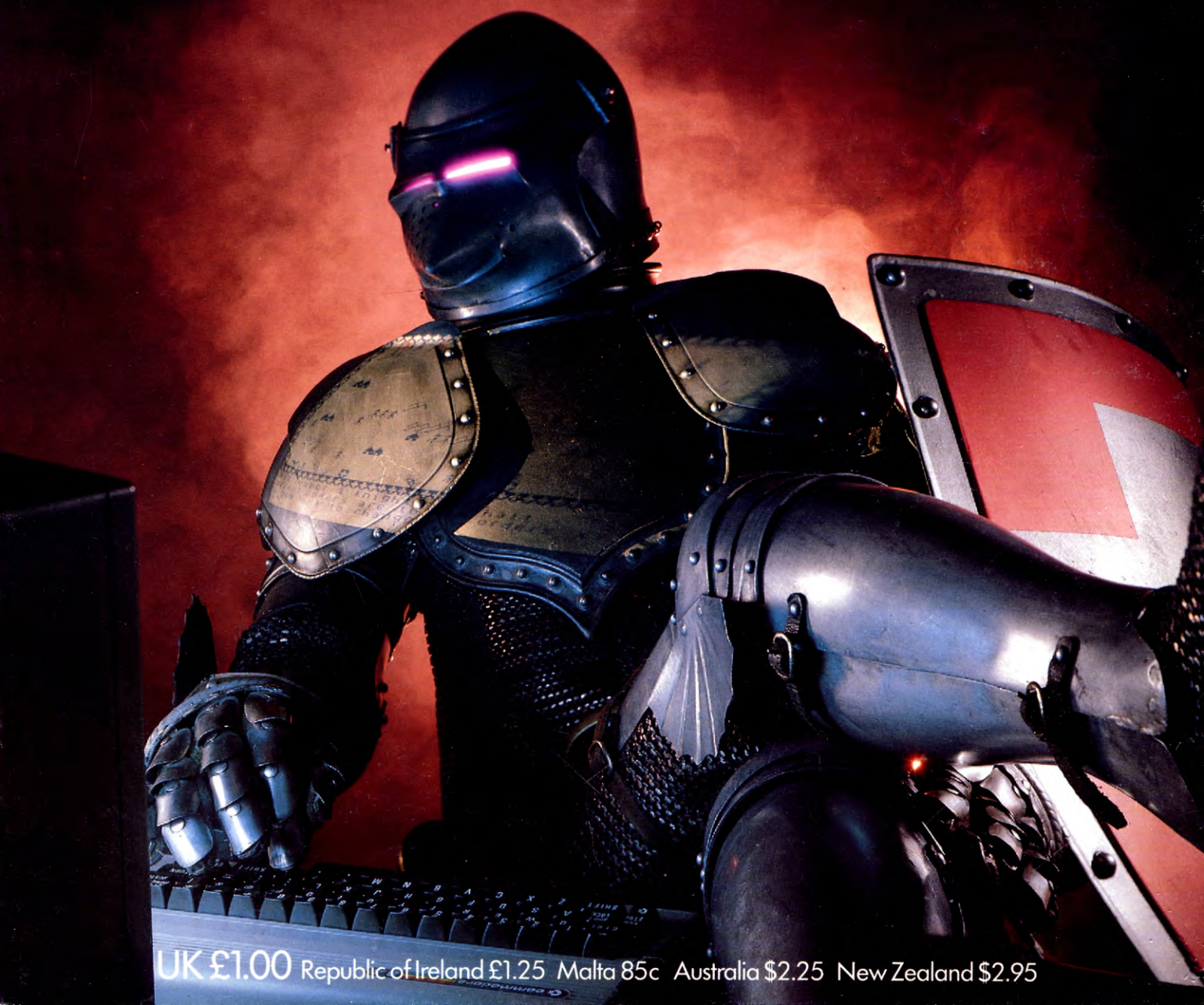
A MARSHALL CAVENDISH 40 COMPUTER COURSE IN WEEKLY PARTS

INPUT

LEARN PROGRAMMING - FOR FUN AND THE FUTURE



UK £1.00 Republic of Ireland £1.25 Malta 85c Australia $2.25 New Zealand $2.95

INPUT

Vol. 4 **No 40**

INDEX

The last part of INPUT, Part 52, will contain a complete, cross-referenced index. For easy access to your growing collection, a cumulative index to the contents of each issue is contained on the inside back cover.

PICTURE CREDITS

Front cover, George Logan. Pages 1237, 1238, 1239, Jeremy Gower. Pages 1240, 1241, The Image Bank/© Production Co./Bernard Fallon. Page 1242, Spectrum. Pages 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, Peter Beard. Pages 1254, 1255, 1256, 1257, George Logan. Pages 1258, 1259, Chen Ling. Pages 1264, 1265, 1266, 1267, Dave King.

Published by Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA, England. Typeset by MS Filmsetting Limited, Frome, Somerset. Printed by Cooper Clegg Web Offset Ltd, Gloucester and Howard Hunt Litho, London.

HOW TO ORDER YOUR BINDERS

UK and Republic of Ireland: Send £4.95 (inc p & p) (IR£5.95) for each binder to the address below:
Marshall Cavendish Services Ltd, Department 980, Newtown Road, Hove, Sussex BN3 7DN

Australia: See inserts for details, or write to INPUT, Times Consultants, PO Box 213, Alexandria, NSW 2015

New Zealand: See inserts for details, or write to INPUT, Gordon and Gotch (NZ) Ltd, PO Box 1595, Wellington

Malta: Binders are available from local newsagents.

There are four binders each holding 13 issues.

BACK NUMBERS

Back numbers are supplied at the regular cover price (subject to availability).

UK and Republic of Ireland:
INPUT, Dept AN, Marshall Cavendish Services, Newtown Road, Hove BN3 7DN

Australia, New Zealand and Malta:
Back numbers are available through your local newsagent.

COPIES BY POST

Our Subscription Department can supply copies to any UK address regularly at £1.00 each. For example the cost of 26 issues is £26.00; for any other quantity simply multiply the number of issues required by £1.00. Send your order, with payment to:

Subscription Department, Marshall Cavendish Services Ltd, Newtown Road, Hove, Sussex BN3 7DN

Please state the title of the publication and the part from which you wish to start.

HOW TO PAY: Readers in UK and Republic of Ireland: All cheques or postal orders for binders, back numbers and copies by post should be made payable to: *Marshall Cavendish Partworks Ltd.*

QUERIES: When writing in, please give the make and model of your computer, as well as the Part No., page and line where the program is rejected or where it does not work. We can only answer specific queries – and please do not telephone. Send your queries to INPUT Queries, Marshall Cavendish Partworks Ltd, 58 Old Compton Street, London W1V 5PA.

INPUT IS SPECIALLY DESIGNED FOR:

The SINCLAIR ZX SPECTRUM (16K, 48K, 128 and +), COMMODORE 64 and 128, ACORN ELECTRON, BBC B and B+, and the DRAGON 32 and 64.

In addition, many of the programs and explanations are also suitable for the SINCLAIR ZX81, COMMODORE VIC 20, and TANDY COLOUR COMPUTER in 32K with extended BASIC. Programs and text which are specifically for particular machines are indicated by the following symbols:

SPECTRUM 16K, 48K, 128, and + **COMMODORE 64 and 128**

ACORN ELECTRON, BBC B and B+ **DRAGON 32 and 64**

ZX81 **VIC 20** **TANDY TRS80 COLOUR COMPUTER**

MACHINE CODE LIFE GAME

- THE RULES OF LIFE
- INITIAL PATTERN
- A GENERATION COUNTER
- CREATING THE COLONY
- MACHINE CODE DATA

Tell your micro a few simple rules and you can simulate the struggle for life of uni-cellular organisms. The trick of this game is to set up the colony for optimum survival

In this game the TV screen represents the world in which cells can live, multiply and die. The screen is divided into invisible squares, each of which can contain one of the cells. If you look at the diagram of the grid you'll see that each cell can have a maximum of eight neighbours. The rules of Life determine whether a cell lives or dies or whether a new cell is born.

The rules are:

- a cell is born into a space that has exactly three neighbours
- a cell survives to the next generation if it has two or three neighbours
- all other cells die

The computer uses these rules to determine the future of every square on the screen and shows how an initial colony of cells develops. In BASIC this would be impossibly slow so the program is written in machine code allowing each generation to be displayed in about one second.

The program also displays successive generations in a different colour so the effect is quite mesmerizing as you watch the original colony spreading out, breaking up into smaller groups, dying out or rejuvenating.

The shape of the initial colony is crucial as some patterns die out after only a few generations while others last for hundreds. Your job is to input the cells that make up the initial colony. One of the aims of this game is to devise an initial pattern that will last the greatest number of generations—there is a generation counter that keeps you informed of how well your colony is doing. But it can be more interesting to create a colony that develops in an intriguing way. The diagrams show a few interesting initial patterns to try out.

CREATING THE COLONY

Here's how to input the initial pattern.

On the Spectrum use the cursor keys to move round the screen, ENTER to create a cell, M to delete a cell and Q to finish.

On the Commodore use the cursor keys to move round the screen, the space bar to enter a cell, DEL to delete and RETURN to finish.

On the Acorn use the Z, X, P and L keys to move the cursor, the space bar to toggle a cell on or off, and RETURN to finish.

Finally, on the Dragon and Tandy use the arrow keys to move, the space bar to toggle a cell on and off, and ENTER to finish.

Now type in and try the program yourself:

```
5 CLEAR 28671: FOR N=USR "A" TO USR
  "A"+7: READ A: POKE N,A: NEXT N
6 DATA 0,24,60,102,102,60,24,0
7 GOSUB 200
10 POKE 23658,8: BRIGHT 1: BORDER 0: INK
   6: PAPER 0: CLS
20 FOR N=0 TO 21: PRINT AT N,0;
   PAPER 1;"□□□□□□□□□□□□□□□□":
   NEXT N
30 PLOT 63,0: DRAW 0,175: DRAW 192,0:
   DRAW 0,-175: DRAW -255,0: DRAW
   0,175: DRAW 63,0
40 PRINT AT 2,1; PAPER 2; INK
   7;"□LIFE□"; PAPER 0; INK 6;AT
   5,1;"□GEN□□";AT 6,1;"□0000□"
70 RANDOMIZE USR 28672: LET X=20: LET
   Y=10
80 PRINT AT Y,X; OVER 1;"■": FOR N=1
   TO 10: NEXT N: PRINT AT Y,X; OVER
   1;"■"
90 LET A$=INKEY$: IF A$="" THEN GOTO
   80
92 IF A$="Q" THEN GOTO 110
94 IF CODE A$=13 THEN PRINT AT
   Y,X;CHR$ 144: POKE 30000+
   ((Y-1)*23)+(X-8),144: GOTO 80
95 IF A$="M" THEN PRINT AT Y,X;"□":
   POKE 30000+((Y-1)*23)+
   (X-8),32: GOTO 80
100 LET X=X-(A$="5")*(X>8)+(A$=
   "8")*(X<30): LET Y=Y-(A$="7")*
   (Y>1)+(A$="6")*(Y<20): GOTO 80
110 OVER 0: RANDOMIZE USR 28711
120 PRINT AT 21,7; FLASH 1;"ANY KEY TO
   RESTART": FOR N=1 TO 200: NEXT N
130 LET A$=INKEY$: IF A$="" THEN
   GOTO 130
140 GOTO 10
200 LET L=500: RESTORE L: FOR N=28672
   TO 28951 STEP 8
210 LET T=0: FOR D=0 TO 7
220 READ A: POKE N+D,A: LET T=T+A:
   NEXT D: READ A: IF A<>T THEN PRINT
   FLASH 1;"DATA ERROR AT LINE□";L:
   STOP
230 LET L=L+10
240 NEXT N
250 RETURN
500 DATA 33,25,117,1,211,3,62,32,484
510 DATA 119,35,13,32,249,5,242,6,701
520 DATA 112,33,48,48,34,0,113,34,422
530 DATA 2,113,33,48,117,34,252,112,711
540 DATA 33,248,118,34,254,112,201,243,
   1243
550 DATA 42,252,112,6,1,197,88,6,704
560 DATA 8,80,62,22,215,123,215,122,847
570 DATA 215,126,35,254,32,40,10,214,926
580 DATA 142,245,62,16,215,241,215,62,1198
590 DATA 144,215,4,120,254,31,32,233,1033
600 DATA 193,4,120,254,21,32,214,42,880
```

```
610 DATA 252,112,237,91,254,112,229,213,
    1500
620 DATA 1,200,1,197,221,33,4,113,770
630 DATA 1,0,7,213,221,94,0,221,757
640 DATA 86,1,229,25,235,225,26,254,1081
650 DATA 32,209,40,1,12,221,35,221,771
660 DATA 35,5,242,107,112,126,254,144,1025
670 DATA 121,56,18,254,2,40,4,254,749
680 DATA 3,32,14,126,254,32,32,11,504
690 DATA 58,251,112,24,6,254,3,40,748
700 DATA 242,62,32,18,35,19,193,13,614
710 DATA 32,185,5,242,99,112,209,225,1109
720 DATA 237,83,252,112,34,254,112,33,1117
730 DATA 4,113,43,126,60,254,58,40,698
740 DATA 4,119,195,203,112,62,48,119,862
750 DATA 195,186,112,62,22,215,62,6,860
760 DATA 215,62,2,215,33,0,113,62,702
770 DATA 16,215,62,6,215,6,4,126,650
780 DATA 35,215,16,251,58,251,112,60,998
790 DATA 254,151,32,2,62,144,50,251,946
800 DATA 112,62,127,219,254,31,218,40,
    1063
810 DATA 112,251,201,149,248,118,48,117,
    1244
820 DATA 48,49,54,49,233,255,234,255,1177
830 DATA 1,0,24,0,23,0,22,0,70
840 DATA 255,255,232,255,0,0,0,0,997
```

```
5 FOR Z=0 TO 278:READ X:POKE
  49152+Z,X:CO=CO+1:NEXT Z
6 IF CO<>279 THEN PRINT "DATA
  ERROR!":STOP
10 POKE 53280,6:POKE 53281,0:POKE
   198,0:POKE 2,1
20 PRINT "■■":FOR Z=0 TO 39:POKE
   1024+Z,160:POKE 1984+Z,160
25 POKE 55296+Z,14:POKE
   56256+Z,14:NEXT Z
30 X=500:C=32
35 N=0:FOR Z=1064 TO 1064+22*40
   STEP 40
40 POKE 832+N,Z-(INT(Z/256)*256):POKE
   857+N,Z/256
45 N=N+1:NEXT Z
100 GET A$
105 POKE 1024+X,C:IF A$=CHR$(13)
    THEN POKE 198,0:GOTO 200
110 IF A$="■" AND X-1>39 THEN
    X=X-1
120 IF A$="■" AND X+1<960 THEN
    X=X+1
130 IF A$="■" AND X-40>39 THEN
    X=X-40
140 IF A$="■" AND X+40<960 THEN
    X=X+40
150 C=PEEK(1024+X):POKE 1024+
    X,102:POKE 55296+X,3
160 IF A$="□" THEN C=87
170 IF A$=CHR$(20) THEN C=32
180 GOTO 100
190 PRINT "■■";:NU=NU+1:SYS
    49152:IF PEEK(197)<>64 THEN RUN 10
200 GOTO 190
210 DATA 169,0,133,251,32,159,192,166,251,
    189,64,3,133,252,133,254,189
220 DATA 89,3,133,253,24,105,190,133,255,
    160,0,177,252,201,32,208
230 DATA 16,200,192,40,208,245,230,251,
    165,251,201,23,208,216,76,178,192
240 DATA 224,0,240,12,192,0,240,8,192,39,
    240,4,224,23,208,3,76,34,192
250 DATA 136,177,254,24,105,1,145,254,200,
    200,177,254,24,105,1,145,254
260 DATA 136,136,166,251,202,189,64,3,133,
    252,189,89,3,24,105,190,133
270 DATA 253,32,15,193,200,32,15,193,200,
    32,15,193,232,232,189,64,3,133
280 DATA 252,189,89,3,24,105,190,133,253,
    136,136,32,15,193,200,32,15
290 DATA 193,200,32,15,193,136,166,251,
    189,64,3,133,252,189,89,3,133,253
300 DATA 76,34,192,162,0,138,157,40,194,
    157,0,195,157,0,196,157,0,197,232
310 DATA 208,241,96,162,0,189,64,3,133,
    252,133,254,189,89,3,133,253,24
320 DATA 105,190,133,255,160,0,177,254,
    201,2,144,21,201,2,240,39,201,4
325 DATA 176,13,177,252
330 DATA 201,87,240,29,169,87,145,252,76,
    230,192,169,32,145,252,165,253
340 DATA 24,105,212,133,253,165,2,145,252,
    165,253,24,233,211,133,253
350 DATA 200,192,40,208,202,232,224,23,
    208,178,230,2,165,2,201,16
```

```
360 DATA 208,4,169,1,133,2,96,177,252,24,
    105,1,145,252,96
```

```
10 MODE1:VDU 19,1,3,0,0,0,19,2,5,0,0,0,23,
   224,0,60,126,102,102,126,60,0
20 DIM MC□300,W1□2440:W1=W1+
   40:W2=W1+1200:T1=&74:T2=&75
30 FOR T=0 TO 7:READ MC?T:NEXT:
   ADD=MC:PROCASS
40 CLS:FOR T=W1-40 TO W1+1199:
   ?T=0:NEXT:X=19:Y=15:PRINT"ENTER
   YOUR PATTERN NOW"
50 VDU 31,X,Y+1:A$=GET$
60 IF A$="Z" AND X>0 THEN X=X-1
70 IF A$="X" AND X<39 THEN X=X+1
80 IF A$="P" AND Y>0 THEN Y=Y-1
90 IF A$="L" AND Y<29 THEN Y=Y+1
100 M=W1+Y*40+X:IF A$="□" THEN
    ?M=&3E OR ?M:VDU (?M*64)+32,8
110 IF A$<>CHR$(13) THEN 50
120 G=0:PRINTTAB(0,0)"GENERATION
    NUMBER"SPC(10):VDU 23;8202;0;0;0;
130 ?T2=(?T2+1) AND 3:IF ?T2=0 THEN
    ?T2=1
140 G=G+1:PRINTTAB(20,0);G:CALL LIFE:
    COLOUR3:*FX15,0
150 IF INKEY(-99) THEN PRINTTAB(0,10)
    "LIFE TERMINATED AFTER□";G;"□GEN-
    ERATIONS""PRESS RETURN" ELSE 130
160 IF GET=13 THEN 40 ELSE 160
170 DATA 0,1,2,40,42,80,81,82
180 DEF PROCASS:FOR T=0 TO 2 STEP
    2:P%=MC+8:[OPT T
190 .LIFE:JSR SET:.L2:JSR CELL:JSR BUMP
200 JSR CHECK:BCC L2:JSR MVE:JMP SHOW
210 .CELL:LDA #0:STA T1:LDX #8:LDA#0
220 STA T1:LDX #8:.C2:DEX:BMI C3:LDA
    ADD,X:TAY:LDA (&70),Y:BEQ C2:INC
    T1:BNE C2:.C3:LDX T1
230 .C3:LDX T1:LDY #41:LDA (&70),Y:CPX
    #2:BEQ C5:CPX #3:BNE C4:CMP
    #0:BNE C5:LDA T2:]:?P%=&2C:
    P%=P%+1:[OPT T
240 .C4:LDA #0:.C5:STA (&72),Y:RTS
250 .SET:LDA #(W1-41)MOD 256:STA &70
260 LDA #(W1-41)DIV 256:STA &71
270 LDA #(W2-41)MOD 256:STA &72
280 LDA #(W2-41)DIV 256:STA &73:RTS
290 .BUMP:INC &72:BNE B2:INC &73
300 .B2:INC &70:BNE B3:INC &71
310 .B3:RTS
320 .CHECK:LDA &70:CMP #(W2-41)MOD
    256:LDA &71:SBC #(W2-41)DIV 256:RTS
330 .MVE:LDY #41:JSR SET
340 .M2:LDA (&72),Y: STA (&70),Y:JSR
    BUMP:JSR CHECK:BCC M2:RTS
350 .SHOW:LDA #30:JSR &FFEE:LDA
    #10:JSR &FFEE:JSR SET:LDY #41
360 .SH2:LDA #17:JSR &FFEE:LDA
    (&70),Y:PHP:JSR &FFEE
370 PLP:BEQ SH3:LDA#224:]:?P%=&2C:
    P%=P%+1:[
380 .SH3:LDA #32:JSR &FFEE:JSR B2:JSR
    CHECK:BCC SH2:RTS:]NEXT:ENDPROC
```

Six generations exist at once

A stable colony lives on

This program will not run if a disk drive is attached, as these use the same memory area:

```
10 CLS0: CLEAR 200, 30999
20 FORK=0TO11:T=0:FORJ=0TO16:READ
   A:POKE31000+K*17+J,A:T=T+A:
   NEXT:READS
30 IF T<>S THEN PRINT"□DATA ERROR
   IN LINE";1000+K*10;"DO NOT RUN
   !!":END
40 NEXT
50 CLS0:FORK=0TO3:PRINT@K*32,
   "generation";:POKE31203+K,48:NEXT:
   POKE65475,0
60 PMODE3:POKE179,128:PCLS
70 DEFUSR0=31000
80 X=2064:Y=1
90 P=PEEK(X):POKEX,(P□OR5*Y)AND(NOT
   (P□AND5*Y))
100 A$=INKEY$:IF A$="□" THEN90
110 POKEX,P
120 IF A$="↑" AND X>1183 THEN
    X=X-32
130 IF A$=CHR$(10) AND X<3040 THEN
    X=X+32
140 IF A$=CHR$(8) AND(X>1152 OR
    (X=1152ANDY=1)) THENY=Y+1:IF
    Y>2 THEN Y=1:X=X-1
150 IF A$=CHR$(9) AND(X<3071 OR
      (X=3071ANDY=2)) THENY=Y-1:IF
    Y<1 THENY=2:X=X+1
160 IF A$=CHR$(13) THEN180
170 GOTO 90
180 H=USR0(0)
190 IF INKEY$<>"Q" THEN 180
200 GOTO50
1000 DATA 182,121,226,139,16,183,121,226,
     142,121,227,108,132,166,128,129,58,2425
1010 DATA 37,9,134,48,167,31,140,121,231,
     37,239,198,4,206,4,11,142,1759
1020 DATA 121,231,166,130,167,192,140,
     121,227,38,247,51,200,28,90,38,238,2425
1030 DATA 142,13,0,204,0,0,237,129,140,28,
     128,37,249,142,4,128,206,1787
1040 DATA 13,64,166,128,133,10,39,2,141,
     47,51,65,133,5,39,2,141,1179
1050 DATA 39,51,65,140,12,0,37,233,142,4,
     128,206,13,64,166,132,52,1484
1060 DATA 2,230,192,134,10,141,52,230,192,
     134,5,141,46,53,2,167,128,1859
1070 DATA 140,12,0,37,231,57,52,2,108,200,
     192,108,200,64,31,48,196,1678
1080 DATA 63,39,8,108,200,191,108,95,108,
     200,63,193,63,39,8,108,200,1794
1090 DATA 193,108,65,108,200,65,53,130,
     165,98,39,19,193,2,39,32,193,1702
1100 DATA 3,39,28,230,98,67,52,2,228,224,
     231,98,32,17,193,3,38,1583
1110 DATA 13,230,98,196,143,250,121,226,
     52,2,234,224,231,98,57,0,0,2175
```

CLIFFHANGER: CLOUDING OVER

Meteorology may not be your strong point, but now's the time to add a bit of weather to the game. Cliffhanger now gives you clouds, sun or sound effects

As if Willie did not have enough problems with the goats and the sea, there is a cloud threatening to rain on his picnic! But don't worry, the cloud is menacing only on the Spectrum and Commodore versions of Cliffhanger.

On the Acorn version of the game, the cloud is a fair weather cumulus which is unlikely to rain on anything. It doesn't even move about, so Acorn owners are being given some special sound effects for their game this week.

And on the Dragon, Willie is even luckier. It is a brilliantly hot day with not a cloud in the sky. But the sun does sail through the summer sky during the course of the game.

The following routine sends the cloud sailing through the sky:

```
        org 58795
cld     ld a,(57347)
        dec a
        ld (57347),a
        cp Ø
        jr z,cdm
        ret
cdm     ld a,6
        ld (57347),a
        ld a,45
        ld bc,16384
        ld hl,(57345)
        ld d,3
        ld e,2
        call blk
        ld a,(57348)
        cp Ø
        jr z,crt
        dec hl
        jr chm
crt     inc hl
chm     ld (57345),hl
        ld bc,57144
        ld a,47
        ld d,3
        ld e,2
        call blk
        ld de,129
        sbc hl,de
        jr nz,cnr
        ld a,Ø
        ld (57348),a
        ret
cnr     ld de,144
        ld hl,(57345)
        sbc hl,de
        jr nz,cnl
        ld a,1
        ld (57348),a
cnl     ret
        org 5897Ø
blk     *
```

Memory location 57,347 contains the cloud delay. This tells the cloud how fast to move across the sky. The speed is specified when the level is initialized by the number loaded into this location.

The cloud delay is loaded into the accumulator, decremented and loaded back. If it has been decremented to zero, the **cp Ø** and the **jr z** instructions jump the return instruction and go on to the rest of the routine which moves the cloud. If the delay hasn't counted down to zero, though, the processor returns and postpones moving the cloud until the routine is called again later.

NOTHING BUT BLUE SKIES

The first thing the routine has to do is reset the cloud delay. If it was left at zero the routine would have to be called 256 times before it counted down to zero again—which would mean that the cloud was moving mighty slow. So the number 6 is loaded into the accumulator and put into 57,347. This puts a fairly stiff breeze behind the cloud so that you can see it moving on the screen.

The number 45 is then loaded into the accumulator. It is going to be carried into the **blk** routine in this register. The **blk** routine prints a block of characters. The dimensions of the block are carried in D and E—D carries the number of columns and E the number of rows. The **blk** routine itself calls the **print** routine that has been used throughout this game. So HL should carry the screen position, BC the data pointer and A the colour. 45 is cyan on cyan—in other words, the cloud is blanked out by printing over it.

BC is loaded with the address of the start of the screen. This will give the appropriate blue sky data. HL is loaded with the contents of 57,345 and 57,346 which are the pointers to the current position of the cloud.

The cloud is three columns wide and two rows deep. So 3 is loaded into D and 2 is loaded into E. Then the **blk** routine is called to print up the block of blue sky.

WHICH WAY BLOWS THE WIND?

Memory location 57,348 is used as a flag to tell the cloud which way the wind is blowing. It is originally set by the initialization routine. If it is set to 1, the wind is blowing to the left. And if it is Ø, the wind is blowing to the right.

The contents of 57,348 are loaded into the accumulator and compared with Ø. If they are Ø, the **jr z** instruction jumps to the **crt** label and the cloud position pointer in HL is incremented—which moves the pointer one character square to the right.

If they are not Ø—in other words, the contents of 57,348 are 1—the **jr z** instruction does not operate and HL is decremented—which moves the cloud position pointer one character square to the left. The next instruction—**jr**—simply jumps over the **inc** instruction.

BRING ON THE CLOUDS

The cloud position pointers in 57,345 and 57,346 are updated by loading the contents of HL back into them. Then the data pointer in BC is loaded with the address of the beginning of the cloud data.

A is loaded with 47, which is white on cyan—the cloud colour. D is loaded with 3 and E is loaded with 2 again. The cloud is the

- CHECKING FOR WIND
- DELAYED ACTION
- REVERSING DIRECTION
- ENVELOPES
- CLOUD SYNCHING

The 'CLIFFHANGER' listings published in this magazine and subsequent parts bear absolutely no resemblance to, and are in no way associated with, the computer game called 'CLIFF HANGER' released for the Commodore 64 and published by New Generation Software Limited.

same size whether you are blanking it out or drawing it in.

Then the **blk** routine is called which prints the cloud in its new position up on the screen.

You must then check whether the cloud has reached the edge of the screen, otherwise it could scroll over the end and start appearing one row up on the other side of the screen when you next try to move it.

First the routine checks to see whether it has reached the extreme left of its path. DE is loaded with 129—the screen position of the left-hand end of its track—and the contents of DE are subtracted from the screen position pointer in HL. If the result is not zero, it has not reached the leftmost end and the **jr nz** instruction jumps on to check whether it has reached the right-hand end.

But if the result is zero, it has reached the leftmost end and the jump is not made. The accumulator is loaded with Ø and that is loaded back into the wind direction flag to tell the cloud to go to the right next time.

The **cnr** routine checks to see if the cloud had reached the extreme rightmost end of its path by subtracting 144 from the position pointer. If it is at the far right, 1 is loaded back in the wind direction flag to tell the cloud to go to the left next time.

ON THE BLOCK

The following routine prints a block of character squares on the screen, D columns by E rows:

```
        org 58970
blk     push hl
blj     push de
        push hl
z       push de
        call print
        inc hl
        pop de
        dec d
        jr nz,z
        pop hl
        ld de,32
        add hl,de
        pop de
        dec e
        jr nz,blj
        pop hl
        ret
        org 58217
print   *
```

Obviously the cloud moving program will not work until this routine is in memory too, as it calls the **blk** routine.

Again, this routine won't work without the **print** routine in memory, as this routine is called as well.

ALRIGHT SQUARE?

The screen position pointer in HL and the numbers of columns and rows in the DE are stored on the stack twice. This is because the routine has to work in two dimensions, using two loops.

Once the parameters have been stored the **print** routine is called which prints the first character up on the screen.

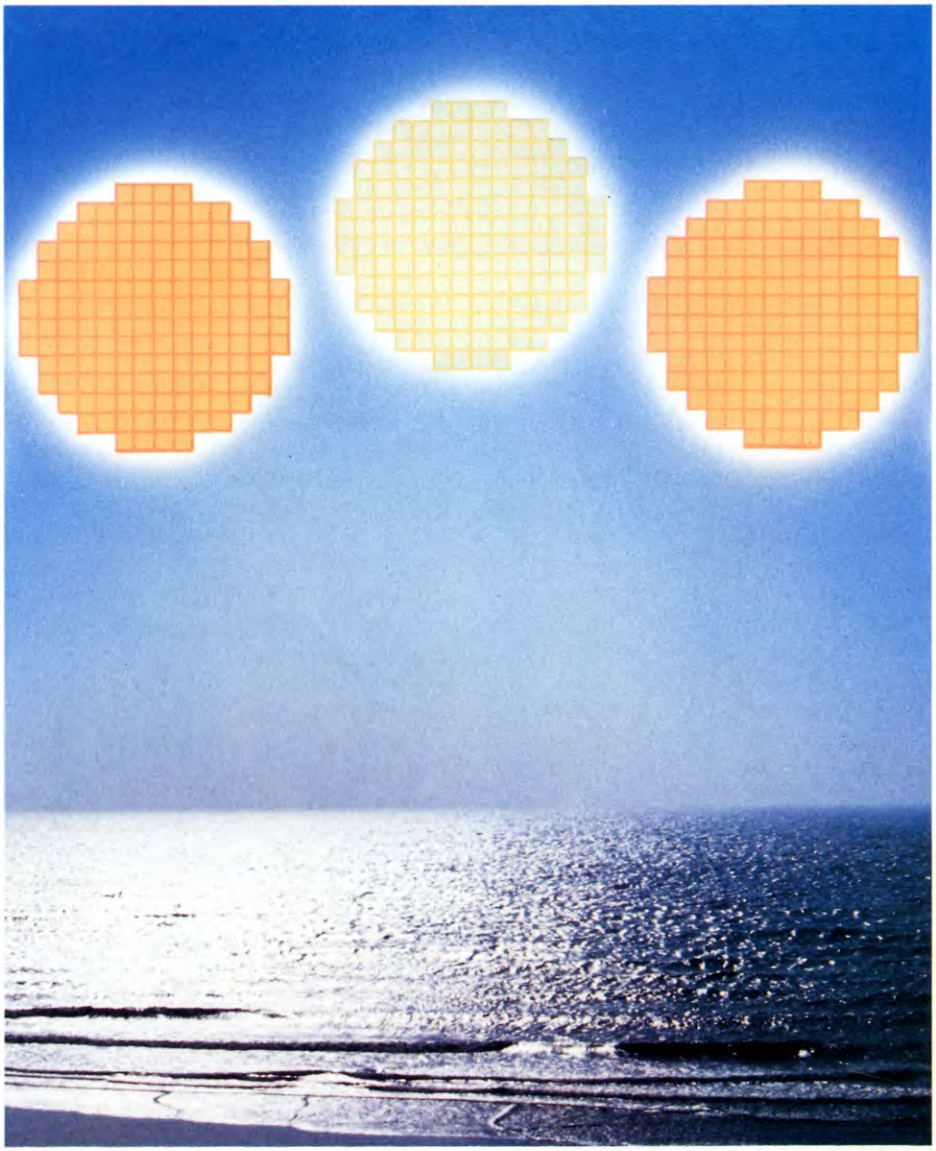
HL is then incremented to move it along to the next character square to the right. The block dimensions are popped off the stack and the horizontal parameter—the number of columns—is decremented. If it has not counted down to zero, the **jr nz** instruction jumps back to the beginning of the loop again to print up the next character square in the row.

When D has decremented to zero, the first row of the block has been completed.

Then HL is popped off the stack and 32 is added to it. This operation is done with the DE register as it is a two-byte addition. The result is in HL so the screen pointer is effectively moved down the screen one line. DE is popped off the stack again to reset D and the E register is decremented. E counts down the rows.

If it has not counted down to zero the **jr nz** instruction sends the processor back to the beginning of the loop to start printing up the next row.

When E has counted down to zero, the processor drops out of the loop and HL is popped off the stack. This resets the screen position pointer.



The following routine moves the cloud back and forth across the screen, depending on which way the wind is blowing:

```
        ORG 22528
        LDA $CØØB
        BEQ LEFT
        INC $DØØ4
        LDA $DØØ4
        CMP #255
        BNE RET
        LDA #Ø
        STA $CØØB
RET     RTS
LEFT    DEC $DØØ4
        LDA $DØØ4
        CMP #6Ø
        BNE RET
        LDA #1
        STA $CØØB
        RTS
```

Memory location $CØØB is used as a flag to tell the cloud which way the wind is blowing. A Ø in that location indicates that the wind is blowing to the left, a 1 means it is blowing to the right.

This flag is loaded into the accumulator. If it is zero the BEQ instruction branches forward to the LEFT routine. If not—that is, if the contents of $CØØB are 1—the processor continues.

The X coordinate of the cloud's position is stored in memory location $DØØ4. The contents of this location are incremented because the cloud is moving to the right.

The next X coordinate is then loaded into the accumulator. The cloud sprite only moves up to the position with an X coordinate of 255 (this saves having to deal with the most significant bit).

If the X coordinate has not reached 255, the BNE RET instruction branches forward to the RTS instruction and the processor returns. But if the cloud has reached its most extreme right-hand position and its X coordinate has reached 255 the branch is not made and the accumulator is loaded with Ø. This is stored back in the wind direction flag to tell the cloud to move to the left next time.

BY THE LEFT

The LEFT routine works in much the same way as the routine that moves the cloud to the right.

This time though the X coordinate is decremented and the result is compared with 6Ø—which is the X coordinate of the extreme left-hand end of the cloud's path. If it has reached this point, a 1 is stored back in the wind direction flag to tell the cloud to move to the right next time.

Being short of cloud movement, this oppor-

tunity has been seized to give you some of the Acorn's extensive range of sound effects. And there will be more in a forthcoming part.

This program produces the crunch, the hiss, and a loud and a quiet envelope for the tune. It also has a provision to turn the sound off, if you want to have a quiet game. Don't forget to set up the computer as normal before you key it in.

```
30 FORPASS=0TO3STEP3
40 RESTORE
80 DATA4,1,0,0,0,0,0,0,126,255,255,255,126,
   126
90 FORA%=&1491TO&149E:READ?A%:NEXT
100 DATA16,0,4,0,5,0,20,0
110 FORA%=&14A0TO&14A7:READ?A%:
   NEXT
120 P%=&14A9
130 [OPTPASS
140 .Crunch
150 JSROff
160 LDA#8
170 LDX#&91
180 LDY#&14
190 JSR&FFF1
200 LDA#7
210 LDX#&A0
220 LDY#&14
230 JSR&FFF1
240 RTS
250 ]
290 DATA0,0,4,0,4,0,20,0
300 FORA%=&14BFTO&14C6:READ?A%:
   NEXT
310 P%=&14C8
320 [OPTPASS
330 .Hiss
340 JSROff
350 LDA#8
360 LDX#&91
370 LDY#&14
380 JSR&FFF1
390 LDA#7
400 LDX#&BF
410 LDY#&14
420 JSR&FFF1
430 RTS
440 ]
510 P%=&152B
520 [OPTPASS
530 .Off
540 LDA&80
550 AND#128
560 BEQLb1
570 PLA
580 PLA
590 .Lb1
600 RTS
610 ]
660 DATA2,1,0,0,0,0,0,0,126,255,255,255,
   126,126
670 FORA%=&1B6ATO&1B77:READ?A%:
   NEXT
680 P%=&1B78
690 [OPTPASS
700 .Loud
710 LDA#2
720 STA&1B6A
730 LDA#8
740 LDX#&6A
750 LDY#&1B
760 JSR&FFF1
770 LDA#3
780 STA&1B6A
790 LDA#8
800 LDX#&6A
810 LDY#&1B
820 JSR&FFF1
830 RTS
840 ]
890 DATA2,2,0,0,0,0,0,0,63,255,255,255,63,63
900 FORA%=&1B95TO&1BA2:READ?A%:
   NEXT
910 P%=&1BA3
920 [OPTPASS
930 .Quiet
940 LDA#2
950 STA&1B95
960 LDA#8
970 LDX#&95
980 LDY#&1B
990 JSR&FFF1
1000 LDA#3
1010 STA&1B95
1020 LDA#8
1030 LDX#&95
1040 LDY#&1B
1050 JSR&FFF1
1060 RTS
1070 ]NEXT
```

To test the sound effects load the rest of the game into memory and call them in turn:

```
?&80=0:CALL &14A9
```

gives the crunch;

```
CALL &14C8
```

makes the snakes hiss;

```
CALL &1B78:?&80=4:REPEAT:CALL &1100:
   UNTIL ?&80=0
```

calls the loud envelope and

```
CALL &1BA3:?&80=4:REPEAT:CALL &1100:
   UNTIL ?&80=0
```

calls the quiet envelope.

CRUNCH

The crunch envelope DATA is in Line 80 and the sound DATA is in Line 100. This is then READ into data tables at &1491 to &149E and &14A0 and &14A7 by Lines 90 and 110.

The assembly language routine starts off by jumping to the subroutine labelled Off. This is the routine that switches the sound off, so the crunch routine quickly checks to see whether the sound effect is required.

A is then loaded with 8. This parameter fixes the envelope when it's output through the routine at &FFF1. X is loaded with &91 and Y is loaded with &14. These are the low and high bytes of the address of the envelope data table. Then the routine at &FFF1 is called and the envelope is defined by the data.

A is then loaded with 7, which gives the SOUND command when &FFF1 is called. X is loaded with &A0 and &14. These are the low and high bytes of the address of the beginning of the sound data table. And when this data is output, the sound of the crunch is made.

HISS

The hiss has its own sound data in Line 290. The same envelope as the crunch is used.

The hiss routine works much the same as the crunch. Off is checked to see if the sound is wanted. The envelope data is output through &FFF1, followed by the hiss data which makes the sound.

THE SOUND OF SILENCE

To switch the sound off, memory location &80 is POKEd with 128—in other words, bit seven is set.

The off routine checks this by loading the contents of &80 into the accumulator and ANDing it with 128. If bit seven is set the processor returns directly from the routine. If not, the stack is decremented twice before the processor returns. This pulls the last return address—that is, the one where the Off routine was called—off the stack. So when the processor hits the RTS at the end of the Off routine, it returns to the place where the sound effect routine was called. And as the Off routine is called before the sound effect routine is executed, no sound is made.

TURN IT UP

The data for the loud envelope is in Line 660. Note that parameters 9, 13 and 14—which control the attack and decay volume—are set to their maximum.

The instructions in Lines 710 and 720 put 2 into the first byte of the envelope data. This is a little superfluous as it already contains 2. But this byte carries the channel number which is going to be changed later. For now, though, the envelope of channel two is being defined.

A is loaded with 8, and X and Y with the low and high bytes of the envelope data as before. Then the subroutine at &FFF1 is called which defines the envelope. Next 3 is loaded into the accumulator and stored in the first byte of the envelope data. The envelope for channel three is about to be defined.

A is loaded with 8 and the X and Y registers are loaded with the low and high byte of the address of the beginning of the data table again. Then the envelope is fixed by calling &FFF1.

The quiet envelope is set in exactly the same way. The DATA is in Line 89Ø. Here, though, the amplitude parameters are set to 63, halfway down their range. And, obviously, the X and Y registers are loaded with the start address of the data of this quiet envelope.

ET

Giving the Dragon and Tandy version of the game a cloud would have involved a lot of extra data. Besides, because of the colour set that has been used the cloud would have had to be blue, red or green, none of which is entirely appropriate. And as this version of the game takes place on an extremely hot day—witness the yellow sky—a cloud is not really necessary. Instead, the following routine moves the sun about:

```
          ORG 19727
MOVSUN    DEC 18258
          BNE SUNRET
          LDA #5
          STA 18258
          SYNC
          LDX #1569
          LDA #3Ø
MOVA      PSHS A
          LDA #2
MOVB      ANDCC #$FE
          PSHS CC
          CLRB
MOVC      PULS CC
          ROR B,X
          PSHS CC
          INCB
          CMPB #14
          BNE MOVC
          LSL ,X
          PULS CC
          ROR ,X
          DECA
          BNE MOVB
          LEAX 32,X
          PULS A
          DECA
          BNE MOVA
SUNRET    RTS
```

To test this routine key in:

```
1Ø EXEC 19426
2Ø EXEC 19727
3Ø GOTO 2Ø
```

Memory location 18,258 contains the so-called sun delay. This stops the sun whizzing about the sky like a flying saucer. The sun delay is set to five at first.

The first instruction of this routine decrements the sun delay. If it has not been decremented to zero, the BNE instruction branches forward to the RTS at the end of the routine and the processor returns. But if it has decremented to zero the BNE instruction does not make the branch and the processor continues. In other words, the movement of the sun is slowed by executing this routine only once every five times it is called.

When the routine is executed, the first thing it does is reset the sun delay by loading 5 into A and storing it back in 18,258.

IN THE SYNC

The SYNC command synchronizes the rest of the routine with the TV scanner. This is to prevent the sun's position from being changed while it is actually on the screen.

X is loaded with the screen address of the top left-hand corner of the sun. A is loaded with 3Ø—the sun is 3Ø lines deep. A is then pushed onto the stack to preserve it. And A is loaded again with 2. This is a loop counter which will cause the sun to be scrolled across the screen by two pixels.

The condition code register is ANDed with $FE. $FE is 1111111Ø, so the seven most significant bits of the condition code register are left unchanged while the least significant bit is cleared. The carry flag is bit zero of the condition code register so this instruction effectively clears the carry flag.

The condition code register is then pushed on the stack to preserve it. And the B register is cleared. This is going to be used as an offset. The routine is now set up ready to do the scrolling.

THE SUN ROTATES

The next part of the routine actually shifts the sun. First the condition code register is pulled off the stack. The first time round this loop this makes sure that the carry flag is clear.

ROR B,X then 'rotates', one place to the right, the bits in the screen location pointed to by X+B. The purpose of a rotate is to shift all the bits along one place: the end place that is emptied by this process is filled by the contents of the carry flag, and the bit that overflows out of the 'other end' of the location is in turn put into the carry flag.

So all the pixels that make up that part of the sun are shifted along one place to the right. The leftmost pixel in that location is switched on or off, depending what is in the carry flag, and the condition of the rightmost pixel is stored in the carry flag. The first time round the loop the leftmost pixel of the first byte is switched off, that is, it is set to the sky colour and the rest of that byte of the sun is shifted along one pixel.

The pixel that has been pushed into the carry flag is then preserved by pushing the carry flag back onto the stack. The loop counter in B is incremented and compared to 14—the part of the sky that the sun is on is 14 character squares wide, so comparing it to 14 checks to see whether the last byte of the area that has to be scrolled has been shifted.

If B has not counted up to 14 yet, the BNE instruction sends the processor back to deal with the next character square. When it has reached the end the processor drops out of the loop and continues.

LOOSE ENDS

This is a northern hemisphere game and the sun moves from left to right. It would be very unnatural if, when the sun reached the end of the screen, or at least bumped into the score, it bounced back and started travelling in the opposite direction.

Instead, the sun is going to scroll round and appear again at the other side of the screen rather smartly. There is no night in Cliffhangerland.

LSL ,X logically shifts, to the left, the contents of the screen position pointed to by X. X has not been updated during this part of the routine, so it still points to the left-hand end of the sun area. This shifts the left-hand-most bit out of the register. The rightmost bit that has been shifted out of the rightmost screen position is pulled off the stack back into the carry flag.

It is then rotated into the leftmost position on the screen by the ROR ,X. The counter in A is then decremented and, if it hasn't yet counted down to zero, the processor branches back to shift the sun along one more pixel.

When the sun has been shifted two pixels LEAX 32,X adds 32 to the contents of X and moves the screen position pointer onto the next line down the sun. The line counter is pulled off the stack back into the accumulator. This is decremented and the BNE instruction sends the processor back to scroll the next row of character squares if it has not counted down to zero.

If it has, the processor moves onto the RTS and returns.

ASKING THE STARS

- MONEY/CAREER
- COMMUNICATION/TRAVEL
- HOME
- AFFAIRS OF THE HEART
- ALL THE DATA YOU NEED
- CHARACTER PROFILES BY STAR SIGN
- PREDICTIONS FOR THE YEAR AHEAD
- FACT OR MUMBO-JUMBO?

Some people say that your character and destiny are determined by the stars. Type in *INPUT*'s horoscope program and find out about the real you, and what's in store in '85

Amuse your friends with *INPUT*'s horoscope program, a fun application for your micro. When you type in the program, you'll be able to retrieve character profile information, and predictions for the coming year. The predictions are under four separate headings—Money/Career, Communication/Travel, Home and Affairs of the Heart. Using a computer is far better than wading through pages of printed material, particularly if you have a number of people with different star signs.

Astrology and horoscopes have intrigued people from the earliest times, and cause much controversy in homes and scientific discussion everywhere. Some famous scientists such as Einstein and Jung put great faith in horoscopes, while most of their colleagues would refuse to have anything to do with them at all. In the last decade, the hornets' nest has been well and truly stirred by research by psychologists in Britain and France, which has, at last, turned up some evidence in favour of astrology which is difficult to refute. But whether a person's personality is actually governed by planetary position at birth is open to argument.

INPUT's program classifies people just by their sun sign—the sign you look under when reading horoscopes in newspapers—rather than the exact time of birth that an astrologer would have to use to give the most 'accurate' reading.

PROGRAM FEATURES

The program asks for your date of birth, and tells you which star sign you are. You then have the choice of asking for a character profile or a prediction for the next year.

If you choose the first option, the program chooses two statements about your star sign from the eight it has stored in memory. Choosing the second option will let you choose either Money/Career, Communications/Travel, Home or Affairs of the Heart. Two predictions will appear. The machine has only two predictions stored for each heading (for each star sign), so there is no variation in output if you choose the same prediction option, unlike the character option.

THE PROGRAM

There is one piece of program for each machine. The DATA lines which follow are common to all, although some small alterations need to be made. See the DATA section for what to do for your machine.

To accommodate all the DATA the Acorn program will only work in MODE 7, so the program will not run on the Electron.

```
5 POKE 23658,8
10 INPUT "ENTER YOUR DATE OF BIRTH
   (e.g.□□12,3,65)□";X,Y,Z
20 IF X<1 OR X>31 OR Y<1 OR Y>12
   THEN GOTO 10
40 GOSUB 700
90 CLS : PRINT AT 1,1;"YOUR STAR SIGN
   IS:□"; INVERSE 1;A$
100 PRINT AT 5,1;"□DO YOU WANT
   :-"'"TAB 3;"<1> A CHARACTER
   PROFILE"'TAB 3;"<2>A PREDICTION
   FOR 1985"'TAB 3;"<N> TO ENTER A
   NEW DATE."
120 LET K$=INKEY$: IF K$<"1" OR K$>
   "5" AND K$<>"N" THEN GOTO 120
130 IF K$="2" THEN GOTO 300
140 IF K$="N" THEN GOTO 10
150 CLS : PRINT AT 0,15-((LEN(A$))/2);A$
160 FOR B=1 TO 2: LET A=INT (RND*4)+1
170 FOR N=1 TO A: READ B$,C$,D$: NEXT N
180 PRINT '': PRINT B$'C$'D$: NEXT B:
   PAUSE 0: GOSUB 700: GOTO 500
300 CLS : PRINT AT 8,8;"WHICH
   PREDICTION :-"'"TAB 4;"<1>
   MONEY/CAREER"'TAB 4;"<2>
   COMMUNICATION/TRAVEL"'TAB 4;
   "<3> THE HOME□"'TAB 4;"<4>
   AFFAIRS□OF□THE□HEART"
310 LET K$=INKEY$: IF K$="" THEN
   GOTO 310
320 IF K$<"1" OR K$>"4" THEN GOTO 310
330 RESTORE (4000+320*(ST-1)): FOR
   T=1 TO (VAL K$)-1: FOR N=0 TO 7:
   READ B$: NEXT N: NEXT T
335 CLS : PRINT AT 1,15-((LEN A$)/2);A$'''
340 FOR T=0 TO 7: READ B$: PRINT B$':
   NEXT T
500 PRINT AT 21,10;"ANOTHER GO ?"
510 IF INKEY$="Y" THEN GOTO 90
520 IF INKEY$<>"N" THEN GOTO 510
530 STOP
700 FOR T=1 TO 12: RESTORE
   (1000+250*(T-1))
710 READ A$,A,B,C,D
720 IF B=Y AND X>=A THEN LET ST=T:
   LET T=12: NEXT T: RETURN
730 IF D=Y AND X<=C THEN LET ST=T:
   LET T=12: NEXT T: RETURN
740 NEXT T: RETURN
```


```
10 POKE 53280,0:POKE 53281,0
20 DIM H$(11,7,3),D(3,11),S$(11),P$(11,3,7)
30 FOR K=0 TO 11:READ S$(K),D(0,K),
   D(1,K),D(2,K),D(3,K)
35 FOR J=0 TO 7:FOR L=0 TO 2:READ
   H$(K,J,L):NEXT L,J,K
40 FOR K=0 TO 11:FOR J=0 TO 3:FOR
   L=0 TO 7:READ P$(K,J,L):NEXT L,J,K
50 PRINT "■INPUT YOUR DATE OF BIRTH
   (EG 24,5,85)":INPUT D,M,Y
55 D=INT(D):M=INT(M)
60 IF D<1ORD>31ORM<1ORM>12OR
   (D>30AND(M=4ORM=6ORM=9OR
   M=10))OR(D>29ANDM=2)THEN50
70 SS=0
80 IF M<D(1,SS)+12*(SS=2ANDM=1)OR
   M>D(3,SS)-12*(SS=2ANDM=12)
   THEN 86
82 IF (M=D(1,SS)ANDD<D(0,SS))OR
   (M=D(3,SS)ANDD>D(2,SS)) THEN 86
84 GOTO 90
86 SS=SS+1:GOTO 80
90 PRINT "■■YOUR STAR SIGN
   IS■";S$(SS)
100 PRINT "■■>■DO YOU WANT:-"
110 PRINT "■■■■■1-□A
   CHARACTER PROFILE"
120 PRINT "■■■2-□A PREDICTION
   FOR 1985":PRINT "■■■■■N-
   TO ENTER A NEW DATE■"
130 GET K$:IF K$<"1" OR (K$>"2" AND
   K$<>"N") THEN 130
140 IF K$="2" THEN 190
145 IF K$="N" THEN 50
150 PRINT "■"SPC(16-LEN(S$(SS))/2),
   S$(SS)"■"
160 C=8:FOR K=1 TO 2:B=INT(RND
   (1)*8):IF B<>C THEN C=B:GOTO 170
165 K=1:NEXT K
170 FOR L=0 TO 3:PRINT H$(SS,C,L):
   NEXT L,K
180 GOTO 240
190 PRINT "■■■>■WOULD YOU
   LIKE:-"
200 PRINT "■■■■1-□A MONEY/
   CAREER PREDICTION"
202 PRINT "■■2-□A
   COMMUNICATION/TRAVEL"
204 PRINT "■■3-□A PREDICTION FOR
   THE HOME"
206 PRINT "■■4-□AFFAIRS OF THE
   HEART■"
210 GET K$:IF K$<"1" OR K$>"4" THEN
   210
220 PRINT "■"SPC(16-LEN
   (S$(SS))/2),S$(SS)"■"
```


```
230 FOR K=Ø TO 7:PRINT P$(SS,VAL
    (K$)-1,K):NEXT K
240 PRINT "■"SPC(12)"■π ANOTHER
    GO (Y/N)?"
250 GET K$:IF K$<>"Y" AND K$<>"N"
    THEN 250
260 IF K$="Y" THEN 90
270 PRINT "♥":END
```

```
10 MODE7
20 VDU26:INPUTTAB(Ø,23)"ENTER YOUR
   DATE OF BIRTH□(e.g. 12,3,65)",X,Y,Z
30 IF X<1 OR X>31 OR Y<1 OR Y>12
   THEN 20
40 ER=Ø:T=-1:REPEAT T=T+1:IF T=12
   THEN ER=1:GOTO 70
50 RESTORE (1000+250*T)
60 READ A$,A,B,C,D
70 UNTIL (B=Y□AND X>=A) OR
   (D=Y□AND X<=C) OR ER=1
80 IF ER=1 THEN 20
90 VDU 12,28,4,23,37,3:PRINTTAB
   (6,Ø)"YOUR STAR SIGN IS :
   "TAB(15-LEN(A$)/2,2)A$
100 PRINT'''"DO YOU WANT:-"''"1-□A
    CHARACTER PROFILE"''"2-□A
    PREDICTION FOR '85"''"N-□ENTER A
    NEW DATE"
110 D=GET:IF D=50 THEN 210
120 IF D=78 THEN 20
130 IF D<>49 THEN 110
140 CLS:PRINTTAB(15-LEN(A$)/2,Ø)A$'''
150 A=RND(8):RESTORE
    (1000+250*T+10)
160 B=RND(8):IF B=A□THEN 160
170 FOR P=1 TO 8:READ B$,C$,D$
180 IF P=A OR P=B THEN PRINTB$'C$'D$'
190 NEXT
200 GOTO 280
210 CLS:PRINTTAB(Ø,5)"WHICH WOULD
    YOU LIKE-"
220 PRINT'''"1- MONEY/CAREER"''"2-
    COMMUNICATIONS/TRAVEL"''"3-
    HOME"''"4- AFFAIRS OF THE HEART"
230 D=GET-49:IF D<Ø OR D>3 THEN
    230
240 CLS:PRINTTAB(15-LEN(A$)/2,Ø)A$'''
250 RESTORE (4000+T*320+D*80)
260 FOR P=Ø TO 7:READ B$:
    PRINTB$:IF P=3 THEN PRINT
270 NEXT
280 PRINTTAB(Ø,20)"ANOTHER GO
    (Y/N)?";:D=GET AND &5F
290 PRINTCHR$(D)
300 IF D=89 THEN 90 ELSE MODE7:END
```

```
10 PCLEAR1
20 DIMH$(11,7,3),D(3,11),S$(11),P$(11,3,7)
30 FORK=ØTO11:READS$(K),D(Ø,K),D(1,K),
   D(2,K),D(3,K):FORJ=ØTO7:FORL=ØTO2:
   READH$(K,J,L):NEXTL,J,K
40 FORK=ØTO11:FORJ=ØTO3:FORL=ØTO
   7:READP$(K,J,L):NEXTL,J,K
50 CLS:INPUT" INPUT YOUR DATE OF BIRTH
   □□□□□□□□□(EG 24,5,85)□";D,
   M,Y:D=INT(D):M=INT(M)
60 IF D<1 OR D>31 OR M<1 OR M>12
   OR (D>30 AND (M=4 OR M=6 OR
   M=9 OR M=10)) OR (D>29 AND
   M=2) THEN 50
70 SS=Ø
80 IF M<D(1,SS)+12*(SS=2ANDM=1)
   OR M>D(3,SS)-12*(SS=2AND
   M=12) OR(M=D(1,SS)ANDD<D(Ø,SS))
   OR(M=D(3,SS)ANDD>D(2,SS)) THEN
   SS=SS+1:GOTO8Ø
90 CLS:PRINT" YOUR STAR SIGN
   IS□";S$(SS)
100 PRINT:PRINTTAB(10)"DO YOU
    WANT:-"
110 PRINT:PRINTTAB(3)"1-□A
    CHARACTER PROFILE"
120 PRINTTAB(3)"2-□A PREDICTION FOR
    1985":PRINT:PRINTTAB(3)"N-□TO
    ENTER A NEW DATE"
130 K$=INKEY$:IF K$<"1" OR (K$>"2"
    AND K$<>"N")THEN 130
140 IF K$="2" THEN 190 ELSE IF K$="N"
    THEN 50
150 CLS:PRINT@16-LEN(S$(SS))/2,S$(SS):
    PRINT
160 C=8:FORK=1TO2:B=RND(8)-1:IF
    B<>C THENC=B ELSEK=1:NEXT
170 FORL=ØTO3:PRINT@L*32+K*128,H$
    (SS,C,L):NEXTL,K
180 GOTO240
190 CLS:PRINT@10,"WOULD YOU LIKE -"
200 PRINT:PRINT"□□1-A MONEY/
    CAREER PREDICTION":PRINT"□□2-A
    COMMUNICATIONS/TRAVEL":PRINT
```

```
"□□3—A PREDICTION FOR THE
   HOME":PRINT"□□4—AFFAIRS OF THE
   HEART"
210 K$=INKEY$:IF K$<"1" OR K$>"4"
   THEN 210
220 CLS:PRINT@16—LEN(S$(SS))/2,
   S$(SS):PRINT
230 FORK=ØTO7:PRINT@64+32*K—32*
   (K>3),P$(SS,VAL(K$)—1,K):NEXT
240 PRINT@448,"□ANOTHER GO (Y/N) ?"
250 K$=INKEY$:IF K$<>"Y" AND
   K$<>"N" THEN 250
260 IF K$="Y" THEN90 ELSECLS:END
```

THE DATA

The DATA lines that follow can be entered as they stand into the BBC machine. If you own a Commodore, Dragon or a Tandy machine, make sure you enclose any DATA containing a colon (:) in inverted commas—see the lines containing commas in the printed listing. Spectrum owners should enclose their DATA in inverted commas except for any numbers.

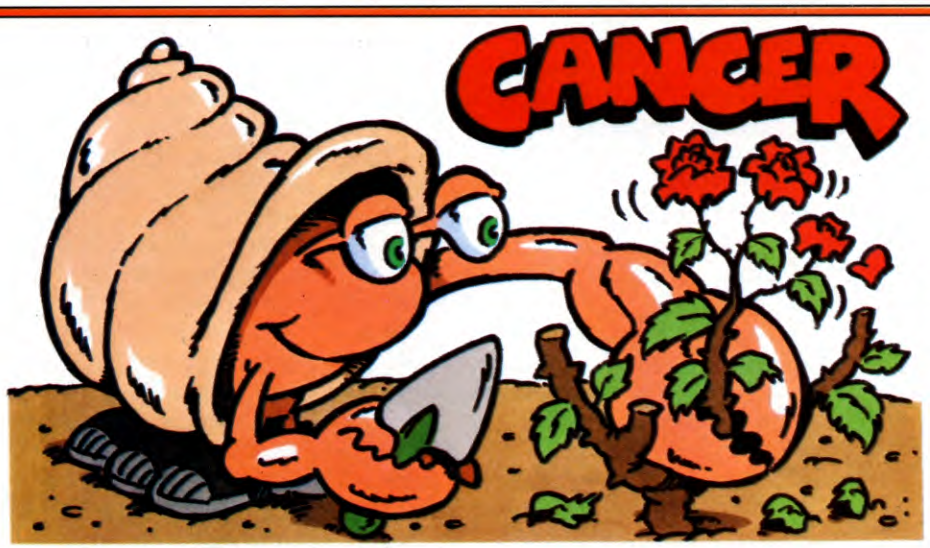
Dragon/Tandy owners should enter the DATA as upper case only.

```
1000 DATA PISCES,20,2,20,3
1010 DATA You are the psychics of the
1020 DATA "zodiac, being the most intuitive"
1030 DATA and sensitive of the signs.
1040 DATA You are compassionate and
1050 DATA empathize readily with other
1060 DATA people's problems.
1070 DATA You absorb atmospheres and need a
1080 DATA "happy environment, mixing with"
1090 DATA those you can relax with.
```


```
1100 DATA You like alcohol and need to be
1110 DATA careful not to over-indulge.
1120 DATA "□"
1130 DATA You are so generous you
1140 DATA frequently don't have enough
1150 DATA money for yourself.
1160 DATA You are highly artistic. Often
1170 DATA "gifted in music, dancing, drawing"
1180 DATA or photography.
1190 DATA "You live and breathe religion,"
1200 DATA but not necessarily in the
1210 DATA conventional sense.
1220 DATA "Compatibility: Virgo, Gemini,"
1230 DATA Sagittarius.
1240 DATA Colours: Sea green/blue.
1250 DATA AQUARIUS,21,1,19,2
1260 DATA You consider your friends to be
1270 DATA as much a part of your family as
1280 DATA your blood relatives.
1290 DATA You are independent from an early
1300 DATA "age, but will often have a house"
1310 DATA full of friends.
1320 DATA You belong to more clubs and
1330 DATA societies than other star signs
1340 DATA and may be a member of CND.
1350 DATA You are both inventive and
1360 DATA "innovative, producing forward"
1370 DATA "looking designs, or ideas."
1380 DATA You find it hard to relate to one
1390 DATA "person, and may be thought"
1400 DATA detached in relationships.
1410 DATA You wear bizarre clothes which
1420 DATA are more interesting than
1430 DATA stylish!
1440 DATA "You choose social work, politics"
1450 DATA and electronic engineering as
1460 DATA careers.
1470 DATA "Compatibility: Capricorn, Libra."
1480 DATA Colour: Electric blue.
1490 DATA "□"
1500 DATA CAPRICORN,22,12,20,1
1510 DATA "You are the most ambitious sign,"
1520 DATA coolly making long term career
1530 DATA plans from an early age.
1540 DATA Capricorn children seem to be
1550 DATA mature serious adults from
1560 DATA birth.
1570 DATA You need to guard against
1580 DATA rheumatism and all ailments
1590 DATA which affect the bones.
1600 DATA Capricorn careers include
1610 DATA "osteopath, mathematician,"
1620 DATA "accountant and financier."
1630 DATA "You are careful with money, but"
1640 DATA what you buy is of quality and
1650 DATA lasts.
1660 DATA You are recognizable by your
1670 DATA serious demeanour and black
1680 DATA looks.
1690 DATA You have more acquaintances than
1700 DATA "friends, your only real allies"
```


```
1710 DATA being your family.
1720 DATA "Compatibility: Taurus, Cancer,"
1730 DATA Aquarius.
1740 DATA Colours: Black and dark shades.
1750 DATA SAGITTARIUS,23,11,21,12
1760 DATA You are a born optimist and are
1770 DATA fun-loving with a good sense of
1780 DATA humour.
1790 DATA You enjoy parties and
1800 DATA "participating in team games, but"
1810 DATA your weak spot is the hip/thigh.
1820 DATA You love travelling abroad and
1830 DATA mixing with the local inhabitants.
1840 DATA "□"
1850 DATA You have a good ear for
1860 DATA "languages, can be fluent in many,"
1870 DATA and love travelling.
1880 DATA "Careers: journalists,"
1890 DATA "broadcaster, lecturers,"
1900 DATA "philosophers, sportsmen."
1910 DATA "You are gregarious, and find it"
1920 DATA difficult to settle in one place
1930 DATA long enough to be monogamous.
1940 DATA "You love animals, and"
1950 DATA "unfortunately, gambling can be a
1960 DATA weakness.
1970 DATA "Compatibility: Pisces, Virgo,"
1980 DATA Gemini.
1990 DATA Colours: Royal Blue/Violet.
2000 DATA SCORPIO,24,10,22,11
2010 DATA "You are the most passionate sign,"
2020 DATA "intense in relationships, and if"
2030 DATA "crossed, vengeful."
2040 DATA You are recognizable by your
```


```
2050 DATA penetrating eyes and your
2060 DATA interesting nose.
2070 DATA Six months before or after the
2080 DATA "birth of a Scorpio, a relative"
2090 DATA dies.
2100 DATA You have an inquisitive mind and
2110 DATA love solving mysteries.
2120 DATA "□"
2130 DATA Scorpio men and women are very
2140 DATA attractive to the opposite sex.
2150 DATA "□"
2160 DATA "Careers: police, surgeons,"
2170 DATA "psychology, social work, and"
2180 DATA positions of power.
2190 DATA You are not the most talkative
2200 DATA "people, but what you say is"
2210 DATA incisive and sometimes mordant.
2220 DATA "Compatibility: Cancer, Libra."
2230 DATA Colour: Maroon.
2240 DATA "□"
2250 DATA LIBRA,24,9,23,10
2260 DATA You are the charmers of the
2270 DATA "zodiac, recognizable by your"
2280 DATA beautiful manners.
2290 DATA You do not like to be left on
2300 DATA your own. You need the
2310 DATA companionship of other people.
2320 DATA "You are excellent hosts, easily"
2330 DATA able to create an atmosphere of
2340 DATA harmony.
2350 DATA "You have dimples on your cheeks,"
2360 DATA and cupid-bow lips. Illnesses
2370 DATA affect your kidneys.
2380 DATA You are sometimes said to be in
2390 DATA love with love and never go for
2400 DATA long periods without a partner.
2410 DATA Libran careers include
2420 DATA "hairdresser, beautician, judge,"
2430 DATA conciliator and lawyer.
2440 DATA Your good sense of style can
2450 DATA achieve harmony in your
2460 DATA clothes by using colours well.
2470 DATA "Compatibility: Aquarius, Aries,"
2480 DATA Scorpio.
2490 DATA Colours: Pastel shades.
2500 DATA VIRGO,24,8,23,9
2510 DATA You are hypochondriacs and never
2520 DATA go anywhere without a box of
2530 DATA pills for any ailment.
2540 DATA "You have a lithe body, but are"
2550 DATA never happy with your figure and
2560 DATA sometimes verge on the anorexic.
2570 DATA You are fastidious and your homes
2580 DATA will always be pristine clean.
2590 DATA "□"
2600 DATA "You are critical by nature,"
2610 DATA paying much attention to detail
2620 DATA in all aspects of life.
2630 DATA "Careers: librarians,"
2640 DATA "statisticians, critics, forensic"
2650 DATA "scientists, or doctors."
2660 DATA You seldom have large families
2670 DATA as disorder makes you physically
2680 DATA ill.
2690 DATA "You are practical by nature, and"
2700 DATA when working in an office you
2710 DATA will be very well-organized.
2720 DATA "Compatibility: Pisces, Gemini,"
2730 DATA Sagittarius.
2740 DATA Colour: Dark blue.
2750 DATA LEO,24,7,23,8
2760 DATA Both sexes wear the most
2770 DATA expensive clothes and jewellery
2780 DATA and are incredibly vain.
2790 DATA "Like lions, you may well have a"
2800 DATA mane of hair in a rich golden
2810 DATA colour.
2820 DATA You enjoy drama and may spend
2830 DATA "your free time at the theatre, or"
2840 DATA at the drama club.
2850 DATA You love of rich food is a
2860 DATA danger to your heart. You are
2870 DATA prone to heart attacks.
2880 DATA You are generous with your money
2890 DATA —provided you have a say in
2900 DATA what's being bought.
2910 DATA "Careers: jewellers, actors, drama"
2920 DATA "teachers, leaders of any kind."
2930 DATA "□"
2940 DATA "Your lair is liable to be a large"
2950 DATA detached house. Neighbours are an
2960 DATA infringement on your space.
2970 DATA "Compatibility: Cancer, Aries."
2980 DATA Colours: Gold and Amber.
2990 DATA "□"
3000 DATA CANCER,22,6,23,7
3010 DATA "You are very shy, hiding your"
3020 DATA green/blue eyes beneath rounded
3030 DATA lids.
3040 DATA You always remain close to your
3050 DATA parents (especially your mother)
3060 DATA even when married.
3070 DATA "You are normally timid, but would"
3080 DATA fight to the death to protect
3090 DATA your family.
3100 DATA Your home is your castle and only
```




```
3110 DATA very special friends are allowed
3120 DATA in.
3130 DATA You are happiest living by
3140 DATA "water—a river, stream, or by"
3150 DATA the sea.
3160 DATA "Careers: nurse, gardener, sailor,"
3170 DATA "cook, teacher and photographer."
3180 DATA "□"
3190 DATA "You cling to the past, and love"
3200 DATA collecting antiques. You have a
3210 DATA photographic memory.
3220 DATA "Compatibility: Leo, Scorpio."
3230 DATA Colours: White and Silver.
3240 DATA "□"
3250 DATA GEMINI,22,5,21,6
3260 DATA "You have a slim figure, and long,"
3270 DATA slender limbs.
3280 DATA "□"
3290 DATA You are a chatterbox and could
3300 DATA blissfully spend all day talking
3310 DATA about nothing.
3320 DATA You are the communicators of the
3330 DATA "zodiac, recognizable by the way"
3340 DATA you talk with your hands.
3350 DATA Both sexes suffer from 'nerves'
3360 DARA "and insomnia, and must guard"
3370 DATA against over-exhaustion.
3380 DATA Your homes will often be found
3390 DATA deserted; you will often be
3400 DATA visiting a friend or neighbour.
3410 DATA You are easily stifled in
3420 DATA "relationships,and avoid clinging"
3430 DATA mates.
3440 DATA "Careers: tele-sales, journalists"
3450 DATA "salesmen, teachers, writers."
3460 DATA "□"
3470 DATA "Compatibility: Sagittarius,"
3480 DATA "Pisces, Virgo."
3490 DATA Colour: Yellow.
3500 DATA ARIES,21,3,20,4
3510 DATA "You spend money liberally, but"
3520 DATA put your own needs first.
3530 DATA "□"
3540 DATA "You are courageous in adversity,"
3550 DATA "quick-tempered, but forgive"
3560 DATA easily.
3570 DATA "You are physically fit, and"
3580 DATA prefer sports where you can excel
3590 DATA individually.
3600 DATA "You suffer from headaches, and"
3610 DATA are prone to cuts and bruises.
3620 DATA "□"
3630 DATA "You enjoy other people's
    company,"
3640 DATA especially those who let you have
3650 DATA your way.
3660 DATA "Careers: forces, butchers,"
3670 DATA sportsmen.
3680 DATA "□"
3690 DATA "Your clothes are very bold, often"
3700 DATA "tight-fitting or low-cut, but"
3710 DATA simple in design.
3720 DATA "Compatibility: Taurus, Libra, Leo."
3730 DATA Colour: Bright red.
3740 DATA "□"
3750 DATA TAURUS,21,4,21,5
3760 DATA You are very affectionate and
3770 DATA romantic.
3780 DATA "□"
3790 DATA "You are good with money, you
    need"
3800 DATA financial security more than any
3810 DATA other sign.
3820 DATA "Normally placid, you can have a"
3830 DATA "fearsome temper, particularly in"
3840 DATA defence of your spouse.
3850 DATA Your self-indulgent nature leads
3860 DATA you frequently to the doctor for
3870 DATA a diet.
3880 DATA You can become very possessive in
3890 DATA "love, and expect your partner to"
3900 DATA reject the opposite sex.
3910 DATA "You are very sensual, enjoying"
3920 DATA the feel of silky fabrics against
3930 DATA the skin.
3940 DATA "Careers: archeologists, banker,"
3950 DATA "farm-workers,market"
3960 DATA stall-holder.
3970 DATA "Compatibility: Aries, Capricorn."
3980 DATA Colours: Pale blue/pink.
3990 DATA "□"
```

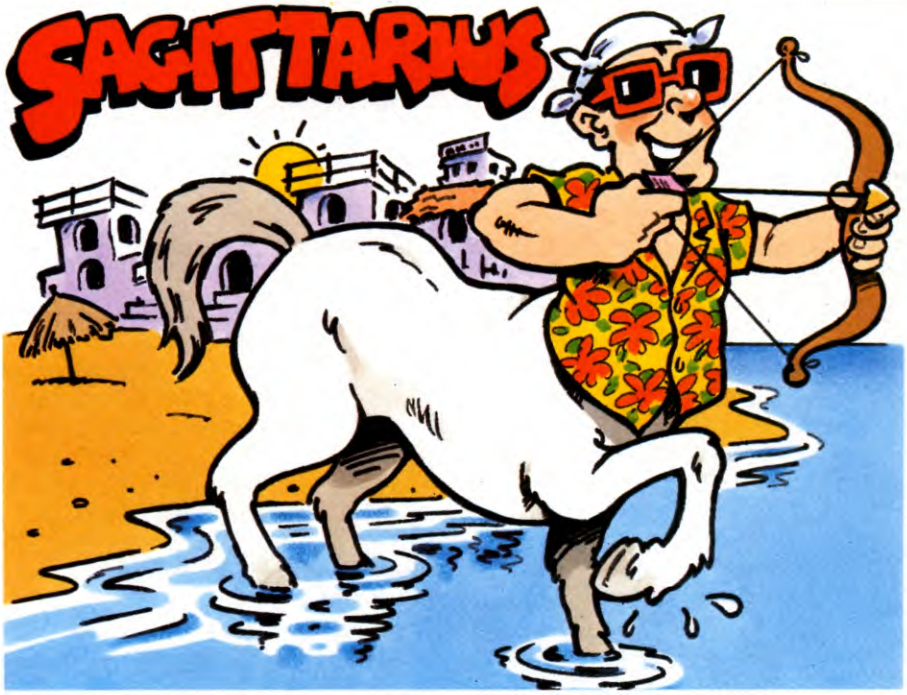
```
4000 DATA From February you will be
4010 DATA working more in seclusion than
4020 DATA "usual, and could develop a"
4030 DATA religious attitude to your work.
4040 DATA "In the Spring, you might find"
4050 DATA you incur more travelling
4060 DATA expenses than usual.
4070 DATA "□"
4080 DATA You would enjoy a late holiday
4090 DATA in October/November digging or
4100 DATA delving into the mysteries of
4110 DATA former empires.
4120 DATA June will witness you adopting
4130 DATA a much more sober and serious
4140 DATA attitude to both day-to-day and
4150 DATA higher matters.
4160 DATA June will see you working hard
4170 DATA "and spending money on your
     home,"
4180 DATA possibly installing some sort
4190 DATA of electrical appliance.
4200 DATA July/August will find you torn
4210 DATA between priorities at home and
4220 DATA at work.
4230 DATA "□"
4240 DATA A relationship which starts in
4250 DATA January could prove lasting.
4260 DATA "□"
4270 DATA "□"
4280 DATA October sees you in an amorous
4290 DATA "mood, enjoying a passing"
4300 DATA relationship which will be at
4310 DATA odds with some of your friends.
4320 DATA For many years to come you
4330 DATA have the advantage of having a
4340 DATA very powerful planet aiding your
4350 DATA career.
4360 DATA Over the past couple of years
4370 DATA you have been wondering if you
4380 DATA have chosen the right path. In
4390 DATA '85 you will know the answer.
4400 DATA The period from February to
4410 DATA April finds you travelling back
4420 DATA and forth short distances to
4430 DATA meet loved ones.
4440 DATA You are likely to holiday late
4450 DATA "in the year, (during September"
4460 DATA "/October), with friends."
4470 DATA "□"
4480 DATA You will be enjoying
4490 DATA entertaining at home during July
4500 DATA and will have to be careful not
4510 DATA to put on weight.
4520 DATA A misunderstanding with
4530 DATA someone in the family could
4540 DATA occur in March.
4550 DATA "□"
4560 DATA An intense relationship could
4570 DATA develop in September from
4580 DATA something which appears to others
4590 DATA to be nothing but a flirtation.
4600 DATA Men may find themselves involved
4610 DATA with a mother figure in
4620 DATA "December, but this is unlikely"
4630 DATA to last into the New Year.
4640 DATA By the end of '85 you could
4650 DATA find that you have reached a
4660 DATA "position of authority, through"
4670 DATA "clear, positive thinking."
4680 DATA You must guard against feeling
4690 DATA too sympathetic for those
4700 DATA "without, and leaving not enough"
4710 DATA for yourself.
4720 DATA Very harmonious communications
4730 DATA in January could make you very
4740 DATA popular at work.
4750 DATA "□"
4760 DATA Foreign sightseeing travel by
4770 DATA "air is likely in September, but"
4780 DATA guard against theft.
4790 DATA "□"
4800 DATA Home will look like a
4810 DATA battleground in March. Try to
4820 DATA avoid confusing statements
4830 DATA which could cause a rift.
4840 DATA You could find yourself
4850 DATA decorating the home in April.
4860 DATA "□"
4870 DATA "□"
4880 DATA There is the possibility of a
4890 DATA "relationship in August, but you"
4900 DATA "could both battle to be dominant,"
4910 DATA neither wishing to give in first.
4920 DATA A lasting relationship could
4930 DATA develop with someone who has a
4940 DATA shared interest or activity
4950 DATA during November.
4960 DATA September's planetary positions
4970 DATA might mean a change of career
4980 DATA which you were beginning to
4990 DATA consider last year.
5000 DATA "You won't be poor in '85,and"
5010 DATA will make small donations
5020 DATA to charities which care for
5030 DATA the elderly.
5040 DATA You will be socializing a
5050 DATA "great deal, all year from"
5060 DATA February.
5070 DATA "□"
5080 DATA You will be joining so many
5090 DATA "societies, particularly higher"
5100 DATA "education, that you may forget"
5110 DATA you have a home.
5120 DATA You will enjoy a romantic
5130 DATA interlude at home in January.
5140 DATA "□"
5150 DATA "□"
5160 DATA February sees the beginning of
5170 DATA a very active year for you
5180 DATA outside the home.
5190 DATA "□"
5200 DATA A romance could develop at
5210 DATA "Christmas, but physical"
5220 DATA attraction might be lacking.
5230 DATA "□"
5240 DATA A deeper relationship which
5250 DATA has its frivolous side could
5260 DATA "develop in June, but this might"
5270 DATA be a fiery/dramatic relationship.
5280 DATA "Despite summer disagreements,"
5290 DATA "you, as last year, are working"
5300 DATA hard consolidating your
5310 DATA position at work.
5320 DATA You must guard against
5330 DATA overspending both on yourself
5340 DATA "and others or house, which may"
5350 DATA be a little extravagant.
5360 DATA You might even take up voluntary
5370 DATA "work in the next few years, with"
5380 DATA a view to counselling others.
5390 DATA "□"
5400 DATA You are likely to want to travel
5410 DATA "in late July, but financially"
5420 DATA it might be better to wait until
5430 DATA August.
5440 DATA You could find that your family
5450 DATA takes a collective interest in
```

```
5460 DATA some form of higher education.
5470 DATA "□"
5480 DATA It's possible that your family
5490 DATA becomes close to a friend who's
5500 DATA an eternal optimist.
5510 DATA "□"
5520 DATA From the second week in November
5530 DATA to December sees the possibility
5540 DATA of an intense relationship
5550 DATA with an older person.
5560 DATA A romantic ethereal affair could
5570 DATA develop at the beginning of the
5580 DATA "year, but this might fade as you"
5590 DATA turn your attentions to work.
5600 DATA The second half of June to
5610 DATA August witnesses you
5620 DATA reassessing your career.
5630 DATA "□"
5640 DATA "Financially, you might find"
5650 DATA yourself consolidating gains
5660 DATA made in '84 — the result of
5670 DATA careful long-term planning.
5680 DATA You might find yourself becoming
5690 DATA more and more interested in
5700 DATA talking about or discussing
5710 DATA political issues.
5720 DATA Foreign travel is probable in
5730 DATA May — possibly sightseeing or
5740 DATA touring.
```


```
5750 DATA "□"
5760 DATA You may feel inclined to move
5770 DATA near to water over the next ten
5780 DATA or so years.
5790 DATA "□"
5800 DATA It's possible that a member of
5810 DATA the family could become
5820 DATA interested in a particular
5830 DATA religion/way of life.
5840 DATA The end of October could see a
5850 DATA long-lasting relationship
5860 DATA developing.
5870 DATA "□"
5880 DATA Permanent commitments may be
5890 DATA made by men at the beginning of
5900 DATA "October, and women at the"
5910 DATA beginning of November.
5920 DATA June sees you enjoying your
5930 DATA "chosen career, travelling to"
5940 DATA meet people in connection with
5950 DATA your job.
5960 DATA October could herald a generous
5970 DATA pay rise as the result of union
5980 DATA negotiations.
5990 DATA "□"
6000 DATA "All this year, as last, you will"
6010 DATA have been talking about your
6020 DATA career.
6030 DATA "□"
6040 DATA October/November see you
6050 DATA becoming a bearable companion
6060 DATA "again, and being light-hearted"
6070 DATA for once.
6080 DATA In '85 you may be sheltering a
6090 DATA "newly divorced friend, or a"
6100 DATA student in your home.
6110 DATA "□"
6120 DATA An unexpected occurrence in
6130 DATA winter '85 — '86 could mean a new
6140 DATA home for you.
6150 DATA "□"
6160 DATA Romance blooms in September
6170 DATA provided your independent
6180 DATA attitude towards setting up a
6190 DATA home doesn't ruin things.
6200 DATA There is a possibility of
6210 DATA marriage or sealing of a
6220 DATA permanent relationship in
6230 DATA October.
6240 DATA The end of May would be a good
6250 DATA time for you to begin a new job
6260 DATA if you feel the need for a
6270 DATA change.
6280 DATA You could find any job change
6290 DATA financially profitable — this
6300 DATA could be the end of an era.
6310 DATA "□"
6320 DATA October/November sees you having
6330 DATA "fun and socializing, but come"
6340 DATA "December, you'll have to take"
6350 DATA care not to be too domineering.
```

```
6360 DATA Spring '85 could find you in
6370 DATA "much warmer climes, although you"
6380 DATA must be careful of a holiday
6390 DATA tiff with a partner.
6400 DATA You may have to care for an
6410 DATA "elderly female relative; outside"
6420 DATA influences should ease the
6430 DATA burden in October and November.
6440 DATA "Enjoy Christmas, but guard"
6450 DATA against broken limbs during
6460 DATA the festivities.
6470 DATA "□"
6480 DATA '85 is a year of friendships
6490 DATA rather than affairs for you.
6500 DATA "□"
6510 DATA "□"
6520 DATA A spring romance could flourish
6530 DATA "briefly while on holiday, but"
6540 DATA both partners will have a
6550 DATA detatched approach.
6560 DATA February/April are significant
6570 DATA "months for your career, with"
6580 DATA the possibility of a pay rise.
6590 DATA "□"
6600 DATA Come September you will be
6610 DATA feeling quite comfortable
6620 DATA financially. You will spend
6630 DATA money socializing.
6640 DATA A dispute at work in September
6650 DATA could lead to you going on
6660 DATA strike or handing your notice
6670 DATA in.
6680 DATA "Foreign travel is very likely,"
6690 DATA especially a first time visit
6700 DATA at the beginning of August.
6710 DATA "□"
6720 DATA June sees you enjoying the
6730 DATA company of male members of the
```

```
6740 DATA "family, and relaxing at home."
6750 DATA "□"
6760 DATA "Come August, you will either be"
6770 DATA gardening or buying antiques
6780 DATA for the home.
6790 DATA "□"
6800 DATA In June you could embark upon
6810 DATA a relationship that could be
6820 DATA "quite physical, the romance not"
6830 DATA starting until October.
6840 DATA "In October, you may conceive"
6850 DATA children.
6860 DATA "□"
6870 DATA "□"
6880 DATA Over the next 13 years you could
6890 DATA find that you will be able to
6900 DATA subtly influence the careers of
6910 DATA many people.
6920 DATA July and August are important
6930 DATA months for money matters —
6940 DATA your purchases in August are
6950 DATA liable to be wise buys.
6960 DATA You must watch what you say in
6970 DATA July/August as you may find
6980 DATA yourself in hot water.
6990 DATA "□"
7000 DATA Travel abroad is possible in the
7010 DATA last two weeks of December and
7020 DATA at the end of the year
7030 DATA generally.
7040 DATA September sees you inviting
7050 DATA strangers or foreigners into
7060 DATA your home.
7070 DATA "□"
7080 DATA You will feel very active in
7090 DATA "September, and will be torn"
7100 DATA between your social life and
7110 DATA doing things in the home.
```




```
7120 DATA You could spend most of
7130 DATA September at home enjoying a
7140 DATA relationship that could lead to
7150 DATA estrangement from others.
7160 DATA Men will feel romantic at the
7170 DATA "beginning of July, and women"
7180 DATA will feel romantic during June
7190 DATA and July.
7200 DATA March to May will see you
7210 DATA sorting out your finances; you
7220 DATA may feel like spending money
7230 DATA impulsively.
7240 DATA In December you are likely to
7250 DATA take a job or choose a career
7260 DATA which will give you more freedom
7270 DATA to communicate.
7280 DATA There should be many
7290 DATA opportunities for foreign travel
7300 DATA throughout '85 especially by
7310 DATA air.
7320 DATA An unexpected journey might come
7330 DATA about in December which might
7340 DATA offer the chance of romance.
7350 DATA "□"
7360 DATA In June you could have vigorous
7370 DATA discussions with the family
7380 DATA about an elderly or invalid
7390 DATA member.
7400 DATA August could see you spending
7410 DATA money on luxury items for the
7420 DATA home.
7430 DATA "□"
7440 DATA A 'behind closed doors'
7450 DATA relationship which had its
7460 DATA beginning in January should take
7470 DATA off in February
7480 DATA A relationship could develop at
7490 DATA "work in October, but the partner"
7500 DATA might become interested in an
7510 DATA older person.
7520 DATA After February you could find
7530 DATA yourself becoming involved in
7540 DATA legal matters which affect your
7550 DATA career.
7560 DATA There is the possibility of a
7570 DATA "small financial gain in June,"
7580 DATA which you may invest in your
7590 DATA own career.
7600 DATA August looks like being a lively
7610 DATA time in the home for you and
7620 DATA you will have some heated
7630 DATA political debates.
7640 DATA If you decide to move in '85
7650 DATA carefully consider all the
7660 DATA "implications, particularly"
7670 DATA during August and April.
7680 DATA The second two weeks of May sees
7690 DATA you making short trips to visit
7700 DATA friends and relations.
7710 DATA "□"
7720 DATA There is a possibility of travel
7730 DATA in July with someone who is
7740 DATA not necessarily your partner.
7750 DATA "□"
7760 DATA A spring romance could end if
7770 DATA you become too possessive. You
7780 DATA could be left for someone more
7790 DATA frivolous.
7800 DATA "For women, a friendship formed"
7810 DATA with a father figure could
7820 DATA develop into something lasting.
7830 DATA "□"
```

WARGAMING: FIRST STEPS

Designing a computer wargame is a fascinating project. Find out what's involved and then mobilize your micro when you start to enter INPUT's own tactical land battle game.

With the advent of computers, wargames have truly come of age. Despite wargames being around for thousands of years, playing them has been, until recently, a cumbersome process involving acres of floorspace and whole phalanxes of small models. A computer consigns all that to the dustbin, brings in a screen-based map, and may even be your opponent.

Wargames have, of course, a serious and deadly purpose, teaching the military how to fight wars more successfully. And though any wargame has, as its main aim, the killing of an enemy, it is certainly true that there is a great deal of interest to be had from creating and playing computer wargames. This interest, though, is more akin to that experienced in playing chess than the zap-pow excitement of arcade-type killing games.

The screen display of a wargame shows a map of the battle or war, usually with the positions of both sides shown. However, a computer game can have a greater degree of realism than traditional types because it is possible, if you wish, to have the opponent's units deployed on the board, but not displayed until their position has been discovered. Computer wargames may be for two players, or like the one you'll see developed in *INPUT*, a game in which one side is controlled by the computer.

The game the generals play tries, of course, to simulate every aspect of a real war as closely as possible. In a home computer game you won't be able to reproduce everything exactly. Nor would you want to—the result of including everything would be a game too horribly realistic for your players to find entertaining enough to play.

DESIGNING A WARGAME

War is about moving 'combat units' (which are usually human beings, but may be tanks or field guns) until they reach enemy units, when they attempt to compel each other to submit, almost always by fighting. The basic ingredients of a wargame, then, are two armies, their movements, and combat between the opposed combat units.

Starting with this as a base, you can consider the broad outline of the design of your wargame. Is the game to take place on land, sea or air? Is it to be a large-scale strategic game with many armies, a smaller tactical game between two armies on a single battlefield, or a skirmish between individual combatants?

Historical period will have a great influence over the type of game—what kind of technology (if any) is available to the combatants, and who are the combatants going to be.

You may wish to try to relive some historical battle, perhaps trying to force Napoleon's armies through to Moscow, or you may wish to invent your own conflict. At this stage, you can let your imagination run riot.

The next stage is to think of the 'rules' of your wargame, making decisions on all the details you want to include in your game, and under what circumstances they will help or hinder each army. For example, you may want your game to include a consideration of armour: wearing armour may protect a soldier during fighting, but it will slow him down as he moves to fight (or retreats after fighting).

It will be tempting to include as many of these components as you can think of, to make your wargame more realistic. But your computer's memory will limit the number of different points of detail you can include, and how complicated you can make your rules.

It's best to stick to the most important components:

- Geographical information on where the troops are and what the terrain is like—how it affects movement, and what cover it provides.
- Troops: how many are there; what weapons and armour do they have; how fast can they move?
- Movement: consider rules about how far units can move, and how far terrain affects movement.
- Orders: a mechanism for giving orders, and perhaps some possibility that the troops will disobey.
- Computer choice: how 'intelligent' will the computer be?
- Combat: what type? Combat is usually divided into missile combat and hand-to-hand combat. Missiles can be anything from throwing axes to intercontinental rockets.

SETTING UP THE GAME

Once the type, period and components of the game are determined you need to consider how they will be represented on your micro. There are two aspects to consider—how the game will appear to the player, and how the game will appear to the computer.

In this series of five articles you will see what is involved as you build a small wargame called Cavendish Field. It is a tactical land battle, fought between two medieval armies. However, the game isn't set in any particular period.

Like most computer wargames, Cavendish Field displays a map showing the disposition of the two armies and the terrain. The players (in this case, you and the computer) act as the commanders of their own units, and must make decisions on strategy, and issue appropriate commands to their men.

Full instructions on how to play the game will be given when the program is completed in part four of this article, but generally, the player is given the option of issuing new orders to each unit, or leaving them as they are. The game proceeds by taking turns to organize the disposition of the troops, which may or may not result in conflict. The outcome of any conflict is determined by the relative type and strengths of the combatants—plus a certain amount of luck. Play continues until one player has reduced the other's forces to an untenable level.

This game will show the important graphics information—the map, with the armies—continuously, but reserve an area for displaying the temporary text instructions and prompts.

It is best to use user defined graphics for the map display. This way, it is simple to handle the two areas of the display by using text-handling instructions for both. In the case of the Dragon and Tandy programs, UDGs have to be set up by DRAWing on the high resolution screen—so this has been used throughout, and a short routine for printing on the high resolution screen has been added.

You need to know what the UDGs will represent. In this game there are four kinds of terrain—plains, villages, woods and hills.

- WARGAMES BEFORE THE COMPUTER
- COMPUTER WARGAMES
- DESIGN CONSIDERATIONS
- HISTORICAL PERIOD
- THE RULES' ROLE
- SCREEN DISPLAY
- SETTING UP THE UDGS
- CLEARING THE TEXT AREA

Microtip

Detailed planning of wargames

Having decided on the broader aims of your wargame, it's very important to get the details right. You should decide on all the factors which are relevant to combat in your game. For example, you might wish to distinguish between missile conflict (whether it's arrows or neutron bombs) and hand-to-hand skirmishing. In most cases, you need at least two different missile types, two different troop types and some representation of range. As a missile weapon will be less effective at longer ranges, it becomes important to know how fast (or how far) the two troop types can move.

This might bring in considerations of armour. Heavy armour slows troops down, but protects them better. Armour will affect not only missile combat, but hand-to-hand combat.

Combat might also be affected by terrain (fighting knee-deep in nettles is harder than fighting in the open), cover (it is harder to hit a target behind a castle wall, than one behind a bush), the number of troops fighting, and so on.

At this planning stage, it's very important to bear in mind that you will almost certainly not be able to include everything you want in your game, even if you have a very large amount of memory available, so you have to balance the features in your game against the amount of RAM you have.

Blank space can be used to represent plains, so no UDGs are needed. Two UDGs each are needed to give a fair representation of hills, villages, units and forest.

Each army has eight units: one leader with his knights, a second body of knights (sergeants), two units of men-at-arms, two units of archers, and two of peasants. The two bodies of knights need a UDG each, but as each of the pairs of units will be armed with the same weapon, the same graphic can be used twice.

This gives a total of nine UDGs, as follows: number one, village; number two, forest; numbers three and four, hills; number five, leader (represented by a flag); number six, knights (a mace); number seven, men-at-arms (a shield); number eight, archers (a bow); and number nine, peasants (a sword).

SETTING UP THE UDGs

This section sets up nine UDGs.

```
210 FOR k=1 TO 9
220 READ a$
230 FOR I=Ø TO 7
240 READ a
260 POKE USR a$+I,a
270 NEXT I
290 NEXT K
2570 DATA "a",16,16,60,126,255,189,231,231
2590 DATA "b",16,56,84,16,56,84,146,16
2610 DATA "c",8,20,34,65,6,8,16,224
2630 DATA "d",0,48,72,132,2,0,0,0
2650 DATA "e",128,240,255,252,143,128,128,
   128
2670 DATA "f",64,240,72,68,68,68,78,68
2690DATA"g",255,231,231,129,129,231,102,
   60
2710 DATA "h",249,70,38,25,9,5,3,1
2730 DATA "i",1,2,4,8,16,160,64,160
```

```
10 CLR
21 PRINT CHR$(142)
22 POKE52,56:POKE56,56:CLR
23 POKE 56334,PEEK(56334)AND254
24 POKE1,PEEK(1)AND251
25 FORI=ØTO583:POKEI+14336,
   PEEK(I+53248):NEXT I
26 POKE1,PEEK(1)OR4
27 POKE56334,PEEK(56334)OR1
28 POKE 53272,(PEEK(53272)
   AND240)+14
29 FORI=ØTO71:READP:POKE(14848+I),P:
   NEXTI
2640 DATA 16,16,60,126,255,189,231,231
2650 DATA 16,56,84,16,56,84,146,16
2660 DATA 8,20,34,65,6,8,16,224
2670 DATA 0,48,72,132,2,0,0,0
2680 DATA 128,240,255,252,143,128,128,128
2690 DATA 64,240,72,68,68,68,78,68
2700 DATA 255,231,231,129,129,231,102,60
2710 DATA 249,70,38,25,9,5,3,1
2720 DATA 1,2,4,8,16,160,64,160
```

```
210 VDU23,224,&1010;&7E3C;&BDFF;&E7E7;
220 VDU23,225,&1C08;&082A;&2A1C;&0849;
230 VDU23,226,&0000;&8870;&0106;
   &0000;
240 VDU23,227,&2418;&01C2;&0806;&00F0;
250 VDU23,228,&F080;&F8FF;&809F;&8080;
260 VDU23,229,&0201;&0804;&E010;
   &A060;
270 VDU23,230,&F040;&4448;&4642;&464F;
280 VDU23,231,&E7FF;&8181;&66E7;
   &183C;
290 VDU23,232,&827D;&2945;&0911;&0205;
300 VDU19,2,4,0,0,0,19,3,2,0,0,0
320 COLOUR 131:COLOUR 0:CLS
```

```
210 FOR K=1 TO 9
220 READ A$
230 UC$(K)=A$
290 NEXT K
2570 DATA BR2D2L2D5RU2R3D2R3U4L2
   UL2D2
2590 DATA BR2R3DRDLDL2ND4L3U2R3DL3
2610 DATA BD3E2R2UD2RFGLG3
2630 DATA BD3E2RF3
2650 DATA ND7FR2FD5UNR3L2
2670 DATA ND7FR4DNR2DNFL3UR2
2690 DATA R5ND6DL5D5R3DL
2710 DATA NR3DRF6U4LHE2DL
2730 DATA BR7G7E2UNF3H2DR
```

The Spectrum program and the Commodore program use the same DATA, but treat it in different ways. The Spectrum sets up the UDGs in characters 124 to 133, but in the Commodore program, a protected block of memory must be reserved in the main program, into which ROM characters can be copied. The new characters are stored from location 14848 onwards, as characters 68 to 77.

The Acorn program simply uses VDU statements to set up the UDGs. In this case, they are set up in characters 224 to 232.

The Dragon/Tandy program uses DRAW instructions elsewhere in the program to set up the characters. In each of the DATA lines there is a string of instructions from which the characters can be outlined.

There is a drawback with using UDGs like this on the Commodore and Acorn machines. On the 64, although the program itself is only 6K in length, when RUN it consumes a great deal of extra RAM, owing to the size of the arrays used. The arrays mean that only a little over 10K of memory remains for any elaborations you may with to add. The problem is even more acute on the Acorns. In order to display a background colour, a colour for terrain and two different colours for the armies, yet have a map large enough for an interesting battle, MODE 1 is needed. This needs 20K of memory by itself, so all REM statements, and unnecessary spaces have been omitted from the listing in order to make sure that the program will run.

SCREEN DISPLAY

These few lines are necessary to clear the area assigned for printing up your instructions:

```
2540 REM Clear text screen
2550 FOR k=17 TO 21: PRINT AT k,0;
   "□□□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□
   □□□": NEXT k
2555 RETURN
```

```
405 PS$="■■■■■■■■■■■■
   ■■■■■■■■■■■■"
2540 REM CLEAR TEXT WINDOW
2545 PRINTPS$;
2550 FOR K=1 TO 5:PRINT "□□
   □□□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□
   □□□□□□□":NEXT K
2555 PRINTPS$;
2560 RETURN
```

```
2540 DEF PROCclean
2550 FOR k=23TO30:PRINT TAB(0,k);
   SPC(39):NEXT
2560 ENDPROC
```

```
2540 REM CLEAR TEXT
   WINDOW
2550 PMODE0,4:PCLS0: PMODE3,1
2560 RETURN
```

The programs treat the screen display as two 'windows'—a text window, and a map window.

The text window has to be cleared and re-written frequently during the game, but the map window has a constant display with occasional small movements of the units it shows.

All four machines see the screen as just one continuous section—not the two that the program needs. So these routines are designed to clear just the area set aside for the text window.

The next part of this article on wargaming deals with the map, and moving the different units around.

COMMODORE COLOUR SPRITES

If you want to play games with your Commodore, they're far better if you use sprites. We have already shown you how to manufacture them, now see how to make them more mobile

Sprites are an essential feature of any game on the Commodore 64. Earlier articles on the subject looked at the creation of high-resolution and multicolour sprites and the various important registers of the VIC-II chip which are used to control them (see pages 168 to 172 and 776 to 783). You will find it useful to refer to these during the course of this article. Now you can find out how to go about actually using and manipulating sprites.

BANKS FOR THE MEMORY

The first thing any program using sprites has to do is to define—or call up from memory—the necessary sprite data. But where is this data kept?

Each sprite, you may recall, requires 64 bytes—of which 63 are taken up by the definition itself (3 bytes per horizontal line, and there are 21 lines). An extra byte is added to the end of the sprite pattern definition to act as a separator and its value is always zero. These 64 bytes can be placed anywhere in memory that's free for such storage.

But a quick look at any memory map will show that large amounts of the Commodore's 64K of total memory is used by things like the video display, character set—and don't forget BASIC program work area and the operating system! So the amount of room left for sprites is restricted.

The memory can conveniently be split into four 'banks' of 16K each as follows (the A value is used to select the bank as explained later):

Bank	Range	A Value
Ø	Ø –16383	3
1	16384–32767	2
2	32768–49151	1
3	49152–65535	Ø

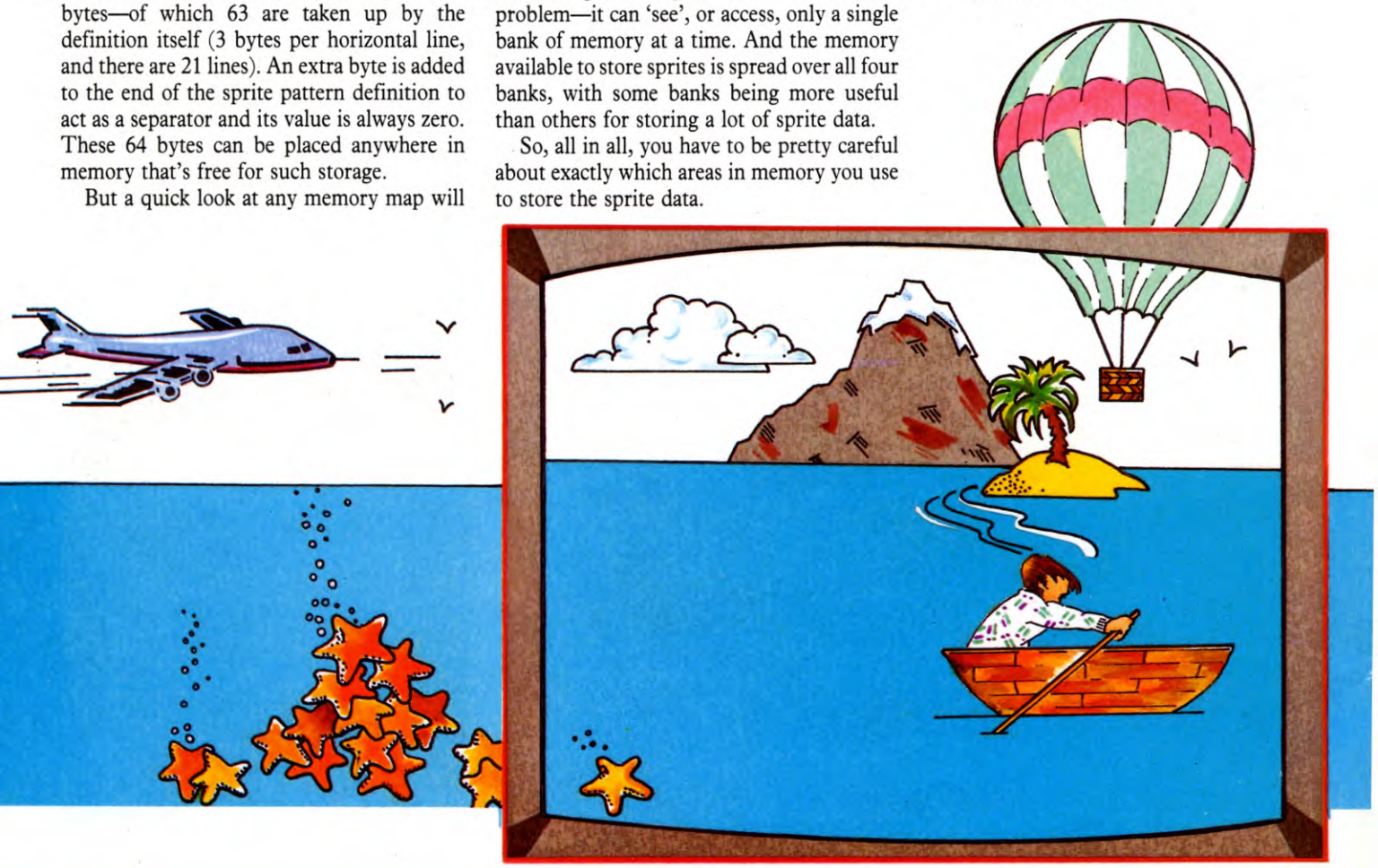
The VIC chip which does all the work of controlling the sprites has one slight problem—it can 'see', or access, only a single bank of memory at a time. And the memory available to store sprites is spread over all four banks, with some banks being more useful than others for storing a lot of sprite data.

So, all in all, you have to be pretty careful about exactly which areas in memory you use to store the sprite data.

SELECTING THE BANK

Unless you tell it otherwise the system defaults to Bank Ø. But with all the system variables, and things like screen memory, space for sprite storage here is fairly tight. But don't neglect possible storage areas away from the main RAM which commences at 2Ø48. There's room for three sprites in the cassette buffer area at 828 upwards (not of much use if a tape unit is used though!), and one somewhere between 679 and 767.

Even allowing for a fairly large BASIC program—say 8K—you've still got a fair amount free above this before the Bank Ø ceiling at 16383 is reached. But a hi-res screen requires another 8K in this bank, although rarely used with sprites.



If you want to store a lot of data you can move BASIC so that it starts very much higher, to give you more room, or you can transfer some to one of the other banks. The other video banks can be accessed by setting what's called the *bank select* bits of register 56578 using the following POKE routine, which may of course be contained within a program:

```
POKE 56578, PEEK(56578)OR3
```

This sets the data direction to output. You can then select the video bank using the 'A' value listed above:

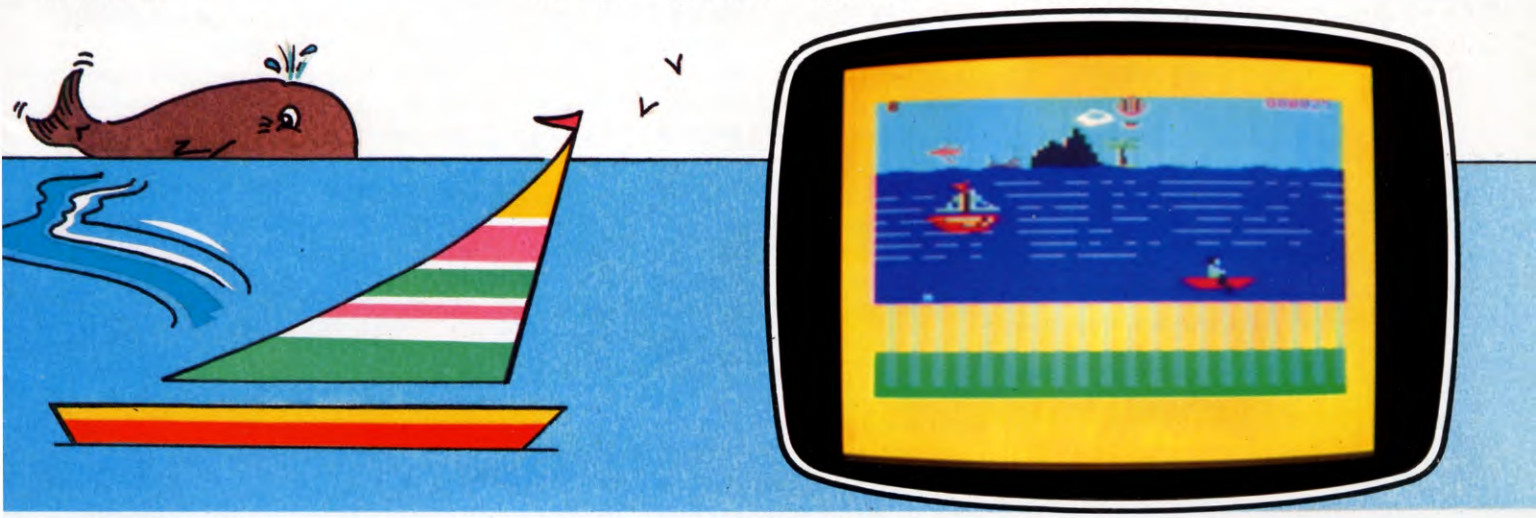
```
POKE 56576, (PEEK(56576)AND252)ORA
```

Take care when storing data for UDGs if you're using these at the same time as sprites. You'll have to find room for both the UDGs and the sprite data.

USING THE DATA

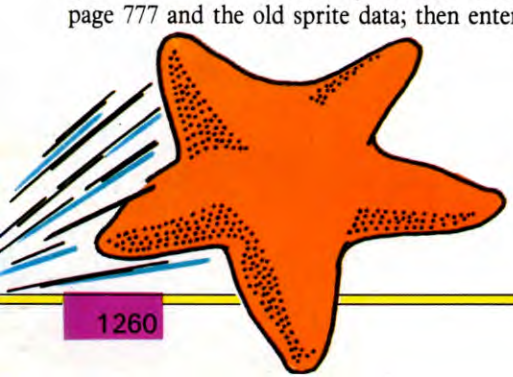
In the previous sprite article, eight sprite definitions were provided for use in the example program below. The program also uses the two other sets of sprite data on page 1261. These are alternative forms of the rowing boat and the whale and are used to create the animation effects. How the program works is explained in each section that follows.

```
1 POKE 51,255:POKE 55,255:POKE 52,47:
  POKE 56,47:CLR
5 POKE 53280,7:POKE 53281,3:V=53248:
  POKE V+21,0
6 FOR Z=49152 TO 49173:READX:POKE Z,X:
  NEXT
7 DATA 162,0,172,4,220,169,160,153,33,5
8 DATA 172,4,220,169,227,153,33,5,232,208,
  237,96
10 A$="□□□□□□□□□□□□□
  □□□□□□□□□□□□□□
  □□□□□□□□□□□□□□
  □□□"
15 PRINT "■■■■■■■■";A$;:FOR
  Z=1 TO 11
20 PRINT "■□□□□□□□□□□□
  □□□□□□□□□□□□□□□
  □□□□□□□□□□□□□□□
  ■";:NEXT Z:PRINT "■";
25 FOR Z=1 TO 140:PRINT "■□■■";
30 IF Z>79 THEN PRINT "■";
35 NEXT Z:FOR Z=0 TO 39:POKE 1984+Z,
  160:POKE 56256+Z,5:NEXT Z:TI$=
  "000000":G=10
38 POKE 2040,192:POKE 2041,197:POKE
  2042,195:POKE 2043,193:POKE 2044,194
39 POKE 2045,196:POKE 2046,199:POKE
  2047,200
40 POKE V+27,82:POKE V+28,213:POKE
  V+37,13:POKE V+38,2
50 POKE V+39,8:POKE V+40,12:POKE
  V+41,4
55 POKE V+42,10:POKE V+43,5:POKE
  V+44,1:POKE V+45,0:POKE V+46,0
70 PRINT"■■";GG
90 GOSUB 1000
92 POKE V+8,180:POKE V+9,76
94 POKE V+10,160:POKE V+11,50
96 POKE V+12,130:POKE V+13,55
98 RX=200:RY=140:TY=81
100 IF TI$>"000100" THEN PRINT
  "■■■>□GAME OVER, (PRESS
  SPACE BAR":GOTO 1300
101 A$=RIGHT$(A$,39)+LEFT$(A$,1):
  PRINT "■■■■■■■■";A$:SYS
  49152
102 PRINT "■■";TAB(33);TI$:PRINT"■
  ■■■■■■■■■■■■■■
  ■■■"SPC(G)"■■□"
103 IFRND(1)>.50 ANDG>1THEN
  G=G-1:GOTO105
104 IFG<38THENG=G+1
105 IF G=INT(RX/8) THEN GG=GG+1:
  G=INT(RND(1)*38)+1:PRINT "■■";
  GG:GOTO 1200
106 PRINT"□"SPC(G)"■■"
110 BX=BX-SP:IF BX<0 THEN POKE
  V+21,PEEK(V+21)AND254:GOSUB 1000
115 POKE V,BX AND 255:POKE V+1,BY
120 Z=INT(BX/256):POKE V+16,(PEEK
  (V+16)AND254)+1*Z
130 TX=TX+2
```



```
135 IF TX>400 THEN POKE V+21,
    PEEK(V+21)AND 253:TX=0
140 POKE V+2,TX AND 255:POKE V+3,TY
145 Z=INT(TX/256):POKE V+16,
    (PEEK(V+16)AND253)+2*Z
150 POKE 2041,197+RND(1)*2
160 IF AY=0 OR AX=0 THEN DD=1:
    AX=150+INT(RND(1)*100)+1
170 AX=AX-1:AY=AY+DD:
    POKE V+4,AX:POKEV+5,AY
180 IF AY>60 AND (PEEK(V+31)
    AND4)=4 THEN DD=-1
190 GET K$:IF K$="■" THEN RX=RX-8
195 IF K$="■" THEN RX=RX+8
200 SYS 49152:IF RX<0 THEN RX=380:
    POKE V+21,(PEEK(V+21)AND127)+128
210 IF RX>380 THEN RX=0:POKE V+21,
    PEEK(V+21)AND127:RX=0
220 Z=INT(RX/256):POKE V+16,(PEEK
    (V+16)AND127)+128*Z
230 POKE V+14,RX AND 255:
    POKE V+15,RY+RND(1)*2
240 IF K$="■" OR K$="■" THEN
    POKE 2047,200+RND(1)*2
250 PX=PX-8
260 IF PX<0 THEN POKE V+21,PEEK
    (V+21)AND247:PX=500:PY=50+
    RND(1)*30
270 Z=INT(PX/256):POKE V+16,
    (PEEK(V+16)AND247)+8*Z
280 POKE V+6,PX AND 255:
    POKE V+7,PY
999 POKE V+21,255:GOTO 100
1000 SP=2:BX=360:BY=80:POKE
    V+23,192:POKE V+29,192
1010 IF RND(1)>.50 THEN SP=4:BY=90:
    POKE V+29,193
1015 IF RND(1)>.75 THEN SP=6:BY=100:
    POKE V+23,193:POKE V+29,193
1020 RETURN
1200 POKE 54296,15:POKE 54278,240:POKE
    54276,33:FOR Z=1 TO 30:POKE 54273,
    Z+30
1210 POKE 54273,30-Z:NEXT Z:POKE
    54276,32:POKE 54273,0:GOTO 110
1300 GET K$:IF K$<>"□" THEN 1300
1310 RUN
```

This program uses the data for the sprites set up on pages 776 to 783 plus the two new sets for the animated sprites. To enter this new sprite data first LOAD in the sprite editor from page 777 and the old sprite data; then enter



the new data into the editor by pressing E and following the prompts. You'll now have ten sets of data which you should SAVE ready for use in this program.

Type NEW to clear the memory then reLOAD the data with:

```
LOAD "sprite file name",1,1
```

Remember to change the first 1 in the command to an 8 if the data is on disk, and use your own file name inside the quotes. Once the data is LOADed type NEW and then enter or LOAD in the program above.

One point to remember is that you must always turn off the sprites before SAVEing any program that uses them. So if you have both the program and sprites in memory and want to SAVE the program first enter:

```
POKE 53248+21,0
```

and only then SAVE the program.

The V+ notation (where V=53248, the start location of the VIC chip) is used extensively in this and the other programs in this article and throughout the explanation here. The table on page 172 which is reproduced here and the descriptions on pages 781 to 783 explain this notation and its uses.

PLAYING THE GAME

The main purposes of the program is to show how the sprites are set up and used, but the picture also forms the basis of a simple game. The object is to manoeuvre the rowing boat to collect the 'starfish' before your time runs out. Use the two cursor keys to move the boat to the left and right. You have to position the boat over the starfish, but you can only pick it up if the starfish is also stationary. You have one minute per starfish and you'll hear a sound each time you catch one.

THE PROGRAM

Here is a brief run through the program so you know what each section does. The principles are explained more fully below. The first part of the program is mostly concerned with setting up the background. Lines 6, 7 and 8 store the machine code for the sea movement, then Lines 10 to 35 set up the rest of the background. Notice the POKE at the end of Line 5. With V equal to 53248 this is the same as the POKE mentioned earlier that turns off all the sprites.

Lines 40 to 55 point to the sprite data and colour the sprites. Lines 92 to 96 position the three stationary sprites—the tree, the cloud and the mountain.

The next section from Line 100 to 106 provides the timing for the game and moves the starfish. After this are the small sections to deal with each of the movable sprites in turn. Lines 110 to 120 are for the yacht, Lines 130 to 150 the whale, Lines 160 to 180 the balloon, Lines 190 to 240 the rowing boat, and Lines 250 to 280 the plane. Lines 1000 to 1200 form a subroutine to move the yacht and alter its size. Finally Lines 1200 and 1210 make a sound when you pick up a starfish and the last two lines let you have another go.

POINTING THE WAY

The sequence of using a sprite starts by calling in the necessary data used for its definition. This is handled by that sprite's pointer—one of values contained in an eight byte block just above the screen location, normally starting at 2040. To call up a sprite you have to POKE its pointer into the correct location shown below. The pointers indicate where the data is stored and you can find out what they are for each sprite on pages 780 to 783. In the program, Lines 38 and 39 set up the pointers for all the sprites.

There are only ever eight sprite numbers and eight pointer locations, but many more sprites can be displayed by altering the pointers in the locations. For example, the whale is animated in Line 150 by randomly changing between pointer 197 and 198, each pointer pointing to different sets of data.

Sprite Number (SN)	Pointer location
0	2040
1	2041
2	2042
3	2043
4	2044
5	2045
6	2046
7	2047

```
0   1   0   0   0   16  0   0   4   0   2   64  0   0   18  0   0   64  0   0
32  96  0   0   112 0   248 56  1   254 60  3   191 62  7   254 127 255 192 255
255 255 243 255 255 1   255 254 0   255 252 0   63  240 0   15  128 0   0   0
0   0   0   0
sprite pointer = 198
start address = 12672
end address = 12735
```

```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   40  0   0   36  0   0   36  0   0   16  0   0   20  0   0   85  0   0
82  0   255 251 255 255 239 255 63  239 252 15  191 240 0   128 0   2   128 0
0   0   0   0
sprite pointer = 200
start address = 12800
end address = 12863
```

This pointer block is always just above the screen location, so if your program relocates the screen, the sprite pointer data also has to move.

Each sprite pointer can hold a value of 0 to 255. The following formula can be incorporated within a program to locate the first byte of the 64 bytes of each sprite definition:

location = (bank*16384) + (pointer value*64)

Thus if the pointer value of sprite 0 is 192 and the default bank 0 is used, as in the example program, the data is stored in location 92*64, equals 12288 onwards.

Rapid adjustment of pointers can be used to create the illusion that many more than the normal eight sprites are being displayed at once.

A simple loop running through a sequence of 64 byte blocks could be used for an animation routine for instance. The whale and the rowing boat are simple examples of this technique using just two sets of data each.

Another little trick which can save a lot of programming effort is to use the same basic design for several sprites. Simply locate the required pattern in a suitable part of memory, copy it into other parts and have more than one of the sprite pointers accessing it! While the main pattern is the same in each case, small parts of the pattern may be edited and copied back. This allows you to make each sprite slightly different. Again this is useful for animation.

SWITCH ON

Once the data for a sprite has been accessed, that sprite has to be turned on (enabled) to make it visible. This is done by location V+21, with each bit looking after one of the eight sprites (SN stands for sprite number).

SN:	7	6	5	4	3	2	1	0
bit:	7	6	5	4	3	2	1	0
'ON' value:	128	64	32	16	8	4	2	1

Any combination of sprites may be activated by POKEing the appropriate value in V+21. Thus POKE V+21,255 would switch on—enable—all eight sprites. POKE V+21,0 would switch them all off. In the program, Line 999 switches on all the sprites.

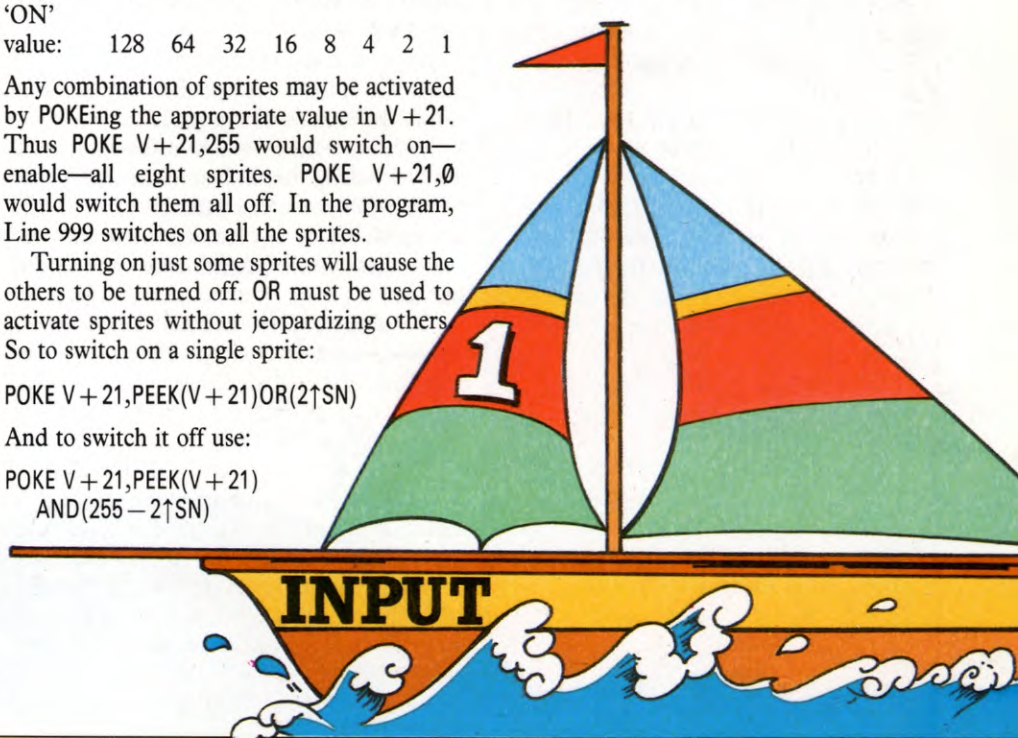
Turning on just some sprites will cause the others to be turned off. OR must be used to activate sprites without jeopardizing others. So to switch on a single sprite:

```
POKE V+21,PEEK(V+21)OR(2↑SN)
```

And to switch it off use:

```
POKE V+21,PEEK(V+21)
   AND(255-2↑SN)
```

Sprites that move need to be turned off once they reach the edge of the screen. You can see this happening in Line 110 which tests when the yacht reaches the left-hand edge. When BX<0 the sprite is turned off and the subroutine at Line 1000 positions it on the other side. A similar thing happens in Line 135 for the whale, Lines 200 and 210 for the rowing boat, and Line 260 for the plane.



VIC-II chip memory locations

This is a handy reference table of the memory locations you will need to access to control the shape, size, colour and position of your sprites.

The V+ value numbers are the most useful and the easiest to remember.

Decimal	V+value	Description
53248	V	Sprite-Ø X position
53249	V+1	Sprite-Ø Y position
53250	V+2	Sprite-1 X position
53251	V+3	Sprite-1 Y position
53252	V+4	Sprite-2 X position
53253	V+5	Sprite-2 Y position
53254	V+6	Sprite-3 X position
53255	V+7	Sprite-3 Y position
53256	V+8	Sprite-4 X position
53257	V+9	Sprite-4 Y position
53258	V+10	Sprite-5 X position
53259	V+11	Sprite-5 Y position
53260	V+12	Sprite-6 X position
53261	V+13	Sprite-6 Y position
53262	V+14	Sprite-7 X position
53263	V+15	Sprite-7 Y position
53264	V+16	MSB of X coordinate
53265	V+17	VIC control register
53266	V+18	Raster register
53267	V+19	(light pen)
53268	V+20	(light pen)
53269	V+21	Sprite display enable
53270	V+22	VIC control register
53271	V+23	Sprite Ø-7 Y-expand
53272	V+24	VIC memory control
53273	V+25	Interrupt register
53274	V+26	Interrupt enable
53275	V+27	Background priority
53276	V+28	Select multicolour
53277	V+29	Sprite Ø-7 X-expand
53278	V+30	Sprite collision
53279	V+31	Background collision
53280	V+32	Screen border colour
53281	V+33	Background colour Ø
53282	V+34	Background colour 1
53283	V+35	Background colour 2
53284	V+36	Background colour 3
53285	V+37	Sprite multicolour 1
53286	V+38	Sprite multicolour 2
53287	V+39	Sprite-Ø colour
53288	V+40	Sprite-1 colour
53289	V+41	Sprite-2 colour
53290	V+42	Sprite-3 colour
53291	V+43	Sprite-4 colour
53292	V+44	Sprite-5 colour
53293	V+45	Sprite-6 colour
53294	V+46	Sprite-7 colour

TYPE, COLOUR AND SHAPE

At some point the general physical characteristics have to be established. Each sprite has its own colour which is set by a register in the following group:

SN:	7	6	5	4	3	2	1	Ø
Register	V+46	+45	+44	+43	+42	+41	+40	+39

The normal range of colour values, Ø to 15, may be POKEd into these locations. Thereafter, each sprite that's turned on will be displayed in the specified colour. Remaining pixels in the sprite area will assume whatever background colour is selected (V+33).

In multicolour mode a sprite may have up to four colours—three plus the background. The main colour of each sprite is set using the registers above. The extra two colours are set in V+37 and V+38 and these colours are the same for all multicolour sprites. Multicolour sprites are turned on by setting register V+28 using the following statement:

```
POKE V+28,PEEK(V+28)OR(2↑SN)
```

And, again, a single multicolour sprite may be switched off using:

```
POKE V+28,PEEK(V+28)AND(255-2↑SN)
```

In the program, the main colours for each sprite are defined in Lines 50 and 55. The two extra colours for the multicolour sprites are set in Line 40; the colours used in this case are 13—light green and 2—red. (The remaining POKE in these lines, POKE V+27,82, determines the 'priority' of the sprites, which is explained later.)

Next, decide whether you want to enlarge the sprite. Any sprite can increase in either the horizontal or vertical dimension—or both—to double the size instantly, a useful way of making the same sprite design 'work' for you. For example a pulsating effect can be achieved simply by looping back and forwards between expanded and unexpanded sprites.

For a single sprite, horizontal expansion is achieved using location V+29 in the following line:

```
POKE V+29,PEEK(V+29)OR(2↑SN)
```

This horizontal expansion can be cancelled using the following:

```
POKE V+29,PEEK(V+29)AND
(255-2↑SN)
```

Similarly the expand/cancel set for the vertical is:

```
POKE V+23,PEEK(V+23)OR(2↑SN)
POKE V+23,PEEK(V+23)AND
(255-2↑SN)
```

In the program you can see this in Lines 1010 and 1015. This is the routine that positions the yacht. To give an impression of perspective and distance, the boat is drawn larger when it is nearer the foreground and smaller when it is nearer the horizon. To add variety its position is determined by the RND function at the start of the lines.

POSITION AND MOVEMENT

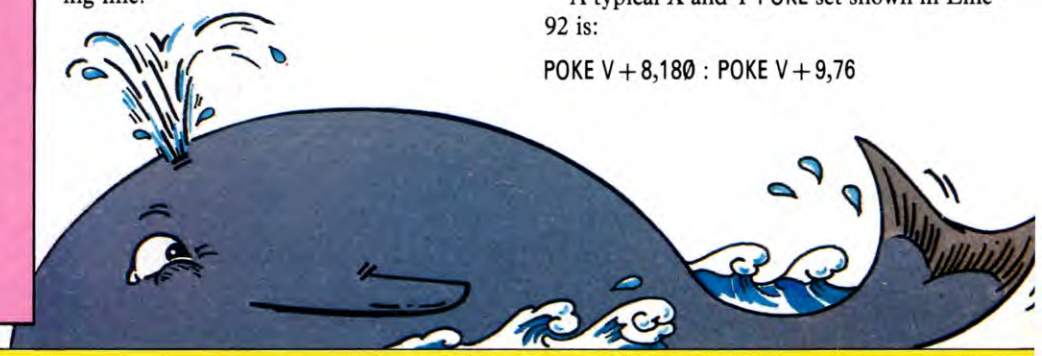
To position a sprite on the screen you have to specify two values: an X and Y coordinate representing, respectively, horizontal and vertical pixel locations. Every sprite has its own unique X/Y POKEs using locations from V to V+16. For example, sprite zero uses locations V and V+1 for the X and Y coordinates; sprite one uses locations V+2 and V+3 and so on.

The X positions run to 360 pixels, and the Y value to 200. This represents the full screen area, but in fact only a window on this represents the visible area. You may have witnessed this on games programs where sprites seem to appear from well inside the apparent viewing areas, starting at about the position where the darker blue inner screen occurs when you first switch on the computer.

The visible window begins at X location 24 and Y location 50. The sprite position is based on its own top left-hand corner and as its size is 24 × 21 pixels, it is conceivable that it could be positioned on the screen but remain completely or partly out of the main viewing area. This is less likely if the sprite is expanded.

A typical X and Y POKE set shown in Line 92 is:

```
POKE V+8,180 : POKE V+9,76
```



This would position sprite 4—the tree—near to the top centre of the screen. Now if you wished to move that sprite, all that you need to do is to POKE a new value in one or both locations. There's no need to erase or otherwise 'tidy up' the old location, and you can see how simple it would be to initiate several loops to control the direction and timing of the movement.

For example, have a look at the routine to move the plane. This is in Lines 25Ø to 28Ø. Line 25Ø decrements the X position by 8 each time and Line 28Ø POKEs the new position in V+6 and V+7.

Have a look, too, at the routine to move the rowing boat. This is in Lines 19Ø to 24Ø. The first two lines check which key is pressed and update the X coordinate accordingly. Line 24Ø also swaps between the two versions of the sprite so the picture is animated and the boatman appears to row.

The only slight problem occurs when the sprite X location is to the right of the screen when values in excess of 255 are encountered. This is where location V+16 is used. This provides a ninth bit facility, seeding a start value of 255 to the X location. Thus:

```
POKE V+16,2↑SN : POKE V+4,45 : POKE
V+5,125
```

This positions sprite 2 at location 3ØØ,125.

The POKEd value of V+16 may of course be for any combination of sprites if more than one is required to be in the right-hand area simultaneously. The 2↑SN value can be ORed to protect other sprite positions. And to return to the left-hand side of the screen you must reset the bit to Ø for each sprite which has used V+16. Thus:

```
POKE V+16, PEEK(V+16)AND
(255−2↑SN)
```

returns sprite 2 to the left-hand side.

You can see this used in several lines in the program—in fact for all the sprites that move left and right. These are the yacht in Line 12Ø, the whale in Line 145, the rowing boat in Line 22Ø, and the plane in Line 27Ø. The other moving sprite, the balloon, moves only up and down and this means that the Y location never exceeds 255.

COLLISIONS AND PRIORITIES

As well as being highly manoeuvrable, sprites have one special characteristic which makes them especially suited to games use—collision detection. This is handled by two special registers, V+3Ø and V+31. As explained in the previous articles, these handle information which can be used to flag collisions between one sprite and any other, or one sprite and the background data (which can be text or graphics).

Each bit of each of the registers looks after one of the eight sprites. And so the familiar 2↑SN value crops up in the PEEKs and POKEs which are used to both detect and act upon collisions (see page 783). Again, several sprite collisions can be set up simultaneously.

In the program, collision detection is used to test when the balloon touches the sea. Look at Line 18Ø. A collision is checked for only if the Y coordinate AY is greater than 60. The test is to look at the value of:

```
PEEK (V+31)AND(2↑SN)
```

The result shows which sprite has collided with the background. The 4 is 2↑2 showing sprite 2 collided.

Remember that a collision is deemed to have taken place only when the register is set for the relevant sprite(s), and when an 'on' part of the sprite *overlays* another. The unused edge of a sprite doesn't count, in other words. Collisions can occur anywhere on the screen, even in the non-visible border area.

Quite which sprite should have visual dominance for collisions purposes is really down to the effect desired and doesn't affect the collision register which is interested in only the sprites involved, not their location or plane. But the effect can be heightened by careful choice of sprite priority, which is achieved by the SN—the lower the SN, the superior it is on display. In the program the priorities were set up right at the start by assigning the pointer for the highest priority sprite to location 2Ø4Ø; the next sprite's pointer to location 2Ø41 and so on.

One cautionary point, however. In multicolour mode, bit pair Ø1 (which sets whatever colour is logged in V+37) is not detected in a sprite to background collision such as handled by V+31—even though it can be present on the screen. But the good thing about this is that you can use the Ø1 bit pattern for colouring objects *not* used in collisions.

Also, note that when V+3Ø or V+31 are read using PEEK, the registers are automatically cleared. So if you wish to retain the value, typically because you want to detect more than one collision, save the value in a variable until it is finished with. Any routine which restarts a sprite collision routine should obviously commence with PEEKs of both locations to clear them of previous collision data.

FURTHER ON

A further area for developing uses of sprites is combining them with other graphics modes. In fact, you can use sprites with all of them—particularly effective is a combination with bit-mapped displays, although this tends to gobble up extravagant amounts of memory.

Even though the very nature of sprites means they are highly controllable from BASIC, most of their real power becomes apparent only when machine code routines are employed. In this way you can achieve very rapid, flicker-free sprite movement and animation. In particular, it's rather limiting to stick to just eight sprites on screen at once when so many more can be defined and consigned to memory.

By using *raster interrupts* many more can appear to be on the screen at once. A full description of interrupts is beyond the scope of this article. Briefly, though, the interrupt feature *appears* to allow several programs to run at once. Every 60th of a second the computer stops what it is doing, scans the keyboard (buffer), does a bit of electronic tidying up and, in the normal course of things, returns to the program in hand. But it can just as well be directed to do something else, for whatever length of time, before making that return.

With a raster interrupt, in effect, a certain amount of the screen is 'drawn' using one routine, before being transferred to another routine for the remainder. Because everything takes place so quickly, parts of two or more separate screens appear as one.

Two VIC registers figure in this: V+17 (high bit) and V+18. By looking at these locations it is possible to tell exactly where the screen is being scanned at any one instant. And when it reaches a particular, specified point, the program can do something else—this is the form of what in machine code programming is called an *interrupt request*.

NEW LANGUAGES: LOOKING AT LOGO

For many children computing is no go without LOGO, a language designed with learning in mind. Seymour Papert's brainchild has revolutionized the classroom

In the 1960s computers were very expensive. The computer power in your micro would have cost tens of thousands of pounds, since even the largest mainframe computers could only store about 144K bytes. For economic reasons computer languages were designed to use as little memory as possible, and were made to be easy for the computer rather than easy for the programmer.

With the appearance of the microcomputer in the 1970s the programming languages of the 60s gained popularity because the new micros, like the mainframe of the 60s, had small memories. The idea originated that 'simple for the computer' meant 'simple for the programmer', and the difficulties of learning languages like BASIC became accepted as part of learning programming.

THE CHOICE OF LANGUAGE

When you switch on the majority of home micros, including all those covered in *INPUT*, they operate in BASIC (Beginners' All-purpose Symbolic Instruction Code), and this is the language that most micro owners stay with. Yet there is no reason for this to be the case. BASIC is only a machine code program which is automatically present in the machine—and in fact, there are still some machines for home use on which BASIC has to be loaded from tape or disk before you start programming.

This means that it is perfectly possible for you to change the language which your computer understands—all you have to do is to load in a machine code program which enables it to recognize the instructions and to perform the appropriate actions. Indeed, if you have tried the program on pages 848 to 855, you will have discovered that it is relatively easy to extend your machine's BASIC with additional commands.

But it is also possible not merely to adapt the existing program used by the BASIC interpreter, but also to replace it completely.

Since the early days of computing, something like a hundred different languages have been developed for various purposes, not to mention 'home-made' languages designed for particular computers. Some of these are so specialized in use that you would never encounter them except at the highest levels of research, but others are at least as practical for the home user as BASIC is—and in some cases more so.

Whether or not a particular language is available to you depends upon whether or not you can obtain the program which enables your micro to operate with it. Alternative languages are usually supplied like any other program—on tape or disk, or sometimes in the form of a ROM cartridge. Their availability depends on which computer you have—many business machines, for example, which have the operating system CP/M can choose between more than a dozen languages, and sometimes several versions of each. And there's a fairly good choice for most home computers, too. The BBC, for example, can work in six languages apart from BASIC, while the unexpanded Dragon has three.

The next few parts of *INPUT* will be examining the most popular alternatives to BASIC to see what advantages and disadvantages they have, and how you can use a new language to extend your programming skills.

LEVELS OF COMMUNICATION

A language is a means of communication between you and the computer. It is something that you both understand—a compromise between natural language (say, English) and binary machine code, which is what the machine actually works on. A language is said to be low-level if it is close to the computer's own. Assembly language is such an example. The highest level languages, like PROLOG and LOGO, can be quite close to natural language. The next generation of computers (called the fifth generation) will probably use something like PROLOG to accept instructions in plain English. BASIC falls somewhere between the two, and in the view of many programmers is not a good compromise, being neither all that easy to understand, nor very quick for the computer to use.

The languages which *INPUT* is covering span the range from high to low level. The four, LOGO, PASCAL, LISP and FORTH, are available for all the computers except the Dragon, which does not have LISP. Tapes, disks or ROMs for all the others should be obtainable from normal software outlets, although they may not be held in stock by smaller stores. But don't worry if you do not want to buy the languages now. You will be able to understand what is involved even if you cannot try the examples. This time, we start with a look at LOGO.

THE BEGINNINGS OF LOGO

In 1967 a research team at the Massachusetts Institute of Technology took a different approach to computing. They set out to create a language that was easy for the programmer rather than easy for the computer. The result was LOGO.

The team was headed by Seymour Papert, an expatriate South African. Papert had worked closely with Jean Piaget, the famous child psychologist, who had said that young children can only understand an abstract concept if it is presented in a concrete form. He believed that a child should learn through its own discoveries, rather than by being told things. This approach was a strong influence on the development of LOGO.

Another influence was the work of Marvin Minsky, a researcher into Artificial Intelligence at MIT in the 1960s. Artificial Intelligence is the science of simulating aspects of human intelligence in machines. Computers are not intelligent, but only obey instructions which have to be given with detailed precision. The factors we consider when solving a problem are immense and varied, and thus to simulate such a process in a computer program is a gigantic task. The world of artificial intelligence needs programming languages with which to simulate human learning and decision-making abilities. LISP is a powerful programming language developed for such a purpose, which will be covered later in the *INPUT* course. LOGO is essentially a dialect of LISP, and although it can handle words and numbers is aimed chiefly at graphics programming.

The name LISP is derived from 'List Processing'. Its basic data structure is a list rather than a numeric array or character string, and because a list can be symbols or other lists, it is easy to process non-numerical data. LISP, however, is not easy to learn.

- COMPUTER LANGUAGES: DESIGN CONSIDERATIONS
- CHOICE AND AVAILABILITY OF LANGUAGES
- COMMUNICATION
- LOGO AND PSYCHOLOGY
- LOGO AND LISP
- ROBOTICS AND THE TURTLE
- PSEUDO-LOGOS
- PROGRAMMING IN LOGO

CONCRETE DEMONSTRATIONS

Seymour Papert and his colleagues looked for a 'gateway' through which children could enter the world of programming. Three areas were discovered, graphics, music and robotics—children were interested in drawing pictures on the computer's monitor, in using the computer to create electronic sounds and tunes, and in driving machines around from the keyboard. Of the three, robotics was most exciting and appealing, which led to Papert creating the robotic Turtle.

The Turtle's development was encouraged because at that time there were no cheap monitors. The Turtle, a floor crawling, pen carrying robot, is controlled by the computer. Its pen can be raised or lowered, allowing the Turtle to draw as it moves. It allows children to relate geometry to their own movements in drawing or walking. Children drawing with a Turtle are often seen turning and walking out routines before telling the Turtle to do them.

The Turtle was named in honour of Grey Walter, a British neurologist and cybernetician who made 'cybernetic tortoises' in the 1950s. These were electrically powered vehicles equipped with devices which measured the power level of their batteries—when supplies ran low they searched for a recharging apparatus and plugged themselves in. Grey Walter's 'tortoise' was one of the first true robots.

The original robotic Turtle at MIT almost became extinct for a while with the advent of personal computers and the facility to display things on screen simply and cheaply. It was succeeded by its two-dimensional descendant, the screen Turtle. This is a cursor, sometimes represented as a chevron, sometimes as a small Turtle. It obeys the same instructions as the robotic Turtle, doesn't break down, and is much cheaper.

The robotic Turtle has made its comeback in schools, however, where it enables a larger number of children to participate in programming activities, providing a model for 'body geometry', and an exciting concrete introduction into an abstract world.

The article on pages 884 to 888 describes a few of the turtles and robots that are available for home computers. These robots are becoming popular in primary schools and are fulfilling their original purpose of introducing children to computer programming in an enjoyable and comprehensible way.

SPEAKING THE LANGUAGE

So what have mechanical toys to do with a computer language? Papert sees the computer as a vehicle for creativity and expression of ideas. He believes the best way for children to learn about computers is to grow up in a computer culture, just as the best way to learn Italian is to spend some time in Italy. At the 1983 annual conference of America's main professional computing association, he called for a scheme to give every American child a computer. He believes that the child should program the computer, rather than the computer program the child. The means he proposed to do this is through LOGO, which in his view gives children control over one of our most powerful resources, provides a framework for problem solving outside computing, and enables complex mathematical ideas to be presented in a fresh, understandable way.

Papert explained his philosophy in his best-selling book, 'Mindstorms, Children, Computers and Powerful Ideas'. Since the book's publication in 1980, versions of LOGO have appeared for most popular microcomputers. It is available for the Spectrum, Commodore 64, Dragon 32, and in four versions for the BBC B—most versions closely resemble the original MIT LOGO. There are also several programs on the market with LOGO type names, such as 'Logo Dart' and 'Logo Graphics'.

These are simulations of Turtle graphics, which only forms a small part of LOGO. The language has full word and list processing facilities, sound, mathematical functions and many other features not found in 'Pseudo-Logo' programs.

LOGO is the first 'user friendly' language. Because it is easy to get into and popular in schools, the idea often arises that it is 'for kids'. Nothing could be further from the truth. Professor Harold Abelson, one of Logo's designers at MIT said, 'In working with LOGO we've discovered some important things.

'A computer language can be simple and powerful at the same time. In fact these two aspects are complementary rather than conflicting because it is the very lack of expressive power in primitive languages such as BASIC that makes it difficult for beginners to write simple programs that do interesting things. More important, we've found that it is possible to give people control over powerful computational resources, which they can use as tools in learning, playing and exploring.'

LOGO is also a growing language. Edinburgh University is developing a version called Control-LOGO, to enable more sophisticated control of robots. Papert wants future versions to include 'worlds' in which children can play with ideas of physics like they play with geometry in Turtle Graphics.

PROGRAMMING IN LOGO

When you load LOGO, either from disk, cassette or a ROM chip or cartridge. LOGO will give you a message like:

```
WELCOME TO LOGO
?
```

The ? is a prompt. LOGO is waiting for you to give it a command. If you type:

```
HELLO LOGO I'VE BEEN LOOKING FORWARD TO MEETING YOU
```

LOGO will reply:

```
I DON'T KNOW HOW TO HELLO
?
```

It has looked at the first word, not recognised it as a command, and informed you.

A command it does recognise is ST. ST stands for Show Turtle. It will call the screen Turtle from the depths of the computer and onto the screen. The Turtle's shape will point in a particular direction. This is its heading. It

is armed with a pen and ready to draw.

The command to move the Turtle forwards is FORWARD which can be abbreviated to FD. FORWARD is a LOGO command which requires an input. The Turtle needs to be told how far to move. Typing FORWARD 100 will send the Turtle 100 units forward. It will draw a line behind it. If you are using a floor Turtle, the Valiant, for example, will move 100 centimetres in the direction it is facing.

If you type FORWARD100 LOGO will reply:

I DON'T KNOW HOW TO FORWARD100

This is because, without the space between the command and the quantity LOGO sees FORWARD100 as a different word from FORWARD and because FORWARD100 is not in its vocabulary it gives a non-recognition prompt. It will give a similar reply if you type FERWARD instead of FORWARD.

There is a simple line editor which enables you to correct mistakes before you press RETURN. The cursor keys allow you to move along the line to the mistake. The delete key removes the character to the left of the cursor. To insert new characters simply type them in. The text to the right will automatically move along to accommodate the addition. Some versions of LOGO allow you to retrieve and edit a line after pressing RETURN.

BACK works in the same way as FORWARD. It is abbreviated to BK. It moves the Turtle backwards, and, as before, it requires a numeric input. You can give any number as an input to FORWARD or BACK. They change the Turtle's position but not its heading.

To turn the Turtle use LEFT and RIGHT. They can be shortened to LT and RT. Like FORWARD and BACK, they require a numeric input. RIGHT 39 will turn the Turtle 39 degrees to the right. LEFT 123 will turn the Turtle 123 degrees to the left.

SOME SIMPLE GAMES

If you are using LOGO with children it is better not to tell them that RIGHT 96 turns the Turtle 96 degrees to the right. Let them experiment with different inputs and discover the effect of alternative values for themselves. There are various games you can play to lead them into discovering values of different angles.

Mark a position on the screen and see how many commands a child takes to stop the Turtle under the mark. Or you can draw a roadway on the screen and get the child to navigate the Turtle along the route, losing a point for each command and a point for each time the Turtle leaves the road. Later in this article you will see how to write a procedure to draw a road.

The program can be stored on disk or tape, and loaded whenever the children want to play the game. If you want to play immediately, draw the route directly on the TV screen with a suitable marker such as a wax crayon or a marker pen of the type used on whiteboards in school classrooms. It will wipe off easily afterwards. This way, it is easier to draw more complicated routes.

There is a variety of games to play with a floor Turtle, based on mazes, knocking things down, collecting objects from various points etc. You can play Shove Turtle, a variation of the pub game Shove Ha'penny using a Turtle instead of a coin.

Shove Turtle can be played on the screen by writing a procedure to draw the board and place the Turtle at the start.

MORE COMMANDS

Playing with the move and turn commands you will need some more LOGO commands or *primitives*. If the screen is not in the 'wrap around' mode, that is, if the Turtle goes off the top of the screen but does not appear back on the bottom, you will need to get it back on the screen.

HOME will bring the Turtle back to its original position and heading in the centre of the screen.

You will also need to clear the screen in order to draw new pictures.

CLEARSCREEN, shortened to CS, will erase any drawings and return the Turtle to the HOME position.

You may want the Turtle to move without drawing. PENUP, shortened to PU, raises the Turtle's pen. If you are using a floor Turtle the pen in its belly is lifted. The screen Turtle merely stops drawing.

PENDOWN, shortened to PD, lowers the floor Turtle's pen and allows the screen Turtle draw.

CLEARSCREEN, HOME, SHOWTURTLE, PENUP and PENDOWN are LOGO primitives which do not require inputs.

Here is an example to illustrate the commands so far. You need to press the RETURN key after entering each line:

```
SHOWTURTLE
LEFT 45
FORWARD 71
RIGHT 135
PENUP
FORWARD 50
PENDOWN
LEFT 45
BACK 71
PENUP
HOME
PENDOWN
```

CLEARSCREEN will erase the picture if you want to try something else.

BUILDING UP A PROCEDURE

The activities so far have all been in the 'immediate' mode. You have been talking directly to the Turtle and it has instantly executed your commands, like a squad of soldiers obeying a Sergeant Major on the parade ground. There is another mode in LOGO, the procedural mode.

In the procedural mode you name a series of commands and write the commands after the name. The name you give to the procedure then becomes part of LOGO's vocabulary. The turtle will respond to that name by executing the commands in its defintion. To define a procedure use TO followed by the chosen name. This can be any name except the name on an existing LOGO primitive. It makes life easier to give a procedure a name that defines its function.

This is best understood using an example. Here is a procedure which teaches the Turtle to draw a zigzag:

```
TO ZIGZAG
FORWARD 20
LEFT 150
FORWARD 20
RIGHT 150
FORWARD 20
LEFT 150
FORWARD 20
RIGHT 150
FORWARD 20
LEFT 150
FORWARD 20
RIGHT 150
END
```

After typing TO ZIGZAG and pressing RETURN the prompt changes from ? to >. This tells you you are in the procedural mode. At the end of the procedure type END and the prompt changes back to a ? telling you you are back in the immediate mode.

ZIGZAG is now part of LOGO's vocabulary. If you type it the Turtle will draw a zig-zag.

LOGO has a repeat command which we can use to save typing the same thing several times. You could rewrite ZIGZAG as:

```
TO ZIGZAG
REPEAT 3 [FORWARD 20 LEFT 150 FORWARD
  20 RIGHT 150]
END
```

The routine to be repeated is enclosed in square brackets, preceded by REPEAT and the number of times the routine is to be repeated.

If you think of a circle as a series of short moves and turns, REPEAT makes it easy to draw curves. For example REPEAT 180 [FORWARD 1 RIGHT 1] draws a semicircle.

PUTTING IT TOGETHER

You now have the ingredients to draw a racetrack. The way to solve a problem with LOGO is to break it into what Seymour Papert calls 'Mind Sized Bytes'. So, starting with the inside edge, first draw a curve:

```
REPEAT 180 [FORWARD 1 RIGHT 1]
```

Then a straight edge:

```
FORWARD 100
```

You can combine two curves and two straight edges in a procedure to draw the inside of the track:

```
TO INSIDE
REPEAT 2 [FORWARD 100 REPEAT 180
[FORWARD 1 RIGHT 1]]
END
```

It is quite legitimate to have a REPEAT within another REPEAT as long as you remember to close all the brackets at the end.

For the outside of the track you need a larger curve, created by increasing the size of the Turtle's steps.

```
REPEAT 180 [FORWARD 2 RIGHT 1]
```

is too large, but something in between:

```
REPEAT 90 [FORWARD 3 RIGHT 2]
```

is about right.

Now, the outside of the track.

```
TO OUTSIDE
REPEAT 2 [FORWARD 100 REPEAT 90
  [FORWARD 3 RIGHT 2]]
END
```

But typing:

```
OUTSIDE
INSIDE
```

doesn't give the ideal result, however.

You can write a procedure to begin the picture in a more suitable place by defining a procedure called BEGIN:

```
TO BEGIN
PENUP
LEFT 40
FORWARD 110
RIGHT 130
PENDOWN
END
```

Another procedure will move the Turtle into a suitable position to draw the inside track and also provides a starting line.

```
TO MOVE
RIGHT 90
FORWARD 30
LEFT 90
END
```

To see the track, call the procedures together by typing:

```
BEGIN
OUTSIDE
MOVE
INSIDE
```

You can then write a procedure to position the Turtle on the starting line.

```
TO START
LEFT 90
FORWARD 15
RIGHT 90
END
```

Once a procedure is in the computer's memory, LOGO allows you to use it in the same way as any other LOGO primitive. This means you can use the procedures to help define new procedures. You can combine the building blocks that you have called BEGIN, OUTSIDE, MOVE, INSIDE and START in a new procedure called GAME:

```
TO GAME
BEGIN
OUTSIDE
MOVE
INSIDE
START
END
```

Whenever you type GAME, the race track will appear with the Turtle on the starting line.

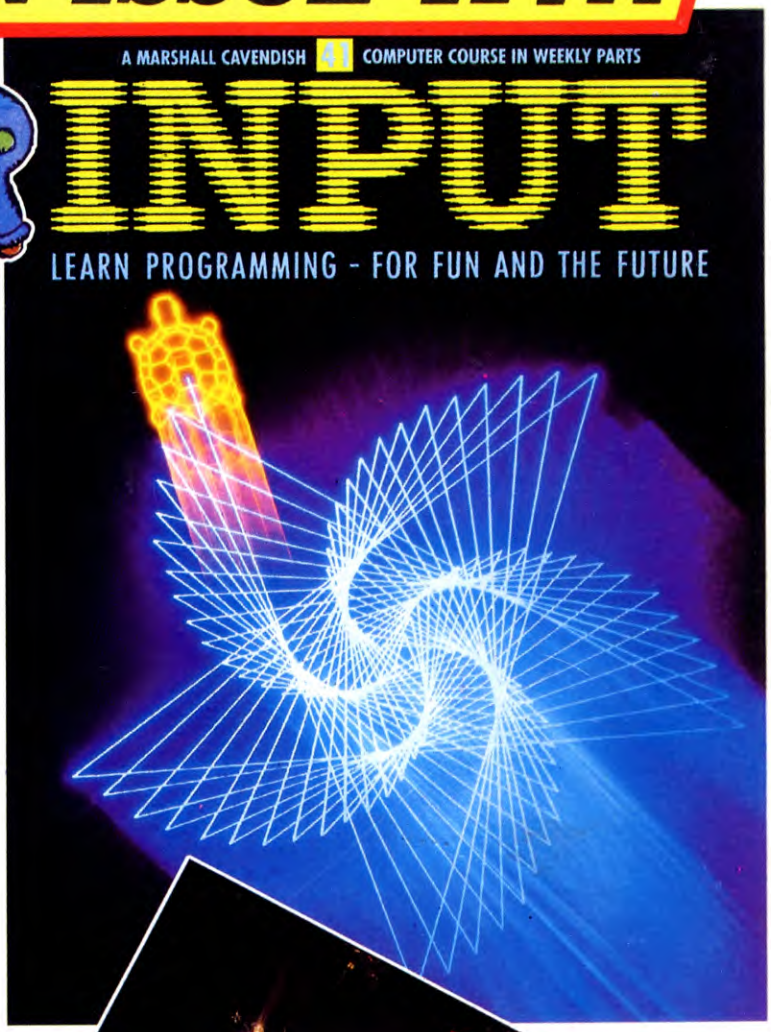
All LOGO programs are constructed in this way. Breaking a program into small building blocks makes it easier to construct and debug. And next time, you'll see how to make up more complicated programs using these techniques.

CUMULATIVE INDEX

An interim index will be published each week. There will be a complete index in the last issue of *INPUT*.

COMING IN ISSUE 41 . . .

- *Get into some advanced programming techniques by learning what's involved in RECURSION. These loops within loops can be used to solve a variety of otherwise intractable problems*
- *Use your micro to take the hard work out of furniture moving, with a ROOM PLANNER program that lets you make sure that everything is going to fit*
- *Continue building up your WAR GAME with the routines that create the map and allow the units to move around the field*
- *LANGUAGES continues by showing you some of the more sophisticated GRAPHICS available in LOGO*
- *In CLIFFHANGER, program in the routine that starts the ROCKS ROLLING DOWN*



ASK YOUR NEWSAGENT FOR INPUT