS
These must have the left graph key pressed and held down, *before* the key is pressed.

LOWER CASE LETTERS
These use the same system, except here you will need to use the right graph key.

MINEFIELD

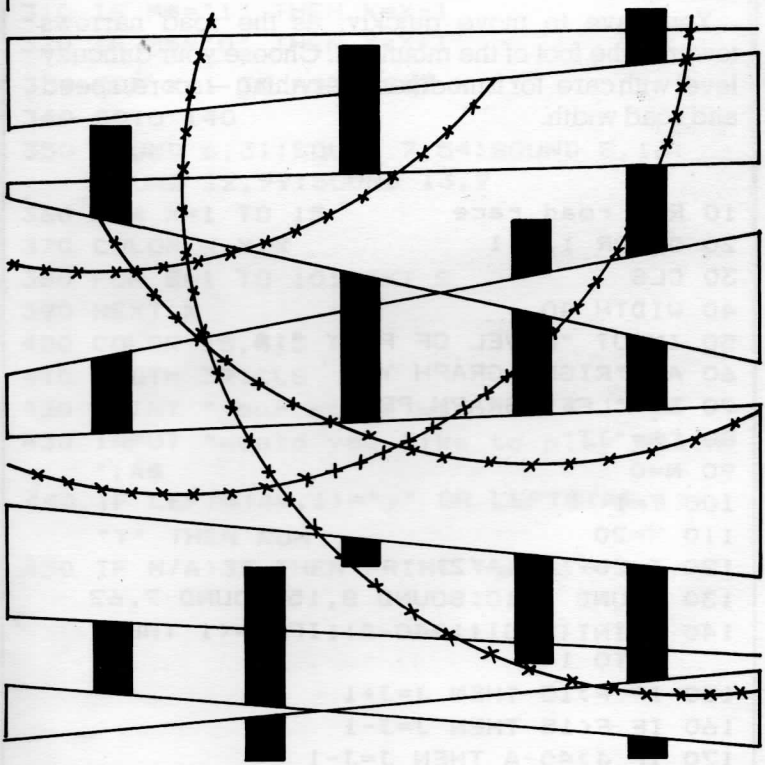
This action-packed game simulates a 'real war' situation. You control a man, represented by a cross, and you must traverse a deadly minefield at an ever-increasing speed to save your comrades at the front.

Move your man around using the 1 key for left and the 0 key for right. You move forward automatically. Your speed across the minefield depends on your level of play.

When you die, as you eventually will, your score will be displayed. A good score is around 30,000. Beware the explosive fence which surrounds the minefield.

```
10 REM MINE FIELD
20 CLS :INPUT "LEVEL  1=HARD
   300=EASY";LEV:IF LEV<1 THEN 20
30 CLS
40 PRINT "PRESS A KEY"
50 L=(RND(1))
60 IF INKEY$="" THEN 50
70 S=0
80 CLS
90 FORI=0TO31
100 LOCATEI,0:PRINT "p"
110 LOCATEI,21:PRINT "p"
120 NEXTI
130 FORI=1TO20
140 LOCATE0,I:PRINT "p";TAB(31);"p"
150 NEXT I
160 FORI=1TO65
170 LOCATE INT(RND(1)*29),INT(RND(1)
    *17)+1:PRINT "p"
180 NEXTI
190 M=20:X=INT(RND(1)*28)+1
200 REM
210 S=S+230
220 IF VPEEK ((X+(M-1)*40)+1)=233 THEN
    300
230 IF M=2 THEN 390
240 LOCATE X,M:PRINT "+"
250 FOR G= 1 TO 200:NEXT G
260 M=M-1
270 IF INKEY$="1" AND X>0 THEN X=X-1
280 IF INKEY$="0" AND X<30THEN X=X+1
290 GOTO 200
300 FOR I=1 TO10
310 LOCATE X,M:PRINT "*"
320 LOCATE X,M:PRINT "+"
330 LOCATE X,M:PRINT " "
340 NEXT I
350 CLS
360 S=S-300
```

```
370 LOCATE 0,10:PRINT "YOU CRASHED....
    SCORE=";S:LOCATE 3,15:PRINT
    "PLAY AGAIN?"
380 GOTO 440
390 FOR  I= 1 TO 25
400 PRINT
410 NEXT I
420 LOCATE 0,10:PRINT "WELL DONE YOU
    SCORED ";S:LOCATE 3,15:PRINT
    "GET READY.."
430 FOR L= 1 TO 1000:NEXT L:LEV=LEV-
    (LEV/3):GOTO 80
440 A$=INKEY$:IF A$="" THEN 440
450 IF A$="Y" THEN 30
460 END
```



ROAD RACE

You are the driver of a high-speed, turbo-charged racing car and you have to try and keep on the road as you race down a twisting mountain road.

Can you become King of the Mountain? Many have tried and failed in this quest, and when you play the game you'll see why.

You steer using the 1 key for left and the 0 key for right, as you attempt to keep the car's wheels on the road. The game is made harder by your increasing speed (you'll get an idea of how fast you're moving by the sound the car makes).

You have to move quickly, as the road narrows towards the foot of the mountain. Choose your difficulty level with care, for it modifies everything — score, speed and road width.

```
10 REM road race
20 COLOR 1,3,1
30 CLS
40 WIDTH 40
50 INPUT "LEVEL OF PLAY ";A
60 A$="RIGHT GRAPH Y."
70 B$="LEFT GRAPH PP."
80 C$="TT"
90 N=0
100 T=1
110 X=20
120 J=20-INT(A/2)
130 SOUND 1,10:SOUND 8,15:SOUND 7,62
140 F=INT(RND(1)*40-A):IF F=<1 THEN
    GOTO 140
150 IF F>15 THEN J=J+1
160 IF F<15 THEN J=J-1
170 IF J>40-A THEN J=J-1
```

```
180 IF J<1 THEN J=J+1
190 LOCATE J,21:PRINT A$:LOCATE J+A,21
    :PRINT  A$
200 PRINT
210 N=N+1:LOCATE 0,0:PRINT N
220 LOCATE X-4,7:PRINT "          "
230 LOCATE X,8:PRINT B$
240 LOCATE X,9:PRINT C$
250 IF N=100 THEN D=2:A=A-1:SOUND 1,8
260 IF N=175 THEN D=1:A=A-1:SOUND 1,7
270 IF N=250 THEN D=0 :A=A-1:SOUND 1,5
280 IF N=350 THEN A=A-2:SOUND 1,3
290 IF VPEEK(10*40+X)<> 0 OR VPEEK(10*
    40+X+1)<> 0 THEN GOTO 350
300 M$=INKEY$
310 IF M$="1" THEN X=X-1
320 IF M$="0" THEN X=X+1
330 FOR K=1 TO A*3:NEXT K
340 GOTO 140
350 SOUND 6,31:SOUND 7,54:SOUND 8,16:
    SOUND 12,99:SOUND 13,9
360 FOR X=1 TO 15
370 COLOR X,X,X
380 FOR S=1 TO 10:NEXT S
390 NEXT X
400 COLOR 15,4,5
410 WIDTH 39:CLS
420 PRINT "your score was ";N/A
430 INPUT "would you like to play again
    ";A$
440 IF LEFT$(A$,1)="y" OR LEFT$(A$,1)=
    "Y" THEN RUN
450 IF N/A>35 THEN PRINT "good driving"
```

SPECTRA-HANG

In this 'educational' game, your life depends on the whim of the sadistic tyrant king. The king has given you one chance of survival: you must correctly guess a word which he, or a partner, has chosen. Your life is over if you do not guess it within 10 moves.

If you fail, the king shouts out the word as the trapdoor beneath your feet flies open. At least you'll die educated!

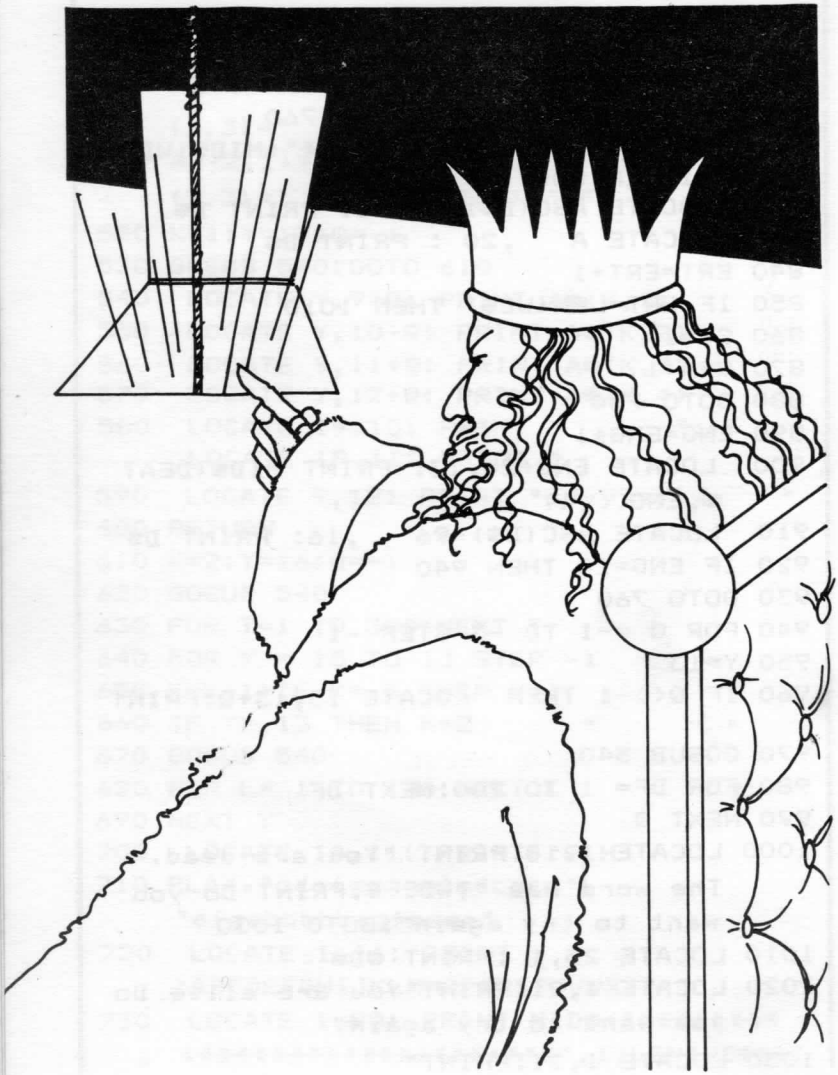
```
10 REM SPECTRAHANG
20 DIM A$(2,4)
30 CLS
40  LOCATE 1,1: PRINT "PRESS A KEY"
50 AS=RND(1)
60 IF INKEY$="" THEN 50
70 CLS
80 DEAT$="I#AM#DEAD"
90 ENG=0
100 CLS
110 DATA black,hole,farm,attack,unit,
120 DATA process,learning,learn,general
    ,november,spectrum,examine,straight
130 DATA algorithms,plausibility,
    indicator,maximum,minimum,operator
    number,arcade
140  COLOR 13,1,2
150  LOCATE 1,1: PRINT "hello welcome
     to SPECTRAHANG",,,"YOUR life is
     hanging from a thread."'"The only
     way to excape is to guess a  word
     that is thought of by the king or
     a partner."
```

```
160  LOCATE 1,6: PRINT "DO YOU WANT
     THE KING TO MAKE A WORD"
170  LOCATE 1,8:INPUT A$
180 IF LEFT$(A$,1)="Y" OR LEFT$(A$,1)
   ="N" THEN 230
190 IF LEFT$(A$,1)="y" THEN 240
200  LOCATE 1,12:PRINT "Will your
     partner now input a word
     WORD=": LOCATE 7,13:INPUT WD$
210 IF LEN (WD$)>17 THEN 200
220 GOTO 300
230  LOCATE 1,21: PRINT "Please enter
   CAPS LOCK":GOTO 170
240 WO=RND(1)*22+1
250 RESTORE 110
260 FOR G= 1 TO WO
270 READ A$
280 NEXT G
290 WD$=A$
300 D$="--------------------------------
      ------------------------------"
310 CLS
320  LOCATE 1,1: PRINT "SPECTRAHANG"
330  LOCATE 1,2: PRINT "-----------"
340  LOCATE 10,4: PRINT "uxxxxi","
           z    z"
350  LOCATE 10,6: PRINT "z      ","
           z      "
360  LOCATE 10,8: PRINT "z      ","
           z      "
370  LOCATE 10,10: PRINT "z      ","
           z      "
380  LOCATE 9,12: PRINT "syyyyyyyya",
390 A$(1,1)=" o ":A$(1,2)="ByN ":A$(1,
    3)=" y ":A$(1,4)="gH "
400 A$(2,1)=" o ":A$(2,2)="FyN ":A$(2,
    3)=" y ":A$(2,4)="FF "
410 FOR Y= 30 TO 19 STEP -1
```

```
420 FOR K= 1 TO 2
430  LOCATE Y,9: PRINT A$(K,1)
440  LOCATE Y,10: PRINT A$(K,2)
450  LOCATE Y,11: PRINT A$(K,3)
460  LOCATE Y,12: PRINT A$(K,4)
470 FOR H= 1 TO 50:NEXT H
480 NEXT K
490 NEXT Y
500 A$(1,1)="  o ":A$(1,2)=" FyH ":A$
    (1,3)="  y ":A$(1,4)=" VF "
510 A$(2,1)="  o ":A$(2,2)=" FyN ":A$
    (2,3)="  y ":A$(2,4)=" FF "
520 K=1:Y=17:Q=-1
530 GOSUB 540:GOTO 610
540  LOCATE Y,9+Q: PRINT A$(K,1)
550  LOCATE Y,10+Q: PRINT A$(K,2)
560  LOCATE Y,11+Q: PRINT A$(K,3)
570  LOCATE Y,12+Q: PRINT A$(K,4)
580  LOCATE 19,10: PRINT "      ":
     LOCATE 19,11: PRINT "        "
590  LOCATE 9,12: PRINT "syyyyyyyya   "
600 RETURN
610 K=2:Y=16:Q=-1
620 GOSUB 540
630 FOR T=1 TO 300:NEXT T
640 FOR Y = 15 TO 13 STEP -1
650 K=K+1:IF K= 3 THEN K=1
660 IF Y= 13 THEN K=2
670 GOSUB 540
680 FOR L= 1 TO 130:NEXT L
690 NEXT Y
700  LOCATE 14,11: PRINT "F H"
710 PLAY "cdefgggggfedcccc",
    "efgabbbbbagfeeee"
720  LOCATE 1,16: PRINT
    "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
730  LOCATE 1,20: PRINT MID$("********
     **********************",1,LEN(WD$))
```

```
740  LOCATE 1,21: PRINT MID$(D$,1,
     LEN(WD$))
750  LOCATE 1,18: PRINT "CHOOSE A
     LETTER"
760  LOCATE 20,21: INPUT D$:IF LEN(D$)
     >1 THEN 760
770 Q=1
780 A=INSTR(Q,WD$,D$)
790 IF (A= 0) AND (SW<>1) THEN 890
800 IF A=0 THEN  SW=0:GOTO 760
810 WD$=MID$(WD$,1,A-1  )+"*"+MID$(WD$,
    A+1,LEN(WD$))
820  LOCATE ASC(D$)-96,16: PRINT D$
830  LOCATE A   ,20 : PRINT D$
840 ERT=ERT+1
850 IF ERT=LEN(WD$) THEN 1010
860 SW=1
870 Q=A+1
880 GOTO 780
890 ENG=ENG+1
900  LOCATE ENG+24,13: PRINT MID$(DEAT
     $,ENG  ,1)
910  LOCATE ASC(D$)-96   ,16: PRINT D$
920 IF ENG=10 THEN 940
930 GOTO 760
940 FOR Q =-1 TO -3 STEP -1
950 Y=13
960 IF Q<>-1 THEN  LOCATE 13,13+Q:PRINT
    "       "
970 GOSUB 540
980 FOR DF= 1 TO 200:NEXT DF
990 NEXT Q
1000 LOCATE 1,18:PRINT "You are dead.
     The word was ";WD  $:PRINT"Do you
     want to try again":GOTO 1030
1010 LOCATE 25,1 :PRINT WD$
1020 LOCATE 1,21:PRINT"You are alive. Do
     you want to try again"
1030 LOCATE 1,21:PRINT"                "
```

```
1040 A$=INKEY$
1050 IF A$="" THEN 1040
1060 ERT=0
1070 IF A$="y" THEN GOTO 30
```



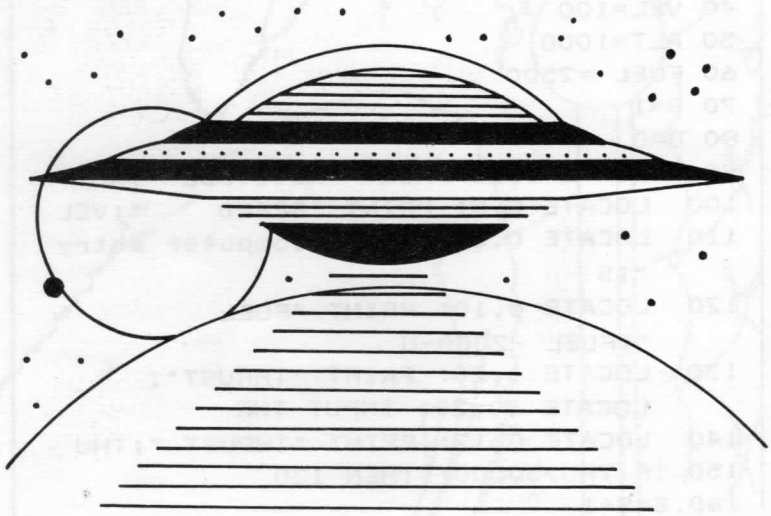
LUNAR ENGAGING MODULE

As you look through the porthole in your lunar module you see the earth approaching very fast. You try to log on to your computer for assistance in landing, but the Spectra does not respond...

You must therefore land manually. The aim of the game is to land your module with a velocity of less than five miles an hour. Certain death is yours if you fail. To slow down your craft increase the thrust.

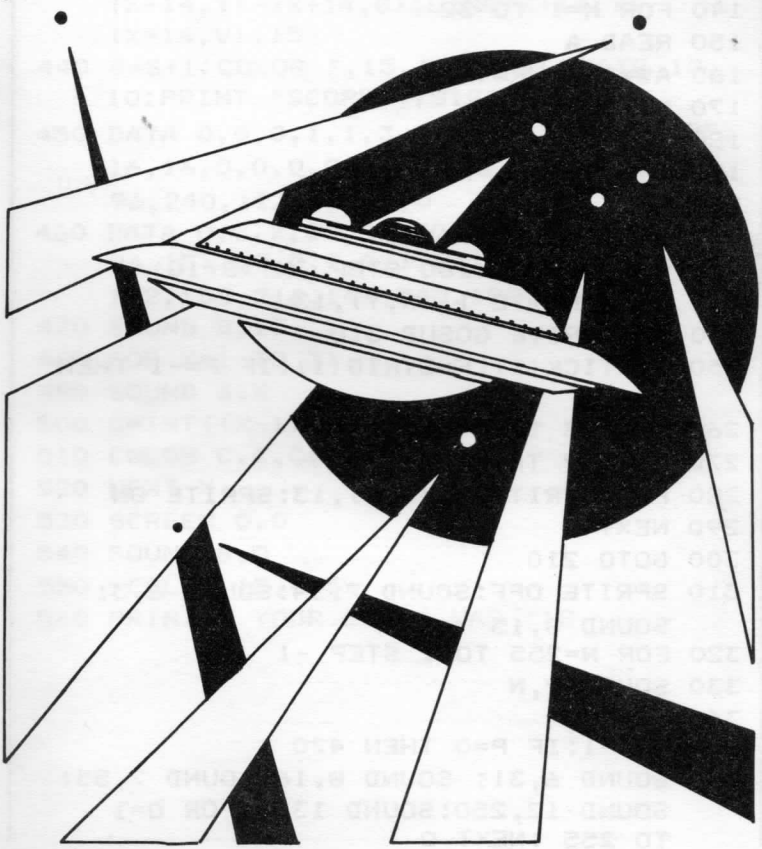
```
10 REM LEM
20 COLOR 15,1,1
30 CLS
40 VEL=100
50 ALT=1000
60 FUEL =2500
70 S=1
80 U=0
90  LOCATE 0,6: PRINT "ALTITUDE ";ALT
100  LOCATE 0,8: PRINT "SPEED    ";VEL
110  LOCATE 0,3: PRINT "computer entry
     ";S
120  LOCATE 0,10: PRINT "FUEL
     ";FUEL -2000-U
130  LOCATE 1,20: PRINT "THRUST":
     LOCATE 10,20: INPUT THR
140  LOCATE 0,12: PRINT "THRUST ";THU
150 IF THU>50000! THEN 130
160 S=S+1
```

```
170 U=THU/50000!*50
180 FUEL=FUEL-U:IF FUEL<=0 THEN 280
190 VEL=VEL-((THU/FUEL)-2)
200 ALT=ALT-VEL
210  IF ALT<=0 AND VEL<5 THEN 250
220  IF ALT<=0  THEN 280
230 SOUND 6,RND(1)*31:SOUND 7,54:SOUND
    8,10
240 GOTO 90
250  LOCATE 3,18: PRINT "WELL DONE
    COMMANDER"
260 PLAY"t255o6cdefgo4go6gggfedcccc"
270 END
280  LOCATE 3,18: PRINT "YOU CRASHED .
     EX-COMMANDER"
290 PLAY "t255eeo5eeo4eeo5eeo4eeo5eeo
    2cccccc":FOR G=1 TO3000:NEXT
   G:SOUND 6,31:
 SOUND 8,16:SOUND 7,54:SOUND 0,0:SOUND
   1,0:SOUND 12,200:SOUND 13,9
300 LOCATE 3,20:PRINT "YOU CREATED A
     CRATER ";FUEL/10."MILES WIDE "
```



STAR STRIKE

You must be quick to move your ship, or you will be wiped out by cosmic fire balls the size of your ship. Use your ammunition wisely — the number of cosmic balls increases as you expend more ammo — and try not to break your joystick in the battle for survival.



```
10 REM STAR STRIKE
20 SPRITE ON
30 X=128:Y=160
40 S=1
50 P=2
60 SCREEN 1,3
70 COLOR 1,15,15
80 CLS
90 FOR X=1 TO 32
100 READ A
110 S$=S$+CHR$(A)
120 NEXT X
130 SPRITE$(1)=S$
140 FOR X=1 TO 32
150 READ A
160 A$=A$+CHR$(A)
170 NEXT X
180 FOR N=2 TO 11
190 SPRITE$(N)=A$
200 NEXT N
210 J=INT(RND(1)*250)
220 FOR V=0 TO 180 STEP INT(S/10)+1
230 PUT SPRITE 1,(X,Y),13
240 ON SPRITE GOSUB 310
250 D=STICK(1):F=STRIG(1):IF F=-1 THEN
    410
260 IF D=3 THEN X=X+1
270 IF D=7 THEN X=X-1
280 PUT SPRITE 2,(J,V),13:SPRITE ON
290 NEXT V
300 GOTO 210
310 SPRITE OFF:SOUND 7,54:SOUND 6,3:
    SOUND 8,15
320 FOR N=255 TO 1 STEP -1
330 SOUND 2,N
340 NEXT N
350 P=P-1:IF P=0 THEN 470
360 SOUND 6,31: SOUND 8,16:SOUND 7,55:
    SOUND 12,250:SOUND 13,9:FOR O=1
    TO 255 :NEXT O
```

```
370 FOR C=1 TO 15
380 COLOR 1,C,C:CLS:FOR DD=1 TO 50:
    NEXT DD
390 NEXT C
400 GOTO 210
410 IF J=X OR J=X-1 OR J=X+1 THEN GOTO
    430
420 SOUND 6,3:SOUND 7,55:SOUND 8,16:
    SOUND 12,20:SOUND 13,9:LINE
    (X+14,Y)-(X+14,0):LINE(X+14,Y)-
    (X+14,0),15:GOTO 250
430 COLOR 6,6,6:SOUND 7,55:SOUND 6,31:
    SOUND 12,250:SOUND 13,9:LINE
    (X+14,Y)-(X+14,0):LINE(X+14,Y)-
    (X+14,V),15
440 S=S+1:COLOR 1,15,15:CLS:LOCATE 10,
    10:PRINT "SCORE=";S:GOTO 210
450 DATA 0,0,0,1,1,3,7,5,4,13,13,31,29,
    16,16,0,0,0,0,0,0,128,192,64,64,96,
    96,240,112,16,16,0
460 DATA 0,0,7,30,42,50,29,41,52,22,29,
    26,15,7,0,0,0,0,192,176,24,172,188,
    172,152,216,144,240,192,128,0,0
470 SOUND 8,15
480 FOR X=1 TO 31
490 SOUND 6,X
500 C=INT((X-1)/2)
510 COLOR C,C,C:CLS
520 NEXT X
530 SCREEN 0,0
540 SOUND 8,0
550  COLOR 15,4,5
560 PRINT " YOUR SCORE WAS ";S
```

ASTROSMASH

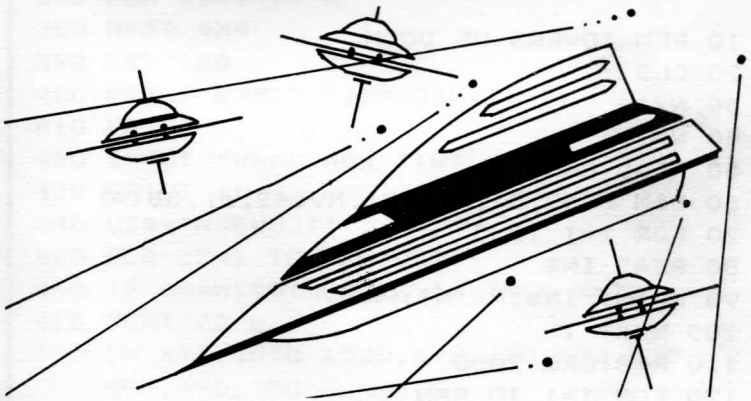
"Oh no...you've led us into an asteroid belt."
"But Captain...I didn't know it was ahead of us."
"Fool..."
As the argument increases in ferocity you, as second in command, must get the USS Compromise out of danger.

To add to your woes, the good doctor reports that freighter ships have been sighted on the Galactic Scanner, approaching at an ever-increasing speed. The freighters are wider and faster than the asteroids.

Use the 1 and 0 keys to steer through the hazards, or you'll go where every man has gone before...star date 11/1/2001.

```
10 CLS
20 COLOR 15,1,1
30 INPUT "TIME DELAY ";Z
40 FOR X=0 TO 13 :SOUND X,0
50 SOUND 6,17
60 SOUND 7,54
70 SOUND 8,15
80 SOUND 11,0
90 SOUND 12,10
100 SOUND 13,16
110 A$="RIGHT GRAPH F."
120 N=0
130 A=0
140 B=0
150 C=0
160 D=0
170 T=1
180 X=12
190 C=0
200 R=INT(RND(1)*27)
210 LOCATE R,21:PRINT A$
220 PRINT
```

```
230 PRINT
240 N=N+T
250 IF N=100 THEN LET  A$="RIGHT GRAPH
    FF."
260 IF N=104 THEN T=2
270 E=D
280 D=C
290 C=B
300 B=A
310 A=R
320 LOCATE X-3,9:PRINT "          "
330 LOCATE X,11:PRINT "RIGHT GRAPH WQ "
340 IF X>=E-1 AND X<=E+T THEN GOTO 400
350 M$=INKEY$
360 IF M$="1" THEN X=X-T
370 IF M$="0" THEN X=X+T
380 FOR K=1 TO Z:NEXT K
390 GOTO 200
400 PRINT "CRASH....."
410 SOUND 6,31:SOUND 8,16:SOUND 12,99:
    SOUND 13,9
420 PRINT "YOUR SCORE WAS ";N
430 INPUT "PLAY AGAIN ";A$
440 IF A$="Y" OR A$="y" OR A$="YES" OR
    A$="yes" THEN RUN
450 PRINT "BETTER LUCK
    NEXT TIME."
```



TOWERS OF DOOM

In this Adventure program you must try to save a princess trapped in a castle by an evil wizard. Full instructions are provided within the program, which nevertheless poses a somewhat difficult problem to solve.

Many monsters, including goblins and ghouls, live in the tower and will make things harder for you. There are 42 rooms in the tower, all of which contain either a treasure or a trap.

Your commands are:

N — north
S — south
E — east
W — west
T — take

You would be well advised to try and draw a map of the tower as you wander around. It will make things much simpler for you.

```
10 REM TOWERS OF DOOM
20 CLS
30 N=1
40 N=1
50 READ RMN,T
60 DIM B(42,3),A$(42),MV(42,6),OB(4)
70 FOR I=1 TO 5
80 READ IN$
90 PRINT IN$:PRINT:PRINT
100 NEXT I
110 RESTORE 2000
120 FOR I=1 TO RMN
```

```
130 READ A$(I)
140 NEXT I
150 FOR I=1 TO T
160 READ HZ$(I)
170 NEXT I
180 FOR I=1 TO RMN
190 READ MV(I,1),MV(I,2),MV(I,3),
    MV(I,4),MV(I,5),MV(I,6)
200 NEXT I
210 REM
220 R=INT(RND(1)*26+9)
230 B(R,1)=RMN
240 FOR I=1 TO (RMN-1)
250 R=INT(RND(1)*RMN+1)
260 IF B(R,1)<>0 THEN 250
270 B(R,1)=I
280 NEXT I
290 FOR I=1 TO T
300 R=INT(RND(1)*34+3)
310 IF B(R,3)<>0 THEN 300
320 B(R,3)=I
330 NEXT I
340 PRINT "Press enter to start":
    INPUT Q$
350 PRINT ::::
360 RESTORE 1600
370 FOR RD=1 TO N
380 READ RM$
390 NEXT RD
400 PRINT RM$;" ";A$(B(N,1))::
410 PRINT
420 INPUT "What now ";Q$
430 PRINT ::
440 VC$="NSEWUDT"
450 FOR ZZ=1 TO LEN(VC$)
460 IF Q$=MID$(VC$,ZZ,1) THEN X=ZZ
470 NEXT ZZ
480 ON X+1 GOTO 1520,490,490,490,490,
    490,490,750
```

```
490 IF MV(N,X)<>99 THEN 520
500 PRINT "You can't go that way"::
510 GOTO 410
520 N=MV(N,X)
530 FLT=0
540 IF N=1 THEN 970
550 RESTORE 1600
560 FOR RD=1 TO N
570 READ RM$
580 NEXT RD
590 PRINT RM$;" ";A$(B(N,1)):PRINT:
    PRINT
600 FL2=0
610 IF B(N,1)<>RMN THEN 720
620 PRINT "Behind a locked door"::
630 FOR I=1 TO 3
640 FOR J=1 TO 3
650 IF OB(I)<>J THEN 670
660 CN=CN+1
670 NEXT J
680 NEXT I
690 IF CN=3 THEN 1490
700 CN=0
710 FLT=0
720 IF B(N,3)=0 THEN 410
730 PRINT ::::
740 GOTO 1190
750 IF FL2=1 THEN 1470
760 IF FLT<>1 THEN 790
770 PRINT "You don't have three keys"::
780 GOTO 410
790 FOR I=1 TO 3
800 IF OB(I)=0 THEN 840
810 NEXT I
820 PRINT "You can only carry three
    objects - leave one behind"
830 GOTO 880
840 OB(I)=B(N,1)
850 B(N,1)=9
```

```
860 PRINT "It's yours"::
870 GOTO 410
880 PRINT "1. ";A$(OB(1)),"2. ";A$
    (OB(2)),"3. ";A$(OB(3)),"4. ";A$
    (B(N,1))::
890 INPUT "Leave which number ";Q
900 PRINT ::
910 IF Q>4 THEN 890
920 IF Q=4 THEN 410
930 TH=OB(Q)
940 OB(Q)=B(N,1)
950 B(N,1)=TH
960 GOTO 410
970 PRINT "Leaving so soon? Value of
    items taken is";
980 FOR I=1 TO 3
990 IF OB(I)<RMN-T+1 THEN 1010
1000 VL=VL+OB(I)*1000
1010 NEXT I
1020 PRINT VL::::
1030 INPUT "Like to try again ";Q$
1040 IF MID$(Q$,1,1)="N" THEN 1180
1050 CN=0
1060 N=1
1070 CLS
1080 FOR I=1 TO RMN
1090 B(I,1)=0
1100 B(I,2)=0
1110 OB(1)=0
1120 OB(2)=0
1130 OB(3)=0
1140 VL=0
1150 B(I,3)=0
1160 NEXT I
1170 GOTO 220
1180 END
1190 PRINT HZ$(B(N,3))::
1200 FOR H=1 TO 3
```

```
1210 IF (OB(H)<30)+(OB(H)>RMN-T)THEN
     1230
1220 RF=RF+OB(H)-30
1230 NEXT H
1240 INPUT "Run or fight ";Q$
1250 PRINT ::
1260 IF MID$(Q$,1,1)="R" THEN 1290
1270 IF MID$(Q$,1,1)="F" THEN 1390
1280 GOTO 1240
1290 CH=INT(RND(1)*3+1)
1300 IF CH<>1 THEN 1330
1310 PRINT "You were caught!"::
1320 GOTO 1420
1330 PRINT "You've escaped but lost
     everything you carried."::
1340 OB(1)=0
1350 OB(2)=0
1360 OB(3)=0
1370 FL2=1
1380 GOTO 410
1390 CH=INT(RND(1)*RF)
1400 RF=0
1410 IF CH>(B(N,3)*2) THEN 1440
1420 PRINT "You lost!"::::
1430 GOTO 1030
1440 PRINT "You won!"::::
1450 B(N,3)=0
1460 GOTO 410
1470 PRINT "No time for that. You're
     too busy running"
1480 GOTO 410
1490 PRINT "You've rescued ";A$(RMN);
     "Now you must escape with any
      treasure you carry!"::::
1500 OB1)=RMN
1510 GOTO 410
1520 PRINT "Use N S E W U D or T
     please"::
1530 GOTO 410
1540 DATA 40,5
```

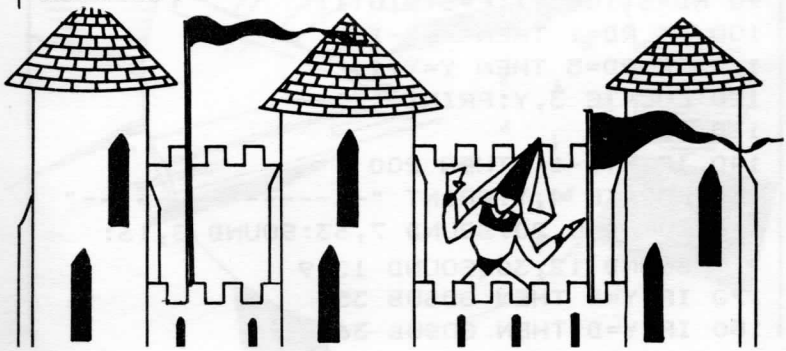
```
1550 DATA TOWERS OF DOOM
1560 DATA You are about to explore the
     evil      wizard's castle. If you
     can you must    rescue the captive
     princess.
1570 DATA To do this you must first
     find three    keys. Commands you
     can use are
1580 DATA N (north) S (south) E (east)
     W (west)  U (up) D (down) T (take).
1590 DATA Around the castle you will
     find weaponswhich will help you to
     defeat hazards  encountered
     (E.G. SWORD) !
1600 DATA You stand at the entrance
     gate to the castle grounds. To the
     north is the     castle. Nearby is
1610 DATA You face the castle to the
     north.        Beneath your feet a
     grille. In front    the draw-bridge.
     Nearby is
1620 DATA You are to the west of the
     castle. On  the ground is
1630 DATA You are behind the castle.
     A guardhouse is to the south.
     On the ground is
1640 DATA You are to the east of the
     castle.       Lying against the wall
     is
1650 DATA You are in the entrance hall
     doors leadN. E. and W. stairs lead
     up. In a        cupboard is
1660 DATA West dining hall door leads
     east. In    here is
1670 DATA The library. A door to the
     East. In     hear is
1680 DATA The west kitchen. A door to
     the east.  A pantry to the west.
```

```
      In here is
1690 DATA The scullery. Doors on all
     sides.      stairs lead down.
   In one corner is
1700 DATA A narrow passage way. Doors
     on all      sides. At one end is
1710 DATA The east kitchen. A single
     door to the west. On the eastern
     side is
1720 DATA an office. A door to the
     west. On the  eastern side is
1730 DATA The east dining hall. Doors
     to the  east and west. In here is
1740 DATA You are at the bottom of the
     stairs.  Doors to E. W. and S.
     Under the stairs is
1750 DATA The gaurd room. A single
     door to the   east. In here is
1760 DATA A store room. One door to
     the west. In a cupboard is
1770 DATA A short hall way. Doors on
     all sides.  In here is
1780 DATA Dungeon. A door to the east.
     In here is
1790 DATA The torture chamber. A door
     to the  west. In an ante-room is
1800 DATA A dungeon. The door to the
     east.          In here is
1810 DATA A passage-way. Doors to n. e.
     and w.   In here
1820 DATA A dungeon. A door to the
     west. In here is
1830 DATA The pantry. The door to the
     E. In one  corner is
1840 DATA An ante-room. The door to the
     west.     A tunnel leads downwards.
     In here is
1850 DATA You are at the top of the
```

```
    stairs. Thereare doors to the N. E.
    and W. In a      cupboard is
1860 DATA A hallway. Doors on all four
     sides.    In a cupboard is
1870 DATA A narrow passage way. Doors
     to the E.  W. and S. At one end is
1880 DATA An attic room. A door to the
     east. In  a cupboard is
1890 DATA An attic room. A door to the
     west. In  here is
1900 DATA A servants room. A single
     door leads   east. In here is
1910 DATA A servants room. A door to
     the west.    In a cupboard is
1920 DATA A bedroom. A door to the
     east. Under    the bed is
1930 DATA The gallery. A door to the
     west.          At one end
1940 DATA The drawbridge. To the north
   a closed  portcullis. At one side is
1950 DATA A guardhouse. Doors to the
     N. and S.    In here is
1960 DATA You are in a field. Sheep
     are grazing  nearby. Under a hedge
     is
1970 DATA You are in a copse. Under
     the trees is
1980 DATA You are deep into the woods.
     In a       clearing is
1990 DATA You are at the base of a
     rocky outcrop.In a cave is
2000 DATA a large key
2010 DATA a small key
2020 DATA a brass key
2030 DATA a cloak
2040 DATA a stone vase
2050 DATA a bottle
2060 DATA a clump of moss
```

```
2070 DATA some rusted armour
2080 DATA a dead rat
2090 DATA a wooden leg
2100 DATA a mouse
2110 DATA a cat
2120 DATA a spider
2130 DATA a glass eye
2140 DATA a cup
2150 DATA a skull
2160 DATA a bell
2170 DATA a body
2180 DATA a bucket
2190 DATA a ring
2200 DATA a skeleton
2210 DATA a goblet
2220 DATA a cobweb
2230 DATA a dragons tooth
2240 DATA a frog
2250 DATA a lamp
2260 DATA a toad
2270 DATA a broom
2280 DATA a stone
2290 DATA a club
2300 DATA a staff
2310 DATA a dagger
2320 DATA an axe
2330 DATA a shield
2340 DATA a sword
2350 DATA a diamond
2360 DATA a bag coins
2370 DATA a ruby ring
2380 DATA a treasure chest
2390 DATA THE CAPTIVE PRINCESS!
2400 DATA GUARD DOGS
2410 DATA A GUARD
2420 DATA A GOBLIN
2430 DATA A TROLL
2440 DATA THE EVIL WIZARD!
2450 DATA 2,1,1,1,99,99,35,1,5,3,99,25,
```

```
      4,2,99,37,99,99
2460 DATA 38,36,5,3,99,99,4,2,40,99,99,
      99,11,99,14,7,26,99
2470 DATA 99,99,6,99,99,99,99,99,11,99,
      99,99,99,99,10,24,99,99
2480 DATA 36,11,12,9,99,15,10,6,13,8,
      99,99,99,99,99,10,99,99
2490 DATA 99,99,99,11,99,99,99,99,25,6,
      99,99,99,18,17,16,10,99
2500 DATA 99,99,15,99,99,99,99,99,99,
      15,99,99,15,22,20,19,99,99
2510 DATA 99,99,18,99,99,99,99,99,99,
      18,99,99,99,99,22,99,99,99
2520 DATA 18,99,23,21,99,99,99,99,99,
      22,99,99,99,99,9,99,99,99
2530 DATA 99,99,99,14,99,2,27,99,34,33,
      99,6,28,26,32,31,99,99
2540 DATA 99,27,30,29,99,99,99,99,28,
      99,99,99,99,99,99,28,99,99
2550 DATA 99,99,27,99,99,99,99,99,99,
      27,99,99,99,99,26,99,99,99
2560 DATA 99,99,99,26,99,99,99,2,99,99,
      99,99,4,10,99,99,99,99
2570 DATA 38,1,3,99,99,99
2580 DATA 39,4,40,37,99,99
2590 DATA 99,38,99,99,99,99
2600 DATA 38,1,99,5,99,99
2610 END
```



NIGHT FIGHTER

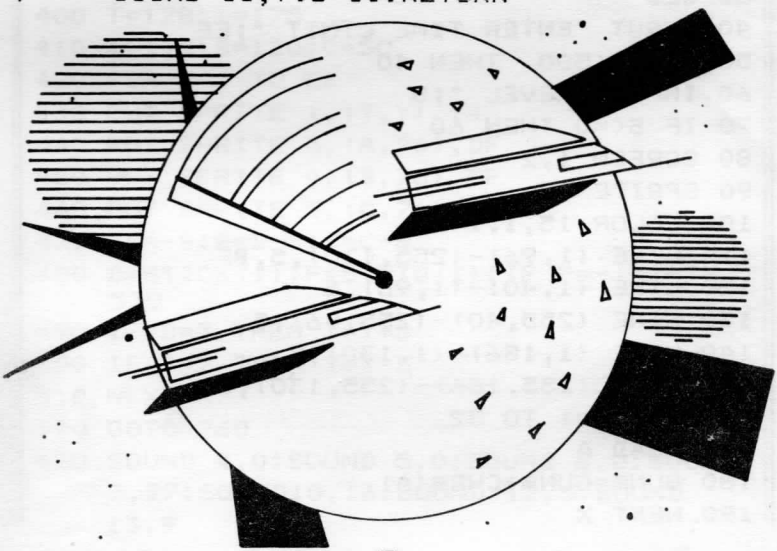
Warning...warning...Incoming aliens have threatened to destroy the earth. How long can you delay their almost inevitable victory?

You have only a flickering, and somewhat faulty, radar screen to warn you of their approach. And 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.

```
5 REM NIGHT FIGHTER
10 CLS
20 INPUT "level ";LEV
30 A=0
40 B=INT (RND(1)*10)+5
50 D=INT (RND(1)*15)+8
60 C=32
70 Y=10
80 REM
90 RD=STICK(1):F=STRIG(1)
100 IF RD=1 THEN Y=Y-1
110 IF RD=5 THEN Y=Y+1
120 LOCATE 3,Y:PRINT "?"
130 CLS
140 IF F<>-1  THEN 200
150 LOCATE 4,Y:PRINT "----------------"
160 SOUND 0,20:SOUND 7,63:SOUND 8,16:
    SOUND 12,30:SOUND 13,9
170 IF Y=B THEN GOSUB 350
180 IF Y=D THEN GOSUB 360
```

```
190 IF B=30 AND D=30 THEN 310
200 PC=PC+1:IF PC<= LEV THEN 80 ELSE
    PC=0
210 IF B<30 THEN B=B+INT(RND(1)*3-1)
220 IF D<30 THEN D=D+INT(RND(1)*3-1)
230 C=C-1
240 CLS
250 IF B<30 THEN LOCATE C,B:PRINT"?;_"
260 IF D<30 THEN LOCATE C,D:PRINT"?;="
270 IF C>2 THEN 80
280 PRINT "THE WORLD HAS BEEN
    DEMOLISHED"
290 PRINT "NIGHT FIGHTER HAD ";A;"
    ATTEMPTS"
300 STOP
310 A=A+1
320 FOR GL=1 TO 5:NEXT GL
330 LEV =LEV-LEV/3
340 GOTO 40
350 SOUND 6,31:SOUND 7,54:SOUND 12,10:
    SOUND 13,9:B=30:RETURN
360 SOUND 6,31:SOUND 7,54:SOUND 12,10:
    SOUND 13,9:D=30:RETURN
```



SHOOT 'EM

The gun is in your hand and the ducks are flapping across the screen, awaiting their destruction. All you have to do is point the gun at them, and shoot.

When you first play the game, set the time limit to around 2000, and the playing level to one — just to give yourself a chance. If your hit/miss rating is over 65 at the end of the game, then you advance to the next level of play. Your score is calculated using the formula: hits, divided by the sum of the time limit, divided by the shots, all multiplied by 100. A score of around 600 is good.

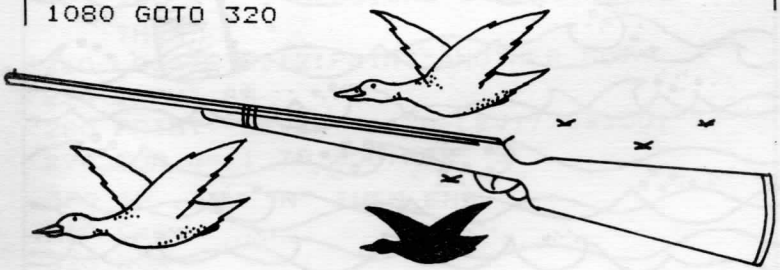
The game contains some very good graphics and sound, which you can incorporate into your own programs. If you show this game when it is running to your friends, they may have trouble believing that it is written in BASIC.

```
10 REM SHOOT'EM
20  SC=0:DF=11
30 CLS
40 INPUT "ENTER TIME LIMIT ";EE
50 IF EE<500  THEN 40
60 INPUT "LEVEL ";S
70 IF S<=0 THEN 60
80 SCREEN 1,2
90 SPRITE ON
100 COLOR 15,1,1
110 LINE (1,96)-(255,130),5,BF
120 LINE (1,40)-(1,96),5
130 LINE (255,40)-(255,96),5
140 LINE (1,186)-(1,130),5
150 LINE (255,186)-(255,130),5
160 FOR X=1 TO 32
170 READ A
180 GUN$=GUN$+CHR$(A)
190 NEXT X
```

```
200 FOR X=1 TO 32
210 READ A
220 DUCK$=DUCK$+CHR$(A)
230 NEXT X
240 FOR X=1 TO 32
250 READ A
260 FACE$=FACE$+CHR$(A)
270 NEXT X
280 FOR X=1 TO 32
290 READ A
300 SHOT$=SHOT$+CHR$(A)
310 NEXT X
320 SPRITE$(1)=GUN$
330 FOR X=2 TO 7
340 SPRITE$(X)=DUCK$
350 NEXT X
360 SPRITE$(7)=FACE$
370 SPRITE$(8)=SHOT$
380 SPRITE ON
390  PLAY "T255cdefgggo5go4gfedcccccdef
     go3ggo5go4gfedcccc","t255co5co4co5
     co4co5co4co5co4co5co4co5co4co5co6c
     o5co4co3co4co5co6co7co4co5co6co4c"
400 T=128:Y=175
410 A=190:B=120:C=50
420 FOR X=1 TO EE
430 PUT SPRITE 1,(T,Y),14
440 PUT SPRITE 3,(A,76),DF
450 PUT SPRITE 4,(B,76),DF
460 PUT SPRITE 5,(C,76),DF
470 A=A-S:B=B-S:C=C-S
480 D=STICK(1):F=STRIG(1):IF F=-1 THEN
    530
490 IF D=3 THEN T=T+S
500 IF D=7 THEN T=T-S
510 NEXT X
520 GOTO 760
530 SOUND 4,0:SOUND 5,0:SOUND 6,0:SOUND
    7,27:SOUND10,16:SOUND 12,5:SOUND
    13,9
```

```
540 SHOOT =SHOOT+1
550 PO=T+2
560 R=14
570 FOR J=158 TO 0 STEP -2*S
580 PUT SPRITE 8,(PO,J),R
590 ON SPRITE GOSUB 710
600 PUT SPRITE 3,(A,76),DF
610 PUT SPRITE 4,(B,76),DF
620 PUT SPRITE 5,(C,76),DF
630 A=A-S:B=B-S:C=C-S
640 D=STICK(1)
650 PUT SPRITE 1,(T,Y),14
660 IF D=3 THEN T=T+S
670 IF D=7 THEN T=T-S
680 NEXT J:SPRITE ON
690 PUT SPRITE 8,(T+2,J),1
700 GOTO 480
710 SPRITE OFF
720 SOUND 0,20 :SOUND 1,0:SOUND 7,62:
    SOUND 8,16:SOUND 12,10:SOUND 13,9
730 SC=SC+1
740 R=1
750 RETURN
760 SCREEN 0,0
770 SOUND 8,0:PLAY "T255cdefgggo5go4gf
    edcccccdefgo3ggo5go4gfedcccc", "
    t255co5co4co5co4co5co4co5co4co5co4
    co5co4co5co6co5co4co3co4co5co6co
    7co4co5co6co4c"
780 CLS
790 PRINT "YOUR TIME IS UP..."
800 PRINT "SHOTS MADE ";SHOOT
810 PRINT "HITS       ";SC
820 PRINT
830 PRINT
840 PRINT "HIT MISS RATIO ";SC/SHOOT*
    100
850 PRINT
860 PRINT
```

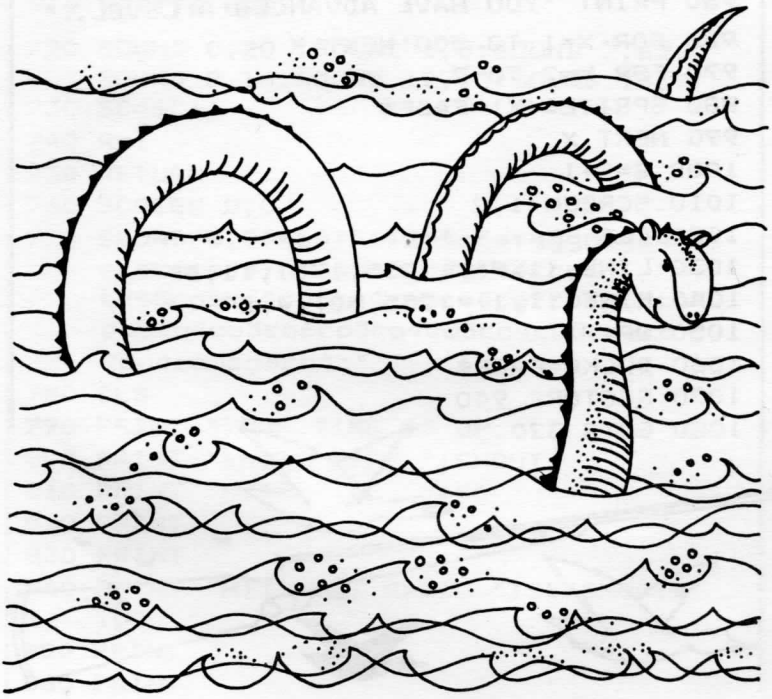
```
870 PRINT "YOUR SCORE IS " ;INT
    (SC/(EE/SHOOT)*100)
880 FOR X=1 TO 6000:NEXT X
890 IF SC/SHOOT*100>65 THEN  950
900 END
910 DATA 0,0,1,1,1,1,1,1,7,1,63,127,
    127,127,62,0,0,0,192,224,192,192,
    192,192,192,192,192,192,160,0,0,0
920 DATA 0,0,0,0,7,11,7,3,6,13,15,12,
    15,7,1,2,0,0,0,0,0,128,128,0,0,240,
    216,56,248,240,64,128
930 DATA 0,32,16,8,7,8,8,10,8,12,13,6,
    6,3,0,0,0,4,8,16,224,16,16,80,16,
    48,176,96,96,192,0,0
940 DATA 0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
950 PRINT "YOU HAVE ADVANCED A LEVEL."
960 FOR X=1 TO 500:NEXT X
970 FOR X=2 TO 7
980 SPRITE$(X)=FACE$
990 NEXT X
1000 S=S+1
1010 SCREEN 1,2
1020 CLS
1030 LINE (1,96)-(255,130),11,BF
1040 LINE (1,1)-(255,50),4,BF
1050 DF=7
1060 DUCK$=FACE$
1070 RESTORE 940
1080 GOTO 320
```



NESS OF LOCHNESS

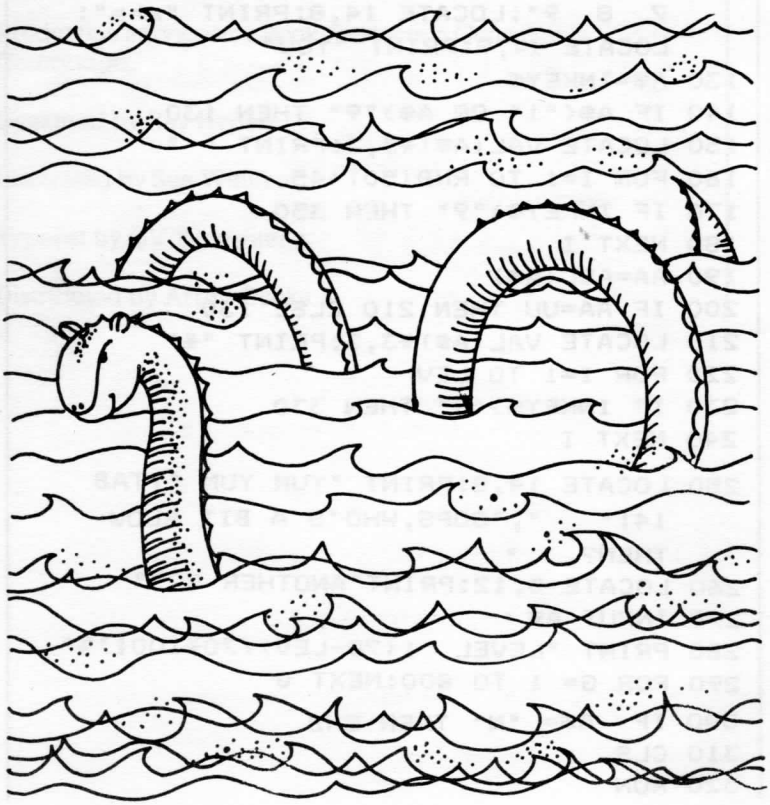
In this unlikely game, Ness is hiding behind a rock. You are a monster hunter for the R.B.Z. (Royal British Zoo). Your task is to shoot Nessie with a sleeping dart. Unfortunately, your weapons are slightly old. One even bears the legend 'Made in 1654'. And if you shoot too quickly, the dart gun will blow up in your face.

To hit your chosen rock, press the corresponding number. To fire, press a letter. However, you must be quick or Nessie will jump out and bite you.



```
10 REM NESS OF LOCHNESS
20 CLS
30  LOCATE 1,3: PRINT "NESSIE IS HERE
    GET HER BEFORE SHE      GETS YOU."
40  LOCATE 1,1: PRINT "NESSIE"
50 AP$=INKEY$:AS=AS+.1:IF AP$=""
   THEN 50
60 FOR T= 1 TO AS
70 QWE=RND(1)
80 NEXT T
90 LEV =150
100 CLS
110 UU= INT(RND(1)*9)+49
120 LOCATE 3,3:PRINT"1  2  3  4  5  6
    7  8  9":LOCATE 14,8:PRINT "ahs":
    LOCATE 14,9:PRINT "YOU"
130 A$=INKEY$
140 IF A$<"1" OR A$>"9" THEN 130
150 LOCATE VAL(A$)*3,3:PRINT " "
160 FOR I=1 TO RND(50)+45
170 IF INKEY$>"9" THEN 350
180 NEXT I
190 AA=ASC(A$)
200 IF AA=UU THEN 210 ELSE 130
210 LOCATE VAL(A$)*3,3:PRINT "*"
220 FOR I=1 TO LEV
230 IF INKEY$>"9" THEN 330
240 NEXT I
250 LOCATE 14,8:PRINT "YUM YUM ";TAB
    14;"    ","OOPS,WHO"S A BIT SLOW
    THEN?    "
260 LOCATE 0,12:PRINT"ANOTHER GO?"
270 INPUT A$
280 PRINT "LEVEL ";(70-LEV)/70*100;"%"
290 FOR G= 1 TO 800:NEXT G
300 IF  A$= "N" THEN END
310 CLS
320 RUN
```

```
330 LOCATE ((UU-48)*3-2),3:PRINT
    "GOT IT"
340 GOTO 370
350 LOCATE 0,5: PRINT "TRIGGER HAPPY
    PERSON"
360 GOTO 260
370 FOR H= 1 TO 900:NEXT H
380 CLS
390 PRINT "OK.  TRY AGAIN ",,,,"THIS
    TIME BE QUICKER"
400 LEV=LEV-5
410 FOR H= 1 TO 900:NEXT H
420 GOTO 100
```



EDUPACK

Here is your chance to answer those critics who claim your computer can only be used for games. Load the program up, and get your mental arithmetic skills into gear. Full instructions are included within the program.

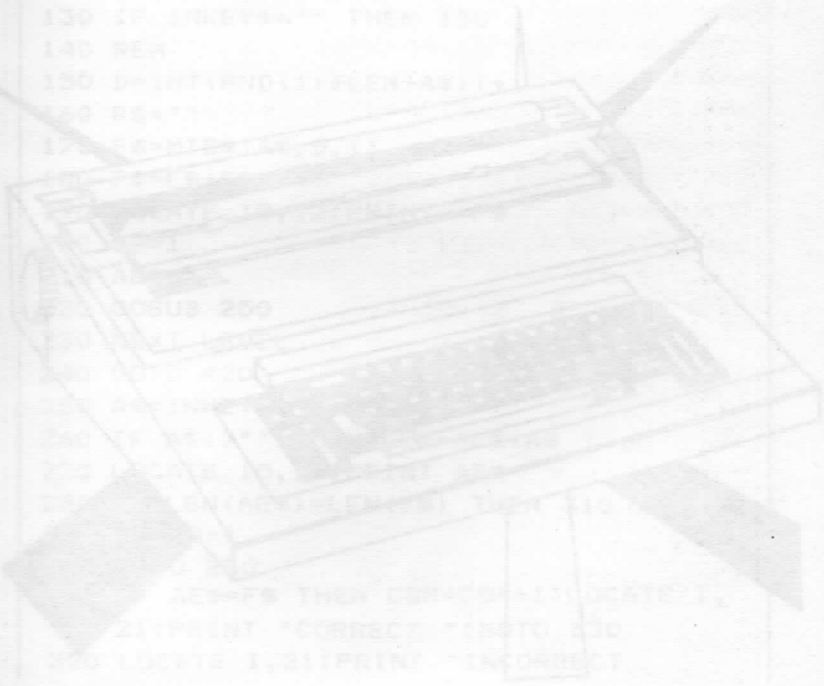
```
10 REM EDUPAK
20 CLS
30 LOCATE 8,5:PRINT "1. multiply"
40 LOCATE 8,7:PRINT "2. division"
50 LOCATE 8,9:PRINT "3. addition"
60 LOCATE 8,11:PRINT "4. subtraction"
70 PRINT :PRINT
80 PRINT "choose a topic. "
90 INPUT Y$
100 IF Y$="1" THEN 150
110 IF Y$="2" THEN 480
120 IF Y$="3" THEN 700
130 IF Y$="4" THEN 940
140 GOTO 90
150 CLS
160 C=0
170 W=0
180 LOCATE 1,21:INPUT "number range
    0 to ";B
190 FOR Q=1 TO 10
200 CLS
210 E=INT(RND(1)*B)
220 F=INT(RND(1)*B)
230 LOCATE 9,10:PRINT "multiply "
    ;E;"*";F
240 LOCATE 1,21:INPUT G
250 K=E*F
260 IF G<>K THEN 310
270 C=C+1
```

```
280 LOCATE 11,15:PRINT "correct ":PLAY "
    cdefg":FOR VV=1 TO 200:NEXT VV
290 NEXT Q
300 GOTO 380
310 LOCATE 5,15:PRINT "    wrong try
    again !!!":PLAY"gfedc"
320 W=W+1
330 LOCATE 9,10:PRINT "multiply "
    ;E;"*";F
340 LOCATE 1,21:INPUT G
350 K=E*F
360 IF K<>G THEN LOCATE 5,3 :PRINT
     "the correct answer was ";K:FOR
     VV=1 TO 999 : NEXT VV:GOTO 290
370 GOTO 280
380 CLS
390 LOCATE 6,10:PRINT
    "out of ten you got..."
400 PRINT TAB(8);C;" correct"
410 PRINT TAB(8);W;" wrong"
420 PRINT
430 PRINT
440 IF C=10 THEN PRINT "excelent score
    keep it up..."
450 IF W>+5 THEN PRINT "you should try
    an easier level.."
460 LOCATE 8,19:PRINT "press any key":
    IF INKEY$="" THEN 460
470 GOTO 20
480 CLS
490 C=0
500 W=0
510 LOCATE 1,21:INPUT "number range 0
    to ";B
520 FOR Q=1 TO 10
530 CLS
540 E=INT(RND(1)*B)
550 F=INT(RND(1)*B):IF F=0 THEN 550
560 LOCATE 9,10:PRINT "divide ";E*F;
```

```
    "/";F
570 LOCATE 1,21:INPUT G
580 IF G<>E THEN 630
590 C=C+1
600 LOCATE 11,15:PRINT "correct ":PLAY
    "cdefg":FOR VV=1 TO 200:NEXT VV
610 NEXT Q
620 GOTO 380
630 LOCATE 5,15:PRINT "    wrong try
    again !!!":PLAY"gfedc"
640 W=W+1
650 LOCATE 9,10:PRINT "divide ";E*F;
    "/";F
660 LOCATE 1,21:INPUT G
670 IF K<>G THEN LOCATE 5,3 :PRINT
    "the correct answer was ";K:FOR
    VV=1 TO 999 :
NEXT VV:GOTO 610
680 GOTO 600
690   PRINT
700 CLS
710 C=0
720 W=0
730 LOCATE 1,21:INPUT "number range 0
    to ";B
740 FOR Q=1 TO 10
750 CLS
760 E=INT(RND(1)*B)
770 F=INT(RND(1)*B)
780 LOCATE 9,10:PRINT "add ";E;"+";F
790 LOCATE 1,21:INPUT G
800 K=E+F
810 IF G<>K THEN 860
820 C=C+1
830 LOCATE 11,15:PRINT "correct ":PLAY
    "cdefg":FOR VV=1 TO 200:NEXT VV
840 NEXT Q
850 GOTO 380
```

```
860 LOCATE 5,15:PRINT "     wrong try
    again !!!":PLAY"gfedc"
870 W=W+1
880 LOCATE 9,10:PRINT "add ";E;"+";F
890 LOCATE 1,21:INPUT G
900 K=E+F
910 IF K<>G THEN LOCATE 5,3 :PRINT
    "the correct answer was ";K:FOR
    VV=1 TO 999 :
NEXT VV:GOTO 840
920 GOTO 830
930  PRINT
940 CLS
950 C=0
960 W=0
970 LOCATE 1,21:INPUT "number range 0
    to ";B
980 FOR Q=1 TO 10
990 CLS
1000 E=INT(RND(1)*B)
1010 F=INT(RND(1)*B):IF F>E THEN 1010
1020 LOCATE 9,10:PRINT "subtract ";E;
    "-";F
1030 LOCATE 1,21:INPUT G
1040 K=E-F
1050 IF G<>K THEN 1100
1060 C=C+1
1070 LOCATE 11,15:PRINT "correct ":
    PLAY "cdefg":FOR VV=1 TO 200:NEXT
    VV
1080 NEXT Q
1090 GOTO 380
1100 LOCATE 5,15:PRINT "     wrong try
    again !!!":PLAY"gfedc"
1110 W=W+1
1120 LOCATE 9,10:PRINT "subtract ";E;
    "-";F
1130 LOCATE 1,21:INPUT G
1140 K=E-F
```

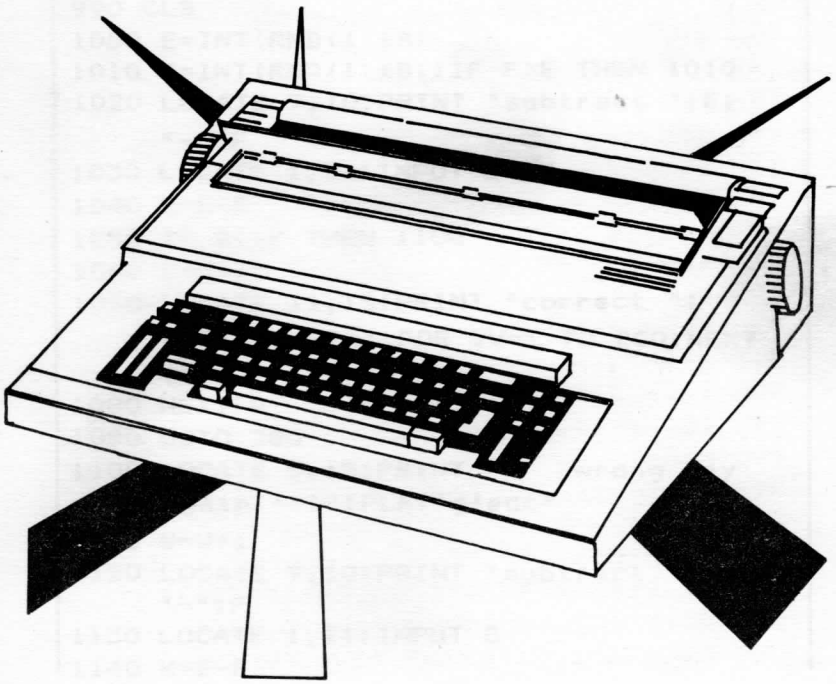
```
1150 IF K<>G THEN LOCATE 5,3 :PRINT
     "the correct answer was ";K:FOR
     VV=1 TO 999  :NEXT VV:GOTO 1080
1160 GOTO 1070
```



TOUCH TYPING

If you're a two-finger typist (or even a three-finger typist) this program will help you increase your speed. You have to copy the letters produced on the screen. The program begins by running through each row, and it then selects letters at random. Any errors that you make, and the time you have taken, are shown to you at the end of the test.

You'll find that your typing skills will increase considerably if you keep your eyes fixed on the screen, and not on the keyboard.



```
10 REM  T.T.
20 CLS
30 DATA 1234567890-=,QWERTYUIOP[]\,
   ASDFGHJKL',ZXCVBNM./,!@#$%^&*()_+,
   qwertyuiop[
   ]\,asdfghjkl',zxcvbnm./,<>?;"{}~_+
40 RESTORE 30
50 FOR LEVEL=1 TO 9 STEP .2
60 FORT=1TOLEVEL
70 READA$
80 NEXTT
90 RESTORE30
100 LOCATE1,1:PRINT "TOUCH TYPING"
110 LOCATE1,3:PRINT "ROW===
    ":LOCATE 7,3:PRINT A$
120 LOCATE 1,21:PRINT "READY.
    PRESS A KEY"
130 IF INKEY$="" THEN 130
140 REM
150 D=INT(RND(1)*LEN(A$))+1
160 F$=""
170 F$=MID$(A$,D,1)
180 F$=F$+F$+F$
190 LOCATE 10,12:PRINT  F$
200 CO=1
210 AE$=""
220 GOSUB 250
230 NEXT LEVEL
240 GOTO 420
250 A$=INKEY$
260 IF A$<>"" THEN AE$=AE$+A$
270 LOCATE 10,19:PRINT AE$
280 IF LEN(AE$)=LEN(F$) THEN 310
290 CO=CO+1
300 GOTO 250
310 IF AE$=F$ THEN CQR=CQR+1:LOCATE 1,
    21:PRINT "CORRECT ":GOTO 330
320 LOCATE 1,21:PRINT "INCORRECT
```

```
    ":INCR=INCR+1
330 FORG=1TO 200:NEXT G
340 RETURN
350 CLS
360 LOCATE 1,1:PRINT "RESULTS"
370 LOCATE 1,4:PRINT "ANSWERS CORRECT
    ";CQR
380 LOCATE 1,6:PRINT
    "ANSWERS INCORRECT ";INCR
390 LOCATE 1,10:PRINT "SPEED RATING "
    ;(CO-10)*8.54518081#
400 FOR H=1 TO 500:NEXT H
410 RETURN
420 G1=6  :CLS:FF=5:Q=25:D=65:GOSUB 440
430 GOTO 560
440 CLS:LOCATE 1,1:PRINT
    "TOUCH TYPING ",,"RANDOM LETTERS "
450 FOR GG=1 TO G1
460 FOR MM=1 TO FF
470 F$=F$+CHR$((RND(1)*Q)+D)
480 NEXT MM
490 LOCATE 10,10:PRINTF$
500 AE$=""
510 GOSUB 250
520 F$=""
530 LOCATE 1,21:PRINT "PRESS A KEY"
    :IF INKEY$="" THEN 530
540 NEXT GG
550 RETURN
560 G1=4:CLS:FF=7:Q=10:D=47:GOSUB 440
570 G1=6:CLS:FF=6:Q=25:D=96:GOSUB 440
580 G1=10:CLS:FF=3:Q=80:D=33:GOSUB 440
590 GOSUB 350
600 LOCATE 1,21:PRINT "PRESS Y TO TRY
    AGAIN ":IF INKEY$="" THEN 600:
    IF INKEY$="Y " THEN GOTO 10
```

CALENDAR

Do you know the actual day of the week when you were born? If not, this program will tell you. It will also tell on which day of the week your birthday, or any other anniversary, will fall in the years to come.

```
10 REM CALENDER
20 CLS
30 PRINT "CALENDER",,"--------"
40 PRINT ,,"ENTER MONTH"
50 INPUT M
60 IF M<1 OR M>12 THEN 50
70 PRINT "ENTER YEAR "
80 INPUT Y
90 PRINT
100 PRINT "MONTH ";M;". YEAR ";Y
110 PRINT
120 GOSUB 310
130 D=DD
140 M=M+1
150 IF M>12 THEN M=1:Y=Y+1
160 GOSUB 310
170 IF DD<D THEN DD=DD+7
180 N=28+DD-D
190 PRINT "SUN MON TUE WED THU FRI SAT"
200 P=4*D
210 FOR X=1 TO N
220 PRINT TAB(P);X;
230 P=(P+4) AND P<21
240 IF POS(3)>26 THEN PRINT
250 NEXT X
260 PRINT ,,,"PRESS ENTER TO GO ON"
270 INPUT A
280 GOTO 20
290 MM=M-2:YY=Y
300 IF MM>0 THEN GOTO 350
310 MM=MM+12
320 YY=YY-1
330 C=INT(Y/100)
340 YY=YY-100*C
350 DD=1+INT(2.6*MM-.19)+YY+INT(YY/4)+
    INT(C/4)-2*C
360 DD=DD-7*INT(DD/7)
370 RETURN
```

BOMB RUN

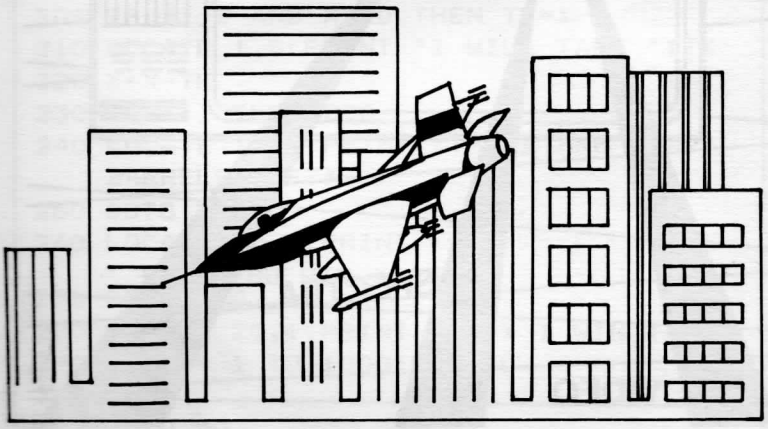
As your plane plummets towards an enemy city, your only hope is to completely destroy the city, and land on its ruins.

If you're a beginner, try level 1, with level 10 reserved for real experts. The level affects the height of your plane at the start of its descent.

It will probably take you many attempts to destroy the city before you manage a single safe landing. The explosion and engine noises are very effective and you may want to use them in other games you write.

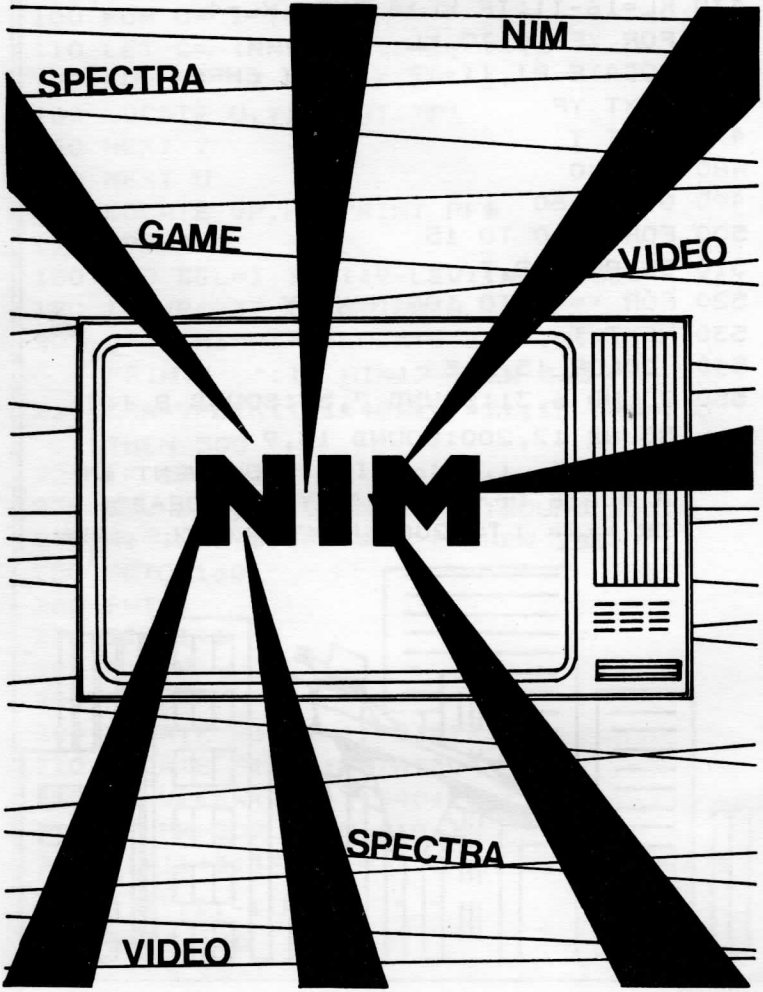
```
10 REM BOMB RUN
20 PF$=" "+CHR$(212)
30 CLS
40  LOCATE 1,5:PRINT "LEVEL"
50  LOCATE 1,6:INPUT LEV
60 VP=2
70 IF LEV >10 THEN 50
80 CLS
90 HI=LEV
100 FOR U= 1 TO 36
110 LET C= (RND(1)):C=C* 7+10
120 FOR Y= 17 TO C STEP -1
130 LOCATE U,Y:PRINT "P"
140 NEXT Y
150 NEXT U
160 LOCATE VP,HI:PRINT PF$
170 VP=VP+1
180 FOR DEL=1 TO (19-LEV):NEXT DEL
190 IF VP<>37 THEN  GOTO  210
200 VP=1:HI=HI+1:LOCATE 37  ,HI-1:
    PRINT"  ":IF HI=17 THEN 360
210 CRA=VPEEK((HI*40)+VP+2):IF CRA=207
    THEN 500
220 IF SWT =1 THEN 290
230 SOUND 3,15:SOUND 7,61:SOUND 9,10
240 A$=INKEY$:IF A$<>"" THEN 260
250 GOTO 160
260 SWT=1
270 I1=HI+1
280 P1=VP
290 REM
300 LOCATE P1,I1-1:PRINT" "
310 LOCATE P1,I1:PRINT"v"
320 PK=VPEEK((I1+1)*40+P1+1)
330 IF PK=207 THEN 410
340 I1=I1+1:IF I1>=17 THEN SWT=0:LOCATE
      P1,I1-1:PRINT " "
350 GOTO 160
360 LEV=LEV+1
```

```
370 IF LEV >10 THEN THR=THR+1+10:LEV=
    10610 FOR VV= 1 TO 600:NEXT VV
380 FOR VV= 1 TO 600:NEXT VV
390 GOTO 80
400 END
410 SOUND 6,30:SOUND 7,54:SOUND 8,16:
    SOUND 12,33:SOUND 13,9
420 FOR T=5  TO -1 STEP  -1
430 KL=16-I1:IF KL>4 THEN KL=4
440 FOR YF= 0 TO KL
450 LOCATE P1,I1+YF :PRINT CHR$(T+160)
460 NEXT YF
470 NEXT T
480 SWT =0
490 GOTO 160
500 FOR T= 0 TO 15
510 COLOR T,2,T
520 FOR Y= 1 TO 100:NEXT Y
530 NEXT T
540  COLOR 15,4,5
550 SOUND 6,31:SOUND 7,54:SOUND 8,16:
    SOUND 12,200:SOUND 13,9
560  LOCATE  1,21: PRINT "YOU WENT IN
    A BLAZE OF GLORY ","YOURE DEAD":
   FOR KKP= 1 TO 2000:NEXT KKP:CLS :RUN
```



NIM

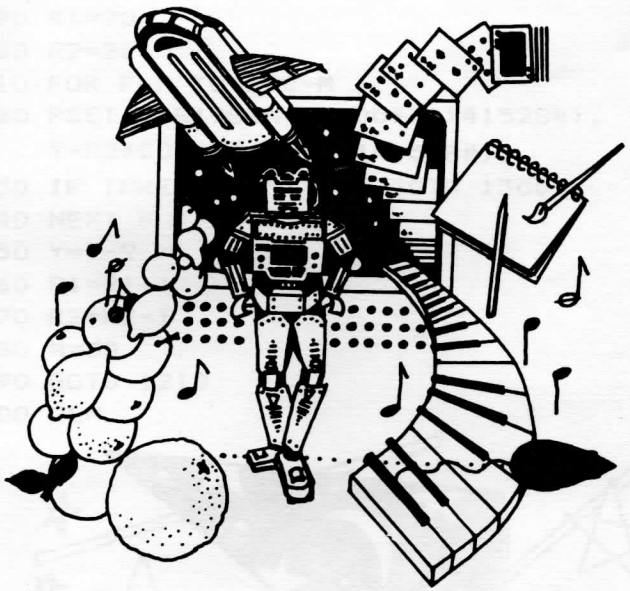
This is the classic game in which you try to force the computer to take the last of a number of chests. The last chest is filled with explosives.



```
10 REM NIM
20 CLS
30  LOCATE 0,21:PRINT "NIM",,"---"
40 FOR T= 1 TO 20:FOR W= 1 TO 100:NEXT
   W:PRINT:NEXT T
50 X= INT(RND(1)*25)+5
60 B$="HFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
      FFFFFFFFFFFFF"
70  LOCATE 3,3: PRINT LEFT$(B$,X)
80 IF RND(1)<.5 THEN 140
85  LOCATE 1,10:PRINT "THERE ARE ";X;"
    BARRELS LEFT     "
90 LOCATE 1,5:INPUT "HOW MANY DO YOU
   TAKE ";TK
100 IF TK<1 OR TK>3 THEN 90
110 X=X-TK
120 IF X= 1 THEN 260
130 LOCATE 1,10:PRINT "THERE ARE ";X;"
    BARRELS LEFT     "
140 TK=3
150 IF X= 2 THEN TK=1
160 IF X= 3 THEN TK=2
170 IF X= 4 THEN TK=3
180 IF X= 5 THEN TK=1
190 IF X= 6 THEN TK=1
200 IF X= 7 THEN TK=2
201 IF X> 8 AND X<10 THEN TK=1
210 LOCATE 1,8:PRINT "I WILL TAKE ";TK
220 X=X-TK
230 IF X=1 THEN 280
240 LOCATE 1,10:PRINT "THERE ARE ";X;"
    BARRELS LEFT"
250 GOTO 90
260 LOCATE 20,1:PRINT
    " **** YOU WIN ****"
270 GOTO 290
280 LOCATE 20,1:PRINT "**** I WIN ****
290 FOR L= 1 TO 1000:NEXT L
300 CLS
310 GOTO 50
```

GRAPHICS DEMONSTRATION — THE SPECTRACULAR

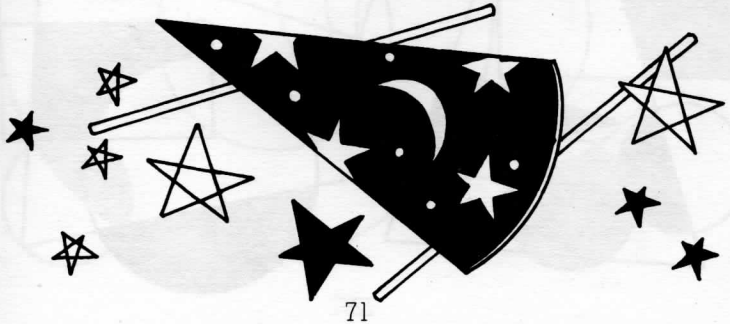
You can enter any of these programs on their own, or you can type a few in, and then add some later. The line numbers have been arranged so that the programs will all fit together to form one long, spectacular demonstration — ideal if you want to show off your computer to your friends.



WIZARD'S HAT

This program produces a picture of Wizard Merlin's magical hat. It is said, in Arthurian legend, that only the cleverest computers can draw it. The main part of the program is in line 1220, which uses PI, SIN and COS in its calculations.

```
1100 FOR X=1 TO 1000:NEXT X
1110 REM WIZARDS HAT
1130 COLOR 1,3,5
1140 CLS
1150 SCREEN1
1160 X=128
1170 Y=88
1180 M=0
1190 R1=70
1200 R2=20
1210 FOR F=M TO 200-M
1220 PSET(X+R1*SIN(F/100*3.141528#),
     Y+R2*COS(F/100*3.141528#))
1230 IF INKEY$=" " THEN GOTO 1300
1240 NEXT F
1250 Y=Y-2
1260 R1=R1-1
1270 R2=R2-1
1280 M=50
1290 GOTO 1210
1300 REM
```

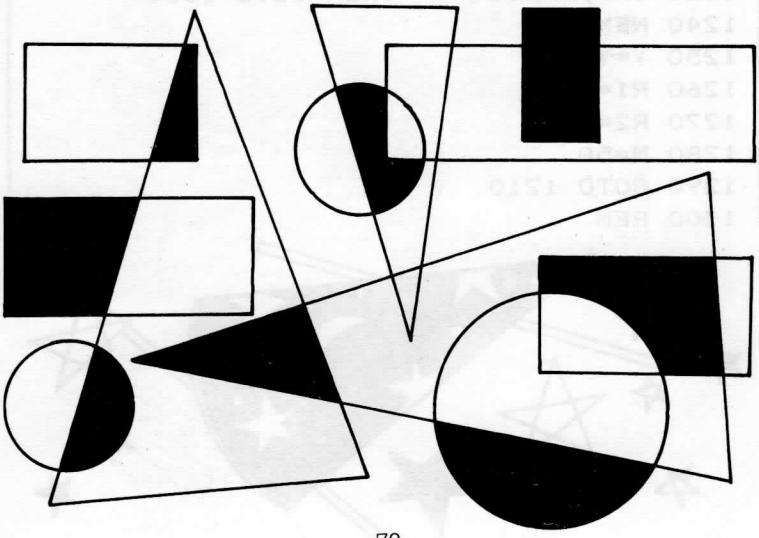


STRINGY

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

Lines 370 to 400 are the heart of the program. You might like to put a variable for the colour, and make up a loop to go through all the 16 colours.

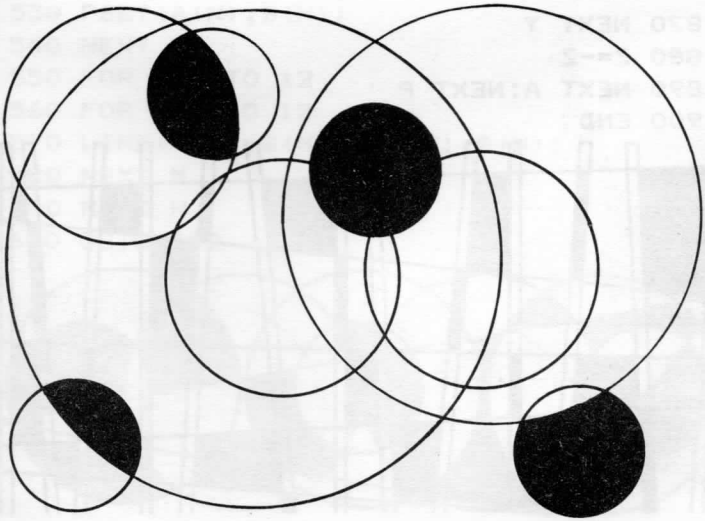
```
270 FOR X=1 TO 1000:NEXT X
280 REM STRINGY
330 CLS
340 COLOR 1,1,1
350 SCREEN 1
360 FOR I=1 TO 175 STEP 4
370 LINE(0,I)-(255-I,I+-I),3
380 LINE(I,0)-(I+(255-I),0+I),3
390 LINE(255-I,175)-(I-255-I,175-I),3
400 LINE(I,175)-(I+255-I,175+-I),3
410 NEXT I
420 GOTO 420
```



CIRCLE DRAW

This program draws random circles on the screen, using different radii and co-ordinates. The program is swift, and quite amazing to watch.

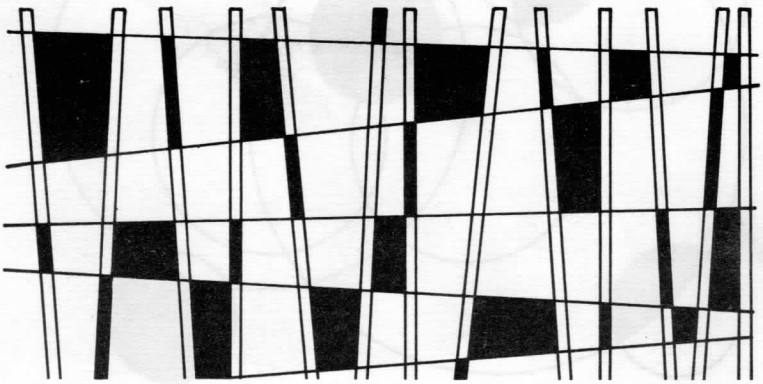
```
1900 FOR X=1 TO 1000:NEXT X
1910 REM CIRCLE DRAW
1940 COLOR 7,3,3
1950 SCREEN 1
1960 CLS
1970 FORC=1 TO 15
1980 R=RND(1)*50
1990 X=R+RND(1)*(255-2*R)
2000 Y=R+RND(1)*(175-2*R)
2010 IF INKEY$=" "THEN 2050
2020 CIRCLE (X,Y),R,C
2030 NEXT C
2040 GOTO 1970
2050 END
```



TIME GATE

This program draws a gate. The main part of the program is in lines 1820 and 1870. To get different patterns, change the STEP commands in lines 1810 and 1850. This program shows how well the LINE command works.

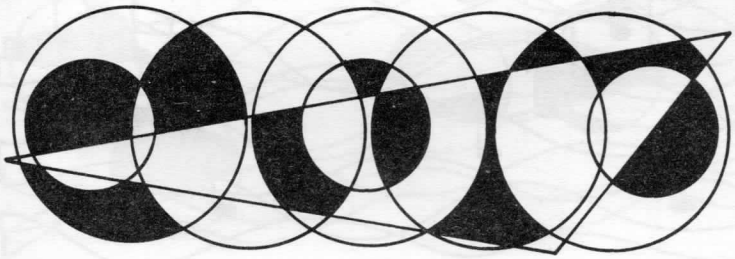
```
1730 FOR X=1 TO 1000:NEXT X
1740 REM TIME GATE
1770 SCREEN 1
1780 CLS
1790 FOR P= 0 TO 15
1800 Z=1
1810 FOR A= 1 TO 2
1820 FORX=0TO254 STEP 2
1830 LINE(128,88)-(128-127*Z+X*Z,88-
     Z*87),P
1840 NEXT X
1850 FOR Y= 0 TO 175 STEP 2
1860 LINE(128,88)-(128+127*Z,88+Z*-87+
     (Y*Z)),P
1870 NEXT Y
1880 Z=-Z
1890 NEXT A:NEXT P
1900 END
```



PATTERN WHEEL

This program shows how DIM statements can be used in graphics programming. The pattern produced is somewhat like a devil's eye. As the hole shrinks, the speed increases, and the wheel develops.

```
420 FOR X=1 TO 1000:NEXT X
430 REM PATTERN WHEEL
440 REM USING DIM STATEMENTS
450 REM FOR GRAPHIC SCREENS
460 SCREEN1
470 COLOR 1,7,7
480 CLS
490 DIM A(12):DIM B(12)
500 FOR N=1 TO 12
510 K=N/6*3.1415927#
520 A(N)=128+80*SIN(K):B(N)=88+80*
    COS(K)
530 PSET(A(N),B(N))
540 NEXT N
550 FOR N=1 TO 12
560 FOR M=1 TO 12
570 LINE(A(N),B(N))-(A(M),B(M))
580 NEXT M
590 NEXT N
600 GOTO 600
```

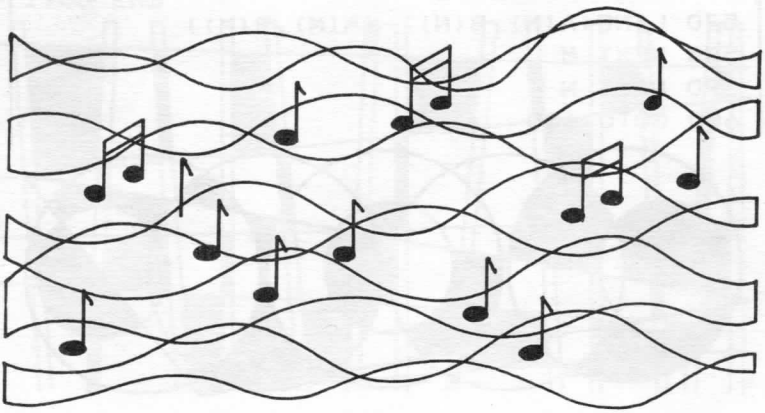


WAVE FORM

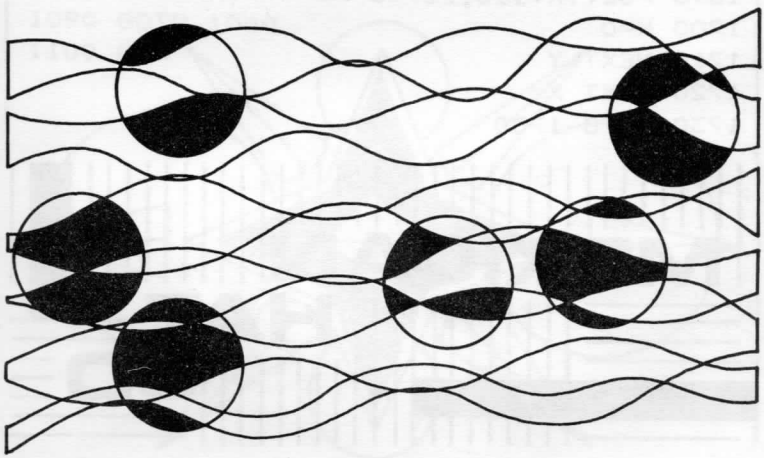
This program, illustrating the power of the LINE command in the high resolution graphics mode, is possibly the best one in this section. The program is extremely quick, and the effect is quite hypnotic.

Wave Form's shape is determined randomly, so it will differ from run to run. You may like to try and modify the program so it produces music in time with the pattern.

```
600 FOR X=1 TO 1000:NEXT X
610 REM WAVE FORM
650 COLOR 1,1,1
660 CLS
670 SCREEN 1
680 X=INT(RND(10)*255)
690 Y=INT(RND(10)*175)
700 L=INT(RND(10)*255)
710 M=INT(RND(10)*175)
720 U=15:V=7
730 DEF FNR(X)=INT(RND(1)*X)
740 GOSUB 920
```



```
750 FOR Q=2 TO 16
760 FOR G=1 TO 150
770 NUM=NUM-1
780 IF NUM=0 THEN GOSUB 920
790 LINE(X,Y)-(L,M),Q
800 IF INKEY$=" " THEN GOTO 930
810 IF X+A>255 OR X+A<0 THEN A=-A
820 IF Y+B>175 OR Y+B<0 THEN B=-B
830 IF L+C>255 OR L+C<0 THEN C=-C
840 IF M+D>175 OR M+C<0 THEN D=-D
850 X=X+A:Y=Y+B
860 L=L+C:M=M+D
870 NEXT G
880 FOR VV=1 TO 300 :NEXT VV
890 CLS
900 NEXT Q
910 IF INKEY$=" " THEN GOTO 930
920 A=FNR(U)-V
930 B=FNR(U)-V
940 C=FNR(U)-V
950 D=FNR(U)-V
960 NUM=FNR(20)+10
970 RETURN
```



THREE-DEE

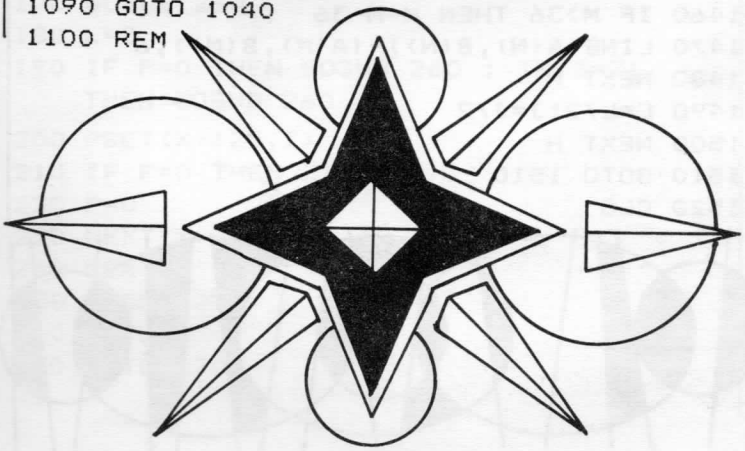
This program draws a three-dimensional Mexican hat on the screen. It takes a while to complete, but the wait is worth it.

```
1530 FOR X=1 TO 1000:NEXT X
1540 REM THREE DEE
1550 REM
1560 CLS
1570 DEF FNA(Z)=32 *SIN(Z/6)
1580 PRINT "RESOLUTION (1 TO 10)"
1590 INPUT R
1600 SCREEN 1
1610 CLS
1620 FOR X= -100 TO 100
1630 J=0:K=1
1640 V=R*INT(SQR((10^4)-X*X)/R)
1650 FOR Y= V TO -V STEP -R
1660 Z=INT(80+FN A(SQR(X*X+Y*Y))-.
     707*Y)
1670 IF Z<J THEN GOTO 1680
1680 J=Z
1690 PSET(X+110,Z)
1700 K=0
1710 NEXT Y
1720 NEXT X
1730 GOTO 1730
```

STAR OF KAZZBAD

This program — although it takes a while to run because the computer has to work out each point — is further proof (if you needed it) of how exciting the Spectravideo's graphics can be.

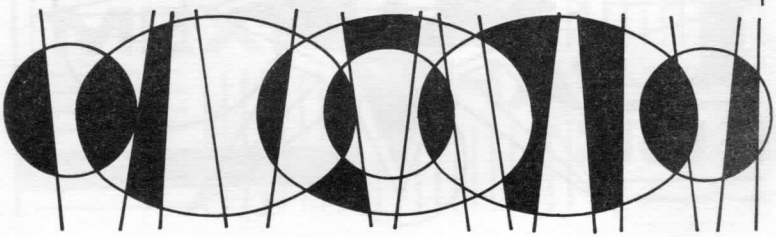
```
970 FOR X=1 TO 1000:NEXT X
980 REM STAR OF KAZZBAD
990 REM IT TAKES A LONG TIME
1000 REM HAVE PATIENCE
1010 CLS
1020 SCREEN 1
1030 X=0:Y=80
1040 FOR N=0 TO 2*3.1415927# STEP
     3.1415927#/180
1050 PSET (128+X*SIN(N),87+Y*COS(N))
1060 NEXT N
1070 X=X+10:Y=Y-10
1080 IF Y=-10 THEN GOTO 1100
1090 GOTO 1040
1100 REM
```



THE EYE OF TIME

Before drawing the graph, the program calculates the point positions. It puts them in a DIM statement, and then completes the graph. The actual calculation is carried out in line 1410.

```
1300 FOR X=1 TO 1000:NEXT X
1310 REM EYE OF TIME
1340 SCREEN1
1350 COLOR 1,1,1
1360 DIM A(36):DIM B(36)
1370 L=120:J=80
1380 FOR H=1 TO 5
1390 FOR N=1 TO 36
1400 K=N/18*3.1415927#
1410 A(N)=128+L*SIN(K):B(N)=88+J*COS(K)
1420 PSET (A(N),B(N)),3
1430 NEXT N
1440 FOR N=1 TO 36
1450 M=N+12
1460 IF M>36 THEN M=M-36
1470 LINE(A(N),B(N))-(A(M),B(M)),6
1480 NEXT N
1490 L=L/2:J=J/2
1500 NEXT H
1510 GOTO 1510
1520 CLS
```



3D GRAPH

This program produces a full, three-dimensional view of a sine curve. As the REM statements indicate, it takes a while to run...but your patience will be rewarded.

```
10 REM 3D GRAPH
20 REM TAKES A WHILE TO RUN
30 REM BUT VERY GOOD EFFECTS
40 REM RESULT.
50 CLS
60 SCREEN 1
70 DEF FNA(Z)=90*EXP(-Z*Z/600)
80 K=5
90 FOR X=-100 TO 100 STEP 1
100 L=0
110 P=1
120 Z1=0
130 Y1=K*INT(SQR(10000-X*X)/K)
140 FOR Y=Y1 TO -Y1 STEP -K
150 Z=INT (80+FN A(SQR(X*X+Y*Y))-.
    707106*Y)
160 IF Z<L THEN GOTO 230
170 GOSUB 260
180 L=Z
190 IF P=0 THEN GOSUB 260 : IF Z=Z1
    THEN GOSUB 260
200 PSET(X+125,Z)
210 IF P=0 THEN Z1=Z
220 P=0
230 NEXT Y
240 NEXT X
250 GOSUB 270
260 RETURN
270 GOTO 270
```

CREATING SOUND EFFECTS ON YOUR SPECTRAVIDEO

We thought it might be useful to give you some information to help you create outstanding sounds to add to your programs. First of all, run this program:

```
10 SOUND 7,49
20 SOUND 6,31
30 SOUND 12,40
40 SOUND 9,7
50 SOUND 10,7
60 IF INKEY$ = "" THEN GOTO 60
70 SOUND 8,10
80 FOR X = 31 TO 0 STEP - 1
90 SOUND 2,50 - X
100 SOUND 4,60 = X
110 SOUND 6,X
120 NEXT X
130 SOUND 8,16
140 SOUND 2,0
150 SOUND 4,0
160 SOUND 6,31
170 SOUND 13,0
180 FOR X = 1 TO 150: NEXT X
190 GOTO 60
```

As you can hear, it certainly demonstrates the sound capabilities of your computer. Here's how the sound works:

REGISTER 0 AND 1:

CONTROL THE FREQUENCY OF THE TONE ON CHANNEL A
R0 IS FINE TUNE
R1 IS COARSE TUNE
THE FORMULA FOR THE FREQUENCY IS...

100000/R0 VALUE +256*R1 VALUE

IF R0 IS SET TO CONTAIN THE VALUE 100 AND R1 SET TO ZERO, A 1KHZ TONE WILL BE PRODUCED.

R2, R3: THE SAME AS ABOVE BUT FOR TONE B.

R4, R5: THE SAME AS ABOVE BUT FOR TONE C.
R6 CAN TAKE ANY VALUE FROM 0 TO 31 AND CONTROLS THE FREQUENCY OF THE NOISE GENERATOR.

REGISTER 7:
THIS IS THE ONE THAT MANY PEOPLE FIND HARD TO UNDERSTAND.

IT IS A MIXER. IT MIXES THE NOISE AND TONE FOR EACH CHANNEL.

THE VALUE IN R7 CAN BE THOUGHT OF AS THE SUM OF:

1	WHEN THE TONE IS NOT CONNECTED TO CHANNEL	A
2		B
4		C
8	NOISE	A
16	NOISE	B
32	NOISE	C

THE VALUE 32+16+4+1 WOULD CORRESPOND TO:
THE NOISE ONLY IN A AND
THE TONE ONLY IN B.

REGISTER 8 IS THE VOLUME CONTROL FOR CHANNEL A; 0 IS MINIMUM AND 15 IS MAXIMUM.

THE VALUE 16 PUTS THE OUTPUT OF THE CHANNEL UNDER THE CONTROL OF THE ENVELOPE GENERATOR.

REGISTER 9:
SAME AS R8 BUT FOR CHANNEL B

REGISTER 10:
SAME AS R8 BUT EFFECTS CHANNEL C
R11 and R12 CONTROL THE PERIOD OF THE WAVEFORM BY THE ENVELOPE GENERATOR.
R11 IS FINE TUNE
R12 IS COARSE TUNE
THE RESULTING ENVELOPE PERIOD BEING .2*(R11 VALUE+256 * R12 VALUE)MILLISECONDS

REGISTER 13 CONTROLS THE SHAPE OF THE ENVELOPE WAVE FORM.

THE VALUES IN REGISTER 13 CAN BE FROM 8 to 15:

8=HIGH ATTACK SLOW DECAY CONTINUOUS
9=FAST ATTACK SLOW DECAY END
10=SLOW DECAY SLOW ATTACK CONTINUOUS
11=FAST ATTACK SLOW DECAY FAST ATTACK END CONTINUOUS
12=SLOW ATTACK FAST DECAY CONTINUOUS
13=SLOW ATTACK END CONTINUOUS
14=SLOW ATTACK SLOW DECAY CONTINUOUS
15=SLOW ATTACK FAST DECAY END

ODD VALUES GIVE A SINGLE ENVELOPE WHICH IS FOLLOWED EITHER BY SILENCE (9.15) OR BY A CONTINUOUS HIGH LEVEL SOUND (11.13).

THE SINGLE SHOT ENVELOPES ARE TRIGGERED EACH TIME R13 IS LOADED WITH THE WAVE FORM VALUE.

How To Write Better Programs

How To Write Better Programs

INVENTING AND DEVELOPING YOUR OWN GAMES PROGRAMS

By Series Editor, 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

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

BIBLIOGRAPHY

Compiled by 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

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 SPECTRAVIDEO

More than 25 challenging programs, each one especially written for the series and guaranteed to provide hours of entertainment.

The games include MINEFIELD (risk your life to save your comrades-in-arms); ROAD RACE (can you control a turbo-charged car?); STAR STRIKE (cosmic fire balls threaten your life and your spaceship!); TOWERS OF DOOM (in this fully-fledged adventure you play the hero, while avoiding the ghouls and goblins); and NIGHT FIGHTER (all that lies between the aliens and the earth's destruction is you and a faulty radar screen). The book ends with a dramatic series of graphic demonstrations, and a brief chapter on making effective use of the Spectravideo's sound.

GAMES FOR YOUR SPECTRAVIDEO 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 extend the programs in the book.

Programs of originality and quality for all the family.

0 86369 047 5

Virgin

United Kingdom £2.95
Australia $9.95 (recommended)