

AMSTRAD DEATH RACE



DEATH CHASE STARTED as a straightforward conversion attempt on a Spectrum program published by *Your Computer* some time ago. In the process some of the internal workings of the Amstrad were revealed and the result is a playable and colourful game. This article gives the finished program and also details of screen display configuration and how to add extra commands to your Basic — lateral scrolling, enlarged printing and screen character reading.

In converting Spectrum Cross to Amstrad Death Race there were three fundamental problems to be faced. First the different size of the text screen, second the way the screen start moves around in memory, and third the way colour is mapped.

The Amstrad has three screen modes. Mode 2 is the high definition, two colour mode.

Mode 1 is medium resolution with 40 columns of text in four colours. Mode 0 is the multi-colour mode with 20 column text. I decided to reduce the number of cars and lorries on the road to enable me to make the most of the colours.

Although the number of columns would be less than the Spectrum's, the number of lines available was more — 25 instead of 22. Since a double height printing routine has long been one of my favourites, the answer was obvious.

The screen display occupies the area of memory from hex &c000 up and is composed of 25 lines each with eight rows of 30 bytes. Some arithmetic tells you that there is a little memory left over at the end of the screen.

This fact, coupled with hardware scrolling of the screen when for example listing a program, means that the byte with address &c000

may not in fact be expected top left-hand corner. It could be anywhere, even off the screen altogether. However, after a mode change the screen is reset so that solves many of the problems.

Now supposing the top left hand corner is &c000 then we might expect the byte underneath to be &c050, but as on the Spectrum that is the top byte of the second line — pixel row nine. Thinking again and given that 25 lines times 80 bytes = &7d0 we might expect that second byte down to be &c7d0. In fact it is &c800. This tidy figure is arrived at by simply tagging roughly half a line on after the bottom of the screen.

Colour mapping is at its simplest in Mode 2. Here the eight pixel width of a character is matched by the eight bits in one byte. Hence 80 characters per line means 80 bytes across


```

10 *****
20 +
30 +          ANTIHOM: UAHN: UAHN:
40 +          By
50 +          C. J. LEIGH
60 +
70 +*****
80
90 after break run 100
100 GOSUB 10000:REM ***machine code**
110 N:=1:GOSUB 9900:REM ***graphics**
120 ENV 1,3,2
130 ENV 2,10,-1
140 ENV 3,15,-1:0
150 GOSUB 9810:REM ***instructions**
160 N:=7:HOME:GOSUB 9800:GOSUB 1000
170 GOSUB 7000:REM ***scores**
200 REM ***start**
210 PRINT 12:POKE 64450,44
220 GOSUB 5000
230 ZDE 12:CHR$(0)=15
240 Y1=23:COL15=12:Y2=13:Y2=Y1
250 LOCATE 1,1:OPEN 5:PAPER 0:PRINT "*****"
260 PEN 0:LOCATE 12,24:PAPER 0:PRINT "*****"
270 LOCATE 12,25:PAPER 0:PRINT "*****"
280 LOCATE 1,24:PAPER 0:PRINT "*****"
290 IF INKEY(0)=0 THEN 290
300 REM ***loop**
310 IF INKEY(0)=0 THEN Y2=Y1-2
320 IF INKEY(1)=0 THEN X2=X1+1
330 IF X2=21 THEN Y2=Y2
340 IF INKEY(0)=0 THEN X2=X1-1
350 IF X2=0 THEN X2=X1
360 COL2=X2:Y2=Y2:AND Y2=15 AND Y2=23-12 AND Y2=3 AND Y2=12:OR AND Y2=31
370 LOCATE X1,Y1:PAPER 0:PRINT "*****"
380 I:SCROLL 1,4,5,0 9,15,20,21
390 I:SCROLL 1,7,13, 9,17,17
400 IF HOME:5 THEN I:SCROLL 12,13,12,13 ELSE I:SCROLL 2,13,12,13
410 IF HOME:5 THEN I:SCROLL 1,13,13,13,13,13,13,13
420 IF HOME:7 THEN I:SCROLL 1,13,13,13,13,13,13,13,13
430 IF HOME:7 THEN I:SCROLL 1,13,13,13,13,13,13,13,13
440 IF HOME:7 THEN I:SCROLL 1,13,13,13,13,13,13,13,13
450 IF HOME:15 THEN I:SCROLL 1,4,5,0,9,15:SCROLL 7,11
460 PAPER 0:LOCATE X2,Y2
470 ICHR$(CHR$(0) OR CHR$(X2)):THEN 1000
480 PEN 0:IF Y2=1 THEN 9900
490 PRINT "*****"
500 IF Y2=1 THEN SOUND 2,30,20,15,2:score=score+5:LOCATE 1,24:PAPER 0:PRINT "*****"
510 IF Y2=1 THEN SOUND 2,30,20,15,2:score=score+5:LOCATE 1,24:PAPER 0:PRINT "*****"
520 FOR N=1 TO score:NEXT
530 GOTO 310
540 REM ***graphics**
550 SOUND 7,0,120,15,2,0,15
560 N:=0:IF N=0 THEN 550:THEN 700
570 IF N=0 THEN N=1:score
580 LOCATE 1,22:PAPER 0:PRINT "*****"
590 N:=0:IF N=0 THEN 550:THEN 700
600 POK 64450,60:GOSUB 700
610 FOR N=1 TO 10:GOTO 600
620 LOCATE 1,21:CHR$(CHR$(0) OR CHR$(X2)):THEN 2000
630 PRINT "*****"
640 GOTO 310
650 IF HOME:16 THEN HOME:8:GOSUB 2
660 GOTO 210
670 REM ***score**
680 DATA 255,1,255,1,198,0,255,0,5,198,0,5,255,0,5,198,0,5,198,1
690 FOR N=1 TO 10:GOTO 680:GOTO 2,0,10000,0,1
700 NEXT
7100 REM ***score**
7200 REM ***score**
7300 REM ***score**
7400 REM ***score**
7500 REM ***score**
7600 REM ***score**
7700 REM ***score**
7800 REM ***score**
7900 REM ***score**
8000 REM ***score**
8100 REM ***score**
8200 REM ***score**
8300 REM ***score**
8400 REM ***score**
8500 REM ***score**
8600 REM ***score**
8700 REM ***score**
8800 REM ***score**
8900 REM ***score**
9000 REM ***score**
9100 REM ***score**
9200 REM ***score**
9300 REM ***score**
9400 REM ***score**
9500 REM ***score**
9600 REM ***score**
9700 REM ***score**
9800 REM ***score**
9900 REM ***score**

```



DEATH RACE

C J Leigh with a very original version of Frogger for the Amstrad.

the screen. Each bit reset is paper colour – usually pen 0 – and each bit set is usually pen 1.

In Mode 1 things are more difficult since the character width covers two display bytes with each pixel pen being represented by the state of two bits, so giving the four colours available. These bits are not consecutive but the byte is split in two with the most significant bit of each half representing the leftmost pixel of the byte.

In Mode 0 the same general idea is followed. The character width covers four bytes, each byte representing two pixels. The byte is split into four with the most significant bit of each quarter representing the left hand pixel of the pair. Each of the quarters contributes a bit to a 4 bit number with the bits weighted 1,4,2,8 indicating the pen number used.

Extensions to Amstrad Basic can be made very easily by setting up a name table and jump table as in listing 1 along with a four byte buffer and the log on routine at &a400. Each command must be preceded by a bar character available on the @ key.

The scrolling routines are straightforward whilst the double height printing routine is used after the Rom has printed to the screen. The machine code reads the bytes of colour off the screen and doubles them up. Screen character reading is done using a Rom system call.

Using machine code in this way is very attractive and is made even easier since it is possible to pass parameters from Basic to your own code and back again. These parameters are pointed to by the IX register pair with the number of parameters given in the A register.

In fact, the IX pair point to the low byte of the last parameter passed and is the base address of a parameter buffer set up for you by the system. If there is only one parameter it is available in the DE register pair and if the parameter is an integer variable preceded by @ then it is the address of the variable that is passed to the machine code. That allows values to be transferred back to Basic.

The graphics characters in the program listing may be entered most easily by entering and running the short routine at line 6000 first. This copies the Spectrum system of making the keyboard produce the appropriate graphic character when Control is pressed with one of the keys A to U.

The routine at 6040 returns the keys to normal if required. Follow the guidance of the

(continued on page 91)

(continued from page 39)

Rem statements when typing in the graphics. In addition you may find it easier to enter the machine code if you first enter Key 138,"" to change the decimal point on the numeric pad to a comma.

Finally, since the user graphics do not allow you to make the most of the Amstrad's colour,

three different ways of going this have been introduced in the screen construction routine. They may be omitted if preferred since they slow down the screen printing.

To slow down the game increase the value of the pause used at the end of the main loop — set at line 160 — or to make it much easier remove some of the traffic. Every time you ge:

four men home you will find extra spiders added and the speed will increase.

You can avoid the chore of typing in the program by sending £3 with your name and address to Death Race, 12 The Bassetts, Cashes Green, Stroud, Gloucestershire GL5 4SJ. Please make cheques payable to C J Leigh.

Listing 1.

```

ORIGIN A400H
0103A4 INITIAL LD BC,COMTAB
210A04 LD HL,BUFFER
CDD1BC CALL BCD10 ; tell the system
C9 RET
00000000BUFFER DEFS 4 ; system work space
1CA4 COMTAB DEFW NMTAB ; command table
C347A4 JP PRINT
C37BA4 JP LSCROLL
C39CA4 JP RSCROLL
C3BDA4 JP CHR
505249 NMTAB DEFM FRI ; command names
4ED4 DEFM A,T+80H
4C534352 DEFM L,S,C,R
4F4CCC DEFM C,L,L+80H
52534352 DEFM L,S,C,R
4F4CCC DEFM C,L,L+80H
4348D2 DEFM C,H,R+80H
00 DEFB 0 ; end marker
DD6E00 ADDRESS LD L,(IX+0) ; calculate screen start
DD23 INC IX
DD23 INC IX
2D DEC L ; start at line zero
2600 LD H,0
29 ADE HL,HL ; multiply by two
29 ADE HL,HL
29 ADE HL,HL
29 ADE HL,HL
E5 PUSH HL
D1 POP DE
29 ADE HL,HL
29 ADE HL,HL
19 ADD HL,DE ; times eighty
C9 RET
CD33A4 PRINT CALL ADDRESS
1103F8 LD DE,F800H ; bottom of first line
19 ADE HL,DE
EB EX HL,DE ; source
215000 LD HL,500H ; move to next line
19 ADE HL,DE
EB EX HL,DE ; destination in DE
0E02 LD C,2 ; two lines
E5 HALF PUSH HL
0604 LD B,4 ; four pixel rows
C5 ROW PUSH BC
0E02 LD C,2 ; one row becomes two
0650 STRETCH LD B,50H ; length of line
E5 PUSH HL
D5 PUSH DE
7E NEXT LD A,(HL) ; read byte
12 LD (DE),A ; write byte
23 INC HL
15 INC DE
10FA DJNZ NEXT ; next character
D1 POP DE
7A LD A,D
D608 SUB B ; up a pixel line
57 LD D,A
E1 POP HL ; recover read start
0D DEC C
20ED JRNZ STRETCH ; do it again
7C LD A,H
D608 SUB B
67 LD E,A
C1 POP BC
10E2 DJNZ ROW ; next read row
E1 POP DE ; destination = source
0D DEC C
20DC JRNZ HALF ; top half
C9 RET
F5 LSCROLL PUSH AF ; save parameter count
CD33A4 CALL ADDRESS
110CC0 LD DE,C000H ; start of top line
19 ADD HL,DE ; screen address
C60E LD B,8
C5 BACK PUSH BC
E5 PUSH HL
D1 POP DE
23 INC HL ; HL to DE
014F00 LD EC,4FH ; line length less one
1A LD A,(EE) ; save first byte
EDB0 LDIE ; move line
12 LD (DE),A ; wraparound byte
01B007 LD EC,7B0H
09 ADD HL,BC ; next pixel row
C1 POP BC
10EE DJNZ BACK
F1 POP AF
3D DEC A ; reduce parameter count
20E0 JRNZ LSCROLL ; next line?
C9 RET
F5 RSCROLL PUSH AF
CD33A4 CALL ADDRESS
114FC0 LD DE,C04FH ; end of top line
19 ADD HL,DE
0608 LD E,8
C5 AGAIN PUSH BC
E5 PUSH HL
D1 POP DE
2B EEC HL ; EL to DE
014F30 LD EC,4FH
1A LD A,(DE)
EDB8 LDIE
12 LD (DE),A
015038 LD EC,850H
09 ADD HL,BC
C1 POP BC
10EE DJNZ AGAIN
F1 POP AF
3D DEC A
20E0 JRNZ RSCROLL
C9 RET
D5 CHR PUSH DE ; save parameter address
CD60BB CALL BB60H ; read screen character
E1 POP HL
77 LD (HL),A ; put value in variable
C9 END RET
END A4C3H

```