Computing with the AMSTRAD

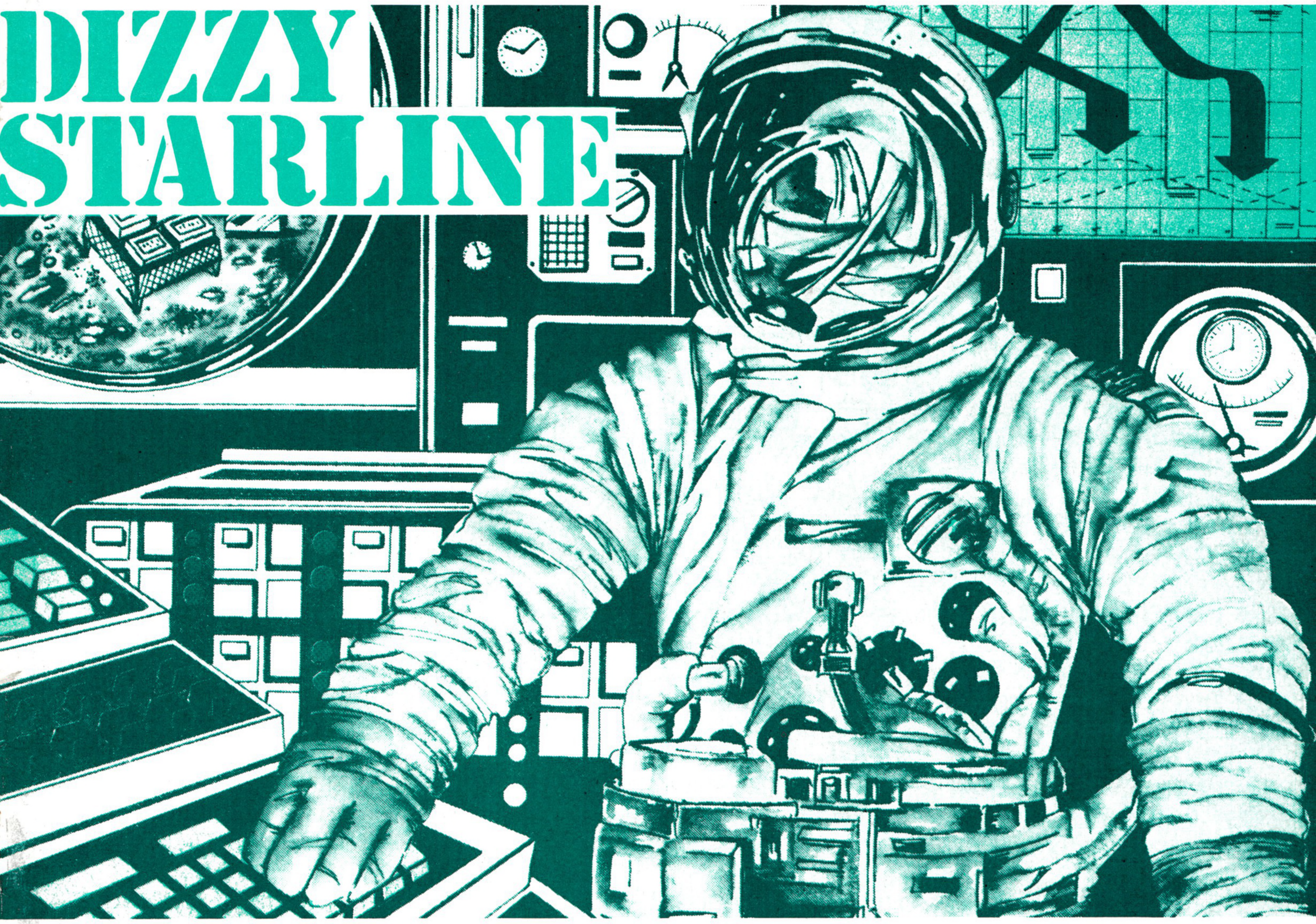
February, 1987

$3.95

A Database Publication

The independent magazine for Amstrad computer users

Registered by Australia Post - Publication No. VBP7811



LISTINGS BONANZA!

DIZZY STARLINE - AN INTERPLANETARY TRADING GAME
SPACE BASE - PROTECT EARTH FROM THE DEADLY ALIENS
TEXT EDITOR - SIMPLE BUT VERSATILE
VARIABLE DUMP - LIST THOSE VARIABLES!
PONTOON - OUR VERSION OF THE GREAT CARD GAME
BIORHYTHMS - WHAT DOES THE FUTURE HOLD FOR YOU?
FUNCTION KEY LISTER - NOW YOU DON'T HAVE TO REMEMBER

TASWORD 464 - THE WORD PROCESSOR

WE HAVE ONLY TWENTY LEFT AND ARE SELLING THESE AT ONLY $29.95 ON TAPE AND $41.95 ON DISK. MANUAL IS COMPLETE WITH ADDENDUM TO ENABLE TRANSFER TO DISK. HURRY! THESE WON'T LAST.

TASWORD
464

The Word Processor

A
Tasman Software Program
for the
Amstrad CPC 464

(002) 29-4377

Strategy
Software **P.O. BOX 11, BLACKMANS BAY, TASMANIA 7152**

Computing With The Amstrad

February 1987

Published monthly by Strategy Software under license from Database Publications Ltd.

Telephone: [002] 29 4377
Telex: AA 58134
Attn. HT163

Mail: P.O. Box 11
Blackmans Bay
Tasmania 7152

Annual Subscription (12 issues)

Australia:	$ 40•
South Pacific (inc. NZ)	$A60

'Computing With The Amstrad' welcomes program listings and articles for publication. Material should be typed or computer printed, and preferably double spaced. Program listings should be accompanied by cassette tape or disk. Please enclose a stamped addressed envelope or the return of material cannot be guaranteed. Contributions accepted for publication by Database Publications or its licensee will be on an all-rights basis.

Contents

Whoops!... I shouldn't have said that....

This is the issue that wasn't. We had planned to bring you our new-look CWTA this month but there have been gremlins at work -again.

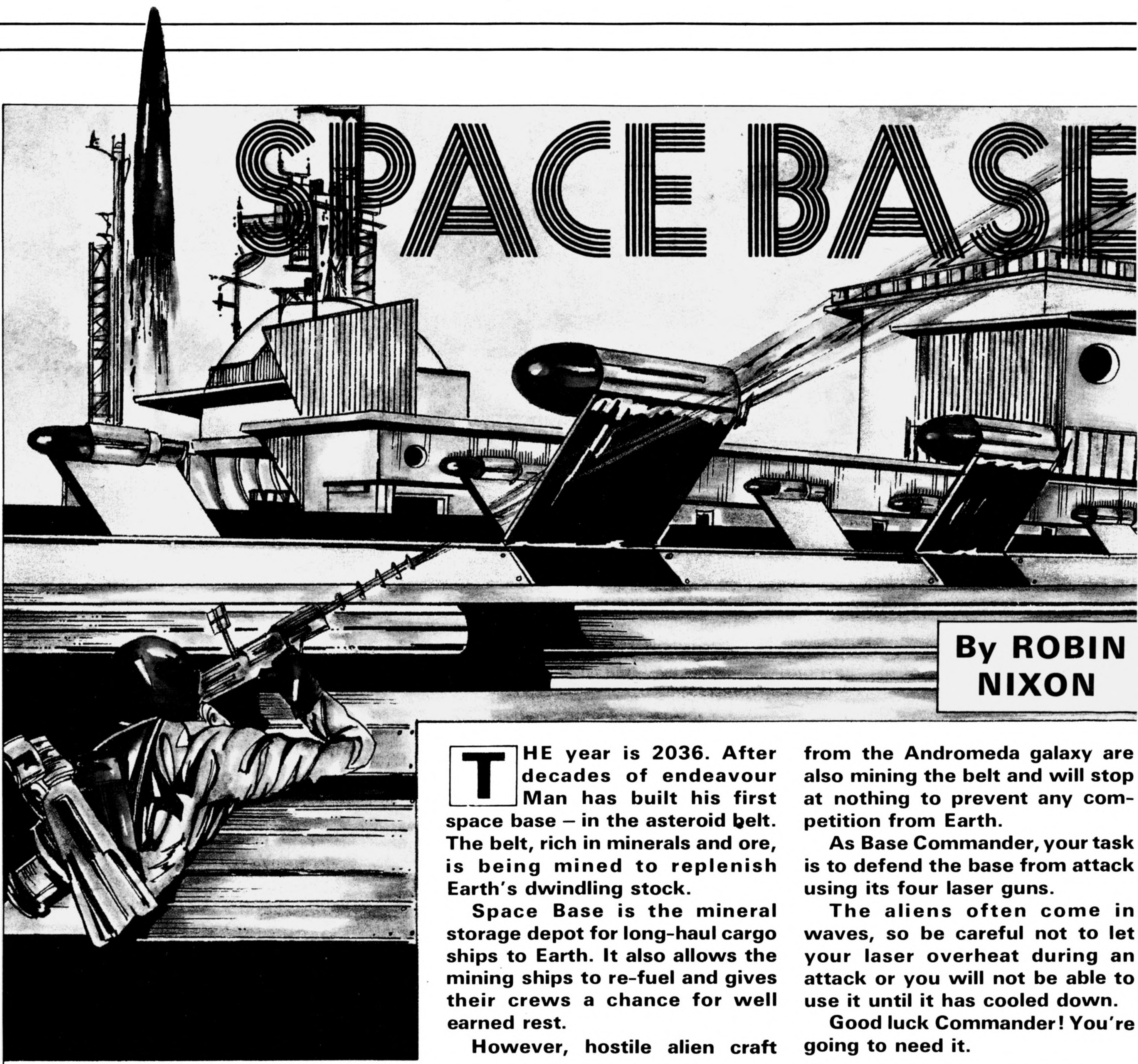
Life hasn't been terribly kind to us since the big move and I guess it can be put down to one thing - lack of planning. Believe it or not, our February issue was all ready and sitting at the printers when I received a 'phone call suggesting that it would be another two weeks before we could despatch it to all and sundry. After I'd been revived I had to make a snap desicion - do we go with the February issue as planned and waste a lot of work by having it arrive at the end of the month or do we knock up a 'quickie' and save what was the February issue until March?

Well by now you'll know the answer. We've tried to make this issue as much value for money as possible but we're disappointed that you won't see our real efforts until the March issue.We think it will be worth waiting for. Every cloud has it's silver lining and this little debacle means that future issues of CWTA will be prepared and printed about a month earlier than previously.

As a kind of a thankyou for persevering with us you'll find a special offer enclosed in the envelope with this magazine. At the time of writing this we don't know what it is but I'm sure you'll be as surprised as we are!

See you next month.

SPACE BASE



By ROBIN NIXON

THE year is 2036. After decades of endeavour Man has built his first space base – in the asteroid belt. The belt, rich in minerals and ore, is being mined to replenish Earth's dwindling stock.

Space Base is the mineral storage depot for long-haul cargo ships to Earth. It also allows the mining ships to re-fuel and gives their crews a chance for well earned rest.

However, hostile alien craft from the Andromeda galaxy are also mining the belt and will stop at nothing to prevent any competition from Earth.

As Base Commander, your task is to defend the base from attack using its four laser guns.

The aliens often come in waves, so be careful not to let your laser overheat during an attack or you will not be able to use it until it has cooled down.

Good luck Commander! You're going to need it.

```
100 REM *************************
110 REM *                       *
120 REM *       SPACE BASE      *
130 REM *                       *
140 REM *     By Robin Nixon    *
150 REM *                       *
160 REM * (c) Computing with the *
170 REM *        AMSTRAD        *
180 REM *                       *
190 REM *************************
200 REM
210 SYMBOL AFTER 220:DEFINT A-Z
220 SYMBOL 220,0,60,24,60,60,60,24
230 SYMBOL 221,0,0,58,126,126,58,0,0
240 SYMBOL 222,0,24,60,60,60,24,60,0
250 SYMBOL 223,0,0,92,126,126,92,0,0
260 SYMBOL 230,0,0,0,0,32,16,8,4
270 SYMBOL 231,0,0,0,0,129,66,36,24
280 SYMBOL 232,0,0,0,0,4,8,16,32
290 SYMBOL 233,34,17,15,1,1,15,17,34
300 SYMBOL 234,36,231,60,36,36,60,231
,36
310 SYMBOL 235,68,136,240,128,128,240
,136,68
320 SYMBOL 236,4,8,16,32,0,0,0,0
330 SYMBOL 237,24,0,0,0,0,0,0,0
340 SYMBOL 238,32,16,8,4,0,0,0,0
350 SYMBOL 240,15,31,63,127,255,255,2
55,255
360 SYMBOL 241,255,255,255,255,255,25
5,255,255
370 SYMBOL 242,129,195,231,255,255,25
5,255,255
380 SYMBOL 243,240,248,252,254,255,25
5,255,255
390 SYMBOL 244,255,127,63,31,31,63,12
7,255
400 SYMBOL 245,255,254,252,248,248,25
2,254,255
410 SYMBOL 246,255,255,255,255,127,63
,31,15
420 SYMBOL 247,255,255,255,255,255,23
1,195,129
430 SYMBOL 248,255,255,255,255,254,25
2,248,240
440 SYMBOL 249,165,90,165,90,90,165,9
0,165
450 SYMBOL 250,24,24,60,24,126,24,255
,24
460 SYMBOL 251,64,80,84,255,255,84,80
,64
470 SYMBOL 252,24,255,24,126,24,60,24
,24
480 SYMBOL 253,2,10,42,255,255,42,10,
2
490 CALL &BC02:INK 0,0:INK 2,2:BORDER
 0:DIM A(5),B(5):HS=2000
500 MODE 1:INK 2,26:SC=0:LV=1:SH=56:P
EN 2:PAPER 3:CLS:LOCATE 16,2:PRINT"SP
ACE BASE"
```

MAJOR VARIABLES

A(5)	Flag to indicate whether an alien is in position.
B(5)	Position of bombs when fired.
SC	Player's current score.
LV	Level.
SH	Shields left.
LZ	Laser left.
LZ1	Flag set if laser overheats.
LZ2	Counter used to simulate the laser's cooling.
LZC	Laser colour.
A1$ A2$ A3$	The aliens.
LB	Position of laser.
R	Used as a random integer in various places.
K1	True if Z pressed or joystick moved left.
K2	True if X pressed or joystick moved right.

PROGRAM STRUCTURE

220-250	Data for bombs. User defined characters.
260-340	Data for aliens. User defined characters.
350-480	Data for base. User defined characters.
490-640	Set up and instructions. Initialise the colours and display instructions.
650-880	Main program loop. Sets up the screen, prints the score, shields remaining, laser temperature and high score. Depending on which level you are at it decides how quickly and how violently the aliens are to attack.
890-960	Set a bomb going in the correct direction.
970-1000	Put an alien on the screen in a random position.
1010-1060	Keeps bombs on course.
1070-1080	Check to see if a bomb has hit the base. If so explosions occur and the shields are lowered.
1090-1270	Move laser.
1280-1370	Fire laser. Destroy any bombs and kill any aliens that are in its path.
1380	Removes a bomb when shot and increases the score.
1390	Removes an alien when shot and increases the score.
1400-1430	Scan keyboard and joystick for an input.
1440	Does sound effects and sets up starry screen background.

```
510 PEN 1:PRINT:PRINT:PRINT"You are t
he captain of Earth's first"
520 PRINT"space base in the asteroid
belt."
530 PRINT:PEN 0:PRINT"Your station ac
ts as a base for mining"
540 PRINT"ships to re-fuel and their
crews"
550 PRINT"to rest."
560 PRINT:PEN 2:PRINT"However, hostil
e alien craft from the"
570 PRINT"Andromeda galaxy are also m
ining the"
580 PRINT"belt and will stop at nothi
ng to prevent";
590 PRINT"any competition from Earth.
"
600 PRINT:PEN 1:PRINT"Your task is to
 defend the base from"
610 PRINT"attack."
620 PRINT:PEN 0:PRINT"Use Z and X to
control your lazer and"
630 PRINT"press ENTER to fire."
640 PAPER 2:PEN 0:LOCATE 16,24:PRINT"
Press a key";:WHILE INKEY$<>"" OR JOY
(0)>0:WEND:WHILE INKEY$="" AND JOY(0)
=0:WEND:INK 2,2
650 WHILE SH>-1:MODE 0:PAPER 3:CLS:PE
N 2:LOCATE 7,13:PRINT"LEVEL";LV;:FOR
Z=1 TO 5000:NEXT:MODE 1:PAPER 0:CLS:L
Z=40:LZ1=0:LZ2=0:LZC=1:HT=0:ERASE A,B
:DIM A(5),B(5)
660 GOSUB 1440:PEN 3:PAPER 2:LOCATE 1
,1:PRINT"LAZER";:PEN 1:PAPER 3:LOCATE
 33,1:PRINT" SCORE ";:PEN 3:PAPER 1:L
OCATE 1,25:PRINT"SHIELDS";:PEN 2:PAPE
R 1:LOCATE 32,25:PRINT"HI SCORE";:PAP
ER 1:LOCATE 1,2:PRINT STRING$(5,32);:
PAPER 2:LOCATE 1,24
670 PRINT STRING$(7,32);:PEN 2:PAPER
0:LOCATE 33,2:PRINT SC;:PEN 3:LOCATE
32,24:PRINT HS;:FOR Z=57 TO SH+1 STEP
 -1:MOVE Z*2,16:DRAW Z*2,31,0:NEXT
680 PAPER 0:PEN 1:LOCATE 18,11:PRINT
CHR$(240);CHR$(241);CHR$(242);CHR$(24
1);CHR$(243);
690 LOCATE 18,12:PRINT STRING$(5,241)
;:LOCATE 18,13:PRINT CHR$(244);STRING
$(3,241);CHR$(245);
700 LOCATE 18,14:PRINT STRING$(5,241)
;:LOCATE 18,15:PRINT CHR$(246);CHR$(2
41);CHR$(247);CHR$(241);CHR$(248);
710 PEN 2:PAPER 1:LOCATE 19,12:PRINT
CHR$(249);:LOCATE 21,12:PRINT CHR$(24
9);:LOCATE 20,13:PRINT CHR$(249);:LOC
ATE 19,14:PRINT CHR$(249);:LOCATE 21,
14:PRINT CHR$(249);
720 A1$=CHR$(230)+CHR$(231)+CHR$(232)
:A2$=CHR$(233)+CHR$(234)+CHR$(235):A3
$=CHR$(236)+CHR$(237)+CHR$(238)
```

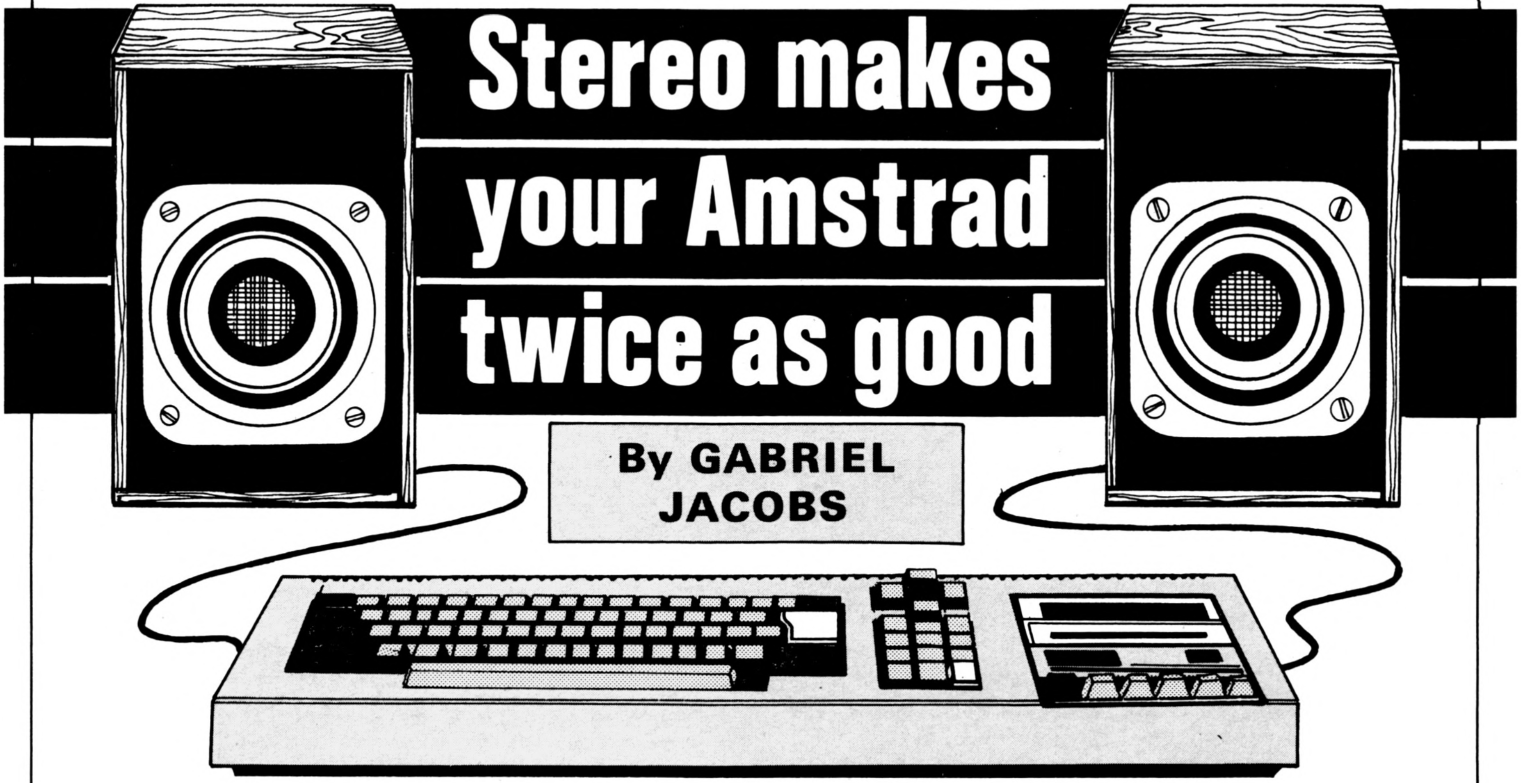
```
730 PEN 3:PAPER 0:LOCATE 20,11:PRINT
CHR$(242);:LOCATE 18,13:PRINT CHR$(24
4);:LOCATE 22,13:PRINT CHR$(245);:LOC
ATE 20,15:PRINT CHR$(247);
740 HT=0:LB=1:IP=0:PEN 3:PAPER 2:GOSU
B 1170:PEN 2:GOSUB 1220:PAPER 0
750 WHILE HT<LV*25+25 AND SH>-1
760 R=RND*(10-LV):IF R<1 THEN GOSUB 8
90
770 FOR X=1 TO 4:IF B(X)=0 THEN GOTO
780 ELSE GOSUB 1010
780 GOSUB 1400:NEXT
790 GOSUB 1090:R=RND*(10-LV):IF R<1 T
HEN GOSUB 910
800 IF INKEY(18)=0 OR JOY(0)>15 THEN
GOSUB 1280
810 LZ2=LZ2+1:IF LZ2=6 THEN LZ2=0:IF
LZ<39 THEN LZ=LZ+1:MOVE LZ*2,368:DRAW
 LZ*2,382,LZC
820 IF LZ<0 THEN LZ1=1:LZC=3
830 IF LZ1=1 AND LZ>14 THEN PAPER 1:L
OCATE 1,2:PRINT STRING$(2,32);:LZC=1:
LZ1=0:PAPER 0
840 WEND:LV=LV+1
850 PEN 2:PAPER 0:FOR Z=SH TO 0 STEP-
1:MOVE Z*2,16:DRAW Z*2,31,0:SC=SC+LV:
LOCATE 33,2:PRINT SC;:SOUND 1,20,1,7:
FOR Z1=1 TO 25:NEXT Z1,Z
860 WEND
870 FOR Z=1 TO 1000:INK RND*3,RND*26:
SOUND 1,Z,1,7:NEXT:INK 0,0:INK 1,24:I
NK 2,2:INK 3,6:MODE 0:PAPER 3:PEN 1:C
L9:LOCATE 6,13:PRINT"GAME  OVER";:FOR
 Z=1 TO 5000:NEXT:IF SC>HS THEN HS=SC
880 GOTO 500
890 R=RND*3+1
900 IF A(R)=0 THEN A(R)=1:PEN 2:FOR Z
=500 TO 0 STEP -23:SOUND 1,Z,1,6:NEXT
:ON R GOTO 970,980,990,1000
910 R=1:WHILE R<5 AND (A(R)=0 OR B(R)
>0):R=R+1:WEND:IF R=5 THEN RETURN
920 FOR Z=200 TO 400 STEP 20:SOUND 1,
Z,1,6:NEXT:ON R GOTO 930,940,950,960
930 B(R)=4:RETURN
940 B(R)=36:RETURN
950 B(R)=22:RETURN
960 B(R)=4:RETURN
970 LOCATE 19,1:PRINT A1$;:GOSUB 1400
:LOCATE 19,2:PRINT A2$;:GOSUB 1400:LO
CATE 19,3:PRINT A3$;:RETURN
980 LOCATE 37,12:PRINT A1$;:GOSUB 140
0:LOCATE 37,13:PRINT A2$;:GOSUB 1400:
LOCATE 37,14:PRINT A3$;:RETURN
990 LOCATE 19,23:PRINT A1$;:GOSUB 140
0:LOCATE 19,24:PRINT A2$;:GOSUB 1400:
LOCATE 19,25:PRINT A3$;:RETURN
1000 LOCATE 1,12:PRINT A1$;:GOSUB 140
0:LOCATE 1,13:PRINT A2$;:GOSUB 1400:L
OCATE 1,14:PRINT A3$;:RETURN
1010 ON X GOTO 1020,1030,1040,1050
1020 LOCATE 20,B(X):GOSUB 1080:IF B(X
)+1=11 THEN GOTO 1070 ELSE B(X)=B(X)+
1:PEN 3:LOCATE 20,B(X):GOTO 1060
1030 LOCATE B(X),13:GOSUB 1080:IF B(X
)-1=22 THEN GOTO 1070 ELSE B(X)=B(X)-
1:PEN 3:LOCATE B(X),13:GOTO 1060
1040 LOCATE 20,B(X):GOSUB 1080:IF B(X
)-1=15 THEN GOTO 1070 ELSE B(X)=B(X)-
1:PEN 3:LOCATE 20,B(X):GOTO 1060
1050 LOCATE B(X),13:GOSUB 1080:IF B(X
)+1=18 THEN GOTO 1070 ELSE B(X)=B(X)+
1:PEN 3:LOCATE B(X),13
1060 PRINT CHR$(219+X);:RETURN
1070 SOUND 1,0,15,5,0,0,5:FOR Z=0 TO
24 STEP 6:INK 1,Z:INK 2,Z+2:INK 3,Z/6
:FOR Z1=1 TO 30:NEXT Z1,Z:INK 2,2:INK
 3,6:MOVE SH*2,16:DRAW SH*2,30,0:SH%=
SH%-1:B(X)=0:PEN 2:GOSUB 1220:RETURN
1080 PEN 0:PRINT" ";:RETURN
1090 GOSUB 1400:IF INKEY(71)=-1 AND I
NKEY(63)=-1 AND JOY(0)<>4 AND JOY(0)<
>8 THEN IP=0:K1=0:K2=0:RETURN
1100 IF IP=1 THEN RETURN
1110 GOSUB 1400
1120 IF K1=0 AND K2=0 THEN RETURN
1130 IP=1:PEN 3:PAPER 0:GOSUB 1170:PE
N 0:GOSUB 1220:IF K1=1 THEN LB=LB-1:I
F LB=0 THEN LB=4
1140 IF K2=1 THEN LB=LB+1:IF LB=5 THE
N LB=1
1150 K1=0:K2=0
1160 PEN 3:PAPER 2:GOSUB 1170:PEN 2:G
OSUB 1220:RETURN
1170 ON LB GOTO 1180,1190,1200,1210
1180 LOCATE 20,11:PRINT CHR$(242):RET
URN
1190 LOCATE 22,13:PRINT CHR$(245):RET
URN
1200 LOCATE 20,15:PRINT CHR$(247):RET
URN
1210 LOCATE 18,13:PRINT CHR$(244):RET
URN
1220 ON LB GOTO 1230,1240,1250,1260
1230 PAPER 0:LOCATE 20,10:GOTO 1270
1240 PAPER 0:LOCATE 23,13:GOTO 1270
1250 PAPER 0:LOCATE 20,16:GOTO 1270
1260 PAPER 0:LOCATE 17,13:GOTO 1270
1270 PRINT CHR$(249+LB):RETURN
1280 GOSUB 1400:IF LZ1=1 THEN RETURN
ELSE MOVE LZ*2,368:DRAW LZ*2,382,0:LZ
=LZ-1:FOR Z=30 TO 10 STEP-3:SOUND 1,Z
,1,5:NEXT:ON LB GOSUB 1300,1320,1340,
1360
1290 PEN 2:GOSUB 1220:RETURN
1300 FOR Z=1 TO 4:GOSUB 1400:MOVE 310
,260:DRAW 310,362,Z:NEXT:IF B(LB)>0 T
HEN SOUND 1,0,20,7,0,0,15:LOCATE 20,B
(LB):GOTO 1380
1310 IF A(LB)=0 THEN RETURN ELSE SOUN
D 1,0,20,7,0,0,1:PEN 0:FOR Z=1 TO 3:L
OCATE 19,Z:PRINT"   ";:GOSUB 1400:NEX
T:GOTO 1390
1320 FOR Z=1 TO 4:GOSUB 1400:MOVE 370
,200:DRAW 580,200,Z:NEXT:IF B(LB)>0 T
HEN SOUND 1,0,20,7,0,0,15:LOCATE B(LB
),13:GOTO 1380
1330 IF A(LB)=0 THEN RETURN ELSE SOUN
D 1,0,20,7,0,0,1:PEN 0:FOR Z=12 TO 14
:LOCATE 37,Z:PRINT"   ";:GOSUB 1400:N
EXT:GOTO 1390
1340 FOR Z=1 TO 4:GOSUB 1400:MOVE 310
,140:DRAW 310,42,Z:NEXT:IF B(LB)>0 TH
EN SOUND 1,0,20,7,0,0,15:LOCATE 20,B(
LB):GOTO 1380
1350 IF A(LB)=0 THEN RETURN ELSE SOUN
D 1,0,20,7,0,0,1:PEN 0:FOR Z=23 TO 25
:LOCATE 19,Z:PRINT"   ";:GOSUB 1400:N
EXT:GOTO 1390
1360 FOR Z=1 TO 4:GOSUB 1400:MOVE 250
,200:DRAW 50,200,Z::NEXT:IF B(LB)>0 T
HEN SOUND 1,0,20,7,0,0,15:LOCATE B(LB
),13:GOTO 1380
1370 IF A(LB)=0 THEN RETURN ELSE SOUN
D 1,0,20,7,0,0,1:PEN 0:FOR Z=12 TO 14
:LOCATE 1,Z:PRINT"   ";:GOSUB 1400:NE
XT:GOTO 1390
1380 PEN 0:PRINT " ";:B(LB)=0:SC=SC+2
*LV:PEN 2:PAPER 0:LOCATE 33,2:PRINT S
C;:HT=HT+1:RETURN
1390 A(LB)=0:SC=SC+1*LV:PEN 2:PAPER 0
:LOCATE 33,2:PRINT SC;:HT=HT+1:RETURN
1400 IF K1=1 OR K2=1 THEN RETURN
1410 IF JOY(0)=4 THEN K1=1:RETURN
1420 IF JOY(0)=8 THEN K2=1:RETURN
1430 K1=INKEY(71)+1:K2=INKEY(63)+1:RE
TURN
1440 PAPER 0:CLS:FOR Z=1 TO 500:SOUND
 1,500-Z,1,5:PLOT RND*620,RND*400,RND
*2+1:NEXT:RETURN
```

Can't hack keying it in?

See Page 38

Stereo makes your Amstrad twice as good

By GABRIEL JACOBS



THERE'S no reason in principle why the Amstrad should not be connected to a conventional hi-fi stereo system. But for most people this is going to involve constantly carrying pieces of equipment around the house and always having to reposition the micro between the two speakers.

So if you've installed your machine in some remote corner to keep it away from uninvited fingers – or yourself away from the rest of mankind – and if you also want to take advantage of the full potential of its stereo capability, then the C-Lect 5000 kit of stereo amplifier, speakers and headphones, designed to sit permanently on your desk, may be just what you need.

The amplifier, purpose-built for the Amstrad, is housed in a sturdy black box measuring 7 by 5 by 3½ inches. It plugs into the stereo output port, taking its power from the mains via a transformer supplied.

It has standard 3.5mm jack sockets for input and output, two separate volume controls, a selector switch for speakers or headphones and comes with all necessary leads.

It will drive any speakers of at least 4 ohms impedance, but the full C-Lect kit includes a pair of Japanese 3 inch double cone, air suspension units. These are capable of handling up to 30 watts each, and so I was surprised to find that the amplifier delivers only 1½ watts per channel – plenty to spare, to say the least.

In practice, however, the amplifier and speakers proved to be well matched. Volume levels will certainly be adequate for all except the very hard-of-hearing, or those determined to damage their eardrums.

Stereo separation is excellent, and the frequency response is fairly flat, giving uniformly good sound across the range. Bass response in particular is impressive for such a small system.

The relatively cheap Korean headphones supplied with it, however, give rather poor results in the bass.

Once you have set up the C-Lect system, or any stereo amplification system, the Amstrad's own unremarkable mono speaker is bypassed, and any sound generated is diverted to the stereo port.

This means that currently available software written with stereo sound, such as Protek's Hunter/Killer, will be automatically implemented as its programmers intended it to be. And if you haven't yet heard some of your games with good stereo sound, you've a real surprise waiting for you.

A tape of a gavotte by Bach, which was sent with the review kit for demo purposes, gave a fair idea of what can be achieved musically with the Amstrad's three sound channels, given the proper equipment.

But we're still waiting for the appearance of a good range of music software comparable with that available for older-established micros.

When Island Logic, for example, produce a version of their Music System for the Amstrad, the stereo results, output through an external amplifier, ought to put the competition in the shade. In the meantime, the C-Lect system only serves to emphasise the present lack of this kind of software.

At the time of writing I know of only one other Amstrad-dedicated stereo amplifier actually available, the one which comes with Dk'tronic's Speech Synthesiser.

This will give you the same basic capabilities as the C-Lect 5000 – and, of course, you get a versatile speech synthesiser thrown in – but it will not give you the same quality of sound.

So if you're less interested in digital speech than in good quality digital music, not to mention good-quality crashes, bleeps and sirens, then the C-Lect system is worth buying.

Cost is £35 for the full kit, and £18.50 for the amplifier and headphones only. Not cheap, but affordable and quite good value for money.

DIZZY STARLINE

By STEVEN MITCHELL

YOU are the captain of a Dizzy Starline cargoship and your company – verging on bankruptcy as usual – has told you to sell your cargo to salvage the firm's finances.

Your cargo of Gowong personal computers have to be sold in each of the 49 sectors in quadrant 10 of the galaxy.

Unfortunately, due to a few defects in manufacture, the Gowong computers, on the orders of the Galactic Trade Federation, are being withdrawn from sale in 49 galactic days.

These same defects make it inadvisable to return to a sector after selling your cargo there.

You have contracts in four of the sectors and must deliver in these sectors on a particular day, not before and not after.

Each hyperspace jump between sectors takes one galactic day. However your starship, manufactured by a subsidiary of the Gowong Company, has a minor defect in its hyperspace drive and can only make jumps similar to a knight's move in chess.

So your task is to work out a route that will visit each of the 49 sectors once and only once and also meet your delivery dates.

To help you, Arnold, the ship's computer, will display a grid map of the quadrant showing your firm delivery dates, your spaceship and a trail of planets to show you which sectors you have visited.

He will also give you full instructions on how to use the map, the nature of the problem and keep track of your moves to stop you if you get stuck, fail to meet a delivery date or try to make an invalid move.

Arnold – who is not manufactured by the Gowong Computer Company – will also, if asked, show you his solution to the problem. See if you can prove your human supremacy and find a different solution.

The program is heavily structured with one main section, the Calling Menu, from line 270-480. This displays the four options available, calls the selected option and ends each game.

The subroutine game loop from line 700 to 780 controls the flow of each game and is clearly REMmed, hopefully making the program logic easy to follow.

All arithmetic is integer, so I've taken the opportunity to use the DEFINT function in line 130, forcing string variables using the $ symbol.

PROGRAM STRUCTURE

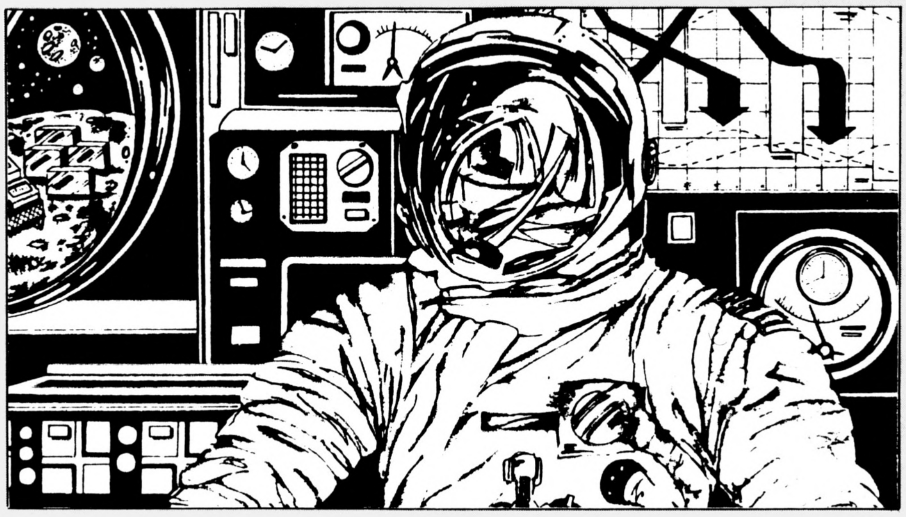
110-250	Declare and fill arrays, define characters and declare two volume and tone envelopes.
270-480	Calling menu.
500-680	Instructions.
700-780	Game loop.
800-950	Draw grid map and reset variables.
970-1150	Accept and validate moves, call routine to flash grid co-ordinates, set flag if user quits or fails to meet a delivery date.
1170-1260	Move ship, leave a planet in the old ship position, increase and display date, update check array and *spos* variable. Call routine to reset grid coordinates.
1280-1300	Draw horizontal ring around planet 1.
1320-1330	Draw diagonal ring around planet 2.
1350-1370	Draw continent and moon on planet 3.
1390-1440	Test for more legal moves, set *flag* if no moves possible.
1460-1510	Computer game – use solution read from data.
1530-1540	Empty keyboard buffer.
1560-1580	Display message, reset inks and await keypress.
1600-1610	Reset mode, set all inks to border colour.
1630-1650	Display error message.
1670-1690	Flash x and y coordinates.
1710-1720	Reset flashing x and y coordinates.
1750	Number of legal moves.
1780	x and y coordinates of background stars.
1810-1870	Legal moves from each square.
1900	Data for computer solution.
1920	Data for sound routine on successful completion.

MAIN VARIABLES

square(49,3)	Holds grid and x and y coordinates of square.
check(49,9)	Holds number of legal moves and their square numbers.
ship$	Ship characters.
planet$	Planet characters.
opt$	User input.
opt	Menu Option.
flag	Flag for why a game has ended.
date	Holds current date.
spos	Holds number of square occupied by ship.
grid	Coordinates along bottom and side of map.
sqto	Holds square you wish to move to.
optx,opty	Hold vertical and horizontal grid co-ordinates of square you wish to move to.
cp	Holds square number of each legal move.
game	Indicates computer game.
x,y	Hold text x and y coordinates.
gx,gy	Hold graphic x and y coordinates.

```
10 REM The Dizzy Starline
20 REM and the Gowong Computers
30 REM
40 REM          By
50 REM
60 REM     S.Mitchell
70 REM
80 REM (c) Computing with
90 REM The Amstrad
100 REM
110 REM * Set up variables *
120 REM
130 DEFINT a-z:DIM square(49,3),check
(49,9):SYMBOL AFTER 248
140 SYMBOL 248,0,48,255,54,55,49,48,6
0
150 SYMBOL 249,0,12,255,108,236,140,1
2,60
160 SYMBOL 250,3,15,31,31,31,31,15,3
170 SYMBOL 251,192,240,248,248,248,24
8,240,192
180 ship$=CHR$(248)+CHR$(249):planet$
=CHR$(250)+CHR$(251)
190 ENV 1,15,14,3:ENV 2,8,1,4:ENT 1,5
0,-1,1:ENT 2,1,0,60,44,-1,2
200 grid=11:x=11:y=16
210 FOR z=1 TO 49:square(z,1)=grid:sq
uare(z,2)=x:square(z,3)=y
220 grid=grid+10:x=x+3:IF INT(grid/10
)=8 THEN grid=11+grid MOD 10:x=11:y=y
-2
230 NEXT
240 FOR z=1 TO 49:READ check(z,1):NEX
T:RESTORE 1810
250 FOR z=1 TO 49:FOR t=2 TO check(z,
1)+1:READ check(z,t):NEXT:NEXT
260 REM
270 REM * Calling Menu *
280 REM
290 BORDER 3:GOSUB 1600:ZONE 1:PEN 3:
LOCATE 2,2
300 PRINT "DIZZY STARLINE & THE GOWON
G COMPUTERS":PRINT,STRING$(37,"-")
310 PEN 2:LOCATE 2,7:PRINT "Press num
ber alongside your choice:-"
320 PEN 1:PRINT:PRINT,"1] To see the
instructions"
330 PRINT:PRINT,"2] To play a game"
340 PRINT:PRINT,"3] To watch the comp
uter play"
350 PRINT:PRINT,"4] End the program"
360 INK 0,14:INK 1,24:INK 2,6:INK 3,1
:GOSUB 1540
370 opt$=INKEY$:IF opt$="" OR INSTR("
1234",opt$)=0 GOTO 370 ELSE opt=VAL(o
pt$)
380 IF opt=4 THEN MODE 1:CALL &BC02:P
```

```
APER 0:PEN 1:CLS:END
390 ON opt GOSUB 500,700,1460:IF opt=
1 GOTO 290
400 WINDOW #1,1,40,21,25:CLS #1:PEN #
1,0:PRINT #1
410 IF flag=2 THEN PRINT #1," You hav
e not met a firm delivery date":FOR z
=1 TO 10:SOUND 1,120,50,7,0,1:NEXT
420 IF flag=3 THEN PRINT #1,TAB(5);"Y
ou can not make any more moves":FOR z
=1 TO 10:SOUND 1,160,50,7,1,1:NEXT
430 IF flag=4 THEN PRINT #1,TAB(7);"A
ll consignments delivered !":RESTORE
1930:FOR z=1 TO 4:READ n1,n2,n3,d1,d2
,d3:SOUND 49,n1,d1,6:SOUND 42,n2,d2,7
,1:SOUND 28,n3,d3,5,2,2:NEXT
440 WHILE SQ(1)>127:WEND:IF opt=3 GOT
O 290
450 GOSUB 1600:PEN 2:LOCATE 15,2:PRIN
T "END OF GAME"
460 PRINT TAB(15);STRING$(11,"-"):PEN
 3:LOCATE 2,5
470 IF date<49 THEN PRINT "YOU MANAGE
D TO DELIVER";date;"CONSIGNMENTS" ELS
E PRINT "WELL DONE !  ALL GOWONG COMP
UTERS SOLD"
480 GOTO 310
490 REM
500 REM * Instructions *
510 REM
520 GOSUB 1600:PEN 2:ZONE 3
530 PRINT:PRINT TAB(15);"INSTRUCTIONS
":PRINT TAB(15);STRING$(12,"-")
540 PEN 1:PRINT:PRINT "1] You have to
 deliver consignments of":PRINT,"'GOW
ONG' computers in each sector."
550 PRINT:PRINT "2] You must visit th
e 49 sectors":PRINT,"in 49 Galactic d
ays."
560 PRINT:PRINT "3] You cannot visit
the same sector":PRINT,"twice."
570 PRINT:PRINT "4] You must deliver
in the following":PRINT,"sectors on t
hese dates."
580 PEN 3:PRINT:PRINT,"Sector 7,1 on
the 13th.":ZONE 5:PRINT,"..   4,4 on
the 25th."
590 PRINT,"..   1,7 on the 37th.":PRI
NT,"..   7,7 on the 49th."
600 ZONE 3:GOSUB 1570:GOSUB 1600:PEN
1
610 PRINT:PRINT "5] Each hyper-space
jump takes one":PRINT,"Galactic day."
620 PRINT:PRINT "6] You spaceship can
 only jump sectors":PRINT,"in the fol
lowing ways :-"
630 PEN 3:PRINT:PRINT,"2 RIGHT, 1 UP
```



```
OR DOWN.":PRINT,"2 LEFT, 1 UP OR DOWN
."
640 PRINT,"2 UP, 1 LEFT OR RIGHT.":PR
INT,"2 DOWN, 1 LEFT OR RIGHT."
650 PEN 1:PRINT:PRINT "7] To move to
a sector enter the":PRINT,"grid co-or
dinates by pressing :-"
660 PEN 3:PRINT:PRINT,"the number alo
ng the bottom first,":PRINT,"next, th
e number on the left side."
670 PEN 1:PRINT:PRINT:PRINT TAB(5);"Y
ou are now in the plotting room":PRIN
T,"working out your course, GOOD LUCK
 !"
680 GOSUB 1570:RETURN
690 REM
700 REM * Game Loop *
710 REM
720 GOSUB 800:REM Set up board
730 GOSUB 970:REM Accept user move
740 IF flag>0 THEN RETURN:REM end gam
e if user quits or date not met
750 GOSUB 1170:REM make move
760 GOSUB 1390:REM test for more lega
l moves
770 IF flag>0 THEN RETURN:REM end gam
e if no more legal moves
780 GOTO 730:REM Go back to accept ne
xt move
790 REM
800 REM * Set up board *
810 REM
820 GOSUB 1600:WINDOW #0,1,40,1,20:WI
NDOW #1,1,31,21,25:WINDOW #2,32,40,21
,25
830 PAPER #1,2:PAPER #2,2:PEN 2:PEN #
1,1:PEN #2,1:CLS #1:CLS #2
840 flag=0:date=1:spos=1:grid=0:RESTO
RE:FOR z=1 TO 49:READ check(z,1):NEXT
850 TAG:FOR gx=152 TO 488 STEP 48:ORI
GIN gx,136:DRAWR 0,224,2
860 ORIGIN gx,120:grid=grid+1:IF grid
<8 THEN PRINT grid;
870 NEXT:grid=0
880 FOR gy=136 TO 360 STEP 32:ORIGIN
152,gy:DRAWR 336,0,2
890 ORIGIN 104,gy+20:grid=grid+1:IF g
rid<8 THEN PRINT grid;
900 NEXT:TAGOFF
910 LOCATE 29,16:PRINT "13":LOCATE 20
,10:PRINT "25":LOCATE 11,4:PRINT "37"
:LOCATE 29,4:PRINT "49":PEN 1:LOCATE
11,16:PRINT ship$
920 FOR z=1 TO 19:READ x,y:LOCATE x,y
:PRINT CHR$(144):NEXT
930 PRINT #2:PRINT #2,"STARDATE":PRIN
T #2:PRINT #2,"37084.00"
940 LOCATE #1,2,4:PRINT #1,"Press 'Q'
 to end game"
950 RANDOMIZE TIME:INK 0,0:INK 1,24:I
NK 2,14:INK 3,6,24:RETURN
960 REM
970 REM * Accept user move *
980 REM
990 LOCATE #1,2,2:PRINT #1,"Move to s
ector ?";CHR$(18)
1000 GOSUB 1530:PEN #1,0
1010 opt$=LOWER$(INKEY$):IF opt$="" O
R INSTR("1234567q",opt$)=0 GOTO 1010
1020 IF opt$="q" THEN flag=1:RETURN
1030 IF sqto$<>"" THEN LOCATE #1,19,2
:PRINT #1,opt$ ELSE LOCATE #1,17,2:PR
INT #1,opt$;",";:PEN #1,1:PRINT #1,"?
"
1040 sqto$=sqto$+opt$:TAG:PLOT 1000,0
,3
1050 IF LEN(sqto$)<2 THEN GOSUB 1680:
GOTO 1000 ELSE GOSUB 1690
1060 sqto=VAL(sqto$):sqto$=""
1070 FOR z=1 TO 49:IF sqto=square(z,1
) THEN sqto=z:z=49
1080 NEXT
1090 IF check(sqto,1)=0 THEN GOSUB 16
30:GOTO 990
1100 FOR z=2 TO check(spos,1)+1:IF ch
eck(spos,z)=sqto GOTO 1120
```

```
1110 NEXT:GOSUB 1630:GOTO 990
1120 IF (sqto=7 AND date<12) OR (sqto
=25 AND date<24) OR (sqto=43 AND date
<36) OR (sqto=49 AND date<48) THEN GO
SUB 1630:GOTO 990
1130 IF (sqto<>7 AND date=12) OR (sqt
o<>25 AND date=24) OR (sqto<>43 AND d
ate=36) THEN flag=2:RETURN
1140 PEN 3:LOCATE square(sqto,2),squa
re(sqto,3):PRINT CHR$(197);CHR$(199):
PEN 1
1150 PEN #1,1:RETURN
1160 REM
1170 REM * Make move *
1180 REM
1190 LOCATE square(spos,2),square(spo
s,3):PEN 3:PRINT ship$
1200 FOR z=2 TO 7:FOR t=3000 TO 0 STE
P-200:SOUND 7,t,1,z:NEXT:NEXT
1210 PEN 1:LOCATE square(sqto,2),squa
re(sqto,3):PRINT ship$
1220 PEN 2:LOCATE square(spos,2),squa
re(spos,3):PRINT planet$
1230 date=date+1:LOCATE #2,6,4:PRINT
#2,USING ".##";date/100
1240 IF game<>1 THEN GOSUB 1710
1250 ORIGIN 0,0:r=INT(RND*3)+1:ON r G
OSUB 1280,1320,1350
1260 check(spos,1)=0:spos=sqto:RETURN
1270 REM
1280 REM * Planet 1 *
1290 gx=square(spos,2)*16-12:gy=16*(2
5-square(spos,3))+10:PLOT gx,gy,1
1300 PLOTR -2,0:PLOTR -2,-2:PLOTR 2,-
2:PLOTR 2,0:PLOTR 2,0:PLOTR 2,0:PLOTR
 2,-2:DRAWR 10,0:PLOTR 2,2:DRAWR 6,0:
PLOTR 2,2:PLOTR -2,2:PLOTR -2,0:RETUR
N
1310 REM
1320 REM * Planet 2 *
1330 gx=square(spos,2)*16-12:gy=16*(2
5-square(spos,3))+2:PLOT gx,gy,1:PLOT
R 0,-2:PLOTR 2,0:MOVER 2,2:DRAWR 14,1
4:PLOTR 0,2:PLOTR -2,0:PLOTR -2,-2:RE
TURN
1340 REM
1350 REM * Planet 3 *
1360 gx=square(spos,2)*16-2:gy=16*(25
-square(spos,3))+12:PLOT gx,gy,1
1370 PLOTR 0,-2:DRAWR 4,0:PLOTR -2,-2
:PLOTR 2,0:PLOTR -2,-2:PLOTR -2,-2:DR
AWR 4,0:PLOTR -4,-2:PLOTR 2,0:PLOTR -
8,4:PLOTR 2,2:PLOTR -8,8,3:RETURN
1380 REM
1390 REM * Test for more legal moves
*
1400 REM
1410 IF date=49 THEN flag=4:RETURN
1420 FOR z=2 TO check(spos,1)+1:cp=ch
eck(spos,z):IF check(cp,1)=0 GOTO 144
0
1430 IF (cp=7 AND date<>12) OR (cp=25
 AND date<>24) OR (cp=43 AND date<>36
) OR (cp=49 AND date<>48) GOTO 1440 E
LSE RETURN
1440 NEXT:flag=3:RETURN
1450 REM
1460 REM * Computer game *
1470 REM
1480 GOSUB 800:CLS #1:LOCATE #1,6,3:P
RINT #1,"Move to Sector"
1490 game=1:RESTORE 1900:FOR y=1 TO 4
8:READ sqto:LOCATE #1,20,3
1500 PRINT #1,LEFT$(STR$(square(sqto,
1)),2);",";RIGHT$(STR$(square(sqto,1)
),1)
1510 GOSUB 1170:NEXT:game=0:flag=4:RE
TURN
1520 REM
1530 REM * Empty keyboard buffer *
1540 WHILE INKEY$<>"":WEND:RETURN
1550 REM
1560 REM * Continue, reset inks *
1570 PEN 2:LOCATE 8,24:PRINT "PRESS A
NY KEY TO CONTINUE"
1580 INK 0,9::INK 1,18:INK 2,0:INK 3,
24:GOSUB 1540:CALL &BB18:RETURN
1590 REM
1600 REM * Reset mode,inks *
1610 MODE 1:FOR z=0 TO 3:INK z,3:NEXT
:PAPER 0:CLS:RETURN
1620 REM
1630 REM * Error message *
1640 SOUND 1,1000,40,7:PEN #1,0:LOCAT
E #1,17,2:PRINT #1,"Invalid move"
1650 FOR t=1 TO 3000:NEXT:GOSUB 1710:
sqto$="":PEN #1,1:RETURN
1660 REM
1670 REM * Flash grid co-ordinates *
1680 optx=VAL(opt$):gx=104+48*optx:OR
IGIN gx,128:PRINT VAL(opt$);:TAGOFF:R
ETURN
1690 opty=VAL(opt$):gy=124+32*opty:OR
IGIN 104,gy:PRINT VAL(opt$);:TAGOFF:R
ETURN
1700 REM
1710 REM * Reset grid co-ordinates *
1720 TAG:PLOT 1000,1,2:ORIGIN gx,128:
PRINT optx;:ORIGIN 104,gy:PRINT opty;
:TAGOFF:RETURN
1730 REM
1740 REM * No. of legal moves from ea
ch square *
1750 DATA 2,3,4,4,4,3,2,3,4,6,6,6,4,3
,4,6,8,8,8,6,4,4,6,8,8,8,6,4,4,6,8,8,
8,6,4,3,4,6,6,6,4,3,2,3,4,4,4,3,2
1760 REM
1770 REM * Star positions *
1780 DATA 2,3,5,5,2,18,15,16,14,4,18,
2,21,14,24,10,22,19,27,6,32,13,35,17,
39,2,39,19,12,19,17,8,29,1,4,13,37,9
1790 REM
1800 REM * Legal moves from each squa
re *
1810 DATA 10,16,11,15,17,8,12,16,18,9
,13,17,19,10,14,18,20,11,19,21,12,20
1820 DATA 3,17,23,4,18,22,24,1,5,15,1
9,23,25,2,6,16,20,24,26,3,7,17,21,25,
27,4,18,26,28,5,19,27
1830 DATA 2,10,24,30,1,3,11,25,29,31,
2,4,8,12,22,26,30,32,3,5,9,13,23,27,3
1,33,4,6,10,14,24,28,32,34,5,7,11,25,
33,35,6,12,26,34
1840 DATA 9,17,31,37,8,10,18,32,36,38
,9,11,15,19,29,33,37,39,10,12,16,20,3
0,34,38,40,11,13,17,21,31,35,39,41,12
,14,18,32,40,42,13,19,33,41
1850 DATA 16,24,38,44,15,17,25,39,43,
45,16,18,22,26,36,40,44,46,17,19,23,2
7,37,41,45,47,18,20,24,28,38,42,46,48
,19,21,25,39,47,49,20,26,40,48
1860 DATA 23,31,45,22,24,32,46,23,25,
29,33,43,47,24,26,30,34,44,48,25,27,3
1,35,45,49,26,28,32,46,27,33,47
1870 DATA 30,38,29,31,39,30,32,36,40,
31,33,37,41,32,34,38,42,33,35,39,34,4
0
1880 REM
1890 REM * Computer solution *
1900 DATA 10,19,28,13,4,9,18,27,14,5,
20,7,12,21,6,11,24,15,2,17,8,3,16,25,
34,47,42,33,48,35,26,39,44,29,38,43,3
0,45,36,23,32,41,46,37,22,31,40,49
1910 REM
1920 REM * Music data *
1930 DATA 119,89,71,70,70,60,106,127,
80,70,70,60,159,127,106,70,70,60,119,
80,60,130,130,130
```

Can't hack keying it in?

See Page 38

Defining keys? Then you might need this function key lister

By ROLAND WADDILOVE

YOU'VE probably always taken it for granted that when you press the A key on your Amstrad you get the letter A reflected on your screen.

However, as this article shows, it ain't necessarily so. You can arrange things so that pressing the A will give you B, C or whatever. Confusing as this seems, it's actually quite useful.

The point is that each key in the Amstrad has its own unique number, in the range 0 to 79. This number simply labels the key switch – not the letter that happens to be printed on it.

It has to be like this to allow for the keys that are duplicated, such as the digits 0-9. How else could the Amstrad tell which of the two number 5s had been pressed for instance?

Fine, but how does the Amstrad know what to interpret the key press as? Well, it maintains a table to help it keep track – and when you switch on it defaults to the normal state of affairs, where pressing a key with the A marked on it is interpreted as A.

In actual fact our normal A key has the number 69 associated with it and, provided we haven't done anything clever, the Amstrad will look up 69 and see that it's linked with 65.

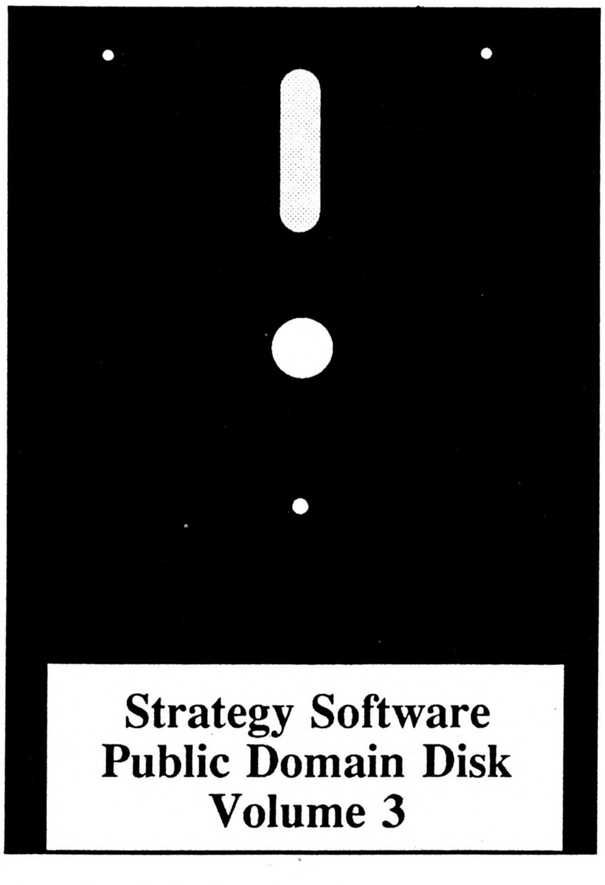
65? Yes, because that's the Ascii for A – remember that the Amstrad likes numbers, not letters. So our table links the key switch numbers with the Ascii code we want it to be interpreted as.

Why not label it as 65 in the first place, you may ask? Well, getting our characters in this way has three advantages:

- *It caters for the operating system.*
- *It allows for duplicate keys.*
- *It lets us redefine keys to give us any value between 0 and 251.*

The code produced by a particular key when pressed is quite easily altered. If for some reason you wanted the A key to produce the letter B when pressed then:

```
KEY DEF 69,1,66
```



Strategy Software CP/M Public Domain Disks available now:

PD1: NSWP & other disk utilities
PD2: Communications software
PD3: Assembler & Disassembler

All $19.95 ea.

Order NOW! before impending price increase

[002] 29 4377 for Bankcard & Mastercard orders

or mail to :

Strategy Software
P.O. Box 11
Blackmans Bay
Tasmania 7152

All software is supplied on 3" disk and is suitable for both PCW and CPC computers.

would tell the Amstrad to produce the Ascii code 66 when key number 69 is pressed, The 1 indicates that the key is to auto repeat if held down. Zero would disable this feature.

Extra codes can be tagged on to the end of the command to indicate which code is to be produced if Shift or Ctrl is held down at the same time.

The codes 128 to 159 have a special significance. The Amstrad stores 32 strings, one for each of these codes. When one of these special codes is produced it is removed from the keyboard buffer and replaced with the whole string. The keys producing these codes are called function keys.

To set up a function key to produce a string we use KEY, for instance:

```
KEY 128,"MODE 1"+CHR$(13)
```

As you can see, any string expression can be used. When the code 128 is encountered it is expanded into the full string.

Any key can be defined to produce the codes using KEY DEF. On power up, or after a reset, the keys on the numeric keypad to the right of the main keyboard produce codes 128 to 140. You'll have to use KEY DEF to redefine some of the other keys on the keyboard to produce the rest of the function key codes.

Although there are 32 strings, the Amstrad only reserves 120 bytes of memory. This will not be enough if all the keys are to be defined, or if long strings are to be used. Fortunately the function key buffer – the memory where the strings are stored – can be any size and anywhere in memory. It can only be relocated from machine code though.

```
LD DE,address
LD HL,length
CALL &BB15
```

will place the buffer at *address* and reserve *length* bytes for the strings.

With 32 function keys it's difficult to remember which have been defined and which haven't and what the definitions are. Accompanying this article is a short machine code utility to list the key definitions.

Run Program I, a Basic listing which pokes in the machine code. Program II is the equivalent assembly listing. Call &A000 to list the definitions.

The routine as it stands will only list keys 128 to 140, the default keys. If you want to list all the definitions then poke &A034 with 160 – the last key number plus 1. As this may produce rather a lot of output you can end at any time by pressing a key.

All the control codes such as carriage return, CHR$(13), are represented by their graphics symbols. This prevents the display from being corrupted.

Function keys are quite handy when typing in listings. As a final note, here's probably the most useful:

```
KEY 129,"CALL &BC02:CALL
&BB4E:MODE 1:LIST"+CHR$(13)
```

It resets the inks, pen and paper, changes to Mode 1 and lists the program in memory, if any.

```
10 REM PROGRAM I
20 REM Function Key Lister
30 MEMORY &9FFF
40 FOR i=0 TO 93
50 READ a$
60 POKE &A000+i,VAL("&"+a$)
70 NEXT
80 DATA CD,09,BB,38,FB,0E,80,CD,39
90 DATA A0,4B,45,59,20,00
91 DATA CD,43,A0,3E,3A,CD,5A
100 DATA BB,06,FF,04,79,68,CD,12,BB
110 DATA 30,07,C5,CD,5D,BB,C1,18,F1
120 DATA CD,09,BB,D8,CD,39,A0,0D,0A
130 DATA 00,0C,3E,8D,B9,20,CF,C9,E1
140 DATA 7E,CD,5A,BB,23,B7,20,F8,E9
150 DATA 41,1E,64,CD,50,A0,1E,0A,CD
160 DATA 50,A0,1E,01,78,16,2F,A7,14
170 DATA 93,30,FC,83,47,7A,C3,5A,BB
```

Program I

```
Pass... 2           ORG &A000

A000:               .keylist
A000:CD 09 BB       CALL &BB09
A003:38 FB          JR C,keylist
A005:0E 80          LD C,128
A007:               .key1
A007:CD 39 A0       CALL string
A00A:               DEF$ 'KEY '
A00E:00             DEFB 0
A00F:CD 43 A0       CALL dec
A012:3E 3A          LD A,58
A014:CD 5A BB       CALL &BB5A
A017:06 FF          LD B,255
A019:               .key2
A019:04             INC B
A01A:79             LD A,C
A01B:68             LD L,B
A01C:CD 12 BB       CALL &BB12
A01F:30 07          JR NC,keydone
A021:C5             PUSH BC
A022:CD 5D BB       CALL &BB5D
A025:C1             POP BC
A026:18 F1          JR key2
A028:               .keydone
A028:CD 09 BB       CALL &BB09
A02B:D8             RET C
A02C:CD 39 A0       CALL string
A02F:0D 0A          DEFW &0A0D
A031:00             DEFB 0
A032:0C             INC C
A033:               .num
A033:3E 8D          LD A,141
A035:B9             CP  C
A036:20 CF          JR NZ,key1
A038:C9             RET
A039:               .string
A039:E1             POP HL
A03A:               .sp1
A03A:7E             LD A,(HL)
A03B:CD 5A BB       CALL &BB5A
A03E:23             INC HL
A03F:B7             OR  A
A040:20 F8          JR NZ,sp1
A042:E9             JP (HL)
A043:               .dec
A043:41             LD B,C
A044:1E 64          LD E,100
A046:CD 50 A0       CALL digit
A049:1E 0A          LD E,10
A04B:CD 50 A0       CALL digit
A04E:1E 01          LD E,1
A050:               .digit
A050:78             LD A,B
A051:16 2F          LD D,&2F
A053:A7             AND A
A054:               .dec1
A054:14             INC D
A055:93             SUB E
A056:30 FC          JR NC,dec1
A058:83             ADD A,E
A059:47             LD B,A
A05A:7A             LD A,D
A05B:C3 5A BB       JP &BB5A
A05E:               END
```

Program II

The ups and downs of sorting out data

ALEATOIRE gives you a lift with understanding quick search algorithms

MANY years ago when computers were mysterious giants that filled up a ballroom and required a dozen people shovelling coal and flicking switches to make them work there were no random access devices. Everything that the machines were fed or regurgitated was sequential, usually on paper or magnetic tape.

Consequently a great deal of time and effort was spent in devising efficient sorting, hence quick searching, algorithms. One of the biggest problems was that computers had very limited memory, or main store, compared with the size of the data records to be sorted, and this gave rise to the elevator puzzle.

Imagine a building with F floors, each holding exactly P people but with the majority of them on the wrong floor. There is a single elevator that can carry, at most, E people and E (the computer memory) is smaller than P, the maximum number of people or records that you want to shift.

The elevator always starts at the ground floor. It can move up and down, loading and dropping people until everyone is where they should be – the elevator then returns to the ground floor.

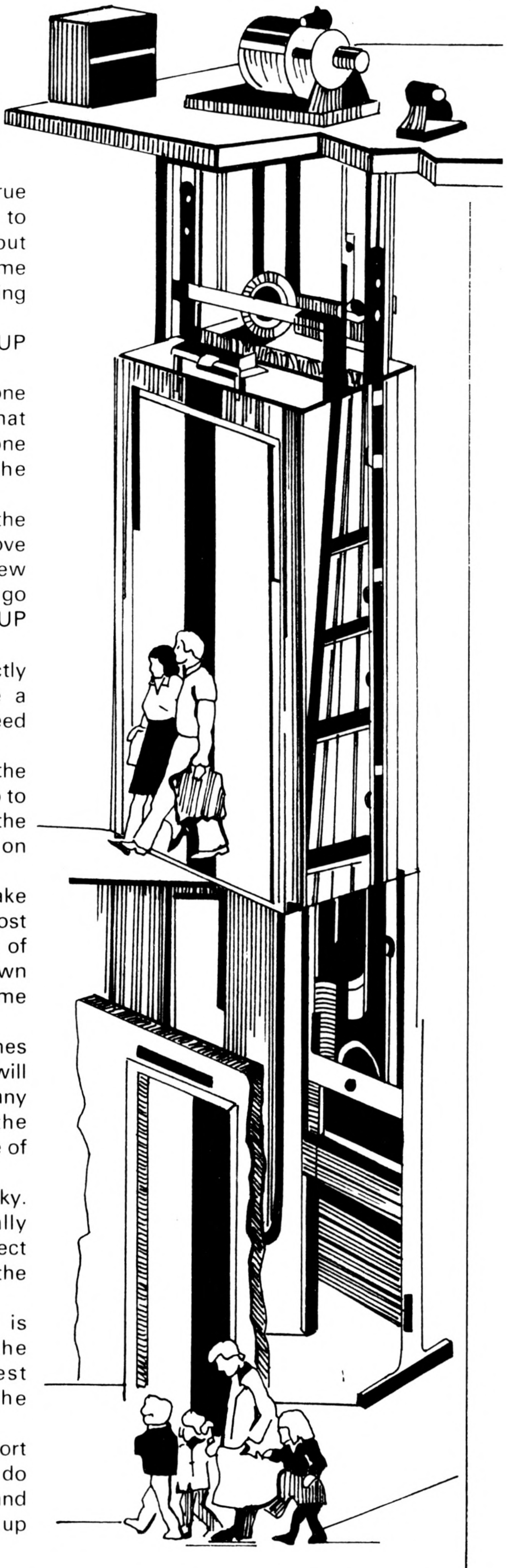
The problem is how to sort all the people in the minimum number of moves, where a move is up or down a floor, that is we want to minimise the distance the elevator has to travel.

Program I will allow you to practice and learn how to solve this classic problem. To make it easier to visualise I have named the seven families (three members each) and the seven floors of the building after the colours of the rainbow with the Reds living at the top and the Violets at the bottom.

To test that it is working remove, or ignore, lines 110-170 and you should

```
10 REM SORT THE RAINBOW
20 MODE 1:DIM r(21),c$(7),i(2)
30 c$(0)="WHITE"
40 FOR i=-7 TO -1
50 READ c$(i+8)
60 FOR j=1 TO 3
70 r(j+i*3+21)=-i
80 NEXT j
90 NEXT i
100 DATA VIOLET,INDIGO,BLUE,GREEN,YEL
LOW,ORANGE,RED
110 FOR i=1 TO 21
120 a=INT(RND*20+1)
130 b=INT(RND*20+1)
140 x=r(a)
150 r(a)=r(b)
160 r(b)=x
170 NEXT i
180 m=0
190 FOR f=1 TO 7
200 u=0
210 FOR i=1 TO f*3
220 IF r(i)>f THEN u=u+1
230 NEXT i
240 m=m+INT((u+1)/2)
250 NEXT f
260 m=m*2
270 r=1
280 i(1)=0
290 i(2)=0
300 PRINT"This is ";c$(INT(r/3+1));"
level. Moves left ";m
310 PRINT"You have    ";c$(i(1));" an
d ";c$(i(2))
320 PRINT"You can see ";c$(r(r));",";
c$(r(r+1));" and ";c$(r(r+2))
330 INPUT"Up,Down or (swop) XY ";a$:a
$=UPPER$(a$)
340 b$=LEFT$(a$,1):PRINT
350 IF (b$="U" AND R<19) OR (b$="D" A
ND R>1) THEN 530 ELSE IF b$="U" OR b$
="D" THEN 330
360 d=0
370 REM drop first colour x
380 REM then pick up y
390 IF b$=LEFT$(c$(i(1)),1) THEN d=1
400 IF b$=LEFT$(c$(i(2)),1) THEN d=2
410 IF d=0 GOTO 330
420 g=0
430 REM now get new colour
440 b$=RIGHT$(a$,1)
450 IF b$=LEFT$(c$(r(r  )),1) THEN g=
1
460 IF b$=LEFT$(c$(r(r+1)),1) THEN g=
2
470 IF b$=LEFT$(c$(r(r+2)),1) THEN g=
3
480 IF g=0 THEN 330
490 x=i(d)
500 i(d)=r(r+g-1)
510 r(r+g-1)=x
520 GOTO 300
530 IF a$="U" THEN r=r+6
540 r=r-3
550 m=m-1
560 IF m>0 THEN 300
570 FOR i=2 TO 7
580 FOR j=1 TO 3
590 IF r(i*3-3+j)<>i THEN 630
600 NEXT j
610 NEXT i
620 IF r=1 THEN PRINT"Well sorted, su
nshine":STOP
630 PRINT"Failed!!"
```

Program I: The Elevator puzzle



find that the families start in the reverse order — that is, Reds at the bottom, Violets at the top – and that you have exactly 40 moves to sort them all out. See Figure I.

7.	Red	V	V	V
6.	Orange	I	I	I
5.	Yellow	B	B	B
4.	Green	G	G	G
3.	Blue	Y	Y	Y
2.	Indigo	O	O	O
1.	Violet	R	R	R

Figure I: The starting pattern of the program test. Sort all the colours back to their correct levels in 40 moves.

The number of moves is the correct and absolute minimum required and is calculated by lines 190-260. The program will check that you do not exceed this limit, but has no idea itself, nor gives any clue, about how to solve the puzzle.

To further test that the program is working you should note that you (playing the elevator) are "empty" at the start (contain two Whites), that you can "see" three Reds and that entering U or D does actually move you up and down the levels correctly.

To swop, hence sort, colours just enter the first letter of a colour you have followed by the first letter of a colour you can see at the current level. For instance, at the start of the test entering WR will swop a White for a RED. Note that a swop does not count as a move. Only U and D are counted.

If all this is working, replace lines 110-170 and the program will mix up the colours randomly, calculate the minimum number of moves to sort them out, obey your commands and check your progress.

If you have never seen this sort of problem before it is very unlikely that you will be able to solve any of the literally thousands of starting patterns it can generate.

You may imagine that you must need to know where all the colours are before you can start. This is true for the program because it has to calculate the number of moves, but you can solve the problem every time without this prior knowledge by using the following, simple algorithm.

You, the elevator, start in the UP state.

1: If in the UP state and anyone needs to go up, then get the two that need to go highest and move up one floor. Otherwise change to the DOWN state.

2: If in the DOWN state, pick the two who need to go lowest and move down one floor. If no one at this new floor, or below this floor, needs to go to a higher floor, change to the UP state.

If you follow these rules exactly and without thinking – just like a computer – you will always succeed in sorting out any pattern.

The things to notice are that at the start you will always go straight up to the 7th level and inevitably drop the two Reds that you will have found on the way.

If you go wrong later on, and take too many moves, you have almost certainly broken the second part of rule 2, that is you have moved down when no one below needed to come higher than the current floor.

With practice the puzzle becomes trivial, but note that the algorithm will work for any number of levels and any number of people, providing the number of people exceeds the size of the elevator.

How it does this is quite spooky. You will notice that occasionally people are taken from their correct level and even made to travel in the wrong direction.

This is because the system is communist, in that it does the greatest good for the greatest number and the rights of the individual are totally ignored.

The elevating moral of this sort story is that rights and privileges do not make the world go round and actually get in the way of it going up or down.

Turn your Amstrad into an electronic typewriter with ROLAND WADDILOVE'S

TEXT EDITOR

HAVE you ever wished you could turn your Amstrad into a typewriter? With its superb editing commands – the electronic equivalent of Tippex – you'd be able to correct your typing errors with ease.

You could of course buy an all singing, all dancing word processor but they cost a fair bit. So we've come up with a text editor. It may not have a lot of frills, but it gets the job done and you don't need a PhD to understand it.

TextEd allows you to enter, print, load, save and manipulate text with a minimum of effort. It is designed to be easy to use, being menu driven with prompts where necessary.

It's also fairly robust and contains a reasonable amount of error checking and mug traps.

Most of the basic functions found in professional word processors have been implemented to increase the power of TextEd.

The only facility excluded is left and right justification – only left hand justification is available. This means that the right hand margin will be uneven. Justification is possible, but you will have to do this yourself.

When the program has been typed in and run a menu will be displayed on a Mode 2 screen.

There are 12 options, each listed with an associated reference character. Option selection is by pressing the key corresponding to this reference character – you don't need to press Enter after the selection.

The 12 options are:

E – Enters the edit mode. This allows the document in memory to be edited. It is also the option you want when starting to write a document. More of this later.

P – Will allow you to print your text. When selected, you will be asked to press the space bar when ready. This allows you to prepare the printer for printing. When space is pressed, the text is printed out exactly as it appears in the edit mode.

L – Load a previously saved text file. If you already have a document in memory you will be prompted with a message asking if you are sure you want to load a new text file.

This is done because the document in memory will be lost as soon as a new file is loaded. Press Y if you wish to load a new file, or N if you want to retain the file in memory – in which case the menu will be displayed again.

If you select Y you will be asked for the new file's name. Enter it, and press Play – making sure you have the correct tape in the tape recorder. Once the file has been loaded the menu will be displayed.

S – Save a text file. This saves the file in memory onto tape. If the file is too short – less than two lines – it cannot be saved, and the message, "not enough text" will appear. Otherwise you will be asked to enter the name of the file.

Now put a blank cassette in the tape deck and wind it to a suitable position. When you are ready to save the file, press a key. The text file will now be saved. The menu will reappear when the saving is complete.

N – New file. This wipes the text file from memory. Since the command is so destructive a safety feature has been added. As with the L command, you must confirm your option. Pressing Y will delete the text file, while N returns you to the menu.

Ø – This option exits from the program. Be careful you don't press Ø if you want the text file in memory. If you do, the text file is lost and the program ends.

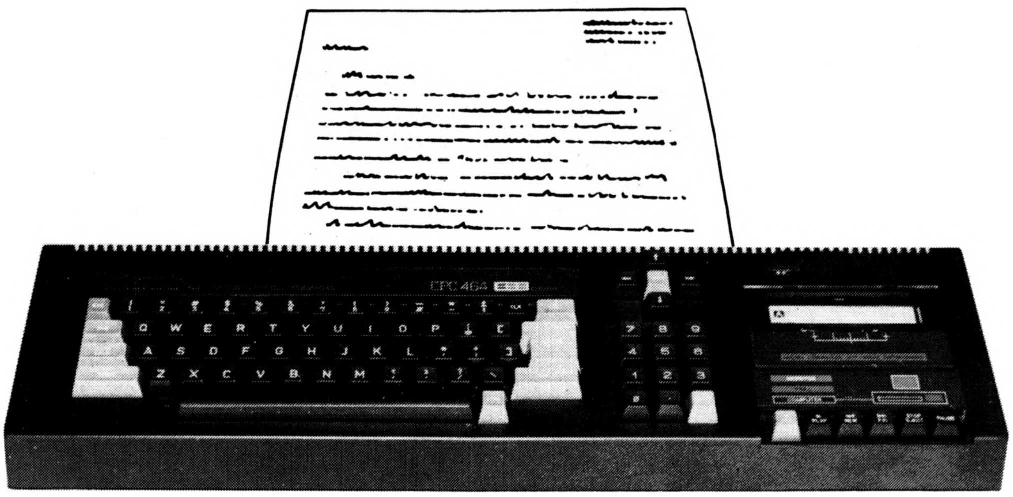
T – Set tab positions. This option allows you to set the four tab positions. You will be asked for the first, second, third and fourth column positions one after another. The tab position must be within the 71 column display – the column width will be less if a left margin has been set.

The tabbing function is invoked by pressing Tab while in the editing mode. This then moves the cursor to the next tab setting you have defined.

M – Set the left margin. This allows

VARIABLES

Option$	Option chosen from menu.
char$	Key pressed.
column%(4)	Tab positions.
start	Start of memory reserved for text.
here	Start of text.
address	Address of cursor.
finish	Address of last character.
last	Last available address for text.
length	Length of line.
insert%	Flag to show mode.
copy	Position of copy cursor.
a$	Start label.
b$	General variable.
i%,j%,k%,i	General variables.
x%	x coordinate of cursor.



you to define the number of characters for the left margin. When the document is viewed each line of text is moved over to the left by the number of characters selected.

This is useful for preparing documents to be punched and filed. Normally the punch holes would destroy part of the printout. This option offers a simple remedy.

R – Re-defines the colours. This allows you to change the colour of the paper, pen and border. If you want to change the pen colour for example, press P. On pressing the key, the colour of the pen is automatically changed.

This colour is the next one available in the palette. The O and B keys change the colour of the paper and border, respectively. Press Enter when you are happy with the colours selected.

C – Send codes to the printer. Some printers allow you to select different character sets, or different printing styles. This option can be used to send the necessary codes to the printer to select them. You will be asked for the character number to be sent to the printer – the Ascii number.

Since most codes consist of several numbers, you will be repeatedly asked for numbers until you enter –1. This is used to terminate the list.

One point to note is that the Amstrad only allows you to send the numbers 0 to 127 to the printer – since only seven bits are used to transfer data to the printer.

H – Help page. This gives information on the keys used in the editing mode – very useful for those with only short term memories.

D – Define a function key. This allows you to program the number keys on the numeric pad – 1 to 9. You will be asked for the key number to be programmed and the string you wish to assign to the key.

This allows the keys to be programmed with words frequently used throughout the text – saving time and wear and tear on the keyboard.

The most important command of the 12 is the Edit command. When this has been selected the editing mode is invoked. Now you can type in your document or edit an existing one. The start of the text is denoted by the message ** start ** – this allows you to keep track of your position in the document.

On the screen you will see a flashing cursor. Moving this, by using the cursor arrows, allows you to position the cursor at the point in the text where you wish to begin editing.

When you have moved to a suitable position in the text you can start typing. You will notice that it is much the same as typing in a program except this time you type in words instead of Basic commands.

The screen acts like a small window. This window can be moved up and down the text by using the cursor up and down keys – allowing you to view and edit any section of the text no matter how large it is.

If the text size were limited to one screen of text, only small documents could be prepared – making the utility useless to people who need large text files.

There are two editing modes, write and insert – toggle selection by pressing CLR. The bottom left corner of the screen shows the current editing mode.

In write mode everything you type is printed at the cursor position. If text is already there it will be overwritten.

Insert mode will create space for the new character entered. This is done by shuffling the existing text down memory, so making space for the new character.

Any word that splits over two lines will look much neater if it is put onto the next line. This can be done by moving the cursor to the start of the split word and pressing the large Enter key – this must be done in the insert mode. The split word will now be dragged onto the next line.

Other significant keys in the editing mode are listed below. This list can also be obtained by selecting option H from the menu:

● The *cursor* can be moved to the next tab position by pressing the TAB key.

● The *large Enter key* moves the cursor to the start of the next line.

● Pressing *Delete* causes the character to the left of the cursor to be deleted – the cursor is moved to the previous position of the deleted character as well.

● The *small Enter key* is used to set a ▶

marker position when copying blocks of text.

Copying text is quite easy. First move the flashing cursor to the start of the text to be copied, and press the small Enter key. Now you should move the cursor to the position you wish to copy the text to.

Pressing Copy will copy the text from the cursor's previous position to the new position in the text – one character at a time. This is very similar to editing a Basic program, except there is no second cursor showing the next character to be copied.

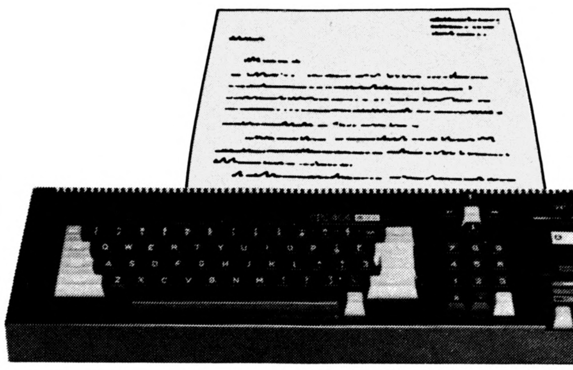
Justification requires extra spaces between words to pad out each line, resulting in a tidy right hand edge. Move the cursor to the position where the pad is required and press the space bar – while in insert mode.

The text will now be shuffled to the left by one character. The pads should be inserted until the required result has been achieved.

The best way to understand how the editing mode works is to try it. It soon becomes second nature.

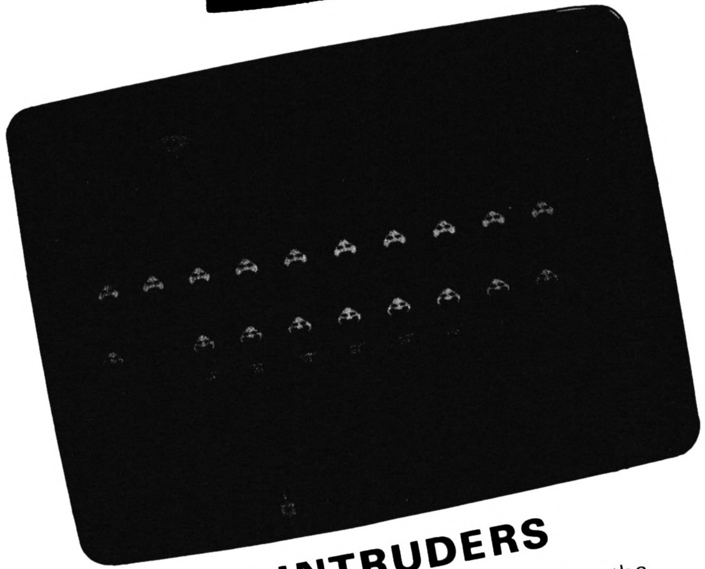
TextEd is an invaluable utility. When used correctly, high quality documents can be prepared with very little effort. And that's what word processors are all about.

```
10 REM ***** TextEd *****
20 REM *By R.A.Waddilove*
30 MEMORY &3FFF
40 MODE 2
50 ON BREAK GOSUB 400 :REM do nothing
60 GOSUB 240 :REM initialise
70 WHILE option$<>"0"
80 GOSUB 1390 :REM menu
90 IF option$="E" THEN GOSUB 420 :REM
 input
100 IF option$="T" THEN GOSUB 1960 :R
EM set tabs
110 IF option$="M" THEN GOSUB 2060 :R
EM set margin
120 IF option$="N" THEN GOSUB 1900 :R
EM new file
130 IF option$="L" THEN GOSUB 1770 :R
EM load
140 IF option$="S" THEN GOSUB 1660 :R
EM save
150 IF option$="R" THEN GOSUB 2150 :R
EM change colours
160 IF option$="H" THEN GOSUB 1530 :R
EM help
170 IF option$="P" THEN GOSUB 2300 :R
EM print
180 IF option$="C" THEN GOSUB 2440 :R
EM printer codes
190 IF option$="D" THEN GOSUB 2560 :R
EM function keys
200 WEND
210 MODE 2
220 END
230 :
240 REM ***** initialise *****
250 SPEED WRITE 1
260 FOR i%=0 TO 120:READ a$:POKE &AB0
0+i%,VAL("&"+a$)::NEXT
270 DATA DD,5E,00,DD,56,01,D5,DD,6E,0
2,DD,66,03,E5,A7,ED,52,44,4D,E1,54,5D
,03,13,ED,B8,CD,06,B9,E1,DD,7E,04,77,
E5,DD,66,08,2E,10,CD,75,BB,E1,DD,46,0
6,7E,E5,C5,CD,D9,BD,C1,E1,23,10,F5,C9
280 DATA DD,5E,00,DD,56,01,D5,DD,6E,0
2,DD,66,03,A7,ED,52,44,4D,03,62,6B,1B
,ED,B0,CD,06,B9,C3,23,AB
290 DATA CD,06,B9,DD,6E,00,26,01,DD,5
E,02,DD,56,03,D5,C3,28,AB
300 DATA 21,00,40,11,01,40,01,FF,6A,3
6,20,ED,B0,C9
310 MOVE 0,75:DRAW 0,398,1:DRAW 639,3
98:DRAW 639,75:DRAW 0,75:DRAW 0,0:DRA
W 639,0:DRAW 639,75
320 DIM column%(4)
330 INK 0,13:INK 1,0:BORDER 10:CALL &
AB6B
340 start=&4000:a$="** START **"
350 FOR i%=1 TO LEN(a$):POKE start+i%
-1,ASC(MID$(a$,i%,1)):NEXT
360 WINDOW #1,2,79,22,24:WINDOW #2,2,
79,2,20:WINDOW 5,75,2,20
370 length=71:here=start+80:finish=he
re:last=43775-length:copy=here
380 column%(1)=10:column%(2)=20:colum
n%(3)=30:column%(4)=40
390 option$="":KEY 139,""
400 RETURN
410 :
420 REM ***** input text *****
430 address=here:copy=here:row%=1:x%=
1:insert%=0
440 LOCATE #1,3,1:PRINT #1,"Mode: wri
te";TAB(35);"Line:";TAB(60);"Bytes fr
ee:":LOCATE #1,3,3:PRINT #1,"CLR chan
ges mode";TAB(33);"column:";TAB(60);"
CTRL for menu":GOSUB 980
450 LOCATE 1,15:PRINT a$
460 CALL &AB59,length*3,0,here,16:CAL
L &AB59,length-1,0,here+length*(row%+
2),19
470 WHILE INKEY(23)=-1:char$=INKEY$
480 LOCATE x%,16:PRINT CHR$(PEEK(addr
ess));
490 IF char$>=CHR$(&F0) THEN GOSUB 98
0
500 IF char$>CHR$(31) AND char$<CHR$(
127) THEN GOSUB 710
510 IF char$=CHR$(13) THEN GOSUB 1130
520 IF char$=CHR$(&7F) THEN GOSUB 790
530 IF char$=CHR$(9) THEN GOSUB 1270
540 IF char$=CHR$(16) THEN GOSUB 1350
550 IF char$=CHR$(&E0) THEN GOSUB 620
560 IF INKEY(6)>-1 THEN copy=address:
SOUND 1,100,5
570 LOCATE x%,16:PRINT CHR$(95);
580 WEND
590 RETURN
600 :
610 REM ***** copy *****
620 IF finish<address THEN finish=add
ress
630 WHILE char$=CHR$(&E0) AND finish<
last
640 char$=CHR$(PEEK(copy)):GOSUB 1210
:char$=INKEY$
650 copy=copy+1-(copy>address AND ins
ert%)
660 WEND
670 GOSUB 980
680 RETURN
690 :
700 REM ***** print *****
710 IF finish<address THEN finish=add
ress
720 WHILE char$>CHR$(31) AND char$<CH
R$(127) AND finish<last
730 GOSUB 1210:char$=INKEY$
740 WEND
750 GOSUB 980
760 RETURN
770 :
780 REM ***** delete *****
790 IF finish<address THEN RETURN
800 WHILE char$=CHR$(&7F) AND address
>here
810 IF insert% THEN CALL &AB3B,x%,len
gth,0,finish,address:finish=finish-1
ELSE POKE address,32:LOCATE x%,16:PRI
NT " ":IF finish=address THEN finish=
finish-1
820 x%=x%-1:address=address-1
830 IF x%=0 THEN x%=length:GOSUB 1090
840 char$=INKEY$
850 WEND
860 GOSUB 980
```



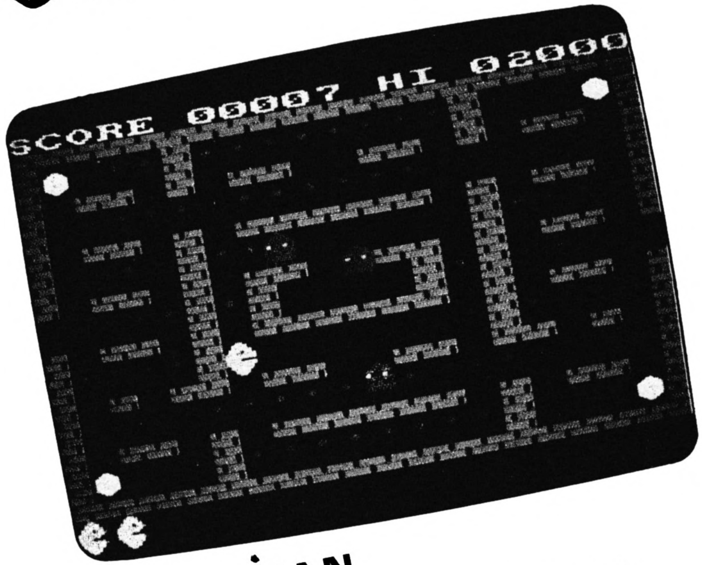
```
870 RETURN
880 :
890 REM ***** move cursor *****
900 IF INKEY(2)>-1 AND address+length
<last THEN address=address+length:GOS
UB 1050
910 IF INKEY(0)>-1 AND address-length
>=here THEN address=address-length:GO
SUB 1090
920 IF INKEY(8)>-1 AND address>here T
HEN x%=x%-1:address=address-1:IF x%=0
 THEN x%=length:GOSUB 1090
930 IF INKEY(1)>-1 AND address+1<last
 THEN x%=x%+1:address=address+1:IF x%
>length THEN x%=1:GOSUB 1050
940 LOCATE #1,40,1:PRINT #1,row%;" "
:LOCATE #1,40,3:PRINT #1,x%
950 RETURN
960 :
970 REM ***** status *****
980 LOCATE #1,40,1:PRINT #1,row%;" "
:LOCATE #1,71,1:PRINT #1,last-finish;
" ":LOCATE #1,40,3:PRINT #1,x%
990 IF NOT insert% THEN RETURN
1000 IF here+length*(row%+2)<last THE
N CALL &AB59,3*length,0,here+length*r
ow%,17:RETURN
1010 IF here+length*(row%+1)<last THE
N CALL &AB59,2*length,0,here+length*r
ow%,17:RETURN
1020 IF here+length*row%<last THEN CA
LL &AB59,length,0,here+length*row%,17
:RETURN
1030 :
1040 REM ***** scroll up *****
1050 row%=row%+1:IF here+length*(row%
+2)>last THEN LOCATE 1,20:PRINT ELSE
CALL &AB59,length,0,here+length*(row%
+2),20
1060 RETURN
1070 :
1080 REM ***** scroll down *****
1090 row%=row%-1:PRINT CHR$(30);CHR$(
11):IF row%>15 THEN CALL &AB59,length
,0,here+length*(row%-16),1 ELSE IF ro
w%=15 THEN PRINT CHR$(30);a$
1100 RETURN
1110 :
1120 REM ***** cr *****
1130 char$=" "
1140 IF finish<address THEN finish=ad
dress
1150 GOSUB 1210
1160 IF x%<>1 AND finish<last THEN 11
50
1170 GOSUB 980
1180 RETURN
1190 :
1200 REM ***** insert *****
1210 IF insert% THEN CALL &AB00,x%,le
ngth,ASC(char$),finish,address:finish
=finish+1 ELSE POKE address,ASC(char$
):LOCATE x%,16:PRINT char$:IF finish=
address THEN finish=finish+1
1220 address=address+1:x%=x%+1
1230 IF x%>length THEN x%=1:GOSUB 105
0
1240 RETURN
1250 :
1260 REM ***** tab *****
1270 char$=" "
1280 IF finish<address THEN finish=ad
dress
1290 GOSUB 1210
1300 IF (x%<>column%(1) AND x%<>colum
n%(2) AND x%<>column%(3) AND x%<>colu
mn%(4)) AND finish<last AND x%<>1 THE
N 1290
1310 GOSUB 980
1320 RETURN
1330 :
1340 REM ***** insert/write toggle **
***
1350 insert%=NOT insert%
1360 LOCATE #1,9,1:IF insert% THEN PR
INT #1,"insert" ELSE PRINT #1,"write
"
1370 RETURN
1380 :
1390 REM ***** menu *****
1400 CLS #1:CLS #2:ZONE 39
1410 LOCATE #2,35,2:PRINT #2,"M E N U
"
1420 LOCATE #2,1,5:PRINT #2,SPC(5)"E.
  Enter and edit text."," T.  Set TAB
 positions.":PRINT #2:PRINT #2,SPC(5)
"P.  Print text."," M.  Set the margi
n."
1430 PRINT #2:PRINT #2,SPC(5)"L.  Loa
d a text file."," R.  Redefine colour
s.":PRINT #2:PRINT #2,SPC(5)"S.  Save
 a text file."," C.  Control codes to
 printer."
1440 PRINT #2:PRINT #2,SPC(5)"N.  New
 file."," H.  Help page.":PRINT #2:PR
INT #2,SPC(5)"0.  End program."," D.
 Define function key."
1450 PRINT #1:PRINT #1,TAB(29);"< Pre
ss a key >"
1460 WHILE INKEY$<>"":WEND:option$=""
1470 WHILE INSTR(" DETPMLRSCNH0",opti
on$)<2
1480 option$=UPPER$(INKEY$)
1490 WEND
1500 CLS #1:CLS #2
1510 RETURN
1520 :
1530 REM ***** help *****
1540 LOCATE #2,35,1:PRINT #2,"H E L P
"
1550 LOCATE #2,1,4:PRINT #2,"CLR.....
..change mode. Write mode overwrites
whatever is already there, insert
       mode will create space by movin
g everything along."
1560 PRINT #2:PRINT #2,"ENTER.....sma
ll key. Note cursor position, when co
py is pressed text is copied
 from here."
1570 PRINT #2:PRINT #2,"ENTER.....lar
ge key. Move to start of next line."
1580 PRINT #2:PRINT #2,"DEL.......del
ete."
1590 PRINT #2:PRINT #2,"TAB.......mov
e to next tab position."
```

4 GREAT GAMES FOR THE PRICE OF ONE!



ALIEN INTRUDERS

With only your laser for protection, destroy the waves of aliens who threaten to engulf you. A quick-fire version of an all-time great!



SNAPMAN

Steer your man around the maze, gobbling up energy pellets while keeping away from the aggressive ghosts who are out to get you.

MAYDAY!
The adventure begins...
Press Space

MAYDAY

Guide the sole survivor of a stricken spaceship through the wreckage of his craft. Recover vital medical supplies . . . or a planet is doomed!

DEADMAN

The time honoured game of suspense – but with some novel endings. Guess the hidden letters or something nasty could happen to you.

Computing with the Amstrad presents

CLASSIC GAMES on the Amstrad

Here's something really special from *Computing with the Amstrad!* We've commissioned four rip-roaring programs that no games collection is complete without – the kind of games that really stand out in the short history of microcomputing.

This value-for-money package includes two top-rate machine code arcade classics plus a traditional word game and a futuristic adventure.

There's hours of enjoyment and something to suit everyone in this superb collection.

Limited quanity only - don't miss out

Use our toll-free order line or order by mail, but hurry - these won't last at this price!

TAPE	$15.95
DISK	$27.95

```
1600 PRINT #2:PRINT #2,"CTRL......ret
urn to menu."
1610 PRINT #2:PRINT #2,"Arrows....mov
e cursor in directions indicated."
1620 PRINT #1:PRINT #1,TAB(29);"< Pre
ss a key >"
1630 WHILE INKEY$="":WEND
1640 RETURN
1650 :
1660 REM ***** save *****
1670 LOCATE 1,2:PRINT "S A V E    T E
 X T    F I L E":PRINT
1680 IF finish-here<length THEN PRINT
:PRINT "Not enough text in the file !
":FOR i%=1 TO 5000:NEXT:RETURN
1690 PRINT:PRINT:LINE INPUT "What is
the name ";name$:PRINT
1700 i=finish-here
1710 SAVE name$,b,here,i
1720 OPENOUT "!"
1730 PRINT #9,length,finish
1740 CLOSEOUT
1750 RETURN
1760 :
1770 REM ***** load *****
1780 LOCATE 1,2:PRINT "L O A D    T E
 X T    F I L E":PRINT
1790 IF finish-here>length THEN LOCAT
E 1,5:PRINT "Loading a new file will
destroy the one at present in the mem
ory.":PRINT:PRINT "Is this ok ? ( Y o
r N ) ":char$="":WHILE char$="":char$
=INKEY$:WEND:IF UPPER$(char$)<>"Y" TH
EN RETURN
1800 CLS #1:CALL &AB6B
1810 PRINT:PRINT:LINE INPUT "What is
the name ";name$:PRINT
1820 OPENOUT name$:CLOSEOUT
1830 LOAD name$,here
1840 OPENIN "!"
1850 INPUT #9,length,finish
1860 CLOSEIN
1870 i%=(80-length)\2:WINDOW i%,i%+le
ngth-1,2,20
1880 RETURN
1890 :
1900 REM ***** new file *****
1910 LOCATE #2,30,3:PRINT #2,"N E W
  F I L E"
1920 LOCATE #2,13,7:PRINT #2,"You wil
l destroy the file at present in the
memory.":LOCATE #2,26,10:PRINT #2,"Is
 this ok ? ( Y or N )":char$="":WHILE
 char$="":char$=INKEY$:WEND:IF UPPER$
(char$)<>"Y" THEN RETURN
1930 finish=here:CALL &AB6B
1940 RETURN
1950 :
1960 REM ***** set tabs *****
1970 LOCATE #2,22,1:PRINT #2,"S E T
  T A B    P O S I T I O N S"
1980 FOR i=1 TO 4
1990 LOCATE #2,30,i*4:PRINT #2,"TAB";
i;".....";SPC(10);STRING$(10,CHR$(8))
;
2000 INPUT #2,b$
2010 IF b$="-" THEN column%(i)=0 ELSE
 column%(i)=VAL(b$)
2020 IF column%(i)<2 OR column%(i)>75
 THEN 1990
2030 NEXT
2040 RETURN
2050 :
2060 REM ***** set margin *****
2070 LOCATE #2,26,2:PRINT #2,"S E T
  M A R G I N"
2080 LOCATE #2,15,10:PRINT #2,"How ma
ny characters wide is the margin ";SP
C(10);STRING$(10,CHR$(8));
2090 INPUT #2,b$
2100 IF b$="-" THEN i%=0 ELSE i%=VAL(
b$)
2110 IF i%<2 OR i%>30 THEN 2080
2120 WINDOW i%,79-i%,2,20:length=80-2
*i%
2130 RETURN
2140 :
2150 REM ***** change colours *****
2160 i%=0:j%=0:k%=0
2170 LOCATE #2,25,2:PRINT #2,"R E D E
 F I N E    C O L O U R S"
2180 WHILE INKEY(18)=-1
2190 LOCATE #2,25,8:PRINT #2,"P.....c
hange pen colour"
2200 LOCATE #2,25,10:PRINT #2,"O.....
change paper colour"
2210 LOCATE #2,25,12:PRINT #2,"B.....
change border colour"
2220 LOCATE #1,25,2:PRINT #1,"Press
ENTER  when finished."
2230 IF INKEY(27)>-1 THEN i%=(i%+1)MO
D 27:INK 1,i%
2240 IF INKEY(34)>-1 THEN j%=(j%+1)MO
D 27:INK 0,j%
2250 IF INKEY(54)>-1 THEN k%=(k%+1)MO
D 27:BORDER k%
2260 WEND
2270 WHILE INKEY$<>"":WEND
2280 RETURN
2290 :
2300 REM ***** print *****
2310 LOCATE #1,25,2:PRINT #1,"Press t
he  SPACE BAR  to print."
2320 WHILE INKEY$<>" ":WEND:CLS
2330 LOCATE #1,25,2:PRINT #1,"      P
R I N T I N G          "
2340 k%=(81-length)\2
2350 FOR i=here TO finish STEP length
2360 FOR j%=1 TO k%:PRINT #8," ";:NEX
T
2370 FOR j%=0 TO length-1
2380 PRINT CHR$(PEEK(i+j%));:PRINT #8
,CHR$(PEEK(i+j%));
2390 NEXT
2400 PRINT #8
2410 NEXT
2420 RETURN
2430 :
2440 REM ***** send codes to printer
*****
2450 LOCATE #2,20,2:PRINT #2,"P R I N
 T E R    C O N T R O L    C O D E S"
2460 LOCATE #2,21,8:PRINT #2,"Type in
 the ASCII codes one at a time....."
2470 LOCATE #2,25,14:PRINT #2,"Type
-1  when you have finished."
2480 i%=0
2490 WHILE i%>-1
2500 PRINT #1,SPC(20):INPUT #1,;"Code
 ";b$
2510 IF b$="-" THEN i%=1000 ELSE i%=V
AL(b$)
2520 IF i%>-1 AND i%<128 THEN PRINT #
8,CHR$(i%);:PRINT #1,"  character sen
t" ELSE PRINT #1,"  <-- ignored"
2530 WEND
2540 RETURN
2550 :
2560 REM ***** define function keys *
****
2570 LOCATE #2,20,2:PRINT #2,"D E F I
 N E   A   F U N C T I O N   K E Y"
2580 LOCATE #2,25,8:PRINT #2,"Which f
unction key ";SPC(10);STRING$(10,CHR$
(8));
2590 INPUT #2,b$
2600 IF b$<"1" OR b$>"9" THEN 2580 EL
SE i%=128+VAL(b$)
2610 LOCATE #2,25,12:PRINT #2,"What i
s the string...";:LINE INPUT #2,b$
2620 KEY i%,b$
2630 RETURN
```

Can't hack keying it in?

See Page 38

ROLAND WADDILOVE'S Variable Dump takes the slog out of error checking

Check your variables to spot your mistakes

VARIABLE Dump is a short machine code routine to add an extra command |VARDUMP as an RSX. It simply lists all the variables that have been used by a Basic program.

Why would you want to list all the variables? Well, suppose you typed in a listing and it didn't behave as expected when run. It's most likely to be a simple typing slip.

One of the features of Locomotive Basic is that it will create an additional variable every time it comes across a word it doesn't recognise. So if the program uses a variable called *score* and you accidentally typed *scare* instead it would add *scare* to its list of variables. By printing all the variables it's easy to spot any that have been entered incorrectly.

The program must be run before using |VARDUMP. After each variable name it waits for a key to be pressed. This is to prevent long lists of variables scrolling off the top of the screen.

To understand how the routine works we need to know how the Amstrad stores its variables. These are stacked up in the memory starting at the end of the program. Arrays are always placed at the end of the variable stack. All other variables are inserted before these.

I'll call the address of the end of the program currently in memory *TOP*. The value of *TOP* is stored at &AE85 if you have Basic 1.0, and at &AE68 if you have Basic 1.1. The CPC464 has Basic 1.0 and the CPC6128 has Basic 1.1.

The lowest point in the memory which is free I'll call *LOMEM*. This is the top of the variable stack. Basic 1.0 stores *LOMEM* at &AE89 and Basic 1.1 stores it at &AE6C.

The end of the normal variables and the start of the arrays is stored at &AE87, Basic 1.0 and &AE6A, Basic 1.1.

Variable Dump prints the ordinary variable names first, if there are any. These are stored between *TOP* and the start of arrays. Of course if these two values are the same then there aren't any.

When Basic stores the variable it ANDs the characters of the name with &DF to convert any letters to upper case. This has the unfortunate effect of converting any numbers in the name to Ascii values below 31 – reserved for control codes. The last letter of the name is marked by setting bit 7 of the Ascii code.

To print out a variable's name the characters are ANDed with &7F, then ORed with &20. This restores the name.

The byte in memory immediately following the variable's name indicates what type it is. An integer is 1, string 2 and real 4. A % sign is printed after integers' names and a $ sign after strings.

The address of the next variable name is found by adding the space taken up by the variable to the address of the type byte. Integers take up 5 bytes, reals 8 and strings 6.

Remember that the actual string is placed at HIMEM. The data in the variable stack is the string descriptor giving the name, length and address of the string.

A separate routine is required to handle arrays. The name and type is printed out in the same way, followed by a pair of brackets to show that it is an array.

The actual space taken up by the array depends on the number of dimensions and elements there are. The two bytes following the type byte tell us how much space has been reserved. The next variable is stored at the address of these two bytes plus their contents plus 4.

If the address of the next variable is equal to *LOMEM* all the variables have been listed.

Program I is a Basic listing with the machine code stored in data statements. Run this and call &A000 to enable the RSX. Program II is an assembly listing of the routine.

You should find debugging a lot easier using Variable Dump to list all the variables. Now all we need is a routine to tell us which line the misspelt variable is actually in.

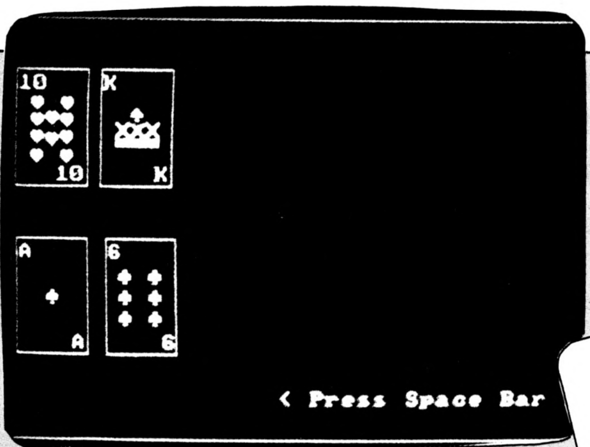
Can anyone help?

Program I

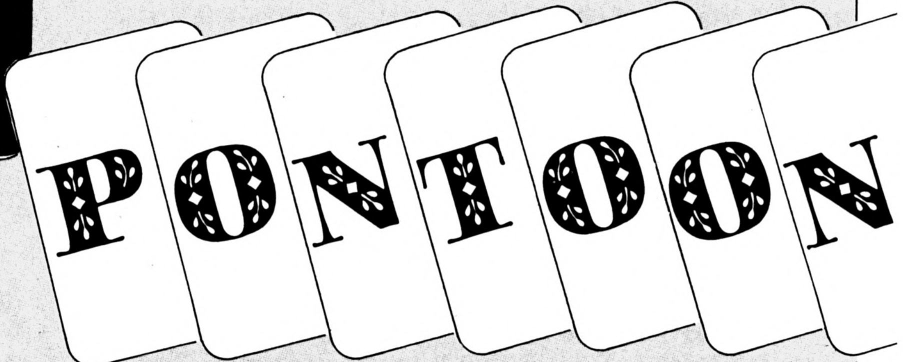
```
10 REM Variable Dump
20 REM By R.A.Waddilove
30 REM(c)Computing With The Amstrad
40 REM CALL &A000 to enable |VARDUMP
50 REM Hold down key to see variables
60 MEMORY &9FFF:address=&A000
70 FOR i=1 TO 19
80 sum=0:READ code$,check$
90 FOR j=1 TO 21 STEP 2
100 byte=VAL("&"+MID$(code$,j,2))
110 POKE address,byte
120 sum=sum+byte:address=address+1
130 NEXT
140 IF sum<>VAL("&"+check$) THEN PRIN
T "Error in data in line ";150+i*10
150 NEXT
160 DATA 214BA0CB4EC0CBCE0146A0,565
170 DATA 2142A0CDD1BCCD00B9F53A,612
180 DATA 02C0A72812216AAE224DA0,3EB
190 DATA 2168AE2252A0216CAE2292,43A
200 DATA A0F1CD0CB9CDC5A04F4B0D,5FC
210 DATA 0A00C956415244554DD000,372
220 DATA 000000003AA0C34CA0002A,2B3
230 DATA 87AEE5ED5B85AEA7ED52E1,75C
240 DATA CA8FA0EB2323CD9BA03E0D,57D
250 DATA CD5ABB3E0ACD5ABBCD06BB,59A
260 DATA 09E5ED52E138E91818CD9B,5C7
270 DATA A0E5CDC5A028290D0A00E1,500
280 DATA CD06BB235E235623232319,30A
290 DATA E5ED5B89AEA7ED52E138DD,740
300 DATA C97EF620E67FCD5ABBCB7E,6ED
310 DATA 2328F37EFE0120083E2501,347
320 DATA 0500C35ABBFE0220083E24,367
330 DATA 010600C35ABB010800C9E1,392
340 DATA 7ECD5ABB23B720F8E90000,53B
```

Program II

```
ORG &A000

;***** initialise RSX's *****
LD HL,flags
BIT 1,(HL)
RET NZ              ;already done?
SET 1,(HL)          ;set flag
LD BC,jump_table
LD HL,workspace
CALL &BCD1          ;log new commands
CALL &B900
PUSH AF             ;enable upper ROM
LD A,(&C002)
AND A
JR Z,init1          ;CPC6128?
LD HL,&AE6A
LD (array),HL       ;change variables
LD HL,&AE68
LD (TOP),HL
LD HL,&AE6C
LD (LOMEM),HL
.init1
POP AF
CALL &B90C          ;restore ROM state
CALL string
DEFB "OK",13,10,0
RET

;******* Name table *********
.name_table
DEFB "VARDUM","P"+&80
DEFB 0
.workspace
DEFW 0
DEFW 0

;****** Jump Table **********
.jump_table
DEFW name_table
JP vardump

.flags DEFB 0

;**** List all variables ****
.vardump
LET array=$+1
LD HL,(&AE87)
PUSH HL             ;get stack top
LET TOP=$+2
LD DE,(&AE85)       ;get array pointer
AND A
SBC HL,DE
POP HL
JP Z,vdump1         ;any variables?
EX DE,HL
INC HL
INC HL              ;HL=first variable
.var1               ;normal variables
CALL vdname         ;print name
LD A,13             ;new line

CALL &BB5A
LD A,10
CALL &BB5A
CALL &BB06          ;wait for key press
ADD HL,BC           ;next variable
PUSH HL
SBC HL,DE
POP HL              ;finished?
JR C,var1
JR vdump1           ;arrays next
.vda1
CALL vdname         ;print name
PUSH HL
CALL string         ;brackets+new line
DEFB "()",13,10,0
POP HL
CALL &BB06          ;wait for key press
INC HL
LD E,(HL)
INC HL
LD D,(HL)           ;jump array space
INC HL
INC HL
INC HL
ADD HL,DE           ;next name
.vdump1
PUSH HL
LET LOMEM=$+2
LD DE,(&AE89)
AND A
SBC HL,DE           ;finished?
POP HL
JR C,vda1
RET

.vdname
LD A,(HL)
OR 32
AND &7F
CALL &BB5A          ;print name
BIT 7,(HL)          ;last letter?
INC HL
JR Z,vdname
LD A,(HL)           ;get type
CP 1
JR NZ,var2          ;integer?
LD A,"%"
LD BC,5
JP &BB5A
.var2
CP 2
JR NZ,var3          ;string?
LD A,"$"
LD BC,6
JP &BB5A
.var3
LD BC,8             ;must be real
RET

;***** Print string *********
.string
POP HL              ;get address
.sp1
LD A,(HL)           ;get char
CALL &BB5A
INC HL              ;next
OR A                ;last one?
JR NZ,sp1
JP (HL)

END
```



Bet you'll lose your shirt in SURJIT RANDHAWA's



PONTOON is a simulation of the well known card game, the object being to hold cards of a higher face count than the bank – in this case the computer.

The computer deals both you and itself two cards. You can see your cards but one of the computer's cards remains concealed.

After placing a bet on the opening strength of your hand, you can increase its total face value by "twisting" further cards until you think you have a total to beat the bank's.

You must do this without exceeding a total of 21 or else you "bust" and lose. When adding up the cards values, 10s to king count 10, ace counts 1 or 11, and all other cards count their face value.

The bank will continue to twist cards to you until you decide to "stick" – when you consider your count is high enough. Once you have chosen to stick the bank will take cards until it either beats your total or busts itself.

You start the game with £10 and must bet a minimum of one pound each time you play a hand.

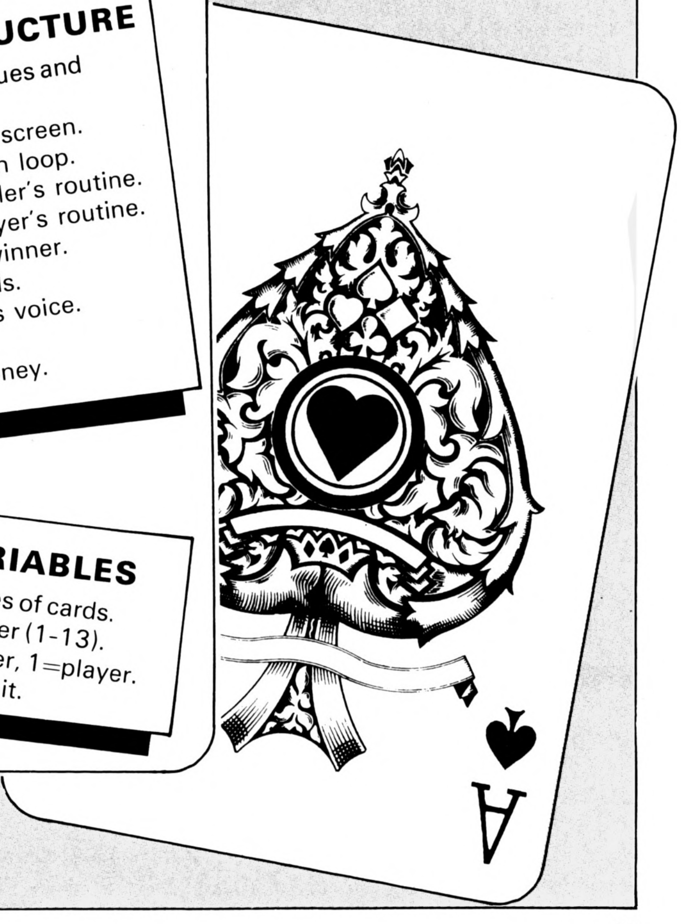
A self-destruct option at the start of the game adds a little extra spice, because should you be brave enough to select it and run out of money not only have you lost your shirt, you've also lost your program and must load it again.

PROGRAM STRUCTURE

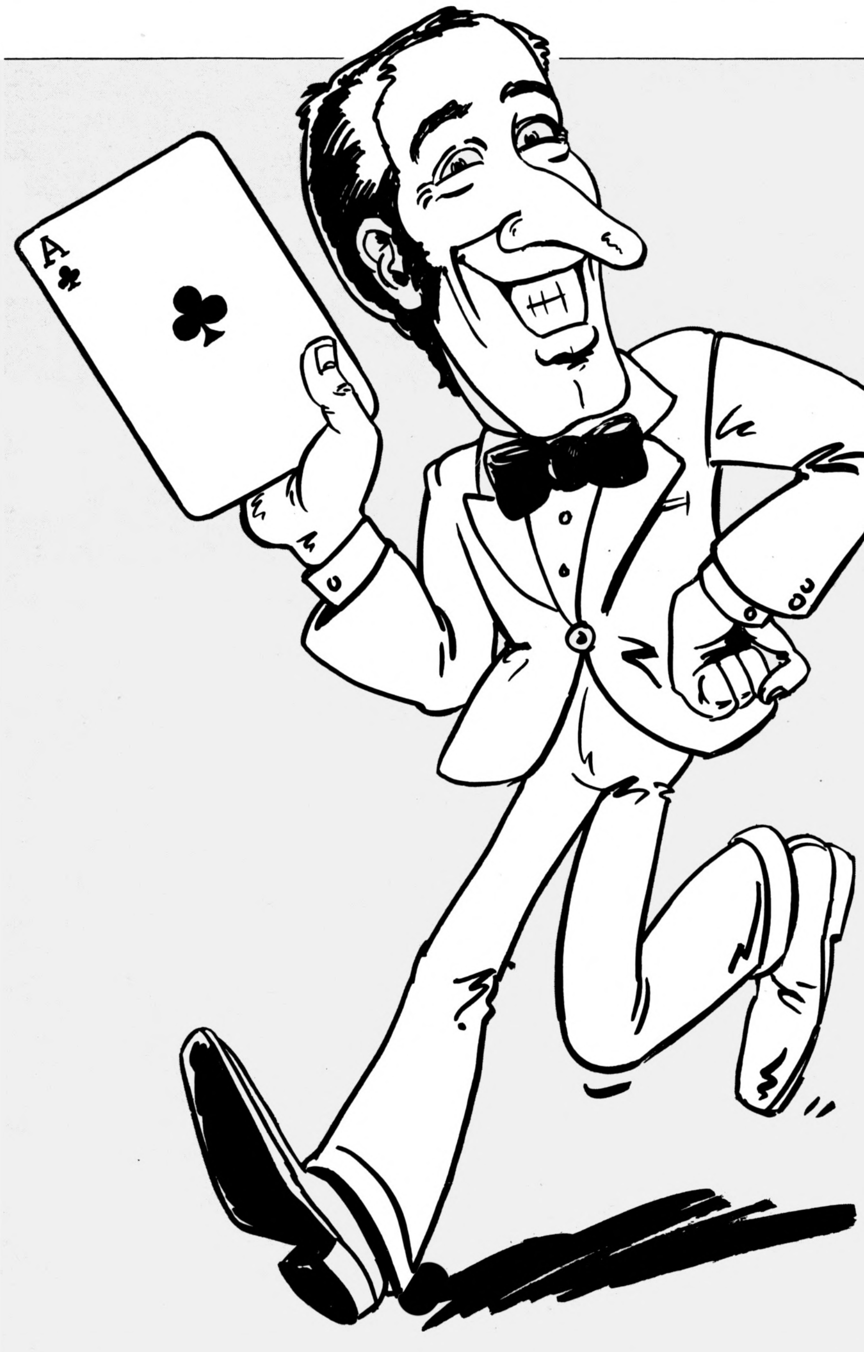
10-60	Read card values and set amount.
70-135	Introductory screen.
140	Start of main loop.
220	Start of dealer's routine.
260	Start of player's routine.
470	Sorts out winner.
570	Draws cards.
870	Computer's voice.
940	Pontoon!
1030	Out of money.

MAJOR VARIABLES

x,y	Coordinates of cards.
c	Card number (1-13).
player	0=computer, 1=player.
s	Indicates suit.



```
5 REM(c)Computing with the Amstrad
10 DIM  card$(13) , pack(4,13)
20 MODE 1 : BORDER 0 : INK 0,0 : INK
1,26 : INK 2,18 : INK 3,22
30 REM read in card values
40 FOR n=1 TO 13 : READ card$(n) : NE
XT
50 DATA A,2,3,4,5,6,7,8,9,10,J,Q,K
60 money=10  : REM thats all you get
!
70 REM ******* start screen *******
80 PEN 2 : LOCATE 15,1 : a$="* PONTOO
N *" : GOSUB 870 : GOSUB 920
85 LOCATE 1,15 : PRINT "Would you lik
e some instructions ?"
86 a$=INKEY$ : IF a$="" THEN 86 ELSE
IF LEFT$(UPPER$(a$),1)<>"Y" THEN 120
90 CLS : PRINT : PRINT"The object  of
  the  game  is  to  add  together th
e face value of the cards    that are
 dealt to you and then compare  them
with the dealers total. "
95 PRINT "If your total is greater th
an the         dealers total then you w
in,but your      total must not exceed
 21 or you're BUST!"
100 PRINT "If you get 21 then you get
 a bonus." : PRINT"The keys to use ar
e :" : PRINT :PRINT"   S ... Stick if
 you are satisfied with          your
total" : PRINT "   T ... Twist if you
 want another card"
110 PRINT:PRINT"To make the game more
 interesting you   have the option of
 auto-destruct .       If you run out
of money the computer    resets!"
114 LOCATE 11,23:PRINT"< Press any ke
y >"
115 WHILE INKEY$="":WEND
120 CLS:LOCATE 7,10 : a$="Auto-destru
ct  ON/OFF " : GOSUB 870
130 INPUT autodestruct$ : autodestruc
t$=UPPER$(autodestruct$)
135 a$="Minimum bet is " : LOCATE 12,
18 : GOSUB 870 : PRINT CHR$(163);" 1"
 : GOSUB 920 : GOSUB 920
140 REM ******* main loop *******
150 playertotal=0 : dealertotal=0 : d
ealeraces=0 : playeraces=0
160 CLS : PEN 1 : INK 0,4 : BORDER 4
: GOSUB 920 :a$=" You have ":LOCATE 1
,10:GOSUB 870:PRINT CHR$(163);" ";mon
ey
170 GOSUB 920 : a$=" How much do you
want to bet " :LOCATE 1,14: GOSUB 870

180 INPUT bet : IF bet < 1 OR bet > m
oney THEN LOCATE 2,18 : a$="What ? Tr
y again.  Bet " : GOSUB 870 : LOCATE
25,18 : GOTO 180
190 CLS : INK 0,9 : BORDER 0 : LOCATE
 1,23 : PEN 2 : PRINT"  Bet";" ";CHR$
(163);bet
200 money=money-bet : LOCATE 31,2 : P
RINT CHR$(163);money
205 LOCATE 2,2 : PRINT "DEALER" : LOC
ATE 2,12 : PRINT "PLAYER"
210 MOVE 0,0 : DRAW 639,0 : DRAW 639,
399 : DRAW 0,399 : DRAW 0,0
220 player=0 : start=1 : x=15 : y=250
 : GOSUB 570 : REM dealers first card
230 x=105 : GOSUB 570     : REM deale
rs second card
240 player=1 : x=15 : y=90 : GOSUB 57
0 : x=110 : GOSUB 570
250 TAGOFF : LOCATE 20,23 : PEN 3 : P
RINT "<S>tick / <T>wist"
260 WHILE playertotal<22
270  reply$=UPPER$(INKEY$) : IF reply
$="" THEN 270
280  IF reply$<>"T" THEN  310
290  SOUND 1,100,5 : SOUND 2,110,10 :
 x=x+90 : GOSUB 570 : REM another car
d
300 WEND
310 WHILE playeraces<>0
320  IF playertotal+10>21 THEN player
aces=playeraces-1
330  IF playertotal+10<22 THEN player
aces=playeraces-1 : playertotal=playe
rtotal+10
340 WEND
345 LOCATE 8,12 : PEN 2 : PRINT playe
rtotal
350 IF playertotal>21 THEN LOCATE 17,
12 : PRINT "B U S T !     " : SOUND 1
,3000,40 : GOTO 510
360 IF playertotal=21 THEN  GOSUB 940
 : REM pontoon !
370 player=0 : x=15: y=250 : c=startc
ard : s=startsuite
380 TAG : GOSUB 700 : x=105 : REM dra
w dealers first card
390 WHILE dealertotal < playertotal
400  IF dealeraces=0 OR dealertotal=2
 THEN 430
410  IF dealertotal+10 > 21 THEN deal
eraces=dealeraces-1 : pace=pace+1
420  IF dealertotal+10 < 22 THEN deal
ertotal=dealertotal+10 : pace=pace+1
 : dealeraces=dealeraces-1 : GOTO 450

430 GOSUB 920 : REM slow things down
a bit !
440  SOUND 1,90,5 : SOUND 2,70,10 : x
=x+90 : GOSUB 570 : REM another card
450 WEND
460 IF pace<>0 AND dealertotal >21 AN
D dealeraces<>0 THEN dealertotal=deal
ertotal-10 : pace=pace-1 : GOTO 390
465 LOCATE 8,2 : PEN 2 : PRINT dealer
total
470 REM ******* who has won ? *******
480 IF dealertotal > playertotal AND
dealertotal < 22 THEN LOCATE 17,12 :
PRINT"I WIN !!!    "
490 IF dealertotal < playertotal OR d
ealertotal >21 THEN LOCATE 12,12 : PE
N 3 : PRINT "YOU HAVE WON ";CHR$(163)
;bet*2 : money=money+bet*2
500 IF dealertotal = playertotal THEN
 LOCATE 15,12 : PRINT"* D R A W *   "
 : money=money+bet
510 IF money < 1 THEN GOTO 1030
520 FOR suite=0 TO 3 : FOR c=1 TO 13
530  pack(suite,c)=0          : REM new
 pack of cards
540 NEXT c : NEXT suite
550 LOCATE 18,23 : PEN 1 : PRINT " <
Press Space Bar >"
560 IF INKEY$="" THEN 560 ELSE CLS :
GOTO 140
570 REM ******* cards *******
580 PEN 1 : MOVE x,y-10 : DRAW x,y+10
0 : DRAW x+75,y+100 : DRAW x+75,y-10
: DRAW x,y-10 : TAG : MOVE x,y+90
590 suite=INT(RND*4) : c=INT(RND*13)+
1 : REM select random card
600 IF suite=0 THEN s=226 : REM clubs
610 IF suite=1 THEN s=227 : REM diamo
nds
620 IF suite=2 THEN s=228 : REM heart
s
630 IF suite=3 THEN s=229 : REM spade
s
640 IF pack(suite,c)=1 THEN 590 : REM
 card has been dealt
650 IF pack(suite,c)=0 THEN pack(suit
e,c)=1
660 IF c>10 THEN cardvalue=10 ELSE ca
rdvalue=c
670 IF player=1 THEN playertotal=play
ertotal+cardvalue : IF c=1 THEN playe
races=playeraces+1
680 IF player=0 THEN dealertotal=deal
ertotal+cardvalue : IF c=1 THEN deale
races=dealeraces+1
690 IF start=1 THEN startsuite=s : st
artcard=c : start=0 : RETURN
700 MOVE x+2,y+95 : PRINT card$(c);
710 IF c=10 THEN MOVE x+42,y+8 : PRIN
T card$(c);: ELSE MOVE x+58,y+8 : PRI
NT card$(c);
720 IF c=1 THEN MOVE x+32,y+50 : PRIN
T CHR$(s);: REM ace
730 IF c=2 THEN MOVE x+30,y+40 : PRIN
T CHR$(s);: MOVE x+30,y+60 : PRINT CH
```



```
R$(s);
740 IF c=3 THEN MOVE x+30,y+70 : PRIN
T CHR$(s);: MOVE x+30,y+50 : PRINT CH
R$(s);: MOVE x+30,y+30 : PRINT CHR$(s
);
750 IF c=4 THEN MOVE x+15,y+70 : GOSU
B 860 : MOVE x+15,y+30 : GOSUB 860
760 IF c=5 THEN MOVE x+15,y+70 : GOSU
B 860 : MOVE x+15,y+30 : GOSUB 860 :
MOVE x+32,y+50 : PRINT CHR$(s);
770 IF c=6 THEN MOVE x+15,y+70 : GOSU
B 860 : MOVE x+15,y+50 : GOSUB 860 :
MOVE x+15,y+30 : GOSUB 860
780 IF c=7 THEN MOVE x+15,y+70 : GOSU
B 860 : MOVE x+15,y+50 : GOSUB 860 :
MOVE x+15,y+30 : GOSUB 860 : MOVE x+3
2,y+50 : PRINT CHR$(s);
790 IF c=8 THEN MOVE x+15,y+70 : GOSU
B 860 : MOVE x+15,y+50 : GOSUB 860 :
MOVE x+15,y+32 : GOSUB 860 : MOVE x+3
2,y+40 : PRINT CHR$(s);: MOVE x+32,y+
60 : PRINT CHR$(s);
800 IF c=9 THEN MOVE x+15,y+75 : GOSU
B 860 : MOVE x+15,y+25 : GOSUB 860 :
MOVE x+15,y+58 : GOSUB 860 : MOVE x+1
5,y+42 : GOSUB 860 : MOVE x+32,y+50 :
 PRINT CHR$(s);
810 IF c=10 THEN MOVE x+15,y+75 : GOS
UB 860 : MOVE x+15,y+25 : GOSUB 860 :
 MOVE x+15,y+58 : GOSUB 860 : MOVE x+
15,y+42 : GOSUB 860 : MOVE x+32,y+40
: PRINT CHR$(s);: MOVE x+32,y+60 : PR
INT CHR$(s);
820 IF c=11 THEN MOVE x+15,y+45 : PRI
NT CHR$(141);CHR$(140);CHR$(142);: MO
VE x+20,y+45 : DRAW x+40,y+70 : DRAW
x+59,y+45 : MOVE x+32,y+55 : PRINT CH
R$(s);
830 IF c=12 THEN MOVE x+15,y+37 : PRI
NT CHR$(131);CHR$(131);CHR$(131);: MO
VE x+15,y+55 : PRINT CHR$(197);CHR$(2
38);CHR$(199);: MOVE x+32,y+70 : PRIN
T CHR$(s);
840 IF c=13 THEN MOVE x+15,y+35 : PRI
NT CHR$(131);CHR$(131);CHR$(131);: MO
VE x+15,y+50 : PRINT CHR$(203);CHR$(2
03);CHR$(203);: MOVE x+32,y+65 : PRIN
T CHR$(s);
850 TAGOFF : RETURN
860 PRINT CHR$(s);" ";CHR$(s);: RETUR
N
870 REM ***** computer voice ! *****
880 FOR t=1 TO LEN(a$) : PRINT MID$(a
$,t,1);
890  FOR delay=0 TO 30 : NEXT delay
900 SOUND 1,20+INT(RND*10),3
910 NEXT t : RETURN
920 REM ******* delay *******
930 FOR delay=0 TO 700 : NEXT delay :
 RETURN
940 REM ******* pontoon ! *******
950 LOCATE 13,12 : PEN 3 : PRINT "PON
TOON !!  BONUS ";CHR$(163);bet
960 money=money+bet
970 FOR loop=1 TO 3
980  FOR note=70 TO 30 STEP -10 : SOU
ND 1,note : SOUND 2,note+loop*5,7
990  NEXT note : SOUND 1,70
1000 NEXT loop
1010 FOR delay=0 TO 3000 : NEXT delay
1020 LOCATE 11,12 : PRINT SPC(25) : G
OSUB 920 : RETURN
1030 REM ***** run out of money *****
1040 FOR t=1 TO 3 : GOSUB 920 : NEXT
1050 CLS : LOCATE 5,12 : PRINT"You ha
ve run out of money !!"
1060 LOCATE 1,23 : GOSUB 920
1070 IF autodestruct$="ON" THEN CALL
0 : REM goodbye !!
1080 LOCATE 5,18 : PRINT"Another game
? Y/N";:INPUT answer$
1085 answer$=UPPER$(answer$)
1090 WHILE answer$<>"Y" AND answer$<>
 "N":WEND
1100 IF answer$="N" THEN END
1110 money = 10:INK 0,0: INK 2,18: GO
TO 120
```

Can't hack keying it in?

See Page 38

HERE is a simple program that will help you to keep track of your biorhythms.

As you probably know your life is ruled by your intellectual, physical and emotional states, and according to biorhythm theory they follow a cyclic pattern from the day you were born.

The intellectual cycle repeats itself every 33 days while the periods of the other two are shorter – 28 for the emotional and only 23 for the physical cycle.

When you run the program it will ask you for your name, date of birth and a month for which you want your biorhythms.

What you see on the screen is a calendar displayed in perspective with a tower on each of the days.

The tower is split into three parts. The bottom section (blue) tells you your intellectual status. The next one up (green) is your physical condition and at the top in brown you will see your emotional status. Thin is bad, fat is good.

If you are wise you will use the biorhythm calendar to plan all your activities. When you've got to do something strenuous such as watering the plants or running a marathon then try to find a day where your physical cycle is at its highest level. Otherwise you will find yourself working below your peak.

Playing chess as well as other mental activities like watching Mastermind or reading this magazine make heavy demands on your brain. So look up your intellectual level and see that it is up to the task.

As far as sex is concerned I'm not sure whether this comes under physical or emotional. Better play safe and check them both!

Occasionally you will see the towers on your calendar shrink down to just a red square.

These are the days when all your levels are at rock bottom. Anything you do will probably end in disaster. Stay in bed and do nothing.

The program is liberally sprinkled with REMs and the structured design should be easy to follow.

The main loop starting at line 100 calls the data entry routine to ask for your name, birth date and month to display. Then comes a call to the calendar routine.

When you have finished looking at the screen you can ask for the next month (N), the previous month (P) or go back to the menu (M).

Dates are entered as 28nov44 or 1 DEC 1850 with or without spaces. They are decoded by lines 910 to 1030. *m* is set to 0 if the three lette. month name is not recognised. Years less than 100 have 1900 added.

The INSTR function in line 130 takes the key pressed converted to upper case and compares it with Space, N, P and M to produce a number 1, 2, 3, 4 or 0 if no match. The result, plus one, is used by the ON GOTO instruction.

PRINT CHR$(23);CHR$(0) in lines 400 and 1310 sets the graphics mode to opaque while CHR$(23); CHR$(3) at line 1230 sets it to OR to avoid overwriting the grid.

However CHR$(23) in line 1190 is just a trick to suppress an unwanted space.

Text printed between lines 1230 and 1310 is tagged to the graphics cursor.

VARIABLES

The DEFINT a-y instruction at line 310 makes all number variables integers except those that begin with z.

bd,bm,by,zb	Date of birth: day, month, year and number of days since 31 Dec 1BC.
dm,dy	Month to display.
name$	Your name.
mth$	An array to hold month names.
wkdy$	Weekday names.
do	Weekday of start of month displayed.
dn	Number of days in month.
wn	Number of weeks to span month.
xo,yo	Bottom left corner of calendar grid.
xg,yg	Top left corner.
xt,yt	Top left text/tower location.
t	Tower height.
d,m,y,z	Work variables for dates.
a,a$,b,b$	General work variables.

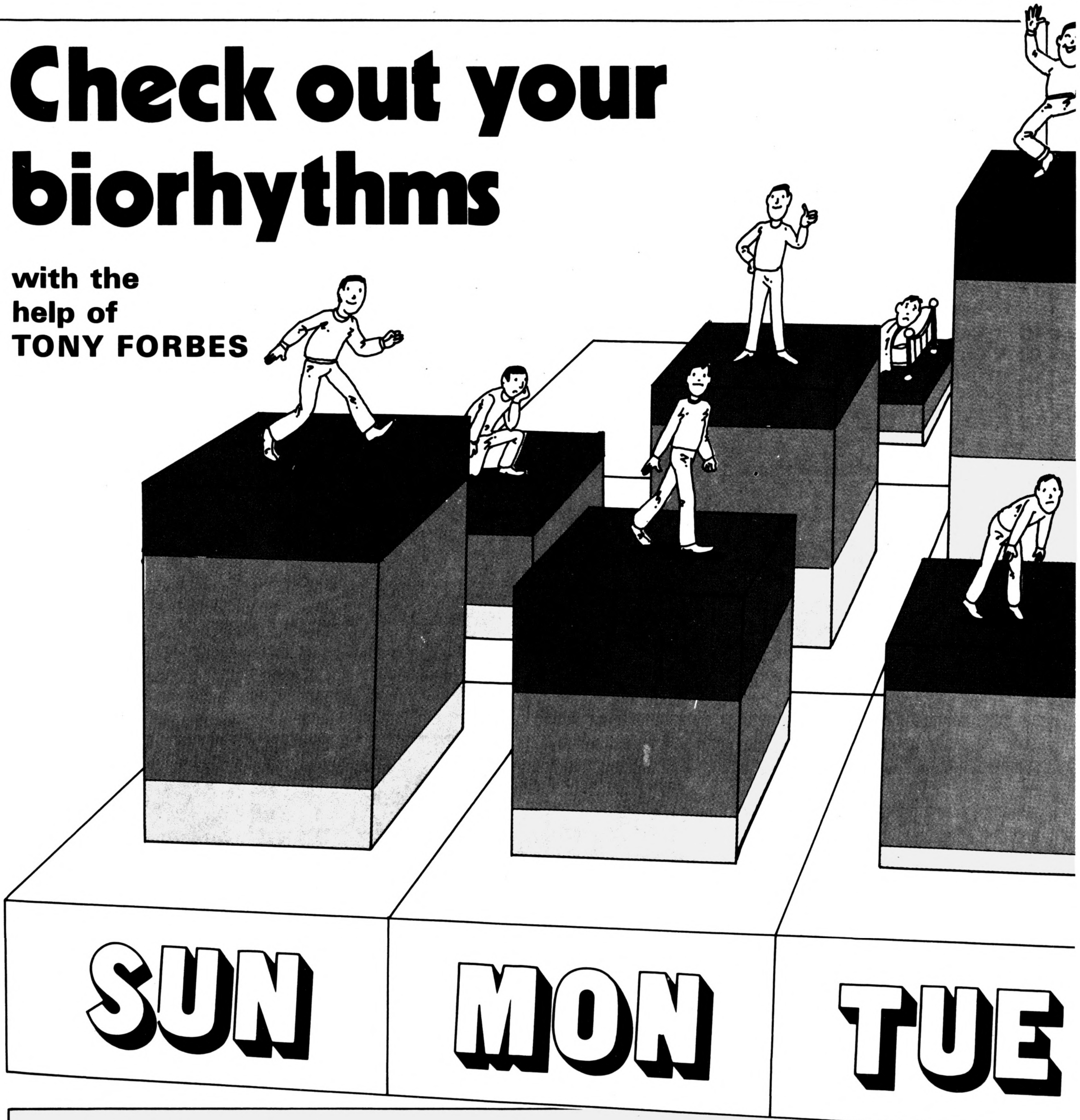
FUNCTIONS

Lines 460 to 640 define all the functions used by the program. They include some useful algorithms for messing around with dates. Include them in your own programs if you want to.

FNleap(y)	Returns –1 if *y* is a leap year, 0 otherwise. Correctly recognises ends of centuries.
FNdmth(m,y)	The trigonometry version of the Thirty days hath September rhyme. The number of days in a month is 31 plus a bit that waves about like the COSine function. February is the exception.
FNz(d,m,y)	Returns the number of days since 31 Dec 1 BC. For simplicity the Pope Gregory calendar change has been ignored. So dates before the 16th century may not give the correct answer.
FNoK(d,m,y)	Returns –1 if the date is valid.
FNwkday(z)	Returns the weekday of day number *z* (result of FNz function). 0 is Sunday, 1 Monday and so on up to 6 Saturday.
FNth$(a)	Returns the correct pair of letters to stick on the end of the number *a*.
FNlev(z,p)	Level of the period *p* cycle for day *z*.
FNxp,FNyp	Where to build a tower.

Check out your biorhythms

with the help of TONY FORBES



```
10 REM *************************
20 REM *  BIORHYTHM CALENDAR  *
30 REM *  ------------------  *
40 REM *     Tony Forbes       *
50 REM *************************
60 REM
70 REM(c)Computing With The Amstrad
80 REM
90 GOSUB 310:REM initialisation
100 REM *** MAIN LOOP
110 GOSUB 660:REM data entry
120 GOSUB 190:REM calendar
130 ON INSTR(" NPM",UPPER$(INKEY$))+1
 GOTO 130,130,140,160,110
140 dm=dm MOD 12+1:dy=dy-(dm=1)
150 GOTO 120
160 dm=(dm+10) MOD 12+1:dy=dy+(dm=12)
170 GOTO 120
180 REM *** calendar
190 MODE 1
200 do=FNwkday(FNz(1,dm,dy))
210 dn=FNdmth(dm,dy)
220 wn=INT((dn+do+6)/7)
230 xg=32*wn:yg=56*wn
240 GOSUB 1050:REM grid
250 xt=xo-22+xg:yt=yo-38+yg
260 GOSUB 1150:REM text
270 xt=xt+22:yt=yt+4
280 GOSUB 1370:REM towers
290 RETURN
300 REM *** initialisation
310 DEFINT a-y
320 GOSUB 470:REM functions
330 BORDER 16
340 INK 0,16:INK 1,9:INK 2,1:INK 3,3
350 RESTORE 380
```

```
360 DIM mth$(12)
370 FOR a=1 TO 12:READ mth$(a):NEXT
380 DATA Jan,Feb,Mar,Apr,May,Jun,Jul,
Aug,Sep,Oct,Nov,Dec
390 wkd$="Sun Mon Tue Wed Thu Fri Sat
"
400 PRINT CHR$(23);CHR$(0);
410 xo=0:yo=16
420 name$="Tamsin"
430 bd=3:bm=5:by=1980
440 dm=4:dy=1985
450 RETURN
460 REM *** FUNCTIONS
470 REM y --> TRUE if y is leap
480 DEF FNleap(y)=y MOD 4=0 AND (y MO
D 100<>0 OR y MOD 400=0)
490 REM m,y --> number of days in mon
th
500 DEF FNdmth(m,y)=INT(31+COS(2.7*(m
-7.5))+(m=2)*(2+FNleap(y)))
510 REM d,m,y --> number of days AD
520 DEF FNz(d,m,y)=y*365+(y+3)\4-(y+9
9)\100+(y+399)\400+INT(30.401*(m-1))-
(m=2)+(m>2 AND m<8)-(m>2 AND FNleap(y
))+d
530 REM d,m,y --> TRUE if date is val
id
540 DEF FNok(d,m,y)=(y>=0 AND y<=9999
 AND m>0 AND d>0 AND d<=FNdmth(m,y))
550 REM z --> weekday 0=Sun, 1=Mon, e
tc
560 DEF FNwkday(z)=z+5-INT((z+5)/7)*7
570 REM 1 --> 1st, 2 --> 2nd, etc
580 DEF FNth$(a)=STR$(a)+MID$("thstnd
rdthththththth",1-2*(a\10<>1)*(a MOD
10),2)
590 REM level in calendar
600 DEF FNlev(z,p)=INT((SIN((z/p-INT(
z/p))*2*PI)+1)*6.999)
610 REM position in calendar
620 DEF FNxp(a)=xt+64*(a MOD 7)-32*(a
\7)
630 DEF FNyp(a)=yt-56*(a\7)
640 RETURN
650 REM *** data entry
660 MODE 1:WINDOW 3,38,4,25
670 LOCATE 1,1:PEN 1:PRINT "B I O R H
 Y T H M   C A L E N D A R":PRINT STR
ING$(35,154)
680 LOCATE 1,5:PEN 2:PRINT "Enter you
r name";
690 LOCATE 4,7:GOSUB 880:REM input
700 IF a$="" THEN 780
710 name$=LEFT$(a$,20)
720 LOCATE 1,9:PEN 2:PRINT "Your date
 of birth (eg  28 nov 44)";
730 LOCATE 4,11:GOSUB 880:REM input
740 IF a$="" THEN 780
750 GOSUB 920:REM convert date
760 IF NOT FNok(d,m,y) THEN PRINT "In
correct date entered - try again";:GO
TO 730
770 bd=d:bm=m:by=y
780 zb=FNz(bd,bm,by)
790 LOCATE 1,13:PEN 2:PRINT "Month to
 display (eg  apr 85)";
800 LOCATE 4,15:GOSUB 880:REM input
810 IF a$="" THEN RETURN
820 a$="1"+a$
830 GOSUB 920:REM convert date
840 IF NOT FNok(d,m,y) THEN PRINT "In
correct date entered - try again";:GO
TO 800
850 dm=m:dy=y
860 RETURN
870 REM *** input
880 PEN 1:INPUT "====> ",a$
890 LOCATE 1,18:PEN 3:PRINT CHR$(20);
900 RETURN
910 REM *** convert date a$ --> d m y
920 d=VAL(a$)
930 a$=UPPER$(a$)
940 FOR a=2 TO LEN(a$)
950 IF INSTR("JFMASOND",MID$(a$,a,1))
>0 THEN 970
960 NEXT
970 b$=MID$(a$,a,3)
980 FOR m=12 TO 0 STEP-1
990 IF UPPER$(mth$(m))=b$ THEN 1010
1000 NEXT
1010 y=VAL(MID$(a$,a+3))
1020 y=y-1900*(y<=99)
1030 RETURN
1040 REM *** grid
1050 MOVE xo,yo
1060 FOR a=0 TO wn
1070 DRAWR 448,0,2:MOVER -448+32,56
1080 NEXT
1090 FOR a=0 TO 7
1100 MOVE xo+64*a,yo
1110 DRAWR xg,yg
1120 NEXT
1130 RETURN
1140 REM *** text
1150 PEN 1:LOCATE 1,1:PRINT "BIORHYTH
M CALENDAR"
1160 PEN 2:PRINT:PRINT name$
1170 PRINT:PRINT MID$(wkd$,4*FNwkday(
zb)+1,3)
1180 PRINT mth$(bm);FNth$(bd)
1190 PRINT CHR$(23);by
1200 WINDOW #1,33,40,21,25
1210 PEN #1,1:PRINT #1,mth$(dm);STR$(
dy)
1220 PEN #1,3:PRINT #1," N Next  P Pr
ev  M Menu";
1230 PRINT CHR$(23);CHR$(3);:TAG
1240 PLOT -8,0,1
1250 MOVE xo+8,yo-4
1260 PRINT wkd$;
1270 PLOT -8,0,3
1280 FOR d=1 TO dn
1290 MOVE FNxp(d+do-1),FNyp(d+do-1)
1300 PRINT d;:NEXT
1310 TAGOFF:PRINT CHR$(23);CHR$(0);
1320 LOCATE 40,12:PEN 3:PRINT CHR$(22
8);
1330 LOCATE 40,14:PEN 1:PRINT CHR$(22
9);
1340 LOCATE 40,16:PEN 2:PRINT CHR$(22
7);
1350 RETURN
1360 REM *** towers
1370 z=FNz(0,dm,dy)-zb
1380 FOR d=do TO do+dn-1:z=z+1
1390 MOVE FNxp(d),FNyp(d)
1400 PLOTR 0,0,2
1410 t=FNlev(z,33):REM Intellect
1420 GOSUB 1590:REM level
1430 PLOTR 0,0,1
1440 t=FNlev(z,23):REM Phisical
1450 GOSUB 1590:REM level
1460 PLOTR 0,0,3
1470 t=FNlev(z,28):REM Emotional
1480 GOSUB 1590:REM level
1490 FOR a=0 TO 3
1500 DRAWR 16,0:MOVER 0,2
1510 DRAWR -16,0:MOVER 2,2
1520 NEXT
1530 MOVER -8,-16
1540 DRAWR 16,0,0
1550 DRAWR 6,14:MOVER -6,-14
1560 DRAW FNxp(d)+16,FNyp(d)
1570 NEXT:RETURN
1580 REM *** level
1590 FOR a=1 TO t
1600 DRAWR 16,0
1610 DRAWR 6,14
1620 MOVER -22,-12
1630 NEXT
1640 RETURN
```

Can't hack keying it in?

See Page 38

Making light of penmanship

DORENE COX tries her hand with four light pens

CHRISTMAS came early for me this year and a lot of fun it was too, playing with a bundle of light pens for the Amstrad.

Under scrutiny were the Amstrad LP–1 at £19.95, DK'Tronics Graphic Lightpen at £24.95, the Electric Studio Light Pen at £19.95 (tape) or £26.95 (disc), and Dart Electronics Light Pen at £39.95, henceforth referred to as "LP", "DK", "Electric" and "Dart".

The LP is colour monitor/TV only, won't work with a serial interface or speech synthesiser present, and plugs into the joystick port.

The others use the disc port and include a green monitor facility. Dart's and DK's interfaces resemble the disc interface, matching the computer livery, and with a through connector for add-ons.

Electric's interface is only slightly larger than the port and has no through connector on the tape version.

Each pen comes with a software graphics package. Dart has built-in disc transfer and relocatable pen code. The LP and DK are deliberately unprotected, intending their routines to be used in your own lightpen programs and encouraging back-up copy or transfer to disc.

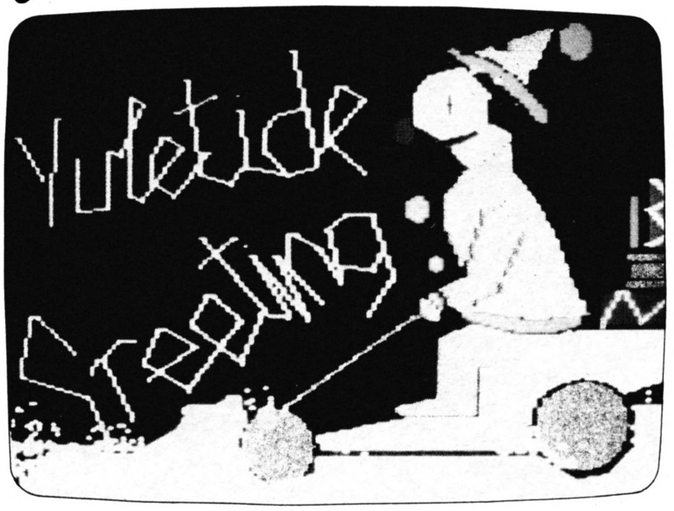
Electric protects against accidental break, and 464 owners will gnash teeth at their inability to connect discs.

All the pens look much the same, like a black biro without the refill. When pointed close to the screen they detect refreshment – and I don't mean beer.

You see, faster than your eye can see, every pixel position is constantly being switched off, updated with the latest information and switched on again.

The pen detects the off/on and signals the computer, which calculates the pen position by the difference in time between start of refreshment and pen signal.

When you consider we're dealing in microseconds here, the wonder is that the position calculation is as near as it is.



But don't expect pinpoint accuracy, although Dart comes very close. The others let you switch to cursor control when you need the extra detail.

Electric allows joystick control as well, but cursor is more accurate. And note that dark backgrounds make pen light detection difficult, although careful screen organisation can overcome this.

All of the packages can operate in Mode 0. The LP offers Mode 1 as well, plus a small light pen game, and Dart provides all modes.

The LP has two full screen menus, with a little white "touch here" box beside each function, taking you to the alternate screen menu, a colour change menu, or the draw screen. Pressing Enter twice returns you to the menu.

Dart has a full screen black on white menu, called by M, with a touch box beside each item which turns blue on selection. Escape returns to Draw Screen.

DK's are ikon driven, overlaying the draw screen centre when called. Each contains three to five pictures – an artist's palette for colour change, a water tap for fill – and includes entrance to the next menu.

As the pen is pointed, large flashing brackets surround the indicated ikon and Enter confirms selection, forwarding to another menu or Draw Screen. Escape returns to the last or previous menu at any time.

Electric is also ikon driven and covers 60 items – including the colour palette – in a single menu, with choice of display left or right of the draw screen.

You point the pen, and when the chosen ikon centre flashes, press Space bar to select function and remove menu. Enter returns to menu.

The lightpens vary considerably when it comes to the manner of DRAWing.

Amstrad expects you to select approximate positions by penpoint and Space bar, providing a small flashing point which can be manipulated by the cursor key.

A large flashing band rushes in from screen edge to leave the small flashing point. Although alarming, it's the reason why dark backgrounds are no problem to the LP pen.

DK provides a small character cross-hair cursor, following pen or cursor as selected and fixed by Enter, although the Space bar sometimes has an extra use.

Electric's cursor is a full screen cross which sometimes looks out of focus and jitters madly. Space bar fixes points and Enter ends functions and returns to menu.

Dart's cursor is a single pixel point, steady enough to allow pixel editing on screen. Functions are switched on/off by specified keys, such as D for Draw and E for Elastic.

In free hand draw, Dart performs superbly, but I couldn't control the others. In fairness, longer familiarity with the pens would probably give better results.

DK and Electric offer re-calibration, but cursor jitter probably makes this purely cosmetic. Electric clearly says use cursor keys on brushes, spraycans and textures.

Only the LP and Dart allow the full choice of 16 from 27 colours although the LP's advantage is somewhat outweighed by their SAVE method which needs a fair degree of programming knowledge to finally heave the saved screen into your own program.

DK confines you to their selection of 10. Not a bad choice, but the two blues look alike on TV.

Electric offers 15 ink colours – chosen by them – and all 27 as paper colours. Again, the choice includes blues and greens which look much the same on TV.

But a listing given in the manual allows the full range – DK, please note. The Clr key step-changes through the ink colours in some functions.

The LP lightpen offers none of the brushes, spraycans and textures users of lightpens on other micros will be used to.

Dart on the other hand has three pen widths, two spraycan widths and a superb thick and thin (depending on direction) pen.

DK has four widths of pen, usable in all directions, and a superb spraycan effect.

Electric's brushes operate up and down only, but have nine widths, obtained from the numeric pad. The original nine width, dot adjustable spraycan indicated in the manual has been altered to a single spraycan with better coverage.

Only Electric offers textures, vertical/horizontal bars, plus two checkerboard types, each varying further via the numeric pad and small Enter. To be honest, some variations look like multi-coloured garbage, but produce interesting patterns. Sadly, there's no texture fill.

The problem here was remembering the controls. With most Electric functions, Space bar turns the function on then follows the pen or cursor, and enter returns to menu.

With texture, however, you Space bar "on", which drops one character space full of texture, Space bar "off", move to the next position, and repeat the process.

If, as is only too easy, you press Space bar and start to move pen or cursor, half the first square is dragged to the second.

The effect can be quite useful, providing you do it on purpose. If not, the air turns an interesting colour without computer aid.

Electric's fleck colour confines its file of splatter patterns to the few colour mixes displayed, and is difficult to control.

With any lightpen package the acid test is the paint fill. Hours of work can be lost if a fill leaks through one open pixel, so it's useful to be able to take quick remedial action.

LP and Dart score full marks for a Delete last fill function, although LP's fill can be a bit hesitant.

DK's fill will stop if you press Escape fast enough, so damage can be minimised.

But if Electric leaks you've had it. And, unlike the others, Electric demands that the outline be the same colour as the fill. This makes filling their 3D shapes in colour a tedious job of changing the colour of every pixel in the outline by hand.

Filling a different colour will often work, but you're playing Russian roulette.

I found banding the most useful function because I am not your "three quick strokes and it's all there" type artist.

I know the right line when I see it, but it sometimes takes hours and many wrong lines to find it. The screens accompanying this article owe everything to banding.

On LP you plot two points, draw a line between, plot another point, and the line continues from the previous point, making shapes and curves relatively easy.

DK go even better with rubber banding. An elastic line follows your pen or cursor, making it easy to see the eventual shape, and you can plot up to 20 points at a time, deleting backwards should you change your mind, and making complex shapes a doddle.

Electric and Dart also have elastic lines, but operating between two points only. To continue to a third you need to plot number two again first.

And if you forget, repositioning to the last plot again is tricky and can lead to leaky filling points.

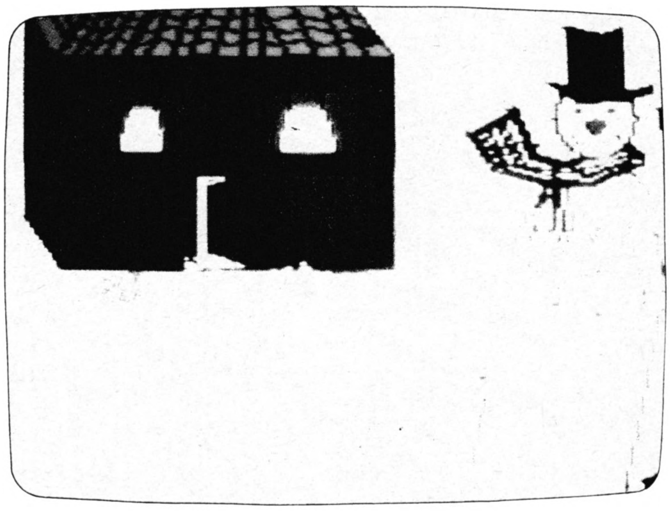
But Electric also offers the ability to plot 12 points and have them transformed into either a 3D figure or a Bezier geometric plane.

Another test of a lightpen package is its range of filled shapes.

Dart provides only circles and rectangles, while the others offer triangles as well, and Electric and the LP include rays.

The LP makes its shapes by plotting opposite corners, or centres and radius, and allows delete. DK happily relies upon its rubber band facility.

All Electric's shapes, ellipses, diamonds, pentagons, hexagons and octagons, are elastic. Plot an approximate



point and the shape appears.

You can then reduce, enlarge, squeeze and elongate with numeric keys, and cursor move to the accurate position before Space bar fixing.

A very nice feature is the 3D box and triangle.

Having drawn your picture, you often want to write on it. LP, Dart and Electric let you type any keyboard available alphanumeric in any position, although Electric has a nasty bug which they're "looking into".

Be warned. You can adjust to the cursor setting jumping forward one space when you Space bar "on" – but the unwanted printed space caused by Space bar "off" means repair work.

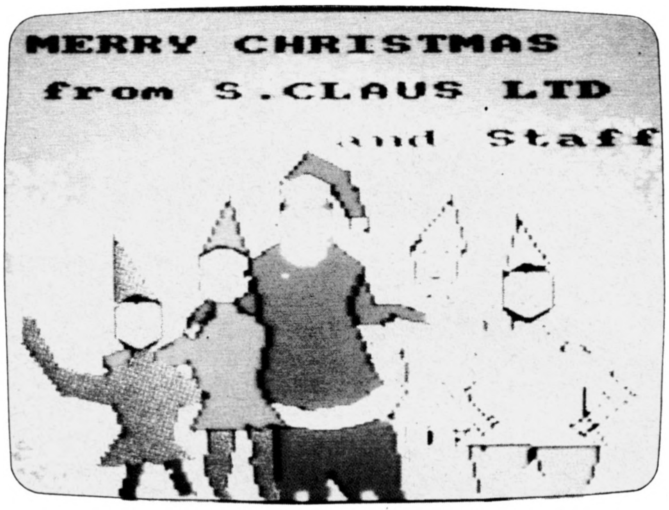
DK offers normal and sideways printing in any position, including all the non-keyboard symbols by using Tab with Ascii code. And it would be simple to user-design these before loading.

Excepting Dart, they all zoom. This is, they can blow up a small selected area of the screen for editing.

LP and DK use the whole screen, Electric uses a smaller overlay on the draw screen and is sometimes inaccurate in return.

Again excepting Dart, they all copy small areas of screen to other positions, but only LP and DK offer a reduce/enlarge by a factor of four function.

Dart has a useful eraser and



Sprites – actually a screen size designer board using a 24 by 24 pixel grid, with all colours to hand. The design can then be placed anywhere on the draw screen.

Deleting dark colours in the first vertical gridline was difficult.

Electric allows small areas to be dragged to a different position, which can be achieved by a combination of functions on the others.

Only Electric has flip/reverse, where you could, for instance, draw one corner ornamentation and process it to match all four corners.

And having created our masterpiece, it's always useful to dump it to a printer. The LP's dump works on the Amstrad DMP-1 only, dumping black/white without shading. A listing for other printers is available from Amsoft. Really it should have been included.

DK gives a separate machine code program, the manual providing enough listings and information to make it work on most printers. The printout is nicely shaded and can be varied in size.

Electric's option is built-in and gives a one size black/white printout without toning.

The Electric has some additional features not found in other packages, such as optional displays of pen/paper numbers, dot or grid backgrounds and cursor x,y positions, which are handy for accurate positioning.

There's also a colourmask facility which changes one colour to another over the whole screen. The vertical/horizontal parallel lines are useful, but the eraser needs practice and the "transfer control to joystick" is a menace if you have no joystick.

With packages as potentially powerful as lightpens, the in-screen instructions and accompanying manuals are very important.

The LP's on-screen instructions offer terse information on lightpen use and programming. The 12-page manual covers the software functions adequately, gives cassette to disc conversion notes and precious little else.

Dart's six page leaflet has DIY listings for putting screens and the relocatable pen operating code into your own programs. It explains the few functions clearly.

DK's 30 pages go step-by-step over all features, give back-up and discs conversion notes, have lots of information on pen and screens, but require you to type in many listings.

Electric's 24 pages explain the functions and offer listings to use the pen or screens in your own programs.

There are some curious omissions. You're left to guess where the pen plugs in, or how to edit in zoom, for instance.

THE BOTTOM LINE...

DART should be ashamed of their graphics package. It works – but I've seen more facilities in magazine listings. The pen does not work on the pixel edge of the screen, which can leave leaky gaps.

On the other hand, it is streets ahead of the others in overall accuracy and control but, comparing graphic packages, is grossly overpriced.

The remaining three pens appear to share the same, lower level of accuracy, differing only in graphic packages.

The LP's non-frilly package has a full colour range and no dark background problems.

But using created screens is difficult and an uninvited, self-printing arrow, apparently due to the joystick port connection, is a nuisance both in loading and editing the program listing.

DK's package is easy to use and offers many frills. Its large ikons make selection reliable and information is generous. But it falls down on its small colour selection and lack of an integral printer dump.

Electric's weakness is its menu. 60 items in 5 by 25 character spaces makes each ikon extremely small and, hindered by cursor jitter, you all too often unknowingly trigger the wrong one and the result can be disastrous.

Monitors improve, but don't cure this problem. And having to fill an area in the same colour fill as its outline is a bore.

But there's a lot of goodies here to make up for the inconvenience of frequent safety saving.

Which would I choose? Well, if we've talking strictly about hardware, I'd go for the Dart because of its accuracy.

However software is a vital part of a lightpen package, so I'd choose the DK'Tronics pen because of its versatility and reliability.

DORENE COX

An excerpt from John Hughes' succesful book Mastering the PCW 8256/8512 - we look at choosing and using applications programs.

Pick the best applications programs

IN the days of computing's stone age back in the 1950's using a computer meant, almost inevitably, writing your own programs for it.

If you couldn't do that you either had to pay someone rather a lot of money to write a special program just for you, or you learned to program for yourself. And if you had neither the money to pay nor the patience to learn you forgot about the computer and made do with pen and paper instead.

Custom-designed programs are still being created, though as programmers now are paid rather more than they were 30 years ago – even allowing for inflation – buying one will set you back quite a lot of money.

All the same, for certain highly specialised purposes like landing a rocket on the moon or handling an airline's world-wide ticket reservation system, research institutes and big companies have no choice but to pay the price.

For most normal commercial and domestic tasks however there are now available an enormous number of ready-made programs for different types of computer. Those which are designed to do something for the user – rather than help him write programs or organise his disc-files – are called applications programs and it is with these that I am now concerned.

However before looking at individual programs it is worth while taking a general look at such programs in order to see what features are most useful and what should be avoided.

First, it has been estimated that there are over 8,000 commercial programs available for CP/M computers. It is fair to point out that most of them – like the majority of books, paintings and whatever else you care to name – are not very good.

Often this is either because they were written a long time ago and have not been properly updated, so that they fail to make full use of the power of modern computers, or because they were designed for a totally different market from the one in which they are now being offered.

For example, accounting programs from the USA may ask you questions about State taxes and Federal taxes which have no relevance in the UK. Even if they avoid this sort of trap they may make it impossible to print money amounts with a pound sign (£) in front of them. Since most computer software comes from the Unted States this sort of problem is more common that you might expect.

Another area that needs care is that of user friendliness, as ease of use is called. The theory is straightforward enough: a user friendly program is one which you can use without needing to remember a string of different commands – perhaps it uses menus, like LocoScript – and which tries to stop you making destructive errors by double-checking that you really mean to do what you have said.

These goals are admirable in themselves, and programs which take no account of them are often awkward to use – the lack of friendliness of some of the older CP/M utilities is a frequent source of complaint.

At the same time some programs are so friendly that they positively get in the way of your work by presenting you with menu after menu, even after you have become quite familiar enough with the program to be able to get along without that help.

In these terms LocoScript offers the sort of compromise which you may find satisfactory – help is available for when you need it, but you can usually avoid it when you don't. Most good software follows the same principles.

Good applications programs almost always involve you in buying a special system for doing whatever task they set out to perform. Because they were designed for general use and not for your own specific purposes you will need to adapt your present practice to fit the new system.

Many small businesses, particularly sole proprietorships, struggle along without any particular system for the various chores which crop up.

In such organisations doing the payroll may involve nothing more complicated than sitting down with the appropriate tax tables and record cards, and keeping accounts sometimes degenerates into a squalid system of putting receipts and cheque stubs into a drawer for the accountant to puzzle over sometime in the future.

More organised concerns will have provided themselves with one or the other of the various excellent manual payroll and accounts systems. It is with these that an accounts or payroll program should be compared. You must therefore be sure that the system which you buy is one that meets your own particular needs, or is easily adaptable to them.

The keynote here is flexibility. An accounts system which insists on a full set of books being kept may well be very rigorous and accurate, but it may be more than you need. Perfectly good accounts for typical small business purposes can be maintained with a simple spreadsheet program.

The other problem is the converse of this – it is common for users to start off with a program which is too limiting for their long-term needs, and to find out too late that converting to what they should have had all along would be a slow and expensive business.

The only advice which will always apply is to look very carefully at a program before you buy it. If this is not possible you may have to rely on reviews in the various business and computing magazines.

At all events it is a sad fact that not all the people who sell programs will be of much help to you. This is partly because they will be largely unaware of your particular problems, and partly because most companies are geared up to the selling of hardware rather than software. The proverbial pinch of salt can be a great help in choosing applications programs.

Until quite recently applications programs assumed that when you wanted to do accounts you would load an accounts program into your computer, when you wanted to write a letter you would use a word processing program and so on.

Gradually this concept has begun to change, so that it is now possible to buy applications programs which will do more than one task.

Imagine writing a quotation for work which you are going to undertake, for example. The majority of the task would count as simple word processing, but in addition you might need to carry out some simple calculations – adding up the cost of several individual operations, perhaps.

Instead of having to stop word processing, load a program to carry out calculations, write down the answers on a piece of paper, reload the word processor and copy the figures off the paper and into the document, some programs now permit you either to carry out word processing in the middle of, for example, a spreadsheet, or to do simple arithmetic in the middle of a word processing job.

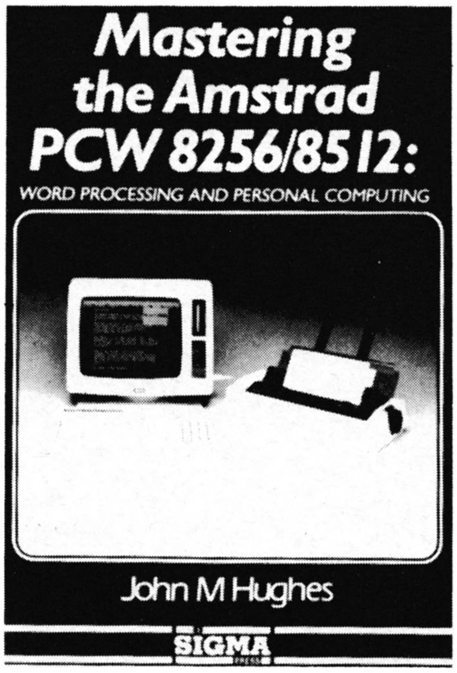
The ultimate in this sort of mixed operation is sometimes referred to as messy desk computing, because it allows you to handle several different sorts of operations simultaneously, just as you can have papers relating to several different jobs on your desk at the same time.

In general, software for this kind of operation tends to be expensive, is sometimes not very user friendly, and no doubt many users fail to make full use of its facilities.

However there is no denying that an ability to do more than one thing in a particular applications program, or at least to import the results of a calculation directly into a word processor without writing things down on a scratch-pad, can be a great time-saver.

Most spreadsheet and database programs have the ability to output text as well as the normal results of their calculations, and this can be a most worthwhile facility.

When changing over from a manual to a computerised system there will inevitably be occasions when things don't work as they should. There are some real horror stories about big companies which computerised their payroll accounting and only then found faults with their new system, which meant that no one could be paid on time.


Mastering the Amstrad PCW 8256/8512:
WORD PROCESSING AND PERSONAL COMPUTING
John M Hughes
SIGMA

The golden rule, particularly where money is concerned, is – never abandon a manual system in favour of a computerised one without a period of running both at the same time.

This may seem wasteful, but the advantages are two-fold: If something goes wrong with the new system – and because it is new and unfamiliar, no one will know what to do if that happens – the old system is still there to fall back on, and if you get different results from the two systems you will know that there is something seriously wrong with either the one or the other.

There is a distressing human tendency to believe what the computer tells us, even if closer scrutiny would show it to be manifestly absurd – to chain your accounts payroll to a system which may be making errors without any way of double checking the result, is extremely dangerous.

Trying to explain to an employee that he has been underpaid this week because of a computer error – which is usually shorthand for operator error – is bad enough, but an error of just a few per cent on a budget forecast, caused perhaps by carelessness in entering formulae on a spreadsheet, could mean the difference between success and receivership. Be warned.

Incidentally it is all too possible when you are converting from a manual to a computerised system that you will be too easily satisfied with what you have and thus make no further progress. This has happened in so many businesses that the bizarre combination of high-tech equipment and old fashioned methods has become a modern commonplace.

The classic army equivalent is the true story of how men were detailed to hold the horses in artillery companies which had been mechanised for decades because nobody understood the consequence of the rule that changes in technology should mean changes in techniques as well. 'Nuff said!

Buying a program is a little like buying a car – usually it works perfectly well at first, but sooner or later you may have problems with it.

The reasons for this are pretty much the same for both cars and programs, and have to do with the fact that both are highly complex products used in the main by non-specialists. Of course a program should never actually wear out in the same way as the parts of a car, but otherwise the analogy is a fair one.

Unexplained errors in the operation of a program are called bugs, and it is safe to assume that nearly every program has them, just as nearly every book has misprints in it. Generally speaking these bugs are pretty harmless, and in most commercial programs they are unlikely to cause you any real inconvenience but you will assuredly meet them from time to time.

Several of the better producers of commercial software offer schemes for maintaining their programs, perhaps in conjunction with telephone advice – often rather dramatically called a hotline service.

In some cases these services are provided free for all registered purchasers of the program, thus helping to freeze out users of illegitimate copies. Sometimes there is a small annual charge to pay.

Either way, the provision of proper after-sales service, or software support, is something which you should investigate when choosing programs – particularly those, like payroll, which may cause embarrassment if there is a substantial delay in running them.

A final problem of computerised systems, and one which is becoming increasingly significant, is that of the security of the system.

It is sometimes assumed that the security implications of computers are not basically any different from those of normal manual practice. This is emphatically untrue.

From the standpoint of small business use, the biggest source of difficulty is the floppy disc. Unlike documents on paper, which take time to copy, the entire contents of a floppy disc, amounting perhaps to several hundred pages, can be copied in a matter of seconds and leave no trace of the copying having taken place.

Various attempts have been made to improve security, and you should at least have an eye towards these when considering the purchase of new software. The classical solution is passwords, combinations of letters and numbers without which it is impossible to gain access to confidential files.

Some sophisticated systems allow different levels of security, with different passwords for each level, so that a user may be able to see some records on a disc but not others – for example, it may be possible for payroll personnel to amend pay records of staff, but not to gain access to personal information of a confidential nature.

Passwords generally provide excellent security if they are properly used. However, as their value depends upon their secrecy, they can be rendered useless by people writing them down to save forgetting them, or choosing easily-guessed words such as their own names or the names of their nearest and dearest as passwords.

Password generator programs are available which will suggest secure and easy-to-remember combinations to avoid these problems.

Even if all due care is taken with choosing and using passwords any user with a moderate level of experience of computing will probably be able to get at confidential information stored on a floppy disc.

Short of using various encryptation techniques to scramble data – and these are available – the best advice in any organisation bigger than a one-man-band is to enforce strict security standards when handling floppy discs, to keep them locked away when not in use and to restrict the number of staff who are permitted to handle them.

Payroll and accounts programs are particularly vulnerable to ingenious and dishonest tampering, and the very nature of the computer makes it less likely that an offender will be detected. There is now a fairly substantial literature about computer fraud, one of the effects of which has inevitably been to popularise it. Don't assume that it won't happen to you.

Modern computing equipment is extremely reliable, and breakdowns are few and far between. However they do happen from time to time. The most vulnerable items are those which have a high proportion of mechanical parts, such as printers and disc drives.

Short of providing actual physical back-ups for these – and a second disc-drive is certainly a very useful thing to have – it is worth considering that programs which give you the option of redirecting printer output to the screen can be a blessing if something does go wrong with a printer.

Obviously a word processor without a printer is practically useless – except in terms of preparing texts which can be printed out later – but many payroll and accounts programs can still be usefully run without a printer if the results are simply backed up on to a disc for printing later.

In this circumstance there is a CP/M Utility program which may prove useful. This is the PUT.COM program, which is used as follows. Enter the instruction:

```
PUT PRINTER OUTPUT TO FILE
filename.filetype
```

and everything that would have gone to the printer will instead be directed to a disc file with the filename and type specified.

To revert to normal output, enter:

```
PUT PRINTER OUTPUT TO
PRINTER
```

The PUT command has a number of forms beyond the two given here – full details of the others are given in the user manual. However the two forms listed are those which are likely to be found most generally useful.

When the printer is again operative you can transfer the contents of the file which you produced with the PUT command on to the printer using our old friend PIP, thus:

```
PIP LST:=filename.filetype
```

LST is CP/M shorthand for the LiSTing device, namely the printer.

To sum up, then: In choosing application programs points which should be borne in mind include being certain that the system will meet your own needs, that adequate support from the manufacturers is available in case you have problems and that the system is suitably secure for your own circumstances.

In starting with a new system it is essential not to abandon the previous manual system immediately, but to run both side by side for a while. In this way errors caused by lack of familiarity with the new system will be trapped, and difficulties caused by hardware or software failure will be eliminated.

Equipment failure can be guarded against to some extent by redirecting the output from the printer to disc files. ■

SUBSCRIPTIONS

CAT #	TITLE	PRICE
5001	MAGAZINE ONLY	$40.00
5002	MAGAZINE + TAPE	$80.00
5003	MAGAZINE + QUARTERLY DISK	$105.00

BACK ISSUES

CAT #	TITLE	PRICE
6008	CWTA PREMIERE EDITION	$4.50
6009	CWTA SEPTEMBER ISSUE	$4.50
6010	CWTA OCTOBER ISSUE	$4.50
6011	CWTA NOVEMBER ISSUE	$4.50
6012	CWTA DECEMBER ISSUE	$4.50
6101	CWTA JANUARY ISSUE	$4.50
6102	CWTA FEBRUARY ISSUE	$4.50

TAPES

CAT #	TITLE		PRICE
7008	DIAMOND DIG	(8/86)	$7.50
7009	ICE FRONT	(9/86)	$7.50
7010	da BELLS	(10/86)	$7.50
7011	DISCMAN	(11/86)	$7.50
7012	ROBOT RON	(12/86)	$7.50
7101	OTHELLO	(1/87)	$7.50
7102	SPACE BASE	(2/87)	$7.50

DISKS

CAT #	TITLE	PRICE
7051	7008/7009/7010 AS ABOVE	$19.95
7151	7011/7012/7101 AS ABOVE	$19.95

BOOKS

CAT #	TITLE	PRICE
3001	AMSTRAD HANDBOOK	$ 9.95
3002	AMSTRAD COMPUTING	$17.95

ORDERING INFORMATION

1. MAIL ORDER

Should you wish to order by mail but not wish to deface your magazine please photocopy P.38 or handwrite your order being careful to include **all** information requested on the order form. Please make sure you have enclosed your name and address (you'd be surprised!) and the correct amount for the goods you require.

If sending a cheque or ordering using Visa, Mastercard or Bankcard please ensure that the date on your cheque is valid (i.e. 1987 not 1986) and that your credit card has not expired.

2. TELEPHONE ORDER [008] 030930

This month we have installed a new toll-free order line. Please follow the instructions below carefully **before** ringing. Note that this number will only be answered by a machine and cannot be used for general enquiries or messages. Anything other than an order will be ignored - you have been warned!

A) Complete the order form on Page 38 as though you were going to order by mail. Do not wait until ringing before deciding which titles you require or trying to find your credit card. The answering machine in use is voice activated and any pause over a couple of seconds will result in the machine hanging up on you.

B) When the machine answers, read the order from your order form slowly, clearly and distinctly giving all the information you have written down. Where possible leave a telephone number just in case we can't understand or hear your order.

C) This service will be in operation 24 hours a day, every day of the year.

D) Allow 28 days for the delivery of your order - orders which we cannot despatch within 2-3 days of receipt will be advised of the likely delay by mail.

All prices include postage & packing

SOFTWARE ORDER FORM

SOFTWARE ON TAPE

CAT #	TITLE	PRICE
1001	TASWORD	$36.95
1006	TOOLBOX	$19.95
1007	FLEXIFREND	$19.95
1008	GRASP	$19.95
1009	CHAOS FACTOR	$15.95
1010	MUZICO	$17.95
1011	DRUMKIT	$16.95
1012	MUSIC COMPOSER	$17.95
1013	EASIDATA II	$29.45
1014	EASIDATA III	$41.45
1015	DATABASE/MAIL LIST	$29.45
1022	QWERTY	$14.95
1024	MYRDDIN FLIGHT	$17.95
1030	AMS-FORTH	$25.00
1208	CLASSIC GAMES	$15.95

SOFTWARE ON DISK

CAT #	TITLE	PRICE
2001	TASWORD	$48.95
2009	CHAOS FACTOR	$27.95
2012	MUSIC COMPOSER	$29.95
2014	EASIDATA III	$53.45
2015	DATABASE/MAIL LIST	$41.45
2016	EASI-WORD COMBO	$49.95
2017	GENESIS	$39.95
2018	EASY MUSIC	$34.95
2019	POT POURRI VOL. 1	$19.95
2020	POT POURRI VOL. 2	$19.95
2022	QWERTY	$26.95
2024	MYRDDIN FLIGHT	$29.95
2208	CLASSIC GAMES	$27.95

PUBLIC DOMAIN DISKS

CAT #	TITLE	PRICE
2801	PD VOL. 1	$19.95
2802	PD VOL. 2	$19.95
2803	PD VOL. 3	$19.95
2804	PD VOL. 4	$19.95

ORDER FORM

CAT #	TITLE	PRICE
	TOTAL	

Bankcard ☐ Mastercard ☐ Visa ☐

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signed ______________ Expiry ☐☐☐☐

Name ______________________

Address ______________________

______________________ 'Phone __________

State __________ Postcode __________

MAIL ORDERS TO:

STRATEGY SOFTWARE
P.O. BOX 11
BLACKMANS BAY
TASMANIA 7152

ENQUIRIES [002] 294377
ORDERS [008] 030930

ORDER TOLL-FREE WITH YOUR
VISA, MASTERCARD OR BANKCARD
SEE SIDE PANEL NEXT PAGE

ALL NEW TITLES AT SPECIAL PRICES

NEW AMSTRAD TITLES AVAILABLE ONLY FROM STRATEGY SOFTWARE

1. GENESIS ADVENTURE CREATOR
DISK ONLY $39.95

Don't pay more! Genesis offers a complete system for writing Text/Graphic adventures with Music and Sound Effects. Features include: Text Compression, synonyms for commands and objects, sentence analyser, multiple graphic windows, variable screen modes, extensive graphic commands and storage of text and graphics on disk for large, disk-based adventures. Ideal for the novice wishing to create his/her own adventures. Used by commercial software houses in the U.K.

2. GRASP PLUS
DISK ONLY $29.95

All the features of Grasp, plus hi-res (mode 2) screen plots, improved labelling, various printer options, faster screen dumps, exploded pie charts and all the advantages of disk operation.

3. MUSICO
TAPE ONLY $17.95

An easy to use system for creating 3 part music and sound effects. Results may be used in your own programs.

4. DRUMKIT
TAPE ONLY $16.95

Percussion Instrument Simulator with eleven standard percussion sounds plus three user-defined sounds. Uses the Amstrad sound chip, no extra hardware required.

5. CHAOS FACTOR

TAPE	$15.95
DISK	$27.95

A graphic adventure game. featuring Nurd Fungus and Narsty & Crutch, the amazing cops who can solve any crime simply by saying 'Hi' a lot! Don't miss it!

6. EASY MUSIC
DISK ONLY $34.95

A combination of MUSICO and DRUMKIT (see above) together with the advantage of disk operation.

7..AMSTRAD POT-POURRI VOLUME 1

TAPE	$ N/A
DISK	$ 19.95

25 great games for your Amstrad. Fantastic value for money at less than 40¢ per program on tape. Volume 1 has a full-length text adventure which alone is worth the money!

8. **AMSTRAD POT-POURRI VOLUME 2**

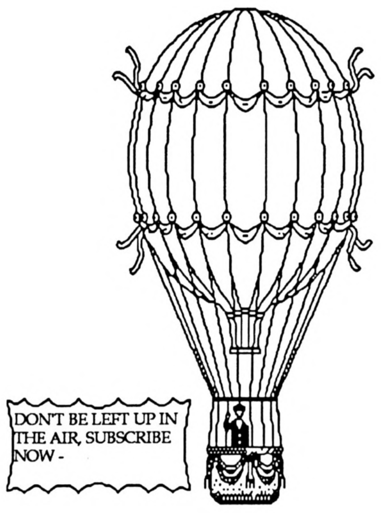
TAPE	$ N/A
DISK	$19.95

25 more previously un-released software on the same basis as Volume 1.Volume 2 includes a great machine-code Space Invaders.

EXTRA SPECIAL!!!

Buy both Volumes 1 & 2 and receive, absolutely free, a tape copier and tape to disk utility. Please note that these programs cannot be guaranteed to work with all tapes.

CALL [002] 29 4377 NOW!

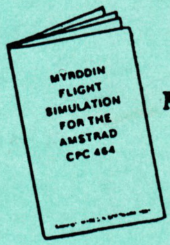


3D LANDMARKS YOU CAN FLY AROUND

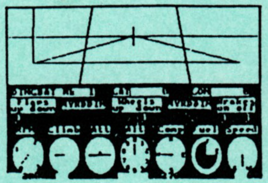
SUPERB REAL TIME SIMULATION

MYRDDIN FLIGHT SIMULATION

AMSTRAD CPC 464

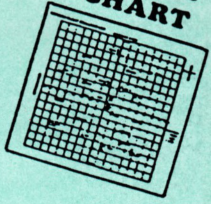


MANUAL

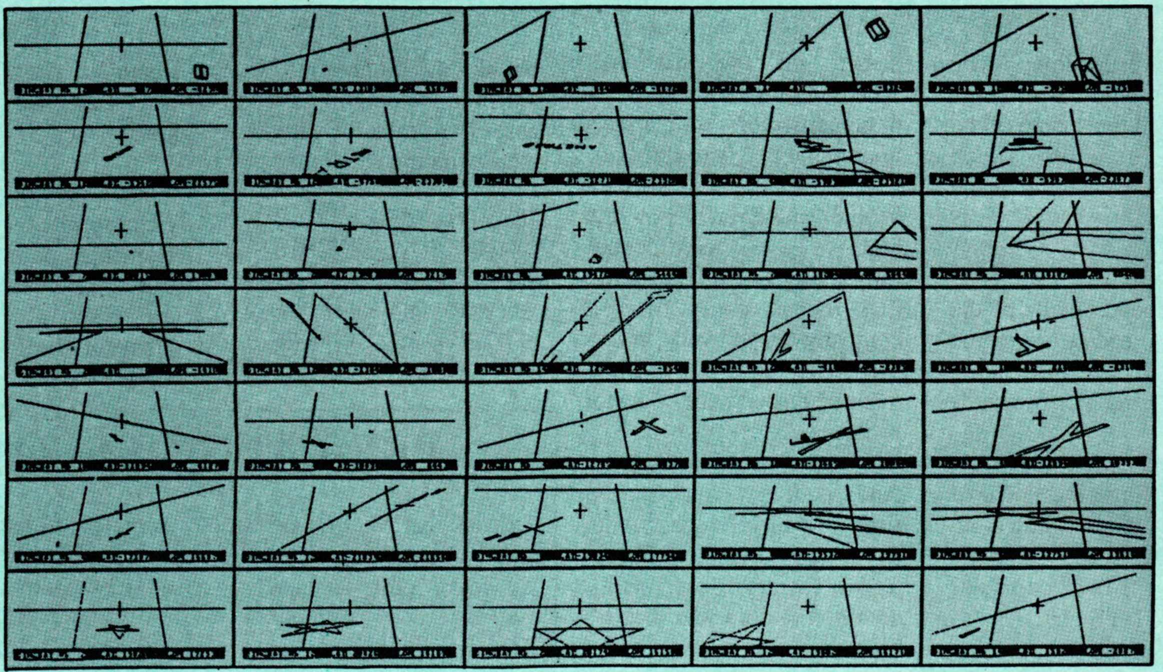


FULL SCREEN DISPLAY

FLIGHT CHART



Here are some screens from a typical flight showing the view from the cockpit (top half of screen) produced as printouts of the actual simulator.



A real time simulation with 3D graphics uses a massive 64000 x 64000 longitude & latitude flying area, making each flight completely different. Developed under pilot instruction to give realistic flight effect. The view through the cockpit gives moving 3D graphics.

Comprehensive instrument panel with moving needle meters & digital displays. 15 aircraft types with varying control sensitivities & speeds of between 100 – 500 knots.

3 runways available for refuelling, take off & landing. Ground and landmark orientation correct with all flying attitudes (rolls etc.).

The 3D graphics are still accurate when you fly upside down.

3D landmarks you can fly around.

Comes complete with manual & fully detailed chart of landmarks & airfields.

Joystick or keyboard operation.

Strategy Software
P.O. Box 11
Blackmans Bay
Tasmania 7152
[002] 29 4377

Tape $17.95 **Disk $29.95**