CREATE GIOCHI ARCADE COL VOSTRO SPECTRUM

con ampie descrizioni di oltre 30 routines e 18 fantastici programmi.

di DANIEL HAYWOOD

edizioni jce

CREATE GIOCHI ARCADE COL VOSTRO SPECTRUM

con ampie descrizioni di oltre 30 routines e 18 fantastici programmi.

di Daniel Halwood

traduzione di Angelo Cattaneo

Via dei Lavoratori, 124
CINISELLO BALSAMO (MI)

Prima edizione: INTERFACE PUBLICATIONS 1983
Pubblicato in Gran Bretagna da:
Interface Publications,
44-46 Earls Court Road,
London, W8 6EJ.

Prima edizione: Giugno 1984

Stampato in Italia da:
Gemm Grafica S.r.l.
Via Magretti-Paderno Dugnano (MI)

INDICE

PREFAZIONE - Tim Hartnell

D' ora in poi non dovrete piu' sacrificare le cento lire in sala giochi;infatti, con l' aiuto dei programmi presentati in questo libro (sempre accompagnati dai chiarimenti tecnici del caso) potrete ottenere gli stessi identici effetti.
L' autore Daniel, nello stendere l' opera, presume che voi conosciate gia' le basi di programmazione dello Spectrum per cui procede spedito presentando i programmi piu' interessanti senza tralasciare pero' la descrizione del perche' venga seguita una certa strada anziche' un' altra.
Il contenuto didattico trova posto alla fine di ogni programma dove e' riportata la spiegazione dei vari blocchi che lo compongono. Ciononostante, siamo certi che il volume verra' apprezzato principalmente per il suo donare all' utente diverse ore di divertimento, lo stesso che abbiamo provato noi nel verificare i listati per la pubblicazione.

Tim Hartnell

Tim Hartnell, oltre ad essere il fondatore del Gruppo Nazionale Inglese degli ZX Users, ha anche scritto una trentina di libri sugli ZX e su altri computers.

Capitolo 1

LETTURA DELLA TASTIERA

Il riuscire a "leggere" la tastiera, e' indispensbile per poter scrivere un programma di giochi. Questo primo capitolo spiega appunto le varie funzioni legate alla tastiera e in quale modo esse agiscano.
INKEY$ rende il carattere del tasto premuto sottoforma di stringa esplorando la tastiera in una frazione di secondo. Cosi' come in altri computers, tale funzione ha una azione "continuativa", cioe' continua a leggere il carattere relativo al tasto per tutto il tempo in cui questo rimane premuto.
INKEY$ agisce anche in combinazione con i tasti di Shift rendendo i relativi caratteri maiuscoli o i relativi simboli.
CODE INKEY$, come e' intuibile, porta il codice del carattere memorizzato da INKEY$ il quale, nel caso non sia azionato alcun tasto, contiene una stringa vuota (ad es." "). In questo caso CODE INKEY$ ritorna il valore zero.
Lo Spectrum possiede anche un secondo comando molto usato : IN. Il vantaggio di "IN" rispetto ad "INKEY$" e' che puo' essere usato per leggere piu' di un tasto contemporaneamente; cosa che INKEY$ non puo' fare.
Provate a battere :

```
10 POKE 23692,-1:PRINT INKEY$;:GO TO 10
```

e rilevate cosa succede premendo due tasti contemporaneamente come potrebbe capitare durante un games di invaders. La risposta e' semplice : non succede nulla in base a quanto sopra spiegato.
La funzione di INKEY$ viene prevalentemente usata per variare le coordinate delle colonne a cui sono riferiti gli oggetti (ad esempio una base missilistica). Un sistema per effettuare dette variazioni e' quello di adottare gli statements IF...THEN, molto piu' comodi che non l' uso di operatori logici. Nella routine e' combinato anche il controllo per evitare che la base missilistica non provochi errori portandosi all' esterno dei margini dello schermo.

'p' contiene le coordinate delle colonne della base missilistica.

Versione 1: IF...THEN

```
100 IF INKEY$="8" AND p<31 THEN LET p=p+1
110 IF INKEY$="5" AND p>0 THEN LET p=p-1
```

Versione 2: Operatori logici

```
100 LET p=p+(INKEY$="8" AND
p<31)-(INKEY$="5" AND p>0)
```

INKEY$ puo' anche venir impiegato per stabilire limiti di tempo. In questo caso e' sottoposto alla funzione PAUSE la quale puo' essere interrotta con la pressione di un tasto qualsiasi per poter far riprendere al programma il suo regolare svolgimento. Una forma del genere e' usata assai spesso. Eccovi un esempio

```
  10
  20
      .
      .
      .
      .
      .
5499 REM Altra partita ?
5500 PRINT AT 5,2;"Vuoi giocare ancora ?";
     AT 7,0;"Hai 5 secondi
per rispondere s/n"
5510 PAUSE 250
5520 IF INKEY$<>"s" THEN PRINT AT 10,13;"O.K.":
     STOP
5530 RUN
```

In unione a GOTO, la funzione INKEY$ sostituisce efficacemente gli statements IF...THEN come potete verificare dalla parte finale del programma che segue

```
   10 GO TO 5000
  300 REM Programma presente
  400 PRINT "Prog..."
  450 BEEP 1,RND*60-30
 5000
 6000 REM Ultimo uomo perso
 7000 GO TO 5500
40000 REM Istruzioni
50000 REM Istruzioni presenti
```

```
5010  PRINT "Istr...."
      BEEP 1,RND*60-30
      :
      REM Inizializzazione
      REM Inizializzazione presen
te .
5610  PRINT "Iniz...."
      BEEP 1,RND*60-30
      :
      GO TO 20
      REM Partita finita
      CLS
      PRINT AT 9,0;" Vuoi le istr
uzioni  s/n  ?"
      GO TO 5610-(610 AND INKEY$=
"s")-(310 AND INKEY$="n")
```

Una particolarita' e' la sua funzione ripetitiva. Nel caso di situazioni tipo "Altra partita (s/n)?", dove "s" e "n" stanno per si o per no, INKEY$ viene usata come risposta . Esempio

```
5550  PRINT AT 7,0;"Vuoi un'altra
 partita (s/n) ?"
5560  GO TO 5560+(30 AND INKEY$="
s")+(INKEY$="n")
5570  PRINT AT 9,2;"Spero tu ti s
ia divertito.";AT 11,9;"Ciao per
 adesso."
5580  STOP
5590  REM Resto del programma....
```

E' possibile adottare una tecnica simile anche per le istruzioni :

```
5000  REM Istruzioni
      :
      REM Inizializzazione
      REM (come prima)
      :
      :
      REM Resto del programma...
      PRINT AT 9,0;"Vuoi le istru
zioni (s/n)?"
      GO TO 5610-(610 AND INKEY$=
"s")-(310 AND INKEY$="n")
```

Sfortunatamente, qui' puo' succedere che, una volta data la risposta, ad esempio "s", non si tolga abbastanza velocemente il dito dal tasto fornendo una analoga risposta, magari

errata, anche ad eventuali domande che seguano. Per evitare questo contrattempo e' sufficiente inserire la linea che segue
5590 IF INKEY$<>""THEN GOTO 5590.
Esaminiamo ora dettagliatamente la funzione IN.
Innanzitutto IN non rende una stringa bensi' un numero e poi non esplora in un sol colpo tutta la tastiera ma prende in considerazione blocchi di cinque tasti alla volta. Ogni singolo tasto sottrae un numero funzione della sua posizione nella fila dei cinque, da un secondo numero fisso caratteristico della tastiera quando non risulta azionato alcun tasto.
Prendiamo come esempio la fila coi numeri dall'1 al 5. La locazione di IN e' 63486, mentre il numero caratteristico e',di solito, 255.
Il tasto "1" sottrae 2 elevato a 0=1; per cui, nel caso sia questo l' unico tasto azionato, il numero reso sara' 255-1=254. Il "2" sottrae 2 elevato a 1=2; per cui da solo rende 255-2=253. Similmente il"3" sottrae 2 elevato a 2=4; il "4" sottrae 2 elevato alla 3=8 e "5" sottrae 2 elevato alla 4=16.
Nel caso in cui ne venga premuto piu' di uno, si avranno le varie combinazioni . In presenza di una azione simultanea su tutti e cinque i tasti, IN 63486 rendera' :
255-16-8-4-2-1=244.
Eccovi la serie completa delle locazioni cui si riferisce la funzione IN :

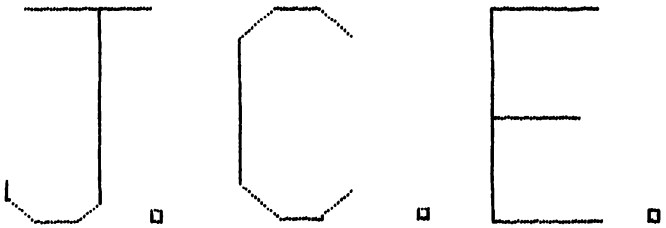
IN 63486 : tasti 1,2,3,4,5
IN 64510 : tasti Q,W,E,R,T
IN 65022 : tasti A,S,D,F,G
IN 65278 : tasti Caps Shift,Z,X,C,V
IN 61438 : tasti 6,7,8,9,0
IN 57342 : tasti Y,U,I,O,P
IN 49150 : tasti H,J,K,L,ENTER
IN 32766 : tasti B,N,M,Symbol Shift,Break/Space

I primi quattro blocchi, si trovano nella meta' sinistra della tastiera ed i relativi valori vanno considerati in ordine di importanza man mano che si va verso l' interno, per cui i tasti verso il bordo hanno valore 1 e quelli posti al centro 16. Per i rimanenti quattro blocchi, il discorso, ovviamente, si inverte.
Per essere piu' chiari, la cifra 1 viene sottratta alla pressione dei tasti "1","Q","p" e "0" come invece la 16 riguarda "G","5","6" e "B".
Un handicap di IN sta' nel fatto che il numero caratteristico non sempre e' 255, per cui possono venir resi anche numeri errati.
Un metodo attendibile per riportare ogni volta il valore caratteristico a 255, e' quello di inserire un BEEP prima della lettura della tastiera.
In linea di massima, e' piu' semplice usare INKEY$ anche se in alcuni casi particolari, come appunto l' azionamento simultaneo di due tasti, la funzione IN si dimostra indispensabile.
Per verificare la versatilita' di IN, provate a far girare il programma che segue riguardante un semplice "Sketchpad" che vi

abilitera' a disegnare sullo schermo. Se possedete una stampante, potrete in ogni momento dare un BREAK al programma e ricavare una copia del quadro tramite il comando COPY.

```
   10  GO SUB 1000
   20
   25  BEEP .1,10
   30
   35  IF m=1 THEN OVER 1
   40  PLOT INK ink;x,y: LET a=x:
LET b=y
   45  REM SU l'issue 3 variare
   46  REM 127 in 255, 169 in 255,
   47  REM 190 in 254, 191 in 255
   50  LET x=x+(IN 65502=190 AND x<1
<1900)-(IN 65502=191 AND x>1)
   60  LET y=y+(IN 64510=189 AND y
<166)-(IN 65278=87 AND y>1)
   70  PLOT INVERSE (m=2); INK ink
; a,b
   80  LET a$=INKEY$
   90  IF a$>"0" AND a$<="7" THEN
LET ink=VAL a$
  100  IF a$="8" THEN PRINT AT 0,m
*6; FLASH 0; OVER 1; INVERSE 0;"
";: LET m=m+1-(3 AND m=22): P
RINT AT 0,m*6; FLASH 1; OVER 1;
INVERSE 0;"
  110  OVER 0: GO TO 30
  120
1000  LET x=127: LET y=87: LET in
k=0: LET m=0
1010  BORDER 0: PAPER 7: INK 0: C
LS
1020  PRINT '" Movimenti:"'" W-s
u";" X-giu ";" A-sin ";" D-dest "
1030  PRINT '" Combinazioni diago
nali
1040  PRINT '" Premi un tasto."
1050  IF INKEY$<>"" THEN GO TO 10
50
1060  IF INKEY$="" THEN GO TO 105
0
1070  CLS
1080  PRINT FLASH 1;"DIS. "; FLAS
H 0;" MUOV. CANC."
1090  PLOT 0,167: DRAW 191,0
1100  RETURN
```

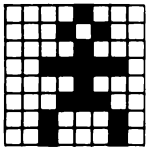
DIS. MUOV. CANC.



Capitolo 2

GRAFICA DEFINITA DALL' UTENTE

Una delle peculiarita' dello Spectrum e' la possibilita' di creare facilmente nuovi caratteri. Ogni carattere, disposto in una griglia 8*8 pixel, e' formato dalla combinazione di 64 punti ciascuno dei quali puo' essere "pieno" o "vuoto" (settato o resettato) a seconda che debba essere visibile o meno.
Considerate l' omino che segue :



Come si puo' notare, e' composto in un array (matrice) di 8*8 punti. Il primo problema da risolvere e' come inserirlo nel computer e visto che questo accetta solo numeri, vediamo di eseguire la trasformazione necessaria.
Il carattere viene definito da 8 numeri : uno riferito alla prima riga, un secondo riferito alla seconda riga e cosi' via.
Il comando interessato e' BIN, forma tronca che sta per "binary", accompagnato da un numero binario formato da tanti zeri quanti sono gli spazi e da tanti 1 quanti sono i punti.
Gli otto numeri dell' omino di cui sopra sono quindi :

```
I    riga - BIN 00001000
II   riga - BIN 00010100
III  riga - BIN 00001000
IV   riga - BIN 00111110
V    riga - BIN 00001000
VI   riga - BIN 00011100
VII  riga - BIN 00100010
VIII riga - BIN 00100010
```

Il manuale riporta che BIN 00000000 puo' venir sostituito da 0 il quale e', evidentemente il suo equivalente decimale, ma non dice che anche qualsiasi altro numero binario ha un suo sostituto in decimale, per cui, se preferite operare in decimale, eseguite ogni volta la conversione b/d. Se andate di fretta, non avrete da far altro che battere il comando diretto PRINT BIN seguito dal numero binario per ricevere all'istante dal computer l'equivalente decimale. Se invece avete un attimo di tempo, soffermatevi a leggere quanto segue sul come ricavarli a mano.

Considerate il decimale 3287, lo possiamo scrivere anche come sottoriportato tenendo presente che il segno "↑" sta' per "elevato alla potenza di" :

```
3*10↑3 (1000)  = 3000
2*10↑2 (100)   =  200
8*10↑1 (10)    =   80
7*10↑0 (1)     =    7
               -------
                 3287
```

E' possibile cioe' scinderlo, moltiplicandolo termine per termine per una potenza di dieci in virtu' del fatto che tale e' la base dei numeri decimali.
Lo stesso discorso vale anche per i numeri binari la cui base e' ovviamente 2. Ad esempio :

```
1*2↑7 (128)  = 128
0*2↑6 (64)   =   0
0*2↑5 (32)   =   0
1*2↑4 (16)   =  16
1*2↑3 (8)    =   8
1*2↑2 (4)    =   4
0*2↑1 (2)    =   0
1*2↑0 (1)    =   1
             ------
               157
```

Altro modo per ottenere la conversione, e' quello di consultare la tabella sotto riportata. I numeri binari con gli 1 nelle posizioni d' inizio sono quelli di valore piu' alto e li troverete verso la fine della lista

```
10 FOR i=0 TO 255: LET j=i
20 LET b$="00000000"
30 FOR n=7 TO 0 STEP -1: IF 2↑
n<=j THEN LET b$(8-n)="1": LET j
=j-2↑n
40 NEXT n: PRINT b$;" --- ";i
50 NEXT i
```

```
00000000  ---  0
00000001  ---  1
00000010  ---  2
00000011  ---  3
00000100  ---  4
00000101  ---  5
00000110  ---  6
00000111  ---  7
00001000  ---  8
00001001  ---  9
00001010  ---  10
00001011  ---  11
00001100  ---  12
```

La grafica viene generata con un loop in cui il numero di ogni riga viene letto (READ) da uno statement di DATA. La routine usata piu' frequentemente e'

```
9000 RESTORE 9100
9002 FOR n=0 TO 7
9003 READ a
9004 POKE USR "a"+n,a
9005 NEXT n
9007 RETURN
9100 DATA 8,20,8,62,8,28,34,34
```

In questo specifico caso si definisce "A" con la sagoma dell' omino di prima.
Volendo definire piu' di un carattere, e' necessario apportare una modifica al programma aggiungendo le linee che seguono e inserendo i dati dalla linea 9010 in avanti. I caratteri vanno definiti dalle lettere UDG che vanno dalla "A" alla "U".

```
9010 FOR i=1 TO n: REM [n' e' il numero dei
     caratteri da definire]
     .
     .
9040 POKE USR CHR$(i+143)+n,a
     .
9060 NEXT i
```

La subroutine usata d' ora in avanti da questo libro e'

```
9000 FOR i=1 TO n
9001 FOR n=0 TO 7
9002 READ a
9003 POKE USR CHR$ (i+143)+n,a
9004 NEXT n
9005 NEXT i
9007 RETURN
9100 DATA n1,n2,n3,n4,n5,n6,n7,n
     8
9110 DATA n1,n2,n3,n4,n5,n6,n7,n
     8
```

Per rendere piu' veloce e piu' agevole la vostra programmazione, ecco una lista delle sagome grafiche usate piu' correntemente con i relativi numeri di DATA.

Per primi, gli invaders "statici", che non devono girarsi, saltare o camminare ma solo fare scena.

```
DATA 129,90,60,90,60,102,153,195

DATA 129,66,60,219,165,24,36,195

DATA 195,165,24,60,90,36,24,231

DATA 129,102,60,255,90,255,129,231

DATA 60,126,219,219,255,126,66,102
```

Per gli invaders in movimento scrivere prima una versione e quindi l' altra presentandole alternativamente :

```
DATA 60,126,235,235,255,126,36,231

DATA 60,126,215,215,255,126,66,102

DATA 66,36,189,173,255,60,36,231

DATA 36,36,60,247,189,189,36,36

DATA 129,189,189,173,239,60,36,231

DATA 231,165,189,181,52,255,129,231

DATA 0,24,60,90,126,24,36,90

DATA 60,90,126,60,24,36,90,0

DATA 198,34,20,42,28,34,17,230

DATA 99,68,40,84,56,68,136,103
```

Ora gli "alieni", ben conosciuti dagli appassionati di giochi spaziali. Con questo sistema l' effetto si crea presentando un carattere alternato ad altri due :

```
DATA 60,126,255,170,170,255,126,60

A sinistra :
DATA 3,7,15,10,10,15,7,3

A destra   :
DATA 192,224,240,160,160,240,224,192
```

Altro metodo, e' quello di stampare gli alieni (tutti lunghi non piu' di due caratteri) di seguito nell' ordine. Si ottiene un effetto assai realistico specialmente a velocita' sostenute.

Primo "alieno"

```
Sinistra  : DATA 63,127,255,73,73,255,127,63
Destra    : DATA 252,254,255,36,36,255,254,252
```

Secondo "alieno":

```
sinistra  : DATA 63,127,255,36,36,255,127,63
Destra    : DATA 252,254,255,146,146,255,254,252
```

Terzo "alieno"

```
Sinistra    DATA 63,127,255,146,146,255,127,63
Destra      DATA 252,254,255,73,73,255,254,252
```

Basi missilistiche

```
            DATA 24,24,24,60,126,255,219,219

            DATA 0,24,24,24,255,255,255,255

            DATA 24,24,24,60,255,255,255,0

Sinistra    DATA 1,1,7,9,31,127,255,255
Destra      DATA 0,0,192,32,240,252,254,254
```

Vediamo qualche astronave

```
Sinistra  : DATA 0,252,32,33,18,127,15,1
Destra    : DATA 0,0,240,8,4,228,255,248

Sinistra  : DATA 31,32,79,255,255,79,32,31
Destra    : DATA 0,0,60,235,235,60,0,0

Sinistra  : DATA 0,248,32,16,15,63,15,3
Destra    : DATA 0,0,0,224,16,8,207,248
```

Alcuni asteroidi assomigliano ad astronavi. Sono formati da due disegni uno per la sagoma orizzontale-verticale, l' altro per quella diagonale.

```
Rivolto verso nord-ovest :
            DATA 192,176,76,67,44,40,16,16
Rivolto verso ovest :
            DATA 0,7,25,98,132,98,25,7,
Rivolto verso sud-ovest :
            DATA 16,16,40,44,67,76,176,192
Rivolto verso sud :
            DATA 198,170,146,68,68,40,40,16
Rivolto verso sud-est :
            DATA 8,8,12,44,194,50,13,3
```

```
Rivolto a est :
            DATA 0,224,152,70,33,70,152,224
Rivolto a nord-est :
            DATA 3,13,50,194,44,12,8,8
Rivolto a nord :
            DATA 16,40,40,68,68,146,170,198

I prossimi grafici sono aerei.

Un tipo di Jet
Sinistra  : DATA 0,0,224,112,120,255,60,15
Destra    : DATA 0,0,0,0,224,248,30,248

Un modello normale :
Sinistra  : DATA 192,192,227,158,127,0,0,0
Destra    : DATA 0,228,244,60,244,36,80,32

Un secondo tipo di Jet
Sinistra  : DATA 14,199,224,255,127,1,0,0
Destra    : DATA 0,0,96,248,252,224,240,112


Un modello particolare quale usato nel programma "Olocausto".

Sinistra  : DATA 192,224,240,248,143,255,15,1
Centrale  : DATA 0,7,9,17,255,255,224,255
Destra    : DATA 0,192,32,16,254,255,30,248


Le prossime otto file di dati si riferiscono ad un aereo
inclinato in ognuna dello otto direzioni della rosa dei venti.


Verso nord :
            DATA 8,8,28,62,127,8,8,28

Verso nord-est :
            DATA 1,2,124,60,28,172,68,32

Verso est :
            DATA 16,24,156,255,156,24,16,0

Verso sud-est :
            DATA 32,68,172,28,60,124,2,1

Verso sud :
            DATA 28,8,8,127,62,28,8,8

Verso sud-ovest :
            DATA 4,34,53,56,60,62,64,128

Verso nord-ovest :
            DATA 128,64,62,60,56,53,34,4
```

Se preferite stare con i piedi per terra, eccovi una sequenza di carri armati e di navi.

```
Sinistra    DATA 0,16,31,31,63,127,63,0
Destra      DATA 0,0,252,0,240,248,240,0


Sinistra    DATA 0,0,63,0,15,31,15,0
Destra      DATA 0,8,248,248,252,254,252,0

DATA 0,0,64,63,112,126,255,126
DATA 0,0,20,252,14,126,255,126
```

Nave da guerra rivolta a sinistra

```
Sinistra    DATA 0,0,2,115,23,255,126,63
Destra      DATA 192,192,224,239,236,255,255,254
```

ora rivolta a destra

```
Sinistra    DATA 3,3,7,247,55,255,255,127
Destra      DATA 0,0,64,204,232,255,254,252
```

Ecco ora i dati per ottenere un carro armato direzionato in vari modi.

Verso nord :

```
            DATA 8,73,73,93,127,93,65,65
```

Verso nord-est :

```
            DATA 33,66,188,60,60,29,2,4
```

Verso est :

```
            DATA 254,16,56,63,56,16,254,0
```

Verso sud-est :

```
            DATA 4,2,29,60,60,188,66,33
```

Verso sud :

```
            DATA 65,65,93,127,93,73,73,8
```

Verso sud-ovest :

```
            DATA 32,64,184,60,60,61,66,132
```

Verso ovest :

```
            DATA 127,8,28,252,28,8,127,0
```

Verso nord-est :

```
            DATA 132,66,61,60,60,184,64,32
```

Concludendo la selezione dei semoventi terrestri ecco qualche auto :

```
In alto a sinistra : DATA 31,31,4,5,55,59,55,6
In alto a destra   : DATA 248,248,32,160,236,220,236,96

In basso a sinistra: DATA 6,6,7,19,31,19,1,7
In basso a destra    DATA 96,96,224,200,248,200,128,224

Sinistra : DATA 224,159,119,239,239,119,159,224
Destra   : DATA 28,242,190,191,191,190,242,28

DATA 0,66,189,239,239,189,66,0

DATA 102,153,239,223,223,239,153,102

Lanciamo le bombe

DATA 32,16,160,92,30,31,15,7

DATA 54,28,8,28,62,62,28,8
```

Qui' di seguito trovate alcuni "pacman" disegnati in modo che ognuno dei primi quattro si alterni col quinto in funzione della propria direzione.

```
Pacman rivolto a destra
DATA 60,127,252,240,240,252,127,60

Pacman rivolto in su'
DATA 66,66,231,231,255,255,126,60

Pacman rivolto a sinistra
DATA 60,254,63,15,15,63,254,60

Pacman rivolto in giu'
DATA 60,126,255,255,231,231,66,60

Pacman con la bocca chiusa
DATA 60,126,255,255,255,255,126,60

Spettro
DATA 56,124,214,214,254,254,170,170


 Ora un po' di frutta


Fragola  : DATA 24,82,247,255,255,126,60,24
Lampone  : DATA 4,8,8,86,171,213,106,60
Ciliegia : DATA 8,8,20,20,34,99,243,96
Pesca    : DATA 44,110,231,247,247,247,102,44
Banana   : DATA 2,3,7,14,30,124,248,0
Prugna   : DATA 8,16,24,60,124,62,60,24
Mela     : DATA 24,82,255,255,255,255,126,36
```

Per concludere la parte della grafica definita dall' utente, presentiamo una serie alternativa di numeri riferiti a forme spesso usate in fasi particolari dei giochi TV.

Un esempio e' il punteggio, inserito in una variabile di stringa con STR$ e quindi scelto per mezzo di un loop tra i vari caratteri, l' ultimo dei quali determinato da LEN. Il codice (CODE) di ciascun carattere viene determinato aggiungendo il numero corrispondente alla differenza tra il CODE del digit interessato e il CODE della grafica stessa definita dall' utente.

Ma poiche' gli esempi valgono da sempre piu' delle parole battete la routine che segue. La variabile "S" contiene il punteggio mentre gli user-graphic da 0 a 9 derivano dalle lettere da A e J impostate in "graphic mode". La variabile S$ contiene il punteggio da ricavare dalla serie di numeri.

```
4000 LET s$=STR$ s
4010 FOR i=1 TO LEN s$
4020 LET s$(i)=CHR$ (CODE s$(i) +
95)
4030 NEXT i
4040 RETURN
```

Eccovi i grafici :

```
0 DATA 0,60,66,66,98,98,98,60
1 DATA 0,8,8,8,12,12,12,12,
2 DATA 0,124,2,2,60,96,96,62
3 DATA 0,124,2,2,60,6,6,126
4 DATA 0,64,96,100,100,126,4,4
5 DATA 0,62,64,64,60,6,6,126
6 DATA 0,62,64,64,124,70,70,126
7 DATA 0,120,8,8,12,12,12,12
8 DATA 0,60,66,66,60,70,70,60
9 DATA 0,60,66,66,60,6,6,126
```

Dopo aver esaminato alcuni possibili caratteri, vediamo il programma sotto riportato che vi permettera' di effettuare la definizione per conto vostro :

```
9500 PRINT TAB 7;"Generatore di
caratteri"
9510 PRINT '"per creare grafici
definiti dall'utente su Una matr
ice di 8x8"
```

```
9520 PRINT '"(Usa i tasti cursor
e,ad es. 5-8 per il movimento de
l cursore)"
9550 PRINT '"Adesso premi un tas
to qualsiasi"
9560 PAUSE 0
9570 CLS : INPUT "Che lettera pe
r il carattere da definire (a-u)
 ?";z$
9580 LET z$=(z$+" ")(1): IF z$<"
a" OR z$<"u" THEN GO TO 9570
9590 CLS : PRINT AT 12,0;"1-Lasc
ia"' ,"Cursori (5-8)"' ,"9-Nero"' ,
"0-Cancella"
9600 FOR x=95 TO 159 STEP 8
9610 PLOT 16,x: DRAW 64,0
9620 PLOT x-79,95: DRAW 0,64: NE
XT x
9630 LET x=2: LET y=2: OVER 1: I
NK 9: PAPER 9
9640 PRINT AT x,y;"x"
9650 LET x1=x: LET x=x+(INKEY$="
6" AND x<9)-(INKEY$="7" AND x>2)
9660 LET y1=y: LET y=y+(INKEY$="
8" AND y<9)-(INKEY$="5" AND y>2)
9670 IF INKEY$="0" THEN PRINT AT
 x,y; PAPER 7; INK 0;" "
9680 IF INKEY$="9" THEN PRINT AT
 x,y; PAPER 0; INK 7;" "
9690 PRINT AT x1,y1;"x"
9700 IF INKEY$="1" THEN GO TO 97
20
9710 GO TO 9640
9720 OVER 0: PRINT AT 11,0;"DATI
": FOR n=1 TO 9: LET a=0
9730 FOR z=2 TO 9
9740 IF ATTR (n,z)=7 THEN LET a=
a+(2↑(9-z))
9750 NEXT z: POKE USR z$+(n-2),a
9760 PRINT a;",";
9770 NEXT n: PRINT CHR$ 8;" "
9780 PRINT '"Il tuo carattere e'
 fatto cosi'":''TAB 15;CHR$ (COD
E z$+47)
9790 PRINT AT 17,0;"Premi"'"1-pe
r definire il carattere"'"2-per
unire i blocchi"'"3-stop"
9795 IF INKEY$<>"" THEN GO TO 97
95
9800 GO TO 9800-(230 AND INKEY$=
"1")+(INKEY$="2")+(199 AND INKEY
$="3")
9810 CLS : PRINT "Premendo il ta
sto -graphic sara'";TAB 12;"stam
pato."''"I tasti direzione muovo
no entro il box."''TAB 10;"1-Las
cia"''TAB 6;"SPACE-Cancella"
9820 PRINT ''''''"Premi qualsias
i tasto": PAUSE 0: PRINT AT 9,0;
"
```

```
9830 PLOT 102,97: DRAW 36,0: DRA
W 0,-36: DRAW -36,0: DRAW 0,36
9840 LET x=10: LET y=13: OVER 1
9850 LET a$=INKEY$: PRINT AT x,y
;"■": IF a$="" THEN PRINT AT x,
y;"■": OVER 0;
9860 IF a$>="a" AND a$<="u" THEN
 PRINT AT x,y;CHR$ (CODE a$+47)
9870 LET x1=x: LET x=x+(a$="6" A
ND x<13)-(a$="7" AND x>10)
9880 LET y1=y: LET y=y+(a$="8" A
ND y<18)-(a$="5" AND y>13)
9890 PRINT AT x1,y1;"■"
9900 IF a$="1" THEN GO TO 9790
9910 GO TO 9850
9995 STOP
```

Come conclusione del capitolo, eccovi un semplice gioco impiegante qualcuno degli UDG sopra descritti. Dal vostro aereo in volo, dovrete colpire con una bomba, una casetta bianca lampeggiante. Lo sgancio avviene premendo il tasto "0".

Elenco delle variabili

P	Variabile di controllo dell' aereo
X	Variable della colonna dell' aereo
bomb	Per verificare se la bomba e' caduta o no
a$	Valuta la decisone del giocatore
yes	Se la bomba ha colpito la casa
R$	Stringa con la disposizione del suolo
r	Usate per creare gli oggetti al suolo
a	
r	Colonna della casa
h	Punteggio
i	Usato nel loop per la definizione dei caratteri e per il lancio della bomba
n	Controllo dei vari loops
SET UP, SCREEN, DROP	Variabili "tokens" delle subroutines.

I caratteri grafici definiti in questo gioco sono :

linea 40	graphic B,C,D.
linea 4110	graphic E.
linea 4500	graphic A.
linea 4520	graphic F,G,H.
linea 4520	graphic I,J,K.

I PRINT grafici inseriti nello statement della linea 4130 sono messi a disposizione dai tasti da 1 a 8.

```
10 GO TO 5000: REM INIT
20 FOR p=1 TO 10: REM PLANE
```

```
  30 GO SUB SET UP
  40 FOR x=1 TO 28: PRINT AT 4,x
; INK 6;" BCD"
  50 IF INKEY$="0" AND bomb THEN
 GO SUB DROP
  60 BEEP .05,(x AND bomb)+(RND*
40 AND yes)-(10 AND NOT yes)
  70 NEXT x
  80 PRINT AT 4,28;"    "
  90 NEXT p
 100 FOR i=1 TO 100: NEXT i
 110 CLS
 120 PRINT AT 5,1;"Tuoi punti ";
h;" dopo 10 voli"
 130 LET a$=("Eccellente" AND h=
10)+("Ottimo" AND h>7 AND h<10)+
("Buono" AND h>5 AND h<8)+("Medi
o" AND h>3 AND h<6)+("Sotto alla
 media" AND h>1 AND h<4)+("Allen
ati" AND h<2)
 140 PRINT AT 7,16-(LEN a$/2);a$
 150 PRINT AT 9,4;"Ancora ? Prem
i ";AT 11,8;"S (si) o N (no)"
 160 GO TO 160+(INKEY$="n")+(514
0 AND INKEY$="s")
 170 CLS : PRINT AT 11,13; FLASH
 1;"CIAOO"
 1999 STOP
 2999 REM ********INIZIO*********
 3000 LET bomb=1
 3010 GO SUB SCREEN
 3020 LET yes=0
 3100 FOR n=1 TO 100: NEXT n
 3190 RETURN
 3999 REM *******SCHERMO********
 4000 LET l$=""
 4010 FOR i=1 TO 32: LET r=INT (R
ND*8+155): LET a=4+(INT (RND*5-2
) AND r<>155 AND r<>156)
 4020 LET l$=l$+CHR$ 16+CHR$ a+CH
R$ r: BEEP .005,i: NEXT i
 4030 LET r=INT (RND*20+7)
 4100 BORDER 5: PAPER 1: INK 7: C
LS
 4110 PRINT AT 17,0;l$: PRINT AT
17,r; PAPER 0; FLASH 1;"E": PRIN
T AT 19,0;"E' il volo ";p;AT 21,
0;"Risultato ora ";h
 4120 PRINT AT 19,20;"PREMI "; FL
ASH 1;"0": PRINT AT 20,20;"E SG
ANCIA": PRINT AT 21,20;"LA BOMBA
"
 4130 PRINT AT 0,0;"
```

```
"
 4190 RETURN
 4499 REM *****CADUTA BOMBE*****
 4500 FOR i=5 TO 17: PRINT AT i,x
+2;"A"
```

```
4510 BEEP .02,50-i*2: PRINT AT i
,x+2;" ": NEXT i
4520 PRINT AT 16,x+1;"FGH": PRIN
T AT 17,x+1;"IJK"
4530 LET bomb=0: LET yes=0
4540 IF x=r-2 THEN LET h=h+1: LE
T yes=1: PRINT AT 21,15;h
4550 RETURN
4999 REM ****INIZIALIZZAZIONE**
5000 GO SUB 9000: REM UDG
5010 LET SET UP=3000
5020 LET SCREEN=4000
5030 LET DROP=4500
5040 LET h=0
5050 GO TO 20
8999 REM ***GRAFICA UTENTE*****
9000 RESTORE 9100
9010 FOR i=1 TO 19: BEEP .01,60-
i*3: FOR n=0 TO 7
9020 READ a: POKE USR CHR$ (i+14
3)+n,a: NEXT n: NEXT i
9030 RETURN
9100 DATA 54,28,8,28,62,62,28,8
9110 DATA 192,224,240,248,143,25
5,15,1
9120 DATA 0,7,9,17,255,255,224,2
55
9130 DATA 0,192,32,16,254,255,30
,248
9140 DATA 8,28,62,126,73,73,126,
[illegible]
9150 DATA 0,3,15,63,63,63,30,0
9155 DATA 0,255,255,255,255,255,
126,60
9160 DATA 0,192,240,252,252,252,
120,0
9170 DATA 1,32,24,6,0,31,0,255
9175 DATA 60,153,90,90,24,189,60
,255
9180 DATA 128,4,24,96,0,246,0,25
5
9190 DATA 8,28,42,73,28,42,8,255
9200 DATA 8,28,28,62,62,28,8,255
9210 DATA 7,56,56,56,198,198,198
,255
9220 DATA 252,196,199,255,252,60
,60,255
9230 DATA 3,63,59,19,59,58,18,25
5
9240 DATA 7,7,7,7,122,122,122,25
5
9250 DATA 0,254,246,240,78,78,78
,255
9260 DATA 248,248,71,71,71,249,2
49,255
```

Note	
10	Inizio
20	Loop per aereo
30	Predispone l' aereo successivo
40-70	Loop per muovere l' aereo
80-100	Aereo successivo
170	Fine della gara
3000-3190	Predispone la corsa dell'aereo successivo
4000-4020	Disegna il suolo
4030	Posizione della casa
4100-4190	Disegna lo schermo
4500-4510	Sgancia la bomba
4520	Esplosione
4530-4540	Setta le variabili di "yes" e"bomb"
5000-5490	Inizializzazione
9000-9030	Grafica definita dall' utente
9100-9260	Dati per la grafica.

Capitolo 3

SUONI

Per produrre i suoni lo Spectrum usa il comando BEEP. Il formato dello statement e' BEEP d,n in cui "d" e' il valore numerico della durata della nota e "n" la sua tonalita'. Il principale vantaggio di tale comando e' l' estrema facilita' d' uso. Pur essendo costante, la qualita' del suono, permette la composizione di effetti assai gradevoli. Il maggior svantaggio e' che, durante l' esecuzione dei suoni, risulta inibita ogni altra operazione con conseguente rallentamento dell' intero programma. Per tale motivo, il suo uso viene ristretto in listati relativamente lunghi.
BEEP puo' venire usato in concomitanza con gli statement di READ e DATA per comporre motivetti coi quali accompagnare i games.
Provate quelli che seguono

```
1000 RESTORE
1010 FOR i=1 TO 7
1020 READ durata,nota
1030 BEEP durata,nota
1040 NEXT i
1100 DATA .2,7,.1,2,.1,2,.2,4,.4
,2,.2,6,.2,7
```

E' altresi' possibile ottenere ottimi risultati impiegando dei loop FOR...NEXT come nei due esempi qui' di seguito :

```
1000 FOR n=10 TO -10 STEP -.5
1010 BEEP .02,n
1020 NEXT n
```

```
1000 FOR n=10 TO -10 STEP -.5
1010 BEEP .02,n
1020 NEXT n
1030 FOR n=-10 TO 10 STEP .5
1040 BEEP .02,n
1050 NEXT n
```

Se vi siete divertiti, battete allora anche questo noto motivetto :

```
1000 LET n=-10: LET m=1
1010 LET m=m-m+(NOT m)
1020 LET n=n+5*m-m-(50 AND n>40)
1030 BEEP .1,n
1040 GO TO 1020
```

Volendo, potete anche dotare il vostro Spectrum di un box sonoro (ottimo quello della Tenkolek) per amplificare il segnale presente sulla presa di EAR. Una buona amplificazione si ottiene anche usando da buffer il registratore stesso. Per far cio', collegare un cavetto tra l' uscita EAR dello Spectrum e l' ingresso MIC del registratore dopodiche' dare il "play".

Capitolo 4

USO DEGLI OPERATORI LOGICI

Gli operatori logici sono :
AND;OR;NOT;=;<>(diverso da);<;>;<=;>=
Tali funzioni vengono completate dalle espressioni matematiche +,-,* (moltiplicato) e / (diviso). Gli operatori logici possono sostituire IF...THEN come nell' espressione IF A=D THEN LET C=1 equivalente a "A=B". Se l' espressione e' vera, cioe' se 'A' e' realmente uguale a 'B', l' azione prosegue attribuendo, nel nostro caso, a 'C' il valore 1. Analogamente agisce LET C=(A=B), il cui nucleo e' "(A=B)", le parentesi non sono necessarie all' atto pratico ma chiariscono maggiormente il concetto. Il computer valuta anche qui l' uguaglianza e se questa risulta vera, rende a C il valore 1, viceversa, cioe' nel caso in cui A sia diverso da B, rende 0. Facciamo un esempio, supponete che tanto A quanto B valgano 5 :

```
C=(A=B)          (oppure C=A=B)
C=(5=5)
E' 5 uguale a 5 ? Si, per cui
C= VERO
C= 1
```

Si nota chiaramente come, l' espressione logica, operi alla stregua degli statement IF...THEN con la sola differenza che, questi mantengono il precedente valore della variabile e non la pongono a 0 come invece succede usando gli operatori logici se questa risulta falsa.
Il segno "=" non funge solamente da comparatore

```
IF A<>B THEN LET C=1

LET C=(A<>B)


IF A<B THEN LET C=1

LET C=(A<B)


IF A>B THEN LET C=1

LET C=(A>B)
```

```
IF A<=B THEN LET C=1

C=(A<=B)


IF A>=B THEN LET C=1

C=(A>=B)
```

Impiegando espressioni matematiche al posto degli operatori logici,ne deriva che tutte le grandezze diverse da 0 vengono considerate vere e tutte quelle uguali a 0, false. I segni matematici possono essere inseriti in statement di IF...THEN come nella linea : IF A=B THEN LET C=1. In questo caso l' espressione A-B viene considerata esclusivamente numero, che se diverso da 0, rende vera l' espressione permettendo l' esecuzione del comando che segue THEN nella fattispecie LET C=1. Puo' sorgere il dubbio che l' espressione A-B, cosi' impiegata, equivalga a A<>B, ma cio' non e' assolutamente vero come capirete battendo il programmino che segue in cui le forme interessate sono evidenziate in carattere inverso per una maggior chiarezza :

```
  10 LIST
  20 LET A=15
  30 LET B=5
  40 IF A<>B THEN LET C=1: GO SU
B 100
  50 IF A=B THEN LET C=1: GO SUB
 100
  60 LET C=(A<>B): GO SUB 100
  70 LET C=(A=B): GO SUB 100
  80 STOP
 100 PRINT "C=";C
 110 RETURN
```

Il risultato e' la dimostrazione che A-B possiede un valore puramente matematico e non logico.
Per esaminare i corrispondenti operatori logici dei segni matematici, dobbiamo prendere in considerazione le funzioni AND, OR e NOT. Anche queste possono trovare posto in statement IF...THEN:

```
IF A=5 AND B=6 THEN...

IF A OR B=2 THEN...
IF A<>0 OR B=2 THEN...

IF NOT A=B THEN...
IF A<>B THEN...
```

qui notiamo che la seconda e la terza linea riguardano la stessa funzione come pure la quarta e la quinta. Il loro impiego permette di inserire espressioni in statement relativi a "operazioni logiche". Nell'esempio,LET C=(A=3 AND B=2), il valore di C e' 1 solamente se A e' uguale a 3 e B e' uguale a 2. La funzione OR raggiunge lo stesso risultato con l' aiuto di IF...THEN. NOT invece, inverte il valore dell' espressione permettendo l' esecuzione sia nel caso di risultato vero (1), sia nel caso di risultato falso (0). In altre parole, trasforma gli "1" in "0" e gli "0" in "1".
Visto cio' e' ora possibile comprendere l' uso dei segni +, * e / come operatori logici. Se consideriamo le espressioni facenti parte di IF (espressione) THEN... , se ne deduce che :

"A-B" equivale a "A<>B"
"A+B" equivale a "A OR B". Notate pero' che se A vale 5 e B vale -5, il risultato della somma e' 0 anche se sono entrambi veri.
"A*B" equivale a "A AND B"
"A/B" equivale ad A stesso, a meno che A sia talmente piccolo rispetto a B che il risultato si approssimi a 0, rivelandosi in tal caso falso.
Gli esempi portati finora per A e B valgono naturalmente anche per le stringhe A$ e B$.
Come esempio, supponiamo ora di voler "simulare" l' espressione IF A=B THEN LET C=3 con operatori logici. La forma equivalente risultera' :

LET C=(X AND espressione)

dove "X" e' il numero (o la stringa) e "C" la grandezza a cui viene attribuito il valore "X" se l' espressione e' vera. Nel caso in cui risulti falsa, "C" varra' 0.
Impiegando l' espressione di cui sopra si otterra'

LET C=(3 AND A=B)

Anche la funzione OR puo' assumere una tale configurazione, naturalmente gli argomenti saranno diversi :

LET C= (x OR espressione)

"X" e' sempre solo un numero in quanto con OR non si possono usare stringhe. Il valore di C in questo caso risulta 1 con espressione vera e "X" con espressione falsa.
A dimostrazione di cio', fate girare il programma che segue mettendo in pratica quanto avete imparato dal capitolo e analizzando i risultati.

```
  10 LIST : LET A=5
  20 INPUT "Inserisci B come 0 o
1 (non-zero)  ";B: PRINT "B=";B
  30 LET C=(A=B): PRINT C
 400 LET C=NOT (A=B): PRINT C
 500 LET C=(B AND A=B): PRINT C
 600 LET C=(A OR B): PRINT C
 700 LET C=(A OR NOT B): PRINT C
 800 LET C=NOT (A OR B): PRINT C:
 900 PRINT ("EUREKA ?" AND B); ("
SEMPLICE" AND NOT B)
```

Nell' impiegare i comandi sopra descritti, tenete ben presente che, nell' eseguirli, il computer valuta la priorita' di uno nei confronti dell' altro; il cui grado di e' mostrato nella lista che segue

"NOT" - Priorita' 4
"AND" - Priorita' 3
"OR" - Priorita' 2

da cui deriva che i NOT sono eseguiti per primi, seguiti dagli AND ed infine dagli OR. La priorita' 5 e' assegnata alle espressioni. Pertanto, incontrandone una del genere, NOT A=B, il computer esegue prima A=B e poi la inverte col NOT. Pensate ad un esempio in cui una astronave aliena di due caratteri, dall' altezza fissa si debba spostare attraverso allo schermo su di una riga ben specifica. Supponiamo che A sia la variabile che si incrementa per muovere la nave da sinistra a destra e che la sagoma sia disegnata sullo schermo con PRINT AT h,A;"astronave" in cui "h" stabilisce l' altezza costante del volo. Sulla parte bassa dello schermo, e' presente una base missilistica lunga un carattere la cui colonna e' memorizzata nella variabile B, mentre S contiene il punteggio (valutato 10 per ogni singolo alieno) e gli spari del giocatore. Con tutti questi elementi potete gia' stendere una linea di programma per incrementare il punteggio :

```
IF B=A OR B=A+1 THEN LET S=S+10
```

Usando gli operatori logici

```
LET S=S+(10 AND A=B OR B=A+1)
```

Dall' analisi vediamo che A=B e' una espressione che, se vera, conferisce al blocco 10 AND A=B il risultato 10 che viene poi comparato col blocco B=A+1 il quale deve essere per forza falso (infatti A=B=vero). Si ottiene in questo caso il test 10 OR 0 che porta al risultato 10.
Al contrario, se A=B risulta falso e B=A+1 vero, sta a significare che il giocatore ha colpito l' alieno di fronte, ma in questo caso il calcolo del punteggio S viene fatto in modo errato, vediamo :

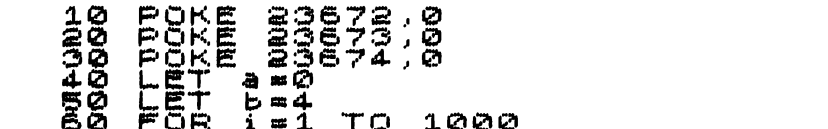
```
1) LET S=S+(10 AND 1=B OR A+1=B)
2) LET S=S+(10 AND 0 OR A=1=B)
3) LET S=S+(0 OR A+1=B)
4) LET S=S+(0 OR 1)
5) LET S=S+(1)
6) LET S=S+1
```

Tutto torna alla normalita' dando priorita' all' esecuzione dell' OR per cui e' necessario l' intervento di una seconda coppia di parentesi :

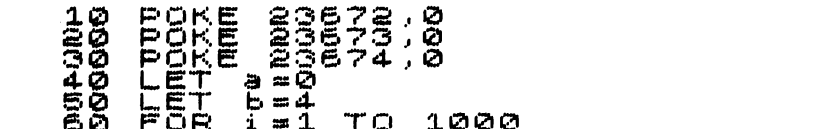
```
LET S=S+(10 AND(A=B OR A+1=B))
```

Gli operatori logici trovano spesso impiego nei programmi dei giochi in quanto :
1) Sono piu' veloci dei loro corrispettivi IF...THEN. Cio' si dimostra facilmente facendo girare i due programmi che seguono e rilevando per ognuno il tempo (in 100esimi di secondo) impiegato per giungere al termine.

Programma 1: usando IF...THEN



Programma 2: usando gli operatori logici



2) Diminuiscono la complessita' del programma. Esaminate la linea che segue :

```
   50 FOR i=1 TO 3: IF a$(i)<>b$
THEN PRINT AT b+i+i,a-(a AND a<0
);a$(m+i,1-(a AND a<0) TO )
```

Volendo ottenere gli stessi risultati con gli statement di IF...THEN e' neccessario impiegare piu' linee :

```
   50 FOR i=1 TO 3: IF a$(i)=b$ T
HEN GO TO 60
   52 IF a<0 THEN PRINT AT b+i+i,
0;a$(m+i,1-a TO ): GO TO 60
   54 PRINT AT b+i+i,a;a$(m+i)
   60 REM ....resto del programma
....
```

3) Provocano un minor assorbimento da parte del banco di memoria in quanto occupano meno spazio di IF...THEN come dimostrato dagli esempi di cui sopra.

Nei programmi racchiusi in questo volume si e' fatto un largo impiego di operatori logici cosi' come si puo' notare anche nel prossimo dal titolo "Bomber". Il game consiste nel bombardare una citta' per potervi creare spazio per l' atteraggio. Le istruzioni dettagliate, sono insite nel programma stesso.

Note

10	Inizializza le variabili, scrive le istruzioni.
20-90	Loop ripetuto per muovere l' aereo.
100-160	Linee eseguite quando la bomba e' sganciata.
170-180	Reset delle variabili quando la bomba termina la caduta.
60	Disegno dell' aereo
140	Disegno della bomba
500-540	Presenta il nuovo punteggio e calcola le bombe lanciate.
3000-3050	Interludio tra due parti.
4000	Dati per il motivetto.
5000-5060	Fine del programma.
8010-8080	Definizione delle variabili.
8110-8170	Scrive le istruzioni, che volendo possono essere riscritte.
8500-8590	Predispone lo scenario disegnando lo schermo.
8560	Mette otto grattacieli in S(29)
9000-9360	Definizione della grafica. A e B - aereo C e D - bomba E - grattacielo F - grattacielo piu' alto G - Resto del grattacielo bombardato.

```
  10 GO SUB 8000: REM iniz
  19 REM routine principale
  20 LET a$=INKEY$: PRINT AT x,y
;"  "
  40 LET y=y+1-(31 AND y=30): LE
T x=x+(a$="q" OR f<1 OR y=0)-(a$
="1" AND x>2 AND f>0): LET f=f-(
2 AND a$="1" AND f>1)
  50 IF ATTR (x,y)<>7 OR ATTR (x
,y+1)<>7 THEN GO TO CRASH
  60 PRINT AT x,y; INK 4;"AB"
  70 IF x=21 THEN GO TO LAND
  80 IF INKEY$="p" AND m AND b T
HEN LET m=0: LET bx=x: LET by=y
  85 LET f=f-(dd/75 AND f>0)
  87 FOR i=f TO ft: PLOT INK 6;i
+40,160: DRAW INK 6;0,7: NEXT i:
 LET ft=f
  90 IF m THEN BEEP .03,RND*10-2
0: GO TO LOOP
 100 PRINT AT bx,by;" "
 110 LET c=c+1: LET cb=cb+mb: IF
 cb=6 OR bx=21 THEN GO TO 160
 120 LET by=by+(c<5 AND cb=0)-(3
0 AND by=30): LET bx=bx+1
 130 IF ATTR (bx,by)<>7 AND mb=0
 THEN LET mb=1: FOR i=s(by) TO b
x: PRINT AT i,by;" ": LET s=s+1:
 NEXT i
 140 PRINT AT bx,by;("C" AND (c>
5 OR cb));("D" AND c<5 AND cb=0)
 150 LET s=s+mb: GO TO LOOP
 160 IF bx<21 THEN PRINT AT bx,b
y; INK ATTR (bx+1,by);"G"
 170 LET m=1: LET mb=0: LET cb=0
: LET c=0: LET b=b-1: GO SUB UPD
ATE
 180 GO TO LOOP
 499 REM aggiornam. punti/bombe
 500 PAPER 5: INK 1
 510 PRINT AT 0,10-LEN STR$ s;s
 520 PRINT AT 0,27;"000";AT 0,30
-LEN STR$ b;b
 530 PAPER 0: INK 7
 540 RETURN
2999 REM atterraggio riuscito
3000 RESTORE 4000: FOR i=1 TO 7:
 READ dur,not: BEEP dur/1.5,not:
 NEXT i
3010 CLS : PRINT AT 10,6; FLASH
1;"MISSIONE COMPIUTA"
3020 PRINT ''"Maggiori ostacoli,
meno carburante."
3030 LET dd=dd+3: LET b=b+dd: LE
T f=215: LET ft=f
3040 FOR i=1 TO 250: NEXT i: GO
SUB SCREEN
3050 GO TO 20
3999 REM sinfonia del 'successo'
4000 DATA 1,7,1,7,.5,6,.5,7,1,9,
```

```
2,4,2,2
4999 REM Caduta
5000 PRINT AT x,y; INK 4;"AB"
5010 FOR i=1 TO 40: BEEP .01,RND
*50-20: NEXT i
5020 FOR i=50 TO -30 STEP -1: BE
EP .01,i: NEXT i
5030 CLS : PRINT AT 8,2;"Tuoi pu
nti ora ";s
5040 PRINT AT 10,8; FLASH 1;"HAI
 FALLITO "
5050 PRINT '"-per cui devi preme
re "''"        RUN per giocare di nu
ovo"
5060 STOP
8000 GO SUB 9000: REM UDG
8010 LET s=0: LET b=100: LET f=2
15: LET ft=f
8020 LET dd=10
8030 LET m=1: LET c=0: LET cb=0:
 LET mb=0
8039 REM ROUTINES
8040 LET CRASH=5000
8050 LET LAND=3000
8060 LET SCREEN=8500
8070 LET UPDATE=500
8080 LET LOOP=20
8100 BORDER 5: PAPER 3: INK 7: C
LS
8110 PRINT TAB 12; FLASH 1;"Bomb
er"
8120 PRINT '"Attento sei in miss
ione. Comandi l'aereo verde "
8130 PRINT "volando sopra la cit
ta'. Devi atterrare senza:a) Sch
iantarti"'"b) Restare senza carb
urante."
8140 PRINT "Devi soffrire qualch
e secondo,"'" scendendo rapidame
nte, ma se sei fortunato sopravv
iverai senza schiantarti."
8150 PRINT '"I tuoi comandi sono
 '1' (su),e '0' (giu'). 'P' sgan
cia le bombe."
8160 PRINT '"Se ti riesce, ti sp
etta la missione piu' dura."
8170 PRINT '"Premi qualsiasi tas
to per bombardare."
8180 PAUSE 0: CLS
8300 DIM s(29): GO SUB SCREEN
8400 RETURN
8499 REM Disegno dello schermo
8500 BORDER 2: INK 7: PAPER 0: C
LS
8510 PRINT AT 0,0; PAPER 6; INK
1;"PUNTI=0000","BOMBE RES.=00","
CAR.:"; PAPER 4,,
8520 FOR i=1 TO 29: LET height=I
NT (RND*12)+5: LET ink=INT (RND*
6)+1
```

```
8530  IF RND>dd/30 THEN GO TO 857
0
8540  FOR n=21 TO height+1 STEP -
1: PRINT AT n,i; INK ink;"E"
8550  BEEP .003,(20-n)*3: NEXT n
8560  PRINT AT height,i; INK ink;
"F"
8570  LET s(i)=height
8580  NEXT i: GO SUB UPDATE
8590  LET x=2: LET y=0
8600  RETURN
9000  REM STORE @300: FOR i=1 TO 7
9001  FOR n=0 TO 7
9002  READ a: POKE USR CHR$ (143+
i)+n,a: NEXT n: NEXT i
9030  RETURN
9300  DATA 192,192,227,156,127,0,
0,0
9310  DATA 0,162,242,62,242,34,80
,0
9320  DATA 34,20,8,20,62,62,20,8
9330  DATA 32,16,160,92,30,31,15,
0
9340  DATA 127,73,73,127,73,73,12
7,0
9350  DATA 0,20,8,20,42,73,73,127
9360  DATA 0,0,0,66,82,80,119,127
```

Elenco delle variabili

x	Numero della riga dell' aereo
y	Numero della colonna dell' aereo
a$	Usato per leggere la tastiera.
f	Carburante usato
b	Bombe sganciate
m	Flag di quante bombe sono state sganciate 1 nessuna
bx	Numero della riga della bomba
by	Numero della colonna della bomba
dd	Grado di difficolta'
ft	Usato per diminuire il carburante
c	Contatore delle bombe che colpiscono. Distruggono 5 piani oppure cadono a vuoto.
cb	Contatore delle bombe che cadono a vuoto
mb	Incremento di "cb". Vale 0 se la bomba colpisce, 1 se va a vuoto.
S	Punti
s(29)	Array contenente otto grattacieli. Viene usato per cancellare l' edificio quando questo e' raso al suolo.
n,i	Variabili di controllo dei loops
a	Legge la grafica definita da utente
height	Usato nel disegno dei grattacieli
ink	Colore dei grattacieli
dur,not	Variabili di lettura per il motivetto

CRASH,
LAND,
SCREEN,
UPDATE,
LOOP "Tokens" per le subroutines

Le lettere maiuscole alle linee 60,140,160,5000,8540 e 8560 vanno battute in "graphic mode" e servono a generare i caratteri richiesti.

Capitolo 5

MOVIMENTO

Gran parte dei games, per non dire tutti, posseggono oggetti in movimento che rappresentano figure spesso bizzarre come i "pacmen" e le navi spaziali. Le difficolta' piu' evidenti nello scrivere programmi nascono dal fatto che :
a) il BASIC non e' sufficientemente veloce per muovere contemporaneamente diverse immagini,
b) l' animazione e' a scatti; piu' a lungo una immagine si ferma sullo schermo e peggio e',
c) gli oggetti non devono travalicare i confini imposti dallo schermo,
d) le varie parti non possono "sapere" in che direzione spostarsi; ad esempio gli invasori spaziali si muovono solo a destra e a sinistra,
e) il computer deve esplorare lo schermo per rilevare le eventuali collisioni di due oggetti (ad esempio, la racchetta con la palla).
Tutto cio' si verifica su uno scenario di sottofondo mantenuto fisso. L' oggetto viene stampato (PRINT) o disegnato (PLOT) in relazione ad una serie di coordinate x e y, quindi cancellati e ridisegnati con una o con tutte e due le coordinate cambiate di valore il che dona l' illusione del movimento.
L' effetto dello spostamento a scatti e' appunto dovuto al fenomeno di cancellazione-ristampa e la sua intensita' dipende dal tempo intercorrente tra le due azioni. Va da se' che piu' rapidamente la figura viene ripresentata dopo essere stata cancellata, piu' si ha la sensazione di un movimento costante.
Il programma che segue, stampa un quadratino nero al centro e lo sposta fino alla fine dello schermo :

```
10 LET y=0
20 PRINT AT 10,y;"■"
30 PRINT AT 10,y;" "
40 LET y=y+1
50 GO TO 20
```

Provatelo e vedrete il quadratino lampeggiare scorrendo verso destra. Come si puo' migliorare un tale effetto ? Un rimedio e' quello di inserire il calcolo di avanzamento, di cui alla linea 40, tra il disegno e la cancellazione dell' oggetto il quale, pertanto, si ferma sullo schermo il tempo necessario al calcolo delle sue nuove coordinate.
Il risultato si ha col programma che segue in cui e' stata prevista una temporanea memorizzazione (linea 20) del vecchio valore di "y" per non far stampare erroneamente la linea 30 :

```
10  LET  y=0
200  LET  t=y
300  PRINT  AT  10,y;"■"
400  LET  y=y+1
600  GO  TO  200
```

Nonostante questo accorgimento, il lampeggiare dell' oggetto e' ancora percettibile. Provate allora ad aggiungere una linea che prolunghi la permanenza dell' immagine per mezzo di un loop FOR....NEXT oppure di un bit musicale :

```
45 FOR N=1 TO 5:NEXT N
45·BEEP .01,RND*20
```

Il programma cosi' scritto conduce pero' ad un errore in quanto porta il computer a scrivere dove non gli e' possibile. Fatelo girare e rilevate il messaggio che appare quando si blocca.
Per ottenere un effetto analogo potete impiegare il loop FOR....NEXT come segue :

```
10  FOR  y=0  TO  31
200  PRINT  AT  10,y;"■"
300  BEEP  .01,RND*20
400  PRINT  AT  10,y;"  "
500  NEXT  y
```

Se lo spostamento del quadrato avviene di una colonna alla volta in linea retta, si impiega di solito uno statement di PRINT che cancelli la figura disegnata dal primo. Notate che l' ultimo quadretto non viene cancellato :

```
10  FOR  y=0  TO  30
200  PRINT  AT  10,y;"  ■"
300  BEEP  .01,RND*20
40  NEXT  y
```

Gli "Asteroidi" e i giochi Arcade rimediano ai limiti di spazio imposti dai bordi facendo girare attorno gli oggetti, per meglio capire, una astronave che esca dal lato destro dello schermo, rientra immediatamente da quello sinistro progredendo nella sua corsa. Sullo Spectrum, questa funzione si implementa facilmente come dimostrato nel programma sotto riportato. La grandezza "y" che e' il numero della colonna nella quale e' disegnato il quadretto, una volta giunto a 32 torna a 0 ripresentando da capo il disegno. Per ottenere un egual risultato in verticale, andra' interessata "x" con valori da 0 a 22.

```
10  LET x=0: LET y=0
20  LET a=x: LET b=y
30  PRINT AT x,y;"■"
40  LET x=x+(y=31): LET x=x-(22
AND x=22)
50  LET y=y+1-(32 AND y=31)
60  PRINT AT a,b;" "
70  GO TO 20
```

Con cio' affrontiamo l' argomento circa la "conoscenza" del percorso da parte degli oggetti. Il prossimo programma mostra un invader, nella fattispecie la lettera "V", che si sposta prima verso destra, poi verso sinistra scendendo in continuazione verso il basso :

```
10  BORDER 4: INK 9: CLS
20  LET x=0: LET y=0
30  LET direzione=1
40  LET a=x: LET b=y
50  PRINT AT x,y; FLASH 1;"V"
60  LET y=y+direzione: BEEP .01
,RND*20
70  IF y=31 OR y=0 THEN LET dir
ezione=-direzione: LET x=x+1: IF
x=22 THEN STOP
80  PRINT AT a,b;" "
90  GO TO 40
```

La variabile "direzione" contiene il numero da addizionare (o da sottrarre se e' negativo) a "y". Alla linea 70 il programma testa se l' invader ha raggiunto uno dei margini dello schermo, se si, la variabile "direzione" viene invertita di segno. La stessa linea incrementa "x" facendo scendere l' invader verso il basso fino a raggiungere l' ultima riga del quadro alla quale il programma si ferma.
Prima di addentrarci nella "conoscenza" del senso di marcia da parte degli oggetti, provate i due programmini che seguono i quali fanno "rimbalzare" l' oggetto stesso sui bordi dello schermo per mezzo di PRINT negativi :

'Rimbalzo' sull' asse y

```
10  LET x=0: LET y=-10
20  PRINT AT x,y;y
30  LET y=y+1: LET x=x+1
40  GO TO 20
```

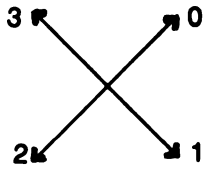
'Rimbalzo' sull' asse x

```
10  LET x=-15: LET y=0
20  PRINT AT x,y;x
30  LET y=y+1: LET x=x+1
40  GO TO 20
```

Queste proprieta' vengono usate in molti giochi tra cui lo "squash" e' quello che le evidenzia maggiormente. Il nome della variabile usata per la direzione e' di solito "d" oppure "dir" e, nell' esempio dell' invader a "V", assume due valori ben precisi :
a) 1, nel qual caso l' incremento vale 1 e l' oggetto si sposta verso destra,
b) -1, in cui si ha un decremento con l' oggetto che si sposta verso sinistra.
Sempre nello stesso esempio, il valore del "marker" di direzione viene inserito direttamente nel calcolo :

```
LET Y=Y+direzione
```

e una tale forma puo' essere usata in qualsiasi momento per indicare dove l' oggetto deve prendere posto ma non per calcolare il movimento. Il valore del "marker" e' compreso tra 0 e 3 ed i numeri singoli si riferiscono alle direzioni :



E' molto difficile manipolare detti numeri per inserirli direttamente nei calcoli per il movimento, pero' e' possibile ricorrere all' aiuto degli operatori logici visti un capitolo addietro. Guardate :

```
'x' e' la coordinata x ( ↓ )
'y' e' la coordinata y (-->)
'd' e' il puntatore di direzione con i valori
       corrispondenti al diagramma.

Cosa succede a 'x' e 'y' per diversi valori.

1000 LET x=x+(d=1 OR d=2)-(d=0 OR d=3)
1010 LET y=y+(d<2)-(d>1)
```

Se un oggetto, nel suo spostamento urta una linea orizzontale (vedremo come piu' avanti analizzando SCREEN$ e ATTR), puo', sempre usando gli operatori logici, rimbalzare cambiando direzione :

```
1999 REM Urto della linea orizzo
ntale
2000 LET d=d+1-(2 AND (d=3 OR d=
1))
```

Se non siete del tutto convinti che una sola linea possa cambiare correttamente i valori di "d", battete quanto segue e confrontatelo col diagramma :

```
  10 INPUT "Il valore di 'd' e'
?";d
  20 PRINT "Vecchio=";d
  30 GO SUB 2000
  40 PRINT "NUOVO=";d
  50 GO TO 10
2010 RETURN
```

Un altro metodo, si presenta con l'adozione delle stringhe di variabili e di VAL. Per vedere l' effetto di dette funzioni, battete questo programma dimostrativo :

```
  10 INPUT "Come ti chiami ?";a$
  20 IF LEN a$>16 THEN PRINT AT
21,0;"Troppo lungo,per favore re
inserisci": BEEP 2,-10: GO TO 10
  30 LET z=0: LET z$="z+1"
  40 POKE 23692,-1: REM scroll
  50 PRINT TAB z; INK RND*5;a$
  60 BORDER RND*7: LET z=VAL z$
  70 IF z=32-LEN a$ THEN LET z$=
"z-1"
  80 IF z=0 THEN LET z$="z+1"
  90 GO TO 40
```

La collisione di due oggetti, come ad esempio un missile ed un invasore spaziale, viene rilevata facilmente testando la variabile dei due oggetti stessi : IF X missile=X invader AND Y missile=Y invader THEN...
Ciononostante, la via piu' facile, resta quella di esplorare direttamente lo schermo per mezzo di uno dei tre sistemi sottoelencati (anche se il terzo lo sconsigliamo) :
1) SCREEN$. Questa funzione rende una stringa e va inserita con la forma "SCREEN$ (x,y)"dove "x" e "y" rappresentano le coordinate cosi'come per lo statement PRINT AT.Non lavora con gli UDG,ma solo con i caratteri che ritornano normali anche nel caso in cui siano inversi.
2) ATTR. Rende un numero che dipende dagli "Attributi" della cella definita dalle coordinate. Gli attributi sono il colore, il flashing e il brightness. Il relativo statement e' "ATTR (x,y)" della medesima fisionomia del precedente.
ATTR e' la somma di :

128 (10000000 binario) se il carattere lampeggia con FLASH 1,
 64 (01000000 binario) se il carattere ha luminosita' piu' accentuata con BRIGHT 1,
  8 il colore di PAPER che occupa il terzo, il quarto e il quinto posto dell' equivalente binario.
ATTR e' molto usato, in quanto non distingue se il carattere che sta esaminando e' un UDG o meno.
3) POINT. Rende 1 oppure 0. La forma del suo statement e' classica "POINT (x,y) in cui "x" e "y" vengono impiegati come in PLOT x,y. Il risultato della funzione e' 1 se le coordinate 'pescano' un punto, mentre e' 0 se riscontrano un vuoto.
Vediamo ora la velocita'. Deve essere tale da mettere in condizione il giocatore di poter sparare agli invaders che scendono spostandosi casualmente a destra o a sinistra. Sia la base che gli alieni, possono uscire dai margini laterali dello schermo per riapparire dal lato opposto nella stessa posizione. Gli invaders, quattro in tutto, vengono presentati dal computer uno per volta rendendo la gara assai interessante anche perche' a loro volta possono sparare alla base.
Le coordinate di ogni invader, sono memorizzate in due arrays, l' array "a" per le coordinate verticali, il "b" per le coordinate orizzontali. I suoi movimenti sono funzione di quelli della base e vengono determinati dalla variabile "i" presente nella linea 20. Fatta questa doverosa presentazione eccovi il programma con relativa documentazione.

Elenco delle variabili

h	Record
a$(,)	Array di caratteri contenente la forma degli alieni.
e$(,)	Array contenente l' immagine dell' esplosione.
s	Punti.
men	Numero delle basi rimaste.
a(,),	
b(,)	Coordinate x e y degli invasori.
p	Numero della colonna della base.
i	Usato nel loop principale per il movimento degli alieni.
ii	Usato per calcolare "i".
t1	Numero di pixel della base nello sparo.
dr	Sviluppa la traiettoria in salita dello sparo della base.
	Usato per calcolare se l' invader e' stato colpito.
t2,	
t1	Numero di pixel delle coordinate x e y quando spara l' alieno.
dr	Sviluppa la traiettoria in discesa dello sparo degli alieni.
x,	
i,	

n	Variabili per il controllo dei loop.
m,	
c$	Usate per la fine della gara.

Le lettere maiuscole nelle linee 140,1010,1020,1040,1510,1520, 7710,7720 riferiti agli statement di PRINT sono tutti caratteri definiti dall' utente(UDG)

Note

10	Inizializza le variabili e scrive le istruzioni.
20-100	Parte del loop per muovere gli alieni e per sparare.
105-160	Parte del loop per muovere la base e per sparare
1000-1060	Calcolo del fuoco
1400-1420	Fuoco
1500-1540	Calcolo del fuoco
1900-1920	Fuoco
2000	Incrementa il punteggio
3000-3020	Esplosione della base
4000-4070	Fine di una parte
4080-4180	Fine della gara
7000-7100	Scrive le istruzioni
7500-7530	Inizializza le variabili permanenti come ad esempio il punteggio.
7600-7800	Inizializza le altre variabili e disegna lo schermo
9000-9030	Definisce i caratteri UDG
9050-9150	Dati per gli UDG
9500	Dati per la fine
9990-9999	Fine.

```
  10 GO TO 7000: REM istr/iniz
  20 LET i1=i: LET i=i+1-(4 AND
i1)+3
  30 IF a(i)>=2 THEN PRINT AT a(
i),b(i);" "
  40 LET a(i)=a(i)+1: LET b(i)=b
(i)-1+INT (RND*3)
  50 IF b(i)=32 THEN LET b(i)=0
  60 IF b(i)=-1 THEN LET b(i)=31
  70 IF a(i)=17 OR a(i)=18 THEN
PRINT AT a(i),0;"
                  "
  80 IF a(i)=20 THEN GO SUB 3000
: REM alterato
  90 IF a(i)>=2 THEN PRINT AT a(
i),b(i);a$(i)
 100 IF a(i)>=2 AND RND>.9 THEN
GO SUB 1500: REM spero
 105 IF m=0 THEN GO TO 4000: R
EM fine della gara
 110 PRINT AT 20,p;" ": LET p=p+
(INKEY$="2")-(INKEY$="1")
```

```
 120 IF p=-1 THEN LET p=31
 130 IF p=32 THEN LET p=0
 140 PRINT INK 2;AT 20,p;"A"
 150 IF INKEY$="0" THEN GO SUB 1
000: REM sparo
 160 GO TO 20: REM loop
 999 REM spari della base
1000 LET t1=p*8+4: PLOT t1,16
1010 IF ATTR (18,p)=6 THEN LET d
r=8: PRINT AT 18,p;"D": GO SUB 1
400: RETURN
1020 IF ATTR (17,p)=6 THEN LET d
r=24: GO SUB 1400: PRINT AT 17,p
;"D": RETURN
1030 FOR x=1 TO 4
1040 IF p=b(x) THEN LET dr=(21-a
(x))*8-16: PRINT AT a(x),p;"D":
GO SUB 1400: PRINT AT a(x),p;" "
: GO SUB 2000: RETURN
1050 NEXT x
1060 LET dr=137
1400 DRAW INK 7;0,dr: BEEP .05,3
0
1410 PLOT OVER 1;t1,16: DRAW OV
ER 1;0,dr
1420 RETURN
1499 REM spari degli invaders
1500 LET t2=(21-a(i))*8-1: LET t
1=b(i)*8+4: PLOT t1,t2
1510 IF ATTR (17,b(i))=6 THEN LE
T dr=t2-32: GO SUB 1900: PRINT A
T 17,b(i);"D": RETURN
1520 IF ATTR (18,b(i))=6 THEN LE
T dr=t2-40: PRINT AT 18,b(i);"D"
: GO SUB 1900: RETURN
1530 IF b(i)=p THEN LET dr=t2-16
: GO SUB 1900: GO SUB 3000: RETU
RN
1540 LET dr=t2
1900 DRAW INK 3;0,-dr: BEEP .05,
0
1910 PLOT OVER 1;t1,t2: DRAW OVE
R 1;0,-dr
1920 RETURN
1999 REM nuovi invader
2000 LET s=s+5*x: PRINT AT 1,(5-
(LEN (STR$ s))); PAPER 4; INK 7;
s
2020 LET a(x)=2: LET b(x)=INT (R
ND*32)
2030 RETURN
2999 REM base distrutta
3000 FOR n=0 TO 10: FOR x=1 TO 2
3010 PRINT AT 20,p;CHR$ (144+x):
 BEEP .05,RND*30
3020 NEXT x: NEXT n
3030 CLS : LET men=men-1
3040 GO SUB 7600
3050 RETURN
3999 REM fine della gara
```

```
4000 PRINT AT 10,11; FLASH 1; PA
PER 3; INK 7;"GARA FINITA"
4010 RESTORE 9500: FOR z=1 TO IN
T (RND*6)+1: READ c$: NEXT z
4020 PRINT AT 11,15-(LEN c$)/2;
INK 0;c$
4030 LET m=0: FOR i=0 TO 31: LET
 b$=SCREEN$ (11,i)
4040 PRINT AT 11,i;CHR$ (151+m):
 BEEP .05,RND*40
4050 LET m=m-m+(NOT m): PRINT AT
 11,i; INK 4;b$
4060 NEXT i
4070 PAUSE 100
4080 BORDER 1: PAPER 6: INK 2: C
LS
4090 PRINT AT 2,8;" Punti      ";
s
4100 PRINT : IF s<=h THEN PRINT
TAB 7;" Il record e' ";h: GO TO
 4130
4110 PRINT TAB 8;" Il nuovo reco
rd!!!"
4120 LET h=s
4130 PRINT : PRINT " Vuoi fare u
n'altra partita ";AT 8,13; FLA
SH 1;"S/N"
4140 PRINT : PRINT TAB 7;"Premi
 y/n"
4150 GO TO 4150+(INKEY$="y")+(58
40 AND INKEY$="n")
4160 IF INKEY$<>"" THEN GO TO 41
60
4170 PRINT : PRINT "Vuoi le istr
uzioni y/n ?"
4180 GO TO 4180+(2970 AND INKEY$
="n")+(2840 AND INKEY$="y")
7000 LET h=0
7010 DIM a$(4): DIM e$(2): GO SU
B 9000: REM UDG
7020 BORDER 6: PAPER 3: INK 7: C
LS
7030 PRINT AT 1,14; FLASH 1;"ZAP
"
7040 PRINT : PRINT "Ecco gli ali
eni ! Distruggili prima che ti r
aggiungano. Punti:"
7050 FOR i=4 TO 1 STEP -1: PRINT
 : PRINT TAB 5;CHR$ (150+i);" ...
 ";i*5: NEXT i
7060 PRINT : PRINT " Tasto '1',
sinistra ";"     - '2', destra
"
7070 PRINT : PRINT " Il pulsante
 di sparo e' '0'."
7080 PRINT : PRINT TAB 2; FLASH
1;" Premi un tasto per continuare
"
7090 IF INKEY$<>"" THEN GO TO 70
90
7100 IF INKEY$="" THEN GO TO 710
```

```
0
7500 LET s=0: LET men=5
7510 DIM a(4): DIM b(4)
7520 GO SUB 7600: REM iniz
7530 GO TO 20: REM Programma pri
ncipale
7599 REM iniz/reiniz
7600 FOR i=1 TO 4: LET a(i)=1: L
ET b(i)=INT (RND*31): NEXT i
7610 LET i=1: LET p=16
7699 REM disegno schermo
7700 BORDER 2: INK 4: PAPER 0: C
LS
7710 FOR n=1 TO 3: PRINT AT 17,n
*11-7; INK 6;"E█F"
7720 PRINT AT 18,n*11-7; INK 6;"
█G█": NEXT n
7730 PRINT INK 7; PAPER 4;AT 0,0
;"PUNTI         RECORD
  00000         00000
"
7740 FOR n=1 TO men-1: PRINT AT
0,30-n*2; PAPER 4; INK 7;"A": NE
XT n
7750 PRINT AT 1,5-(LEN (STR$ s))
; PAPER 4; INK 7;s
7760 PRINT AT 1,16-(LEN (STR$ h)
); PAPER 4; INK 7;h
7800 RETURN
9000 RESTORE 9050: FOR i=1 TO 11
: FOR n=0 TO 7: READ a: POKE USR
 (CHR$ (143+i))+n,a
9010 IF i>1 AND i<4 THEN LET e$(
i-1)=CHR$ (143+i)
9020 IF i>7 THEN LET a$(i-7)=CHR
$ (143+i)
9030 NEXT n: NEXT i
9050 DATA 24,24,24,60,126,255,21
9,219: REM base
9060 DATA 36,41,150,80,42,140,50
,65: REM esplosione 1
9070 DATA 137,74,32,23,232,38,81
,144: REM esplosione 2
9080 DATA 4,128,33,64,0,129,8,65
: REM esplosione schermo
9090 DATA 1,3,7,15,31,63,127,255
: REM s1
9100 DATA 128,192,224,240,248,25
2,254,255: REM s2
9110 DATA 255,255,255,255,255,23
1,129,0: REM s3
9120 DATA 129,90,60,90,60,102,15
3,195: REM invader 1
9130 DATA 129,66,60,219,165,153,
36,195: REM invader 2
9140 DATA 129,90,60,255,90,255,1
29,231: REM invader 3
9150 DATA 60,126,219,219,255,126
,66,102: REM invader 4
9160 RETURN
```

```
9500 DATA "V I T T O R I A","GLI
 INVASORI HANNO VINTO","LA TERRA
 E' MORTA","SALVE.IMPERATORE","T
ROPPO FACILE"
9990 PAPER 7: BORDER 7: INK 0: C
LS : PRINT AT 10,12; FLASH 1;"CI
AOOO"
9999 STOP
```

Il programma che segue e' il classico "Space Invaders" con tanto di laser surriscaldabile e di invasori danzanti. Vengono usate delle stringhe in quanto gli alieni, disposti su tre file vanno colpiti uno alla volta. Le file, memorizzate in arrays, vengono duplicate onde permettere il movimento tipico che caratterizza i giochi riguardanti gli "Space Invaders".

Elenco delle variabili

b	Altezza degli invaders
h	Altezza di partenza degli invaders
m	Usato per l' alternarsi degli invaders
i	Loop di controllo per stampare gli invaders
a	Numero della colonna del primo invader
s,	
si	Usati per variare la velocita' della base in rapporto agli invaders
p	Numero di colonna della base
in	Usato per leggere la tastiera
k	"Marker" per rilevare se il laser e' surriscaldato o no
c	Numero degli invaders colpiti
l	Livello di temperatura del laser
d	Variabile per la direzione degli invaders
g	Livello di terra per vedere se gli invaders hanno toccato il fondo
men	Numero di basi rimaste
wave	Numero delle alternanze
v,	
x	Usati per i dati del motivetto
sc	Punti
l	Distanza della traiettoria degli spari
r	Numero di pixel della base
t	Usato per controllare la stringa e per vedere se ogni invader e' stato colpito
z	Usato assieme a "t"
a$(,)	Arrays per la memorizzazione degli invaders
b$	Stringa vuota per controllare la comparazione ad a$ di una riga di invader
n	Tono della nota suonata
clr	Usato per cancellare la tastiera
FIRE,	
HIT,	
SCREEN,	
SCORE	Memorizzano per chiarezza i numeri di partenza delle subroutines

Vi sono degli user defined-graphics alle linee :
90,5160 e 5170

Note

10	Inizializzazione e istruzioni
30-60	Muove gli invaders
70-120	Movimento della base e sparo
130-160	Laser
170	Rileva se gli invader sono alla fine della linea
180-250	Inizializzazione per una nuova ondata e una nuova base se gli invader sono atterrati
800-830	Suona il motivetto
900-930	Usato per la fine della partita
2000-2050	Prepara lo sparo
2060-2080	Laser
2100-2200	Fuoco e esplosioni
3000-3060	Considera quando l' invader risulta colpito
4000	Presenta il punteggio
5010-5030	Considera le variabili permanenti come i punti
5150-5280	Considera le altre variabili
5300	Chiede se lo schermo e' disegnato
6000-6100	Disegna lo schermo
8000-8010	Dati per il motivo
9000-9030	Definisce gli UDG
9100-9160	Dati per UDG

```
    10  GO TO 5000: REM iniz
   800  LET b=h
   810  REM [LOOP]
   400  LET m=m-m+(3 AND NOT m)
   410  BEEP .01,n+b
   500  FOR i=1 TO 3: IF a$(i)<>b$
THEN PRINT AT b+i+i,a-(a AND a<0
);a$(m+i,1-(a AND a<0) TO )
   600  NEXT i
   700  LET s=s+$i
   710  REM SUll'istruzione 3 cambiare
   720  REM 189 in 253,190 in 254
   800  LET in=IN 63486: IF in=254
OR in=253 THEN PRINT AT 20,p;" "
   900  LET p=p+(in=253 AND p<28)-(
in=254 AND p>3): PRINT AT 20,p;
INK 5;"▲"
  1000  IF IN 61438=254 AND NOT k T
HEN GO SUB FIRE: IF c=24 THEN GO
 TO 8000
  1110  IF l>130 AND NOT k THEN LET
 k=10: PRINT AT 5,7; FLASH 1;"LA
SER SURRISCALDATO"
  1200  IF s>1 THEN LET s=s-1: GO T
O 800
  1300  LET l=l-(2+(3 AND k) AND l>
3)
  1400  LET k=k-(k<>0)
  1500  PAPER 2: INK 7: PLOT 244,15
```

```
5: DRAW INVERSE 1;l-130,0: PAPER
 1
 160 IF NOT k THEN PRINT AT 5,7;
""
 170 IF CODE (SCREEN$ (b+2,1+(29
 AND d=1)))=32 THEN LET a=a+d: G
O TO 30
 171 REM FINE DEL LOOP
 180 LET d=-d: LET a=a+d
 190 FOR i=1 TO 3: IF a$(i)<>b$
THEN PRINT AT i+i+b,a-(a AND a<0
);b$(1-(a AND a<0) TO )
 200 NEXT i
 210 LET b=b+1: IF b<g THEN GO T
O 30
 220 LET men=men-1: LET wave=wav
e+1
 230 FOR i=1 TO 10: BORDER RND*7
: PAPER RND*7: CLS
 240 BEEP .01,RND*40: NEXT i
 250 GO TO 900+(4250 AND men>0)
7999 REM Nuova ondata
8000 RESTORE 8000: FOR i=1 TO 16
8010 READ v,x: BEEP v/10,x: NEXT
 i
8020 LET h=h+(h<9): LET wave=wav
e+1
8030 GO TO 5150
8999 REM Invaders a terra
9000 PRINT AT 10,11; FLASH 1; IN
K 6; PAPER 0;"GARA TERMINATA"
9010 PRINT AT 13,10; INK 2; PAPE
R 7;"Punti= ";sc*10
9020 BORDER 2: PAPER 7: INK 1
9030 STOP
1999 REM Fuoco
2000 LET q=126: LET r=p*8+4
2010 LET t=p-a+1: IF t<1 OR t>24
 THEN GO TO 2050
2020 FOR z=3 TO 1 STEP -1
2030 IF a$(z,t)<>" " THEN GO SUB
 HIT: GO TO 2060
2040 NEXT z
2050 LET z=-b/2+2
2060 LET l=l+10: IF l>130 THEN L
ET l=135
2070 PAPER 2: INK 7: PLOT 112,15
5: DRAW l,0
2080 PAPER 1
2100 INK 2: PLOT r,16: DRAW 0,q
2110 FOR v=1 TO 4: PRINT AT b+z+
z,p; INK 6;CHR$ (146+v): BEEP .0
05,v*10: NEXT v
2130 INK 1: PLOT r,17: DRAW 0,q
2140 INK 7
2200 RETURN
2999 REM Distruzione invaders
3000 LET q=(21-b-z*2)*8-17
3010 LET a$(z,t)=" ": LET a$(z+3
,t)=" "
```

```
3020 LET c=c+1: LET si=1-c/23: L
ET n=c*3-40
3030 LET g=g+(2 AND a$(z)=b$)
3040 LET sc=sc+4-z: GO SUB SCORE
3060 RETURN
3999 REM Punti
4000 PRINT AT 1,11-LEN STR$ sc;
INK 4; PAPER 0;sc
4010 RETURN
5000 GO SUB 9000: REM UDG
5010 DIM a$(6,24): DIM b$(24)
5020 LET sc=0: LET men=3
5030 LET wave=1: LET h=5
5100 LET FIRE=2000
5110 LET HIT=3000
5120 LET SCREEN=6000
5130 LET SCORE=4000
5150 FOR i=1 TO 6
5160 IF i<4 THEN LET a$(i)="  ▓
▓  ▓  ▓  ▓  ▓  ▓  ▓"
5170 IF i>3 THEN LET a$(i)="  ▓
▓  ▓  ▓  ▓  ▓  ▓  ▓"
5180 NEXT i
5200 LET p=INT (RND*26)+3
5210 LET d=1: LET k=0
5220 LET m=0: LET l=70
5230 LET a=3: LET c=0
5240 LET s=0: LET si=1
5250 LET n=-30
5260 LET clr=IN 63486: LET clr=I
N 61438
5280 LET g=14
5300 GO SUB SCREEN
5400 GO TO 20
5999 REM Disegno schermo
6000 BORDER 0: PAPER 1: INK 4: B
RIGHT 1: CLS : PAPER 0
6010 PRINT PAPER 2;''''
6020 PRINT AT 1,1;"PUNTI 00000":
 PRINT AT 1,13;"ONDA    ";AT 1,18
;wave: PRINT AT 1,21;"UOM.  ";men
6040 PRINT AT 2,1;"LASER TEMP:"
6050 INK 7
6060 PRINT AT 2,13; PAPER 2;"███
█"
6070 PAPER 2: PLOT 112,155: DRAW
 l,0
6080 PAPER 1
6100 RETURN
7999 REM Dati per la melodia
8000 DATA 3,0,6,7,1,5,1,4,1,2,3,
12,6,7,1,5,1,4,1,2
8010 DATA 3,12,6,7,1,5,1,4,1,5,3
,2
9000 RESTORE 9100: FOR i=1 TO 7
9010 FOR n=0 TO 7: READ a: POKE
USR CHR$ (i+143)+n,a: NEXT n
9020 NEXT i
9030 RETURN
9100 DATA 129,189,189,173,239,60
,36,231
```

```
9110 DATA 231,165,189,181,52,255
,129,231
9120 DATA 0,8,8,8,28,28,62,127
9130 DATA 129,66,60,36,36,60,66,
129
9140 DATA 129,126,66,66,66,66,12
,129
9150 DATA 255,195,165,153,153,16
,195,255
9160 DATA 0,0,0,0,0,0,0,0
```

Il prossimo game sfrutta un effetto ottico con ottimi risultati. Ha solamente tre parti in movimento : il limite di tempo, la bocca e la coda del serpente. Le istruzioni sono incluse nel programma stesso.

Elenco delle variabili

h	Contatore degli elementi dell' array comprendente le coordinate della testa
i	Contatore degli elementi dell' array comprendente le coordinate del resto del corpo (stabilisce anche il colore del serpente)
a$	Ingresso da tastiera
d$	Memorizza la direzione attuale
x,	
y	Coordinate della testa
c,	
d	Coordinate della mela che deve essere mangiata
w	Usata per l' attesa dopo aver mangiato la mela
u	Altezza aggiornata del limite di tempo
b	Decremento del limite di tempo in funzione del grado di difficolta' scelto
t	Contatore degli elementi compresi nell' array delle coordinate aggiornate della testa
x(.)	Array per la coordinata x del corpo
y(.)	Array per la coordinata y del corpo
m	Usata per il movimento della bocca e per i colori del serpente
l	Lunghezza aggiornata del serpente
s	Punteggio
h(.)	Array contenente i primi cinque record ad ogni livello di abilita'
h$(.)	Array contenente i nomi dei recordmen ad ogni livello di abilita'
v,	
len,	
note	Usate per il motivetto
v,	
n	Usate per definire i nomi iniziali nella tabella dei record
b$,	
b	Usate per inserire un nuovo nome
n,	

v,
i,
l Variabili di controllo per i loops ecc.

I caratteri grafici da definire battendo le maiuscole in "graphic mode", trovano posto nelle linee :
300,"B",310,"U","D","L","R",330,"0",510,"A",1200,"C",5000,"0".

Note

10	Inizializzazione e istruzioni
20-50	Tastiera
60-70	Testa per vedere se qualcosa e' stato colpito
90	Tratta il limite di tempo
100-110	Tratta la coda
120	Inserisce i valori nell' array
300-310	Subroutine per il disegno della testa
500-550	Disegna una nuova mela, il punteggio e suona il motivetto
1000-1010	Predispone i record
1020-1050	Istruzioni
1060-1080	Istruzioni circa i limiti di tempo

Nel caso possediate uno Spectrum a 16K, dovete apportare le modifiche che seguono :
Linee 20,30,110,520,5020,1170,1180. Cambiare ogni numero 500 in 180 e ogni 499 in 179. Se non fate cio' il programma impegna troppa memoria e si blocca con l' errore 4 "Out of memory".

1090-1150	Selezione dei livelli di abilita'
1160	Colori, che potete anche variare
1170-1190	Inizializzazione
1200-1220	Disegna lo schermo
4000-4020	Definisce gli UDG
4030-4100	Dati per gli UDG
5000-5050	Non disegna la coda alla fine della gara
5100-5110	Punti
5120	Presenta il record precedente se non e' stato battuto
5130-5240	Presenta il nuovo record nel caso sia stato ottenuto
5250-5320	Presenta la tavola dei record e il "bit per la fine"
9000	Dati per il motivetto

Serpenti

```
   10 GO SUB 4000: GO TO 1000
   20 LET h=h-1: IF h=0 THEN LET
h=500
   30 LET j=j-1: IF j=0 THEN LET
j=500
   40 LET a$=INKEY$: IF a$>"4" AN
D a$<"9" THEN LET d$=a$
```

```
  50 LET x=x+(d$="6")-(d$="7"):
LET y=y+(d$="8")-(d$="5")
  60 IF x=c AND y=d THEN GO SUB
300: GO TO 500
  70 IF SCREEN$ (x,y)<>" " THEN
GO TO 500: GO SUB 300
  80 IF w<-4 THEN LET u=u-5: PLO
T INK 5; INVERSE 1;240,u: DRAW I
NK 5; INVERSE 1;7,0: IF u=0 THEN
 GO TO 6000
 100 IF w<1 THEN PRINT AT x(t),y
(t);" ": LET t=t-1
 110 LET w=w-1: IF t=0 THEN LET
t=1500
 120 LET x(h)=x: LET y(h)=y: GO
TO 200
 300 IF m=1 THEN PRINT AT x,y;"B
";
 310 IF m=3 THEN PRINT AT x,y;("
U" AND d$="7");("D" AND d$="6");
("L" AND d$="5");("R" AND d$="8"
)
 320 LET m=m+(2 AND m=1)-(2 AND
m=3)
 330 PRINT AT x(j),y(j); INK m;"
0"
 340 RETURN
 4440 REM mela
 4450 LET c=INT (RND*19)+2: LET d
=INT (RND*28)+1: IF SCREEN$ (c,d
)<>" " OR c=x(t) AND d=y(t) THEN
 GO TO 5500
 5100 PRINT AT c,d; INK 4;"A"
 5200 LET l=l+5: IF l<500 THEN LE
T w=w+5
 5300 LET s=s+1: PRINT AT 0,9-LEN
 (STR$ s);s: IF s>h(lv,1) THEN
PRINT AT 0,16-LEN (STR$ s);s;" d
i ??????"
 5400 RESTORE 9000: FOR v=1 TO 5:
 READ len: READ note: BEEP len,n
ote: NEXT v
 5500 GO TO 90
 1000 DIM h$(5,5,6): DIM h(5,5)
 1010 FOR v=1 TO 5: FOR n=1 TO 5:
LET h$(v,n)=" Eroe ": NEXT n: N
EXT v
 1017
 1018 REM istruzioni
 1019
 1020 BORDER 4: PAPER 0: INK 7: C
LS
 1030 PRINT TAB 12; FLASH 1;"SERP
ENTI": FLASH 0;;;"Dovete guidare
 il serpente verso la mela verde
; il serpente diventa sempre piU
 LUNGO."
 1040 PRINT "e appena divorata la
```

```
 mela,ne appare subito un'altra.
Dovete stare attenti che il serp
ente non divori se stesso,oppure
 il bordo."
1050 PRINT ''"Per muovere il ser
pente usate i tasti cursore 5-8"
;,,"     "; FLASH 1;"Un tasto per
 continuare"
1060 LET a$=INKEY$: IF a$="" THE
N GO TO 1060
1070 CLS : PRINT "Ah,dimenticavo
 di dirvi che","c'e' un limite d
i tempo ,e che","piu' il grado d
i difficolta' e' alto,meno tempo
 avete per mangiare le "
1080 PRINT "mele.Il tempo a disp
osizione e'","mostrato dalla car
ta alla sinistra","dello schermo
.Lo scorrere del tempo si arrest
a mentre state mangiando la mela
 per rendere piu' interessante l
a gara."
1087
1088 REM grado di abilita'
1089
1090 PRINT ,,"Scegli il grado di
 abilita':"
1100 PRINT ,,"1)-Almeno bravo qu
anto l'autore 2)-Super Eroe",,"3
)-Eroe",,"4)-Abbastanza bravo","
5)-Principiante"
1110 PRINT ,,"Premi 1-5 "
1120 LET a$=INKEY$: IF a$="" OR
a$<"1" OR a$>"5" THEN GO TO 1120
1130 LET lv=VAL (a$): PRINT ,,"H
ai scelto il livello ";lv;" - ";
("CORAGGIOSO" AND lv<3);("PAVIDO
" AND lv=3);("VILE" AND lv>3);"
!"
1140 FOR i=1 TO 200: NEXT i
1150 LET u=168: LET b=1/lv
1157
1158 REM colori
1159
1160 BORDER 2: PAPER 7: INK 1: C
LS
1170 DIM x(500): DIM y(500)
1177
1178 REM variabili
1179
1180 LET d=-30: LET w=0: LET m=1
: LET x=10: LET y=15: LET h=499:
 LET j=500: LET t=j: LET s=-1: L
ET d$=STR$ (INT (RND*4)+5): LET
l=5
1190 LET x(t)=3: LET y(t)=3
1197
1198 REM disegno dello schermo
1199
1200 LET a$="CCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCC": PRINT "Punti=000 H
```

```
i=000 di ";h$(lv,1);" Tempo";AT
0,15-LEN (STR$ h(lv,1));h(lv,1)
1210 PRINT a$;"-"; INK 5;"█"; FO
R i=2 TO 20: PRINT AT i,0;"C";TA
B 29;"C-"; INK 5;"█"; NEXT i: PR
INT a$;"0"; INK 5;"█"
1220 GO TO 500
3998
3999 REM definizione caratteri
4000
4010 FOR n=1 TO 8: READ z$
4020 FOR l=0 TO 7: READ z: POKE
USR z$+l,z: NEXT l: NEXT n
4029 REM dati
4030 DATA "b",60,126,255,255,255
,255,126,60
4040 DATA "u",66,66,195,231,231,
255,126,60
4050 DATA "d",60,126,255,231,231
,195,66,66
4060 DATA "l",60,254,31,7,7,31,2
54,60
4070 DATA "r",60,127,248,224,224
,248,127,60
4080 DATA "a",24,16,82,255,255,2
55,126,36
4090 DATA "c",129,90,36,153,66,3
6,153,165
4100 DATA "q",60,66,153,165,165,
153,66,60
4110 RETURN
4997
4998 REM fine della gara
4999
5000 PRINT INK m;AT x(j),y(j);"Q
"
5010 PRINT AT x(t),y(t);" ": LET
 t=t-1: PRINT AT x,y;"▚": BEEP .
05,12
5020 IF t=0 THEN LET t=500
5030 IF t=h THEN GO TO 5050
5040 PRINT AT x,y;"▞": BEEP .05,
0: GO TO 5010
5050 PRINT AT 10,10;"Gara finita
": FOR v=1 TO 300: NEXT v
5100 BORDER 4: PAPER 0: INK 7: C
LS
5110 PRINT "Hai mangiato ";s;"
mele";" a grado ";lv
5120 IF s<=h(lv,5) THEN PRINT
"La tavola dei record per questo
 grado e'",,: GO TO 5250
5130 FOR v=10 TO 30: BEEP .01,v:
 NEXT v: FOR v=29 TO 11 STEP -1:
 BEEP .01,v: NEXT v
5140 PRINT ,,"Hai stabilito il r
ecord         ";"    Batti il tuo no
me          ",TAB 5;"(massimo 6 lett
ere)",TAB 11;"<------>"
5150 LET b$="": LET b=12
5160 IF INKEY$<>"" THEN GO TO 51
```

```
50
5170 LET a$=INKEY$: IF a$="" THE
N GOTO 5170
5180 BEEP .005,10: IF CODE a$=13
THEN GO TO 5210
5190 IF CODE a$=12 THEN LET b=b-
(b>12): LET b$=b$(1 TO b-12): PR
INT AT 5,b;" ": GO TO 5160
5200 IF b<18 THEN LET b$=b$+a$:
PRINT AT 5,b;a$: LET b=b+1: GO T
O 5160
5210 FOR v=1 TO 5: IF s>h(lv,v)
THEN GO TO 5230
5220 NEXT v
5230 FOR n=5 TO v+1 STEP -1: LET
h(lv,n)=h(lv,n-1): LET h$(lv,n)
=h$(lv,n-1): NEXT n
5240 LET h(lv,v)=s: LET h$(lv,v)
=s$
5250 PRINT OVER 1;AT 6,0;"Pos.";
AT 6,0;" Nome";AT 6,
13;"_____ Punti";AT 6,25;"
_____";: FOR v=1 TO 5: PRINT AT 8
+v,0;v;TAB 12;h$(lv,v);TAB 20;h(l
v,v): NEXT v
5260 PRINT '' "Vuoi giocare ancor
a (y/n)"
5270 GO TO 5270+(2730 AND INKEY$
="y")
5280 IF INKEY$<>"" THEN GO TO 52
80
5290 PRINT '' "Vuoi le istruzioni
(y/n)"
5300 IF INKEY$="y" THEN GO TO 10
5310 IF INKEY$="n" THEN BORDER 4
: PAPER 0: INK 7: CLS : GO TO 10
5320
5330 GO TO 5300
5400
5410 REM fuori tempo
5420
5430 FOR v=60 TO -20 STEP -1: BE
EP .05,v: NEXT v
5440 PRINT AT 10,6;" Tempo scadu
to": FOR v=1 TO 300: NEXT v
5450 GO TO 5060
5460 CLS : PRINT AT 10,13; FLASH
1; BRIGHT 1; PAPER 1; INK 7;" C
IAO": STOP
5470
5480 REM dati per una nuova mela
5490
5500 DATA 0.05,12,0.025,7,0.025,
7,0.05,9,0.05,7
```

L' ultimo programma di questo capitolo riguarda una specie di "squash", trattandosi di ribattere una palla. Gli oggetti in movimento sono solo due ed i movimenti assai veloci. Vi sono

due "variabili di direzione", una per le coordinate "x" (variabile dx) e una per le coordinate "y" (variabile dy), le quali contengono un numero che puo' essere sia positivo che negativo dalla cui grandezza dipende la velocita' con la quale la palla viaggia nelle varie direzioni.
Se per voi il ritmo della gara e' troppo elevato, variate il contenuto delle linee che calcolano "dx" e "dy". La palla va colpita con la racchetta ogni qualvolta venga respinta dal muro. Il movimento della racchetta si ottiene con i tasti 5 e 8 in accordo con le relative frecce direzionali.

Elenco delle variabili

x	Coordinate x della palla
y	Coordinate y della palla
a	Valore temporaneo di "x" per cancellare
b	Valore temporaneo di "y" per cancellare
dx	Incremento/decremento di "x"
dy	Incremento/decremento di "y"
h	Altezza aggiornata della palla
p	Battuta del giocatore
s	Punti
true	"Marker" normalmente a 1, assume il valore 0 quando la palla viene mancata
ball	Numero aggiornato delle palle rimaste

Note

10	Inizializza
20-30	Disegna la palla, memorizza i vecchi valori e calcola i nuovi
40-60	Test per vedere dov' e' la palla
70-90	Muove la racchetta
100	Cancella la palla
110	Ritorno del loop
120-130	Palla mancata
140	Test per vedere se la partita e' finita
200-220	Preparazione della palla successiva
2000-2200	Nuovi valori delle variabili, colcolati quando la palla tocca il fondo. Notate la linea 2020
3000-3200	Nuovi valori calcolati per i bordi
4000-4200	Nuovi valori delle variabili inerenti al muro
5000-5100	Partenza della sola inizializzazione
5300-5700	Inizializzazione di ogni palla
6000-6400	Disegna lo schermo, i punti e le palle rimaste
8000-8030	"Bit" di fine gara

```
  10  GO SUB 5000: REM iniz
 110  REM inizio del loop
 120  PLOT x,y: LET a=x: LET b=y
 130  LET x=x+dx: LET y=y+dy
 140  IF y<6 THEN GO SUB 2000
 150  IF x<6 OR x>248 THEN GO SUB
3000
```

```
  600  IF  y>h  THEN  GO  SUB  4000
  700  PLOT  p,6:  DRAW  20,0
  800  LET  p=p+(5  AND  INKEY$="8"  A
ND  p<235)-(5  AND  INKEY$="5"  AND
p>0)
  900  PLOT  p,6:  DRAW  20,0
 1000  PLOT  a,b
 1110  IF  true  THEN  GO  TO  20
 1111  REM  fine  del  loop
 1200  LET  ball=ball-1:  LET  s=s-1
 1300  PRINT  AT  0,10-LEN  STR$  s;  O
VER  0;s
 1400  BEEP  1,0:  IF  ball=0  THEN  GO
TO  8000
 2000  PLOT  p,6:  DRAW  20,0
 2100  GOSUB  5300
 2200  GOTO  20
 3000  REM  racchetta  colpita
 0000  LET  dx=RND*SGN  dx*4+SGN  dx*
0110  LET  dy=RND*4+2
0000  LET  true=(x>=p  AND  x<p+21)
0000  LET  y=7
0700  LET  s=s+1:  PRINT  AT  0,10-LE
N  STR$  s;  OVER  0;s
1000  BEEP  .01,20
2000  RETURN
9000  REM  collisione  con  fianchi
0000  LET  dx=-(RND*SGN  dx*4+SGN  d
x*2)
0110  LET  dy=RND*SGN  dy*4+SGN  dy*
0220  LET  x=6+(242  AND  x>248)
1000  BEEP  .01,0
2000  RETURN
3000  REM  collisione  col  fondo
4000  LET  dy=RND*-4-2
4010  LET  dx=RND*SGN  dx*4+SGN  dx*
4000  PLOT  6,h:  DRAW  240,0
4030  LET  h=h-(2  AND  h>30)
4040  PLOT  6,h:  DRAW  240,0
4050  LET  y=h-1
4100  BEEP  .01,40
4200  RETURN
4900  REM  **inizializzazione**
4999  REM  solo  partenza
5000  LET  ball=3:  LET  s=0
5100  LET  h=166
5100  BORDER  4:  PAPER  1:  INK  7:  C
LS
5200  REM  Anche  reiniz
5300  LET  x=127:  LET  y=h-1
5310  LET  dx=RND*10-5:  LET  dy=-(R
ND*2+2)
5320  LET  p=115
5330  LET  true=1
5600  GOSUB  6000:  REM  schermo
5700  RETURN
6000  REM  Disegno  dello  schermo
```

```
60000  OVER 0
60100  PRINT AT 0,1;"PUNTI=000
     PALLE= 000"
60200  PRINT AT 0,10-LEN STR$ s;s;
  PRINT AT 0,31-ball;" FLASH 1;"O"
61000  PLOT 5,0: DRAW 0,167: DRAW
 45.0: DRAW 0,-167
61500  PLOT p,6: DRAW 20,0
62000  OVER 1
64000  RETURN
70000  REM **Fine della gara**
80000  PRINT AT 10,11; FLASH 1;"GA
     F INITA"
80010  BEEP 1,-12: BEEP 2,-24
80200  OVER 0: BORDER 7: INK 9
80300  STOP
```

Capitolo 6

SCROLLING

"Scrolling" e' il nome dato all' effetto per cui tutti gli oggetti disegnati sullo schermo (siano questi parole, spazi, simboli o altro) si muovono assieme nella medesima direzione. E' chiaramente visibile dando il comando LIST in presenza di programmi di una certa lunghezza.
Il computer lista una "videata", quindi si arresta presentando la richiesta 'scroll?' a cui l' operatore puo' rispondere a seconda delle proprie necessita' premendo un tasto qualsiasi per lo scrolling dello schermo verso l' alto, oppure digitando "n", "STOP" o "BREAK" (CAPS SHIFT+SPACE) per arrestare il programma. Questa particolare proprieta' viene sfruttata di buon grado in diversi games quando, ad esempio, si voglia ottenere una astronave in viaggio attraverso ad uno stormo di meteore. Facendo scorrere il paesaggio e mantenendo ferma, o quasi, la navetta, si ha infatti l' impressione che questa si muova in una certa direzione. In condizioni normali, pero', lo Spectrum presenta la scritta 'scroll?' dopo lo scorrimento di ogni 22 righe, il che, nel bel mezzo di un game, non e' molto ben accetto. A tale inconveniente si rimedia semplicemente effettuando una POKE, come avviene nella linea 20 del prossimo programma, e facendola seguire da uno statement di PRINT (linea 50) che porta la posizione di scrittura al limite inferiore dello schermo.
Siete al comando di una astronave che dovete pilotare con i tasti 5 e 8 i quali hanno il potere di cambiare la direzione di marcia per evitare collisioni con le meteore. Attenti perche' l' astronave che lascia dietro a se una coda luminosa, effetto dello scrolling, puo' rimbalzare sui fianchi dello schermo.
La lettera "A", presente alla linea 50, va battuta in "graphic mode".

```
   10 GO SUB 5000
   20 POKE 23692,-1: PRINT TAB RN
D*30;31; INK 2+RND*5.4;CHR$ (145+RN
D*3.4): LET a$=INKEY$
   30 IF a$="5" OR a$="8" THEN LE
T d=-1+(2 AND a$="8")
   40 IF p=0 OR p=31 THEN LET d=-
d
   50 LET p=p+d: PRINT AT 10,p; I
NK 8;"A";AT 21,0: LET n=n+1-(50
AND n=40): BEEP .01,n: LET s=s+1
: GO TO 20+(40 AND ATTR (10,p)<>
7)
   60 PRINT AT 12,12; FLASH 1;"GA
RA FINITA": FOR i=-20 TO 50: PRI
NT AT 10,p; INK RND*5+2;CHR$ (14
4+RND*4): BEEP .01,i: NEXT i
   70 FOR i=1 TO 22: POKE 23692,-
1: PRINT AT 21,0: PRINT : NEXT i
```

```
: PRINT AT 11,10;"Tuoi punti ";s
: STOP
50000 GO SUB 9000: REM UDG
50010 LET s=0: LET p=15: LET d=0:
LET n=0
50020 BORDER 0: PAPER 0: INK 7: C
LS : RETURN
9000 RESTORE 9100: FOR i=1 TO 5
9010 FOR n=0 TO 7: READ a
9020 POKE USR CHR$ (i+143)+n,a
9030 NEXT n: NEXT i: RETURN
9100 DATA 99,85,73,34,34,20,20,8
9110 DATA 102,153,65,66,132,132,
66,66
9120 DATA 7,121,129,130,66,33,34
,28
9130 DATA 32,80,76,66,129,130,68
,60
9140 DATA 110,146,132,66,65,129,
178,76
```

Ricorrendo al codice macchina (m/c), e' possibile ottenere anche lo scrolling a destra e a sinistra. Inserite la routine che segue e vedrete che tutti gli oggetti che escono dallo schermo in un determinato punto, riappaiono tali e quali nel punto diametralmente opposto dando adito ad un singolare effetto. Tenete presente che la subroutine gira sullo Spectrum da 48K, se avete la versione a 16K, cambiate i numeri che seguono :

Linea 10	da 64999 a 31999
Linea 8000	da 65000 a 32000
	da 65045 a 32045
Linea 8019	da 65000 a 32000
Linea 8049	da 65023 a 32023

Naturalmente i numeri piu' bassi sono dovuti alla minor memoria del 16 rispetto al 48K.

```
10 CLEAR 64999: REM salva lo s
pazio per m/c
8000 GO SUB 8000: REM m/c
8000 REM **scroll m/code**
8000 RESTORE 8100: FOR i=65000 T
O 65045: READ a: POKE i,a: NEXT
i: RETURN
8000
8019 REM RANDOMIZE USR 65000-
8020 REM SCROLL SINISTRO
8030
8049 REM RANDOMIZE USR 65023-
8050 REM SCROLL DESTRO
8060
8100 DATA 1,255,2,17,0,88,33,1,8
8,237,176,1,0,24,17,0,64,33,1,64
,237,176,201
```

```
8110 DATA 1,255,2,17,255,90,33,8
54,90,237,184,1,0,24,17,255,87,3
3,254,87,237,184,201
```

Uno dei pochi svantaggi dello scrolling, e' che transla tutto quanto riportato sullo schermo ivi compreso il punteggio, che, come si sa, permane costantemente nella parte alta.
Pertanto, usando un tale effetto, e' necessario memorizzare il risultato senza presentarlo se non alla fine del gioco.
Ed ora eccovi alcuni interessanti programmi basati sullo scrolling. Iniziamo da "Grand Prix". Siete alla guida di un bolide blu attraverso un tracciato di cui dovete evitare i bordi e nello stesso tempo le auto piu' lente che vi si presentano davanti. Lo spostamento coi soliti tasti 5 e 8.

Elenco delle variabili

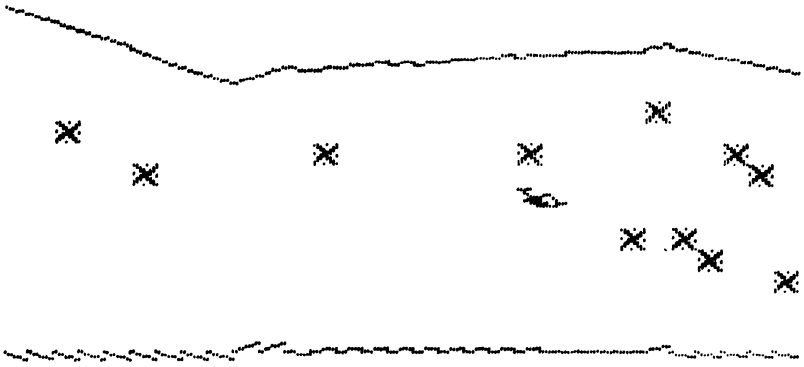
p	Coordinate della colonna dell' auto del giocatore
s	Punti
x(,)	Usato nella cancellazione del fumo
car	Posizione prossima dell' auto da corsa
ink	Colore dell' auto da corsa
rnd	Numero random per lo snodarsi della pista
life	Auto a disposizione (inizialmente 5)
l	Coordinata della colonna della pista
i, n a, END, LOOP	Variabili di controllo dei loop

Gli UDG presenti in questo programma, corrispondono alle lettere maiuscole di cui alle linee :
30,60,110,130,2000 e 5120.

```
  10 GO SUB 5000: REM iniz
  19 REM **Routine Principale**
  20 POKE 23692,-1: PRINT : REM
SCROLL
  30 PRINT INK 0;AT 9,p;"FG": PR
INT AT 10,p;" "
  40 LET p=p+(INKEY$="8")-(INKEY
$="5")
  50 IF ATTR (11,p)<>113 OR ATTR
 (11,p+1)<>113 THEN GO TO END
  60 PRINT INK 1;AT 10,p;"AB";AT
 11,p;"CD"
  70 LET s=s+1
  80 PRINT AT 7,x(1);"  "
  90 LET x(1)=x(2): LET x(2)=x(3
): LET x(3)=p
 100 LET car=INT (RND*9+l+2): LE
T ink=INT (RND*4+2)
```

```
  110 IF s/3=INT (s/3) THEN PRINT
 AT 20,car; INK ink;"AB": PRINT
AT 21,car; INK ink;"CD"
  120 LET rnd=INT (RND*3-1): LET
l=l+rnd-(l=19)+(l=1)
  130 PRINT INK 0;AT 21,l;"E": PR
INT INK 0;AT 21,l+12;"E"
  200 GO TO LOOP
 1999 REM **Fine della gara**
 2000 PRINT AT 10,p; INK 1;"AB":
PRINT AT 11,p; INK 1;"CD"
 2010 FOR i=20 TO -20 STEP -1: BE
EP .01,i: NEXT i
 2020 LET life=life-1
 2040 CLS : PRINT AT 8,6;"Tuoi pu
nti ";s;" con ancora"
 2050 PRINT AT 10,10;life;" autom
obili"
 2060 FOR i=1 TO 200: NEXT i
 2070 IF life>0 THEN GO SUB 5050:
 GO TO 20
 2080 PRINT 'TAB 9;"Altra gara y/
n ?"
 2090 GO TO 2090+(7900 AND INKEY$
="n")+(INKEY$="y")
 2100 GO SUB 5020: GO TO loop
 4999 REM **inizializzazione**
 5000 GO SUB 8000: REM UDG
 5010 LET END=2000: LET LOOP=20
 5020 LET s=0: LET life=5
 5050 LET p=15: DIM x(3): LET x(1
)=15
 5060 LET l=9
 5100 BORDER 0: PAPER 6: BRIGHT 1
: INK 1: CLS
 5110 FOR i=0 TO 20
 5120 PRINT INK 0;AT i,9;"E": PRI
NT INK 0;AT i,21;"E"
 5130 NEXT i
 5140 FOR i=9 TO 12: PRINT AT i,1
0;"▚▚▚▚▚▚▚▚▚▚▚": NEXT i
 5500 RETURN
 7999 REM *******UDG*********
 8000 RESTORE 8100: FOR a=1 TO 7
 8010 FOR n=0 TO 7: READ z
 8020 POKE USR CHR$ (143+a)+n,z
 8030 NEXT n: NEXT a
 8040 RETURN
 8099 REM AUTO
 8100 DATA 31,31,4,5,55,59,55,6
 8110 DATA 248,248,32,160,236,220
,236,96
 8120 DATA 6,6,7,19,31,19,1,7
 8130 DATA 96,96,224,200,248,200,
128,224
 8135 REM BORDI
 8140 DATA 255,129,165,153,153,16
5,129,255
 8149 REM FUMI
 8150 DATA 0,3,15,62,61,31,31,6
 8160 DATA 16,50,254,251,183,246,
```

```
254,28
9990 BORDER 6: PAPER 3: INK 7: B
RIGHT 0: CLS
9999 PRINT AT 10,13; FLASH 1;" C
IAO ": STOP
```



Note

10	Inizializzazione
20	Scroll dello schermo
30-90	Tratta il movimento dell' auto, delle nuvolette di fumo e rileva se la vettura ha picchiato
100-200	Sistema l' auto successiva e la pista
2000-2020	L' auto ha urtato qualcosa
2040-2060	Fine di una gara
2080-2100	Nuova partita
5010-5060	Inizializzazione variabili
5100-5140	Predisposizione dello schermo
8000-8040	Definizione UDG
8099-8160	Dati per gli UDG
9990-9999	Fine

Il prossimo game si avvale del codice macchina per effettuare lo scroll orizzontale. E' la versione semplificata del noto "Scramble". Dovete attraversare una lunga caverna le cui pareti si fanno sempre piu' strette senza incocciare nelle mine sospese qua' e la'. La navetta si pilota per mezzo dei tasti W,A,D e Z disposti nelle quattro direzioni cardinali; azionandone due assieme si ottiene lo spostamento diagonale. Tale effetto deriva dall' uso della funzione IN come sopra descritto. Il tasto A (sinistra) produce l' illusione del rallentamento della marcia, mentre il D, quello dell' accelerazione.

"s$","r$" e "q$" contengono rispettivamente una stringa vuota, la navetta e le mine. Nelle relative stringhe sono racchiusi i caratteri di controllo degli attributi in modo che le varie parti assumano l' appropriato colore per mezzo di semplici PRINT.
PRINTando "q$" si ottiene ad esempio una mina rossa lampeggiante.
"a$" usa il carattere di controllo di AT per presentare il disegno nella parte bassa dello schermo onde prevenire l' effetto di "wrap around" (gia' visto come la scomparsa di un oggetto da un lato e la sua riapparizione dal lato opposto) causato dalla subroutine in codice macchina.
POKE 23624,n viene spesso e volentieri impiegato per variare le due linee piu' basse dello schermo; qui' copre un effetto causato dal codice macchina visibile modificando la linea 5330.

Elenco delle variabili

x	Coordinata verticale della navetta
y	Coordinata orizzontale della navetta
s$	Stringa per cancellare la vecchia navetta
l	Variabili fittizie per eseguire la subroutine in codice macchina
a$	Stringa per eliminare il "wrap-around"
r$	Stringa per disegnare la navetta
s	Punti
g	Coordinata verticale del suolo della caverna
w	Altezza della caverna
d	Usato per dare forma ai muri della caverna
r1	Coordinata verticale random delle mine
r2	r1 convertito in numero di pixel per far capire al programma il rapporto tra r1 stesso e la larghezza della caverna
q$	Stringa per disegnare le mine
men	Numero delle navette rimaste
b$	Stringa fittizia usata in INPUT per dare il via ad una nuova ondata
i, n	Variabili di controllo per i loop FOR...NEXT.

Gli UDG usati nel programma sono tre. Eccoli : "AB" alla linea 80, "AB" alla linea 5120 e "C" alla linea 5130.

```
   10 CLEAR 31999: GO SUB 6000
   15 REM SUI issue 2.variare
   16 REM 251 in 167,253 in 169
   17 REM e 254 in 190
   20 PRINT AT x,y;s$: LET x=x-(I
N 64510=253)+(IN 65278=251): LET
 y=y-(IN 65022=254 AND y>1)+(IN
65022=251 AND y<220)
   30 LET l=USR 32000: PRINT a$;A
T x,y;r$: LET s=s+1: PLOT 239,9:
```

```
 DRAW 8,d: PLOT 239,g+w: DRAW 8,
d: LET g=g+d
  40 LET r1=INT (RND*21): LET r2
=(21-r1)*8: PRINT AT r1,30;(q$ A
ND r2>g+8 AND r2<g+w-8)
  50 IF g<16 OR g+w>159 THEN LET
 d=-d: GO TO 70
  60 IF RND>.8 THEN LET d=RND*10
-5
  70 LET w=w-(w>30 AND s=INT (s/
10)*10): GO TO 20+(60 AND (ATTR
(x,y)<>6 OR ATTR (x,y+1)<>6))
  80 FOR i=0 TO 40: PRINT AT x,y
; INK RND*4+3;"AB": BORDER RND*7
: BEEP .01,i: NEXT i
  90 BEEP 1,-10: LET men=men-1:
PRINT AT 0,3;"PUNTI=";s;"RESTO=
";men
 100 IF men>0 THEN POKE 23624,17
6: INPUT "ENTER per altra ondata
 "; LINE b$: GO SUB 5200: GO TO
 20
 110 POKE 23624,7
 120 PRINT AT 21,12; FLASH 1; IN
K 7;"GARA FINITA"
 130 PAUSE 100
 140 STOP
4999 REM **inizializzazione**
5000 GO SUB 9000: REM UDG
5010 GO SUB 8000: REM m/c
5020 LET s=0: LET w=100: LET men
=3
5100 LET a$=CHR$ 16+CHR$ 6: FOR
i=0 TO 21: LET a$=a$+CHR$ 22+CHR
$ i+CHR$ 31+" ": NEXT i
5110 LET s$=CHR$ 16+CHR$ 6+"  "
5120 LET r$=CHR$ 16+CHR$ 8+"AB"
5130 LET q$=CHR$ 16+CHR$ 2+CHR$
18+CHR$ 1+"C"
5200 LET x=12: LET y=5
5210 LET g=(176-w)/2: LET d=RND*
10-5
5300 BORDER 0: PAPER 0: INK 6: C
LS : INK 4
5310 PLOT 0,16: DRAW 150,0: DRAW
 0,(143-w)/2
5320 PLOT 0,159: DRAW 150,0: DRA
W 0,(w-143)/2
5330 POKE 23624,0: INPUT ,,
5400 BEEP 1,10
5410 RETURN
7999 REM **scroll m/code**
8000 RESTORE 8100: FOR i=32000 T
O 32022: READ a: POKE i,a: NEXT
i: RETURN
8100 DATA 1,255,2,17,0,88,33,1,8
8,237,176,1,0,24,17,0,64,33,1,64
,237,176,201
8999 REM UDG
9000 RESTORE 9100
```

```
9010 FOR i=1 TO 3
9020 FOR n=0 TO 7: READ a: POKE
USR CHR$ (i+143)+n,a: NEXT n
9030 NEXT i
9040 RETURN
9100 DATA 0,248,32,16,15,63,15,3
9110 DATA 0,0,0,224,16,8,207,248
9120 DATA 201,99,54,156,57,103,1
03,147
```

Note

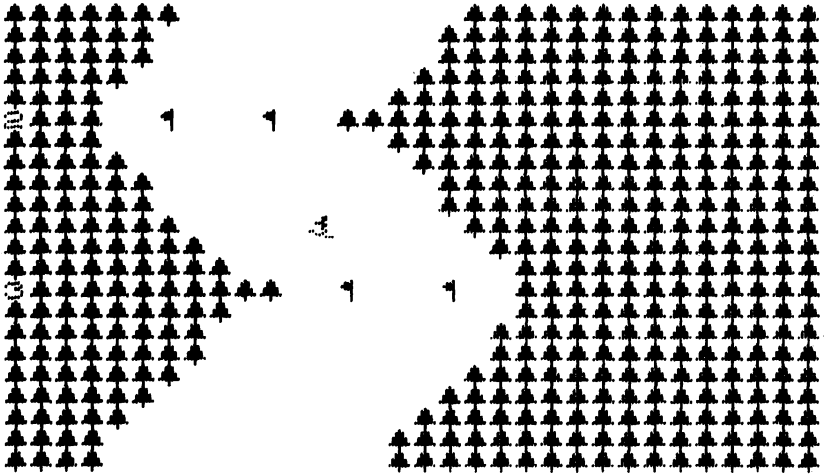
10	Salvataggio della memoria per il codice macchina e inizializzazione
20	Cancella la navetta, predispone gli INPUT e muove
30	Effettua lo scroll, aggiorna il punteggio e disegna la caverna
40	Disegna le mine
50	Controlla se la caverna esce dallo schermo
60	Varia l' inviluppo della caverna
70	Riduce l' ampiezza e rileva se la navetta ha urtato qualcosa
80-90	Urto
100	Predispone una nuova ondata
110-140	Fine della gara
5020-5130	Variabili da non ridefinire
5200-5210	Altre variabili
5300-5330	Predispone lo schermo
5480	Introduce il suono
8000-8100	Predispone la routine di m/c
9000-9040	Definisce gli UDG
9100-9120	Dati per gli UDG

L' ultimo game di questo capitolo, presenta uno scrolling verticale. Dovete controllare uno sciatore rosso facendolo scendere in slalom attraverso 25 porte. Se urta una bandierina, un albero o salta una porta, la gara ha termine.

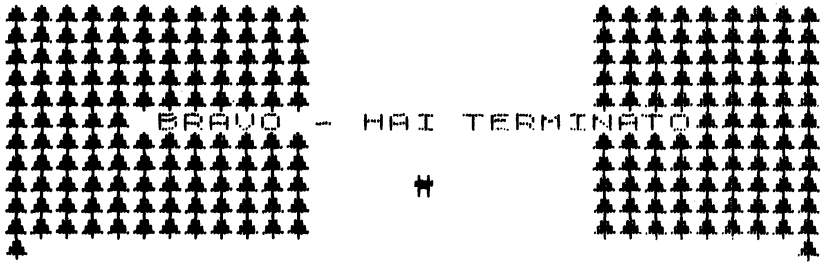
I controlli a vostra disposizione sono : F-sinistra, G-giu' e H-destra. Una volta premuto il tasto, lo sciatore prosegue nella stessa direzione fino a che non azionate un altro tasto.

Elenco delle variabili

n	Loop dello sciatore prima che appaia la porta
p	Coordinata della colonna dello sciatore
a$	Stringa contenente l' input da tastiera
d	Direzione dello sciatore
g	Posizione delle porte
r	Numero random relativo alla posizione della porta successiva
r1	Aggiunto a "g" per cercare la posizione in cui piazzare gli alberi



z$	Stringa contenente una fila intera di alberi presentata con dei PRINT
a(.)	Array relativo al motivetto
m	Si alterna tra 0 e 8 per suonare la prima o la seconda meta' della melodia
e	Marker per rilevare se lo sciatore si e' schiantato o no
s	Loop che predispone il conteggio delle porte
g$	Stringa contenente la porta. Contiene i caratteri di controllo del colore. Lo sciatore deve passarvi in mezzo ma non attorno
n, i	Usate come variabili di controllo dei loops



Caratteri UDG sono presenti alle linee 310-graphic B; 320 graphic E; e 5120-graphic D.

```
   10  GO TO 5000: REM Iniz
   19  REM Loop principale
   20  FOR n=1 TO 8
   30  PRINT AT 10,p;" ": POKE 236
92,-1: PRINT AT 21,0: PRINT
   40  LET a$=INKEY$: IF a$>"e" AN
D a$<"i" THEN LET d=CODE a$-101
   50  LET p=p+d-2: IF NOT p OR p=
31 THEN LET d=2
   60  PRINT AT 10,p; INK 8;CHR$ (
143+d)
   70  LET g=g+r1
   80  PRINT AT 21,0; INK 4;z$( TO
 g-5);: INK 2;TAB g+6; INK 4;z$(g
+7 TO )
   90  BEEP (.05 AND n<>8),a(m+n)
  100  IF ATTR (10,p)<>56 THEN LET
 e=1: LET n=8
  110  NEXT n: RETURN
  149  REM Loop di controllo
  150  FOR s=1 TO 25: GO SUB 20
  160  IF e THEN GO TO 500
  170  LET m=m+(8 AND NOT m)-(8 AN
D m)
  180  LET r=INT (RND*25-12): GO T
O 120+(g+r>5 AND g+r<24)
  190  LET r1=r/10
  200  PRINT AT 21,0; INK 4;z$( TO
 g-4);g$; INK 4;z$(g+7 TO ): PRI
NT AT 21,0: FLASH 1;s
  210  NEXT s
  211
  249  REM Fine gara
  250  LET r1=0: FOR i=1 TO 2: GO
SUB 20: IF e THEN GO TO 500
  260  NEXT i
  261
  300  REM Fatto !
  309  FOR i=1 TO 10: BEEP .1,RND*
30: BORDER RND*7
  310  PRINT AT 0,p;" ": POKE 2369
2,-1: PRINT AT 10,p;"B";AT 21,0
  320  PRINT INK 4;"E";TAB 31;"E":
 NEXT i
  330  PRINT INK 4;z$: PRINT AT 5,
5; FLASH 1;" BRAVO - HAI TERMINA
TO"
  400  STOP
  499  REM CRASH!
  500  FOR i=1 TO 25: BORDER RND*7
: BEEP .05,i: PRINT AT 10,p; INK
 RND*8;CHR$ (143+d): NEXT i
  510  PRINT AT 5,5; FLASH 1;"AUGU
RI PER LA PROSSIMA"
  520  STOP
 5000  GO SUB 9000
 5100  LET m=8
 5110  LET z$="EEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEE"
 5120  LET g$=CHR$ 16+CHR$ 0+"  D"
+CHR$ 16+CHR$ 2+"    "+CHR$ 16+CH
```

```
R$ 0+"D   "
5150 DIM a(16): RESTORE 8500
5160 FOR i=1 TO 16: READ a(i): N
EXT i
5200 LET p=15: LET d=2
5210 LET r=0: LET r1=0
5220 LET e=0: LET g=15
5300 BORDER 4: PAPER 7: INK 2: C
LS
5490 GO TO 150
5500 REM Dati per motivo
5510 DATA 0,0,0,0,9,9,11,0,0,0,0
,5,5,7,0
9000 RESTORE 9000: FOR i=1 TO 5:
 FOR n=0 TO 7: READ a: POKE USR
CHR$ (i+143)+n,a: NEXT n: NEXT i
: RETURN
9100 DATA 48,48,28,48,16,37,28,2
4
9110 DATA 36,36,126,126,60,36,36
,36
9120 DATA 12,12,56,12,8,196,56,7
9130 DATA 4,28,60,28,4,4,4,4
9140 DATA 8,28,28,62,62,127,8,8
```

Note

20-110	La routine principale e' una subroutine usata in due occasioni
30	Cancella il vecchio sciatore ed effettua lo scroll
40-50	Movimento dello sciatore
60	Disegna il nuovo sciatore
70-80	Disegna gli alberi
90	Suono
100	Rileva l' eventuale urto
150-210	Loop delle porte che richiama la routine principale
180-200	Disegna la porta
250-260	Presenta lo sciatore alla fine, dopo l' ultima porta
300-400	Fine
500-520	Routine dell' urto e fine
5000	Definisce gli UDG
5110-5120	Stringhe per disegnare alberi e porte
5130-5160	Inserisce i dati nell' array per il motivetto
5200-5220	Definisce le variabili
5300	Colori
9000	Definisce gli UDG
9100-9140	Dati per gli UDG

Capitolo 7

USO DEGLI ARRAY

Gli array (matrici) sono delle variabili e come tali vanno usati anche se e' possibile affidargli diversi altri compiti. Esistono, come per le variabili, due tipi di array : quelli numerici che contengono unicamente numeri e quelli di caratteri che memorizzano lettere o stringhe. Per comprenderne bene la struttura, immaginateli disposti come una tabellina divisa in settori ciascuno dei quali ospita un numero o un carattere.
Prima di poterli usare, vanno dimensionati per mezzo 'dello statement DIM :

```
10 DIM A (500)
```

In tal modo si prenota lo spazio in memoria per la "tabella" di nome A, la quale inizialmente ha tutti i suoi elementi a 0.
Il nome e' seguito da un numero racchiuso tra parentesi chiamato "subscript" della variabile.
Analogamente avviene per gli array di caratteri.

```
10 DIM A$ (500)
```

significa che l'array A e' formato da 500 elementi ognuno dei quali contiene un carattere che inizialmente e' uno spazio.
Provate ad aggiungere alla linea di cui sopra :

```
20 LET A$ (1)="Ciao"
```

e quindi date il RUN. Fatto cio' inserite il comando diretto, PRINT A$ (1) e vedrete che la risposta sullo schermo non sara' "Ciao" bensi' solo "C" in quanto ogni singolo elemento non puo' memorizzare piu' di un carattere. Volendo pertanto sistemare "Ciao" in un array dovrete ricorrere a piu' di una dimensione come dimostrato facilmente da :

```
10 DIM A$(100,5)
```

In questo modo avrete dimensionato una matrice da 100*5=500 elementi in totale. Battete ora :

```
20 LET A$(1)= "Ciao"
```

fatelo girare e quindi date PRINT A$(1). A questo punto la risposta sara' "Ciao" in virtu' del fatto che A$(1) viene considerato come un array a se' stante lungo 5 caratteri. L' analisi di ogni sua parte e' facilmente ottenibile studiando lo statement :

```
30 PRINT A$(1),A$(1,3)
```

il quale mostra sia A$(1) per intero, sia il suo terzo carattere. Riscrivendo A$(1,3) come A$(1)(3), otterrete una ulteriore tavola di altre due dimensioni. Cio' per dimostrare come un array possa avere quante dimensioni si voglia. Ad esempio, DIM A$(3,6,7,2) ha quattro dimensioni e possiede 3*6*7*2=252 elementi. Naturalmente, ogni elemento deve venire caricato con lo stesso numero di "subscript"; non ha senso, ad esempio,

```
10 DIM A(10,2)
20 LET A(1)=7
```

poiche' il computer non sa se caricare il 7 in A(1,1) oppure in A(1,2).
Usando gli array con un loop continuo, potete generare motivetti con dei BEEP random disponendo appropriati valori in ogni elemento :

```
  10 GO SUB 5000
  20 BEEP .2,a(m): LET m=m+1-(64
AND m=64)
  30 REM Resto del programma
 200 GO TO 20
5000 DIM a(64)
5010 LET m=1
5020 RESTORE 8000
5030 FOR a=1 TO 64
5040 READ a(a)
5050 NEXT a
5490 RETURN
8000 DATA -5,-3,-5,0,-5,0,-5,0,-
5,0,-5,4,4,2,4,0
8010 DATA -5,2,-5,5,-5,5,-5,2,-
5,2,-5,5,5,4,5,2
8020 DATA -5,-3,-5,4,4,4,4,2,2,2
,0,0,2,4,5
8030 DATA -5,-3,-5,2,-5,4,-5,5,-
5,7,-5,-5,-3,-1,2,0
```

Constaterete voi stessi questo piacevole effetto nell' ultimo programma del capitolo.
Gli array possono anche contenere le coordinate degli oggetti da muovere sullo schermo. Un esempio di cio', lo potete avere sia nel game "Zap", in cui l' incremento di ogni singola variabile muove a turno un invader, sia nel programma che segue intitolato "Gara delle pulci".

```
10 GO SUB 5000
20 FOR i=1 TO 5
30 PRINT AT i*3+2,x(i);" "
40 LET x(i)=x(i)+RND*3
50 PRINT AT i*3+2,x(i);"A"
60 IF x(i)>27 THEN GO TO 90
```

```
  70 BEEP .05,RND*30-15: NEXT i
  80 GO TO 20
  90 PRINT AT i*3+2,30; FLASH 1;
 BRIGHT 1;"B"
 100 BEEP 1,0
 110 CLS
 120 PRINT AT 3,9;"Numero ";i;"
vince"
 130 FOR n=1 TO 2
 140 IF p(n)=i THEN PRINT AT 5,8
;"Gioc. ";CHR$ (n+64);" vince £
";b(n): LET c(n)=c(n)+b(n)*2
 150 IF c(n)=0 THEN PRINT 'TAB 8
;"Gioc. ";CHR$ (64+n);" perde "
''TAB 7;"Gioc. ";CHR$ (65+NOT (
n-1));" vince "
 160 NEXT n
 170 IF c(1)=0 OR c(2)=0 THEN ST
OP
 180 GO SUB 5100
 190 GO TO 20
4999 REM iniz:
5000 DIM x(5)
5010 DIM p(2)
5020 DIM b(2)
5030 DIM c(2)
5040 LET c(1)=25: LET c(2)=25
5050 GO SUB 9000
5060 GO SUB 5100
5070 RETURN
5100 FOR i=1 TO 5: LET x(i)=2: N
EXT i
5110 FOR i=1 TO 2
5120 PRINT '"Gioc. ";CHR$ (i+64
);", con numero (1-5) ? ";
5130 LET a$=INKEY$: IF a$<"1" OR
 a$>"5" THEN GO TO 5130
5140 LET p(i)=VAL a$
5150 IF i=2 AND p(1)=p(2) THEN G
O TO 5130
5160 PRINT a$''" Quanto scommett
i       ";CHR$ (i+64);" ?"'" Hai
     £";c(i),"        ";
5170 INPUT b(i)
5180 IF b(i)<>INT b(i) OR b(i)>c
(i) OR b(i)<1 THEN GO TO 5170
5190 PRINT "£";b(i)
5200 LET c(i)=c(i)-b(i)
5210 NEXT i
5220 BEEP 1,10
5300 BORDER 2: PAPER 4: INK 0: C
LS
5310 PRINT 'TAB 10;"Gara delle p
ulci"
5320 FOR i=1 TO 5: PRINT AT i*3+
2,0;i;" A": NEXT i
5330 RETURN
9000 RESTORE 9100: FOR i=1 TO 2:
 FOR n=0 TO 7
9010 READ a: POKE USR CHR$ (i+14
3)+n,a
```

```
9020 NEXT n: NEXT i: RETURN
9100 DATA 102,102,24,129,126,0,3
0,231
9110 DATA 51,51,204,204,51,51,20
4,204
```

Gli UDG sono insiti nel listato alle linee 50-graphic A, 90-graphic B, 5320-graphic A.
Ogni giocatore puo' scommettere su quale pulce vincera'; se entrambi sbagliano, perdono la cifra puntata mentre se vincono incrementano il bottino.
Le matrici possono essere impiegate anche come "marker" dello stato di qualsiasi cosa presente sullo schermo. Lo vedremo tra poco nell' impegnativo programma "ICBM".
Gli array di stringhe trovano particolare uso nella presentazione del display. Battete il programmino che segue il quale, anche senza l' uso di array, da' l' impressione del movimento continuo :

```
10 LET a$=""
20 LET i=1
30 BORDER 0: PAPER 0: CLS
40 PAPER 7: INK 1
50 PRINT AT 10,1;a$(i TO 30);a
$(1 TO i)
60 LET i=i+1-(30 AND i=30)
70 GO TO 50
```

Sullo stesso principio, il prossimo che di linee in movimento ne presenta tre :

```
10 LET a$=""
20 LET i=1
30 BORDER 0: PAPER 0: CLS
40 PAPER 7: INK 1
50 FOR n=1 TO 3
60 PRINT AT 8+n*2,1;a$(i TO 30
);a$(1 TO i)
70 NEXT n
80 LET i=i+1-(30 AND i=30)
90 GO TO 50
```

Piu' difficile si presenta il discorso quando le tre linee devono scorrere in direzioni differenti. Il programmino che segue impiega due variabili denominate "i" per sfilare la linea di centro verso destra e le due laterali verso sinistra.

```
10 LET a$=""
20 LET il=1: LET ir=1
```

```
   30  BORDER 0: PAPER 0: CLS
  400  PAPER 7: INK 1
  500  FOR n=1 TO 3
  505  LET it=(il AND n<>2)+(ir AN
D n=2)
  600  PRINT AT 8+n*2,1;a$(it TO 3
0);a$(1 TO it)
  700  NEXT n
  800  LET il=il+1-(30 AND il=30)
  805  LET ir=ir-1+(30 AND ir=1)
  900  GO TO 50
```

Il risultato e' accettabile, ma desiderando variare la velocita' di scorrimento e' necessario apportare complicazioni al programma. E qui subentrano gli array. Per ogni riga, sono a disposizione diversi "quadri" ordinati in modo che la loro sequenza dia l' impressione del movimento. I "quadri" sono memorizzati in un array e le righe in un' altra dimensione dello stesso array.
Per ottenere il movimento di tre righe separate, dovrete ricorrere allo statement DIM A$(3,n1,n2) in cui "n1" e' il numero dei quadri prima che la sequenza si ripeta e "n2" e' la lunghezza di ogni quadro nell' array di stringhe. I due programmi che seguono, mostrano gli array in azione. Nel primo, "Snakey", disponete di un serpente (formato da una fila di "V") che voi dovete guidare in uno stretto cunicolo badando di evitare i blocchi che vi si fanno incontro. La testa del serpente, rappresentata dalla prima "V", la spostate a sinistra con il tasto 5 e a destra con il tasto 8. La partita ha termine dopo aver urtato dieci blocchi ognuno dei quali viene segnalato a fianco con il numero relativo lampeggiante.

Elenco delle variabili.

a(.)	Array con i dati del motivo
c	Contatore per la parte di array interessata
a	Posizione dei blocchi
p	Numero della colonna con la testa del serpente
n	Numero dei blocchi urtati
s	Punti
i	Controllo dei vari loop
v	Durata della nota del motivetto
x	Tono della nota
x$	Scrive i messaggi a fine gara
a	Variabile di controllo per predisporre a(.)

```
   10  CLS
  200  GO SUB 5000
  300  LET a=INT (RND*4)+10
  400  POKE 23692,-1: PRINT AT 21,
0; PRINT
   50  LET p=p+(INKEY$="8" AND p<1
3)-(INKEY$="5" AND p>10)
   60  IF ATTR (16,p)=15 THEN LET
n=n+1: BEEP .03,24: PRINT AT 21,
```

```
15; FLASH 1;n: IF n=10 THEN GO T
O 200
  700 PRINT AT 21,9;" ";TAB 14;"
";AT 16,p;"U": PRINT AT 21,a; PA
PER 2;"
  800 BEEP .03,a(c): LET c=c+1-(6
4 AND c=64)
  900 LET s=s+1
 1000 GO TO 30
 2000 PRINT AT 16,p;"U";AT 21,9;"
 "
 2100 RESTORE 8100: FOR i=1 TO 12
: READ v,x: BEEP v,x: NEXT i
 2200 PRINT AT 10,18;"Tuoi punti=
";AT 12,18;s
 2300 PRINT AT 14,18;"Altra gara?
";AT 16,18;"Premi Y o N"
 2400 GO TO 2400-(230 AND INKEY$="
y")+(INKEY$="n")
 2500 CLS : RESTORE 8200: PRINT A
T 8,10;: FOR i=1 TO 14: READ v,x
,x$: PRINT x$;: IF i/3=INT (i/3)
 THEN PRINT TAB 10;
 2600 BEEP v,x: NEXT i
 4000 STOP
 5000 DIM a(64): RESTORE 8000
 5010 FOR a=1 TO 64: READ a(a): B
EEP .01,a: NEXT a
 5020 LET s=0: LET n=0: LET c=1
 5300 LET p=12
 5400 RETURN
 8000 DATA 7,0,12,12,7,0,12,12,7,
0,12,-1,11,11
 8100 DATA 5,-1,11,11,5,-1,11,11,
0,11,-1,11,0,12,12
 8200 DATA 7,16,12,12,7,16,12,12,
0,12,16,12,14,11,11
 8300 DATA 5,14,11,11,5,14,11,11,
0,11,14,11,0,12,12
 8100 DATA .1,7,.1,4,.2,0,.2,0,.1
,0,.1,2,.1,4,.1,5,.2,7,.2,7,.2,7
,.1,4
 8200 DATA .1,14,"Arri",.1,16,"ve
de",1.1,12,"rci"
 8210 DATA .1,9,"Arri",.1,7,"vede
",1.1,14,"rci"
 8220 DATA .4,19,"La",.3,17,"cri"
,2.1,14,"me"
 8230 DATA .4,19,"Lacrime",.3,17,
"nei",.1,14,"miei"
 8240 DATA .1,16,"occhi",1.2,12,"
iii !"
```

Note

```
20          Inizializzazione
40          Scroll
50          Mosse del giocatore
60          Rivela gli urti
```

70	Disegna il serpente e i blocchi
80	Musica
90	Punti
200-210	Fine della partita
220-240	Bit di fine gara
250-4990	Fine del programma
5000-5010	Inserisce i dati nell' array
5020-5030	Definisce le altre variabili
8000-8030	Dati per il motivo durante la gara
8100	Dati per il motivo di fine gara
8200-8240	Dati per messaggio a fine programma

Il secondo programma,"ICBM", e' una versione del piu' noto "Missile command".
Scopo del game, e' il salvataggio di sei citta', mostrate sottoforma di blocchi, da uno stormo di missili che si presentano ad ondate successive di venti elementi, ognuna piu' veloce della precedente. Potete distruggere i missili in caduta posizionandovi al disotto e sparando con lo 0.
La vostra posizione varia in senso antiorario col tasto "6" e in senso orario col "7". Se avete qualche perplessita', consultate la freccia nella parte superiore dello schermo. Il game ha termine con la distruzione delle sei citta'. Ad ogni 10.000 punti vi viene regalata una citta', il numero dei punti aumenta ad ogni ondata e varia a seconda dei missili colpiti e delle munizioni risparmiate...il resto lo scoprirete da voi.

Elenco delle variabili

mis	Missile in discesa dal cielo
f(.)	Array relativo ai missili
m(.)	Punto di origine dei missili situato sulla parte alta dello schermo
x(.)	Coordinata orizzontale della testa del missile
y(.)	Coordinata verticale della testa del missile
d(.)	Numero dei pixel necessario al missile per raggiungere la citta' da distruggere
b(.)	Citta' selezionata dal missile. Se e' la 3 il missile centra la base
c(.)	Array che riporta quale delle citta' sia stata distrutta
a(.)	Citta' selezionata (in b(.)) sottoforma del numero di pixel orizzontale
c	Numero delle citta' distrutte
x	Coordinata orizzontale della base
y	Coordinata verticale della base
b$	Usata per l' input
dir	Direzione della base 0=su, 1=su a destra, 2 a destra ecc...
move	Numero dei pixel della base mossi ogni volta (diminuisce man mano che la gara si fa piu' difficoltosa)
ammo	Munizioni residue
count	Missili per ogni ondata

ink	Colore di INK
z	Usato come controllo dei loop per vedere se ogni missile colpisce (linee 420-460)
n1	Valore per predisporre un nuovo missile
sc	Punti
dd	Grado di difficolta'
paper	Colore di PAPER
te	Valore temporaneo di b(.) per calcolare la scrittura della colonna per l' esplosione della citta'
city	Quale citta' e' esplosa
mark	Usato per calcolare se tutte le citta' sono state colpite
wave	Conto delle ondate
abm	Punti delle munizioni rimaste
city	Punti delle citta' rimaste
down	Numero dei pixel relativi ai missili caduti dopo ogni ondata
ink 1	Colore delle munizioni a terra
sc 1	Punteggio in 10.000, usato per calcolare quando dare una citta' gratis
xc	Memorizzazione delle citta' elargite. Se tutte le citta' sono ancora in piedi dopo aver raggiunto i 10.000 punti, la citta' viene conservata per ulteriori bisogni
nc	Posizionamento random della nuova citta'
a1, a2, a3	Usati per la casualita' dei colori di INK e PAPER

```
   10 GO TO 3000: REM INIZ
   30 GO TO 5000
   90 REM LOOP
  100 LET mis=mis+1-(4 AND mis=4)
  110 IF f(mis) THEN GO TO 200
  120 INVERSE 1
  130 PLOT m(mis),163: DRAW x(mis
),-m(mis),y(mis)-163
  140 LET y(mis)=y(mis)-down
  150 LET x(mis)=x(mis)+(d(mis)*d
own)
  160 INVERSE 0
  170 IF y(mis)<16 THEN GO SUB 60
0: GO TO 190
  180 PLOT m(mis),163: DRAW x(mis
),-m(mis),y(mis)-163
  190 IF c=6 THEN GO TO 4000: REM
FINE
  200 INVERSE 1: PLOT x-2,y: DRAW
4,0: PLOT x,y-2: DRAW 0,4
  210 LET b$=INKEY$: IF b$="6" TH
EN LET dir=dir-1+(8 AND dir=0)
  220 IF b$="7" THEN LET dir=dir+
1-(8 AND dir=7)
```

```
  230  LET x=x+move*((dir>0 AND di
r<4) AND x<248)-(dir>4 AND dir<=7
 AND x>7))
  240  LET y=y+move*(((dir=7 OR di
r=0 OR dir=1) AND y<150)-(dir>2
AND dir<6 AND y>23))
  250  INVERSE 0: PLOT x-2,y: DRAW
 4,0: PLOT x,y-2: DRAW 0,4
  260  IF INKEY$="0" AND ammo>=5 T
HEN GO SUB 400
  270  PRINT AT 0,10; INK 2; PAPER
 6;CHR$ (140+dir)
  300  IF countt<24 THEN GO TO 100
  301  REM fine del loop
  310  GO TO 1000: REM riavvio
  400  REM ********FUOCO********
  410  INK 9: PLOT 127,16: DRAW x-
127,y-16: INK ink: INVERSE 1
  420  BEEP .005,10+RND*20
  430  FOR z=1 TO 4: IF y+10<y(z)
OR y>y(z)+10 OR x+8<x(z) OR x-8>x(z
) THEN GO TO 460
  435  PLOT m(z),163: DRAW x(z)-m(
z),-163
  440  LET n1=z: GO SUB 700
  445  LET sc=sc+dd*10+40
  460  NEXT z: LET ammo=(ammo-5 AN
D ammo>=5)
  470  INK 9: PAPER ink1: PLOT 127
,16: DRAW x-127,y-16: PLOT 240,0
: DRAW ammo-199,0
  480  INVERSE 0: INK ink: PAPER p
aper: GO SUB 500: RETURN
  490  REM PUNTI
  500  PRINT AT 0,30-LEN STR$ sc;
 sc: INK 2: RETURN
  550  REM atterraggio del missile
  600  INK 9
  610  IF b(mis)=3 THEN GO TO 800
  620  LET te=b(mis)*4+3
  630  PRINT AT 20,te;":";AT 20,t
e-1;"CD": PRINT AT 19,te;"AB"
  640  BEEP .008,-10
  650  INK ink: PRINT AT 20,te;"MN
": BEEP .008,-20
  660  LET city=b(mis)+(b(mis)<3)
  665  IF c(city)=0 THEN LET c(cit
y)=1: LET c=c+1
  670  LET n1=mis: GO SUB 700: RET
URN
  699  REM **nuovi missili**
  700  LET count=count+(NOT f(n1))
  710  IF count>20 THEN LET f(n1)=
1: GO TO 750
  720  LET m(n1)=INT (RND*256): LE
T b(n1)=INT (RND*7)
  730  LET a(n1)=b(n1)*32+32: LET
 d(n1)=(a(n1)-m(n1))/140
  740  LET x(n1)=m(n1): LET y(n1)=
```

```
100
  7550  RETURN
  8000  LET  ammo=(ammo-50  AND  ammo>
=500):  INVERSE 1
  8110  PLOT  246,3:  DRAW  ammo-199,0
:  INVERSE  0
  8200  BEEP  .01,10
  8300  GO TO 500
  9000  REM  ****Riavvio****
 10000  LET  marker=0:  LET  wave=wave
+1:  IF  wave=1  THEN  LET  wave=-1:
LET  dd=dd+(dd<6)
 1010  LET  abm=5*dd:  LET  icbm=25*d
d:  LET  city=100*dd
 1020  FOR  i=1  TO  6:  IF  c(i)  THEN
LET  marker=1:  GO  TO  1070
 1030  PRINT  AT  20,i*4+(4  AND  i>3)
-1;"_"
 1040  LET  sc=sc+city
 1050  GO SUB  600
 1060  BEEP  .3,-30
 1070  NEXT  i
 1080  LET  down=down+2:  IF  down=12
 THEN  LET  move=move-(move>2):  LE
T  down=4
 1090  FOR  i=ammo  TO  1  STEP  -1:  LE
T  sc=sc+abm:  BEEP  .0025,i/4:  PLO
T  INK  9:  PAPER  ink1:  INVERSE  1;i
+4;3:  GO  SUB  500:  NEXT  i
 1100  IF  sc-sc1<1e4  AND  xc=0  THEN
 GO  TO  1190
 1109  REM  Città  libera
 1110  LET  sc1=sc1+(1e4  AND  sc>=sc
1+1e4):  LET  xc=xc+1
 1130  LET  mark=1:  FOR  i=1  TO  6:  I
F  c(i)  THEN  LET  mark=0
 1140  NEXT  i
 1150  IF  mark  THEN  GO  TO  1190
 1160  LET  nc=INT  (RND*6)+1:  IF  c(
nc)=0  THEN  GO  TO  1160
 1170  LET  c(nc)=0
 1180  LET  xc=xc-(2  AND  xc>1):  GO
TO  1100
 1190  GO  TO  3200
 1999  REM  Disegno  dello  schermo
 2000  LET  a1=INT  (RND*4):  LET  a2=
-(INT  (RND*(4-a1))):  LET  a3=INT
 (RND*2)
 2010  INVERSE  0:  BRIGHT  0:  FLASH
 0:  OVER  0
 2020  LET  ink=(a1*a3)+(a2*NOT  a3)
:  LET  paper=(a2*a3)+(a1*NOT  a3)
 2030  LET  ink1=INT  (RND*8):  IF  in
k1=ink  OR  ink1=paper  THEN  GO  TO
 2030
 2040  INK  ink:  PAPER  paper
 2050  BORDER  ink:  CLS
 2060  PRINT  "direzione                PU
NTI:";
 2070  PRINT  AT  21,0;  PAPER  ink1;
```

```
INK 9;" mun.
2080 GO SUB 500
2090 INK 9: PLOT 47,1: DRAW 0,5:
 PLOT 247,1: DRAW 0,5: PLOT 47,3
: DRAW ammo,0
2100 FOR i=0 TO 6
2110 INK ink: IF i=3 THEN INK in
k1
2120 IF c(i+(i<3)) AND i<>3 THEN
 PRINT AT 20,i*4+3;"MN": GO TO 2
160
2130 FOR n=1 TO 8
2140 PLOT i*32+23+n/(2-(i=3)),7+
n: DRAW 16-n*(1+(i=3)),0
2150 NEXT n
2160 NEXT i
2500 RETURN
2999 REM Inizializzazione
3000 GO SUB 9000: REM UDG
3009 REM Istruzioni
3099 REM Array della citta'
3100 LET c=0: DIM c(6)
3110 LET sc=0: LET sc1=0: LET mi
s=0
3120 LET down=2: LET move=6: LET
 dd=1
3130 LET wave=-1: LET xc=0
3140 LET icbm=25
3199 REM Iniz per ICBM
3200 DIM m(4): DIM x(4): DIM y(4
)
3210 DIM b(4): DIM a(4): DIM d(4
): DIM f(4)
3220 LET count=0: LET ammo=200
3230 LET dir=INT (RND*8)
3240 FOR i=1 TO 4: LET n1=i: GO
SUB 700: NEXT i
3299 REM Iniz per base
3300 LET x=127: LET y=30: LET y1
=y: LET x1=x
3400 GO SUB 2000
3500 GO TO 100
3999 REM Fine della gara
4000 STOP
8999 REM UDG
9000 RESTORE 9100: FOR i=1 TO 14
: FOR n=0 TO 7
9010 READ a: POKE USR CHR$ (i+14
3)+n,a: NEXT n: NEXT i
9020 RETURN
9099 REM Esplosione
9100 DATA 7,63,127,127,255,255,2
55,247
9110 DATA 224,252,254,254,255,25
5,255,239
9120 DATA 99,3,3,7,7,15,31,255
9130 DATA 198,192,224,224,240,24
8,252,255
9199 REM Frecce
```

```
9200  DATA  0,8,28,42,73,8,8,8
9210  DATA  0,62,8,10,16,34,64,128
9220  DATA  0,16,8,4,254,4,8,16
9230  DATA  128,64,34,18,10,8,62,0
9240  DATA  16,16,16,146,84,56,16,
0
9250  DATA  1,2,68,36,72,80,96,124,0
9260  DATA  8,16,32,126,32,16,8,0
9270  DATA  0,124,96,80,72,68,8,1
9280  REM  Rovine  (della citta')
9300  DATA  0,0,0,0,4,14,159,223
9310  DATA  0,0,0,0,2,6,39,255
```

Note

10	Inizializzazione
30	Fuori gioco (succede!)
100	Partenza del loop.
100-180	Movimento del missile
190	Testa se tutte le citta' sono state distrutte
200	Cancella l' attraversamento della base
210-240	Valuta l' input e calcola le nuove coordinate
250	Disegna la base
260-300	Testa lo sparo e disegna la freccia
310	Partenza di una nuova ondata
400	Disegna il laser
420	Testa la collisione
430	Cancella il missile
440	Predispone un nuovo missile
450-470	Aggiorna il punteggio, testa il nuovo missile scala le munizioni
490	Stampa il nuovo punteggio
500	Stampa i punti
610	Test per vedere se il missile ha colpito la base
630	Esplosione
670-680	Cancella la citta dall'array
690	Predispone un nuovo missile
720-740	Predispone i nuovi valori per il missile
800-830	Il missile ha colpito la base,non esplode ma il punteggio cala
1010	Nuovo valore dei punti
1020-1070	Punti per le citta' rimanenti
1080	Aumento delle difficolta'
1090	Punti per le munizioni rimanenti
1110-1180	Aggiunge una nuova citta' se si sono superati 10000 punti
1190	Inizializza il riposo, per una nuova ondata
2000	Colori random
2020-2050	Predisposizione dei colori random
2060-2070	Disegna lo schermo
2080	Stampa il punteggio
2090	Munizioni rimaste
2100-2160	Disegno della base e delle citta'
3000	Definizione degli UDG

3100-3140	Inizializzazione delle variabili (anche 3220-3230)
3200	Array per i missili
3240	Predispone i missili
3300	Inizializza l'attraversamento del cursore
3400	Disegna lo schermo
4000	Fine della gara
9000-9020	Definisce gli UDG
9100-9130	Dati per gli UDG

Capitolo 8

PEEK e POKE

Sono senza dubbio due tra i comandi piu' usati dello Spectrum. Per capire bene le funzioni che essi svolgono, e' necessario conoscere l' hardware della macchina e il suo funzionamento. Come gia' saprete, ogni computer possiede una memoria la quale si divide in due tipi ben distinti. La ROM (abbreviazione che sta per Read Only Memory = memoria a sola lettura) contiene le informazioni necessarie al funzionamento della CPU (Central Processing Unit = Unita' centrale di processo o piu' semplicemente microprocessore) le quali rimangono memorizzate anche quando l' apparecchio viene spento. Il contenuto della ROM non puo' venire alterato in alcun modo.
Il secondo tipo di memoria si chiama RAM (Random Access Memory = Memoria ad accesso casuale) e memorizza i dati provenienti dalla CPU. Le informazioni e i dati in gioco, sono tutto quanto presente in un programma e visualizzato sullo schermo a partire dalla grafica definita dall' utente (UDG) per arrivare alle variabili del sistema. Al contrario di quanto accade per la ROM, il conenuto della RAM va perso quando si toglie corrente e puo' venir alterato in qualsiasi momento.
Vediamo ora in quale modo viene memorizzata una informazione. Immaginatevi la memoria suddivisa in compartimenti ognuno dei quali in grado di contenere un numero; sia la quantita' dei primi che quella dei secondi dipende dal tipo di computer. Lo Spectrum dispone di 65536 (da 0 a 65535) compartimenti, chiamati bytes capaci ognuno di memorizzare un numero compreso entro la gamma 0-255. Per comodita' i bytes vengono raggrupati a mille per volta e ognuno di tali gruppi e' conosciuto col nome di "Kilobyte", abbreviato con "K". Per essere piu' precisi, un K equivale a 1024 bytes.
Il comando PEEK vi permette di accedere a questi bytes con la forma "PEEK n" dove "n" e' appunto il byte da esaminare.
Il comando POKE, invece, vi abilita addirittura a cambiare il contenuto per mezzo dello statement "POKE n,m" in cui "n" e' il byte, o meglio l' indirizzo, presso il quale effettuare la variazione e "m" il valore del nuovo dato compreso tra 0 e 255. POKE viene quindi usato per memorizzare nuovi dati, in una area di memoria riservata e PEEK per andarli a ripescare.
Vi sono diversi modi per riservare la memoria onde proteggere i dati da alterazioni accidentali. Uno di questi sistemi prevede l' uso dello statement di REM.
Se la prima linea e' :

```
1 REM AAA
```

La memoria viene prenotata sotto il segno delle A. Provate a battere questa linea facendola seguire dal comando diretto POKE 23760,127 e vedrete che, listando nuovamente la linea, al posto della prima A troverete il simbolo di copyright corrispondente

appunto al carattere 127. Provate con altri valori. L' indirizzo 23760 si riferisce appunto al primo carattere dopo il REM presente alla linea 1. Per pochi dati potete quindi utilizzare lo statement di REM dotandolo di tanti caratteri quanti sono i dati per poi POKare questi ultimi nei vari indirizzi messi a disposizione dallo statement stesso.
Una seconda via per proteggere i dati la mette a disposizione CLEAR il quale li preserva anche dal comando NEW che ha il potere di azzerare tutto quanto.
Di solito i caratteri a disposizione dell' utente, che sono in tutto 21, occupano gli ultimi 168 bytes della memoria in quanto ogni carattere ne impiega 8. Nello Spectrum da 16 K, essi occupano gli indirizzi da 32600 a 32767, in quello da 48 K gli indirizzi impegnati partono da 65368 e arrivano fino a 65535.
Lo statement "CLEAR n" predispone una variabile di sistema (che vedremo piu' avanti) all' indirizzo "n".
Se "n" e' piu' alto di 32600 nello Spectrum da 16 K si perde un carattere-user, predisponendolo piu' basso, si prenota una zona di memoria da n+1 a 32599. Analogamente accade nella versione da 48 K con i relativi indirizzi.
Con POKE si predispongono le variabili di sistema le quali vengono considerate dalla CPU per valutare quanto sta succedendo e prendere determinate decisioni. Usando correttamente questa funzione e' possibile creare un gran numero di effetti speciali. Pokando, ad esempio, 23609 col numero 100 potete generare, alla pressione di qualsiasi tasto, un beep piu' consistente del normale che puo' risultare utile nelle operazioni di INPUT dei games. POKando un numero maggiore di 1 all' indirizzo 23692, il computer non vi chiedera' piu' lo "scroll?" ogni 22 linee di programma. Per cambiare il colore delle scritte relative ai messaggi sulla parte bassa dello schermo, non dovrete far altro che eseguire un POKE 23624 seguito da un numero a caso. POKE 23624 altera quindi gli ATTRibuti e i colori delle scritte o degli oggetti presenti nella parte alta dello schermo. Dando POKE 23659,1:CLS riempirete lo schermo di strisce la cui ampiezza dipende dal valore POKato in 23693.
Gli indirizzi 23677 e 23678 riguardano la funzione DRAW la quale puo' tracciare linee in tutte le direzioni partendo dall' ultimo punto stabilito da PLOT. Nell' esempio che segue, la posizione di plot orizzontale non e' 20 bensi' 120 (100+20), cosi' come quella verticale non e' 60 ma 140 (80+60).

```
10 PLOT 100,80
20 DRAW 20,60
```

Perche' una eventuale linea 30 possa iniziare a tracciare dai valori 0,0 e' necessario l' intervento di alcuni calcoli. Per mezzo delle variabili di sistema 23677 e 23678 potete simulare un DRAW assoluto. La linea 20 puo' essere riscritta nel modo che segue :

```
20 DRAW 120-PEEK 23677,140-PEEK 23678
```

e se prima avreste dovuto battere

```
30 DRAW -120,-140
```

ora dovete inserire

```
30 DRAW 0-PEEK 23677,0-PEEK 23678
```

Lo Spectrum puo' disporre del tempo in modo assai ben definito grazie alle tre variabili di sistema 23674,23673 e 23672. La piu' bassa delle tre viene incrementata ogni 50esimo di secondo fino a che non raggiunge 255 dopodiche' torna a 0 facendo partire il conteggio nell' indirizzo 23673 il quale, una volta raggiunto 255 abilita il terzo che, a sua volta raggiunge 255, resettandosi e facendo ripartire tutto da capo.
Il calcolo del tempo (espresso in secondi) e' assai semplice :

```
LETsecs=(65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50
```

Questa funzione e' di grande aiuto in games tipo auto da corsa e in qualsiasi caso in cui la gara debba essere portata a termine nel piu' breve tempo possibile. Alcune volte puo' succedere che il calcolo dei secondi dia un risultato sbagliato. Nel caso in cui, ad esempio, la locazione 23673 contenga 0 in conseguenza del fatto che lo 23672 abbia raggiunto 255 e poi si sia a sua volta azzerata, il computer puo' valutare contemporaneamente i valori 0 delle due variabili e di conseguenza considerare un tempo doppio di quello reale falsando i risultati. Per ovviare a tale fenomeno, e' necessario POKare a 0 ognuno degli indirizzi.
Oltre ai 22 caratteri,lo Spectrum ne puo' definire altri 96. Il segreto sta' nel bit del POKE il quale, se impiegato negli indirizzi 23606 e 23607, vi permette di cambiare il punto di partenza del set dei caratteri presente da CHR$ 32 (spazio) a CHR$ 127 (marchio di copyright).
Ecco come dovete fare :

1) Disegnate i nuovi caratteri da mettere al posto dei vecchi e calcolatene i dati.

2) Decidete i caratteri del set originale da ridefinire e prendete nota di quelli che dovranno ridefinirli. Di solito i caratteri da ridefinire (UDG=User-defined graphic) vengono scelti tra i primi del set originale. Se volete definire lettere e numeri tipo "era spaziale" e' bene segnarsi sempre l' accoppiamento tra i vecchi numeri con i nuovi e tra le vecchie lettere con le nuove. Questo perche', mentre state scrivendo lo statement di PRINT vi ricordate le equivalenze, ma, una vola fatto girare il programma, i caratteri ordinari lasciano il posto a quelli ridefiniti scomparendo dal contesto delle linee.

3) Prenotate la memoria usando CLEAR n (come descritto poco fa') o come comando diretto o inserendolo nel programma stesso. Il numero di bytes da riservare deve corrispondere a quello degli UDG compreso il primo e l' ultimo del set. Il numero va naturalmente moltiplicato per 8 essendo ogni carattere formato da altrettanti numeri. La cifra "n" viene ricavata sottraendo il numero cosi' trovato da 32599 per lo Spectrum a 16 K e da 65367 per quello da 48 K.

4) Predisponete gli indirizzi 23606 e 23607 in modo che il primo UDG risulti a "n"+1. Per far cio' procedete come segue :

a) Trovate "n"+1
b) Contate quanti caratteri ci sono prima di quello da ridefinire; esempio, se il primo carattere da ridefinire e' CHR$ 38, avrete 38-36=6 caratteri.
c) Moltiplicate questa cifra per 8.
d) Supponiamo di chiamare quanto ottenuto,"char", per cui i valori da POKEare sono dati da
POKE 23606,char-256*INT (char/256)
POKE 23607,INT (char/256)

5) Qualsiasi UDG, puo' cambiare anche caratteri del set alternativi, come potete vedere dal programma che segue il quale ridefinisce "#" con un carattere grafico.

```
    1 REM Per lo Spectrum da 16K
    2 CLEAR 32592: REM Valore di
"n"
    3 LET num chars=1
    4 LET bytes reserve=num chars
*8
    5 LET n=32599-bytes reserve
    6 LET char before first=CODE
"#"-32
    7 LET byte before first=char
before first*8
    8 LET char=n+1-byte before fi
rst+256
    9 LET chr in 23606=char-256*I
NT (char/256)
   10 LET chr in 23607=INT (char/
256)
   20 FOR i=n+1 TO n+bytes reserv
e: READ a: POKE i,a: NEXT i
   30 DATA 255,36,36,24,24,36,36,
255,0
 9000 POKE 23606,chr in 23606
 9010 POKE 23607,chr in 23607
 9020 PRINT AT 11,15;"#"
 9030 POKE 23606,0
 9040 POKE 23607,60
```

Differenze per lo Spectrum a 48 K

```
2 CLEAR 65380: REM valore di "n"
.
5 LET n=65367-riserva bytes
```

Esistono diverse variabili di sistema che non vengono usate normalmente come le altre ma che restano libere a disposizione. I loro indirizzi sono 23681, 23728 e 23729.
Molti computer posizionano gli oggetti sullo schermo per mezzo di PEEK e POKE, con lo Spectrum cio' non e' possibile. Se volete una dimostrazione fate girare il programma che segue e osservate gli strani risultati :

```
   10 REM Per dimostrare l' impos
sibilita' di pokare lo schermo
   200 LET screen=16384
   300 LET length=24*32*8
   400 FOR i=screen TO screen+leng
th
   500 POKE i,35
   600 NEXT i
   700 BEEP 1,10
   100 REM Per dimostrare l' impos
sibilita' di pokare lo schermo
   200 LET screen=16384
   300 LET length=24*32*8
   400 FOR i=screen TO screen+leng
th
   500 POKE i,35
   600 NEXT i
   700 BEEP 1,10
```

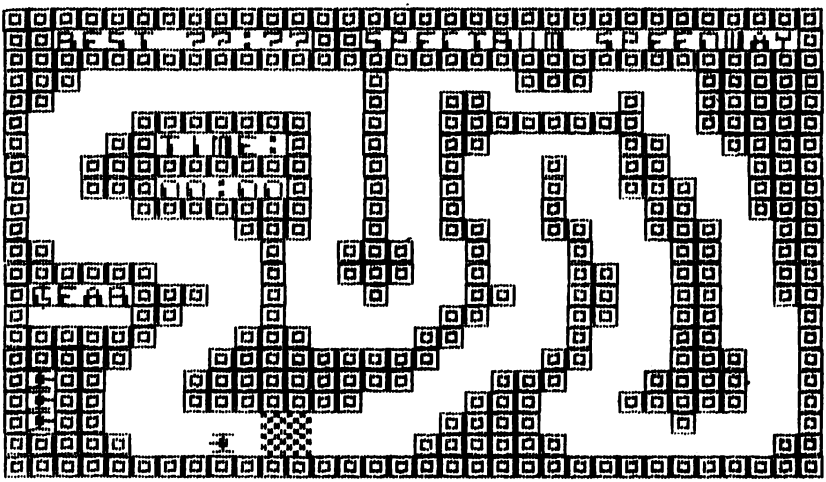
Lo Spectrum risolve la cosa usando PRINT AT al posto di POKE e ATTR, POINT, SCREEN$ al posto di PEEK.
Il nostro prossimo programma, di nome "Circuit", fa un esteso uso di PEEK e POKE e vi permette di guidare la vostra auto attraverso un intricato circuito sterzando nelle quattro direzioni cardinali e nelle quattro diagonoli.
Al posto dell' auto si era pensato di disegnare un carro armato, ma i veloci cambi di direzione non si addicevano a tale mezzo. I tasti sono codificati per mezzo della funzione IN onde poterne azionare due contemporaneamente. L' auto dispone di quattro marce ed, alla partenza, e' innestata naturalmente la prima come potete constatare sul lato sinistro dello schermo. Se volete ingranare marce superiori (e quindi accelerare) premete "Q", per scalare "A". La guida e' molto realistica, col tasto "P" ruoterete in senso orario, se ad esempio siete diretti verso nord volterete verso nord-est oppure se siete orientati verso ovest girerete verso nord-ovest ecc...; col

tasto "O" la ruotazione avverra' in senso antiorario. Il game prevede il conteggio del tempo che viene pero' visualizzato solamente dopo aver concluso il giro oppure dopo esservi schiantati da qualche parte, per non rallentare eccessivamente lo svolgimento della gara. A vostra disposizione avete tre auto con le quali dovrete cercare di completare il tracciato; ogni volta che picchierete, dovrete ripartire daccapo. Il numero delle auto a vostra disposizione lo trovate nell' angolo in basso a sinistra.
La caratteristica piu' importante del gioco, e' sicuramente il set di caratteri definiti, pensate che, oltre agli ottanta UDG (di cui dieci sono numeri) sono state ridefinite, come visto sopra, anche diverse lettere comprese tra "a" e "z". Poiche' il tutto occupa diverso spazio in memoria, abbiamo suddiviso il programma in due tronconi per permettere il caricamento del secondo blocco (il gioco vero e proprio) anche sugli Spectrum da 16 K.
La prima parte predispone il set dei nuovi caratteri, definisce gli UDG e inoltre carica il secondo blocco. Se avete uno Spectrum da 48 K, potete congiungere assieme le due parti.
Molto interessante e' l' effetto di PRINT #0;"....." presente alle linee 8030 e 8060, infatti tale funzione abilita alla scrittura anche nelle due linee piu' basse dello schermo riservate normalmente ai commenti e ai messaggi di errore. Quanto scritto da PRINT 0, viene cancellato dagli INPUT alle linee 8020 e 8040 perche' non vizi il contenuto del display al sopraggiungere dello scrolling. Ora che conoscete alcuni segreti del programma, battetene la prima parte e registratela sul nastro con :SAVE "Circuit" LINE 10. Dopo aver eseguito il VERIFY, battete anche la seconda parte e registratela con : SAVE "C" LINE 10. Verificate quindi anche quest' ultima.

```
8000 REM Definizione caratteri
8999
9000 CLEAR 32399: RESTORE 9200
9010 PRINT AT 9,2;"FERMA IL REGI
STRATORE"
9020 PRINT AT 11,10; FLASH 1;"AT
TENDI"
9030 FOR i=32400 TO 32599
9040 READ a: POKE i,a: NEXT i
9050 FOR i=1 TO 10: FOR n=0 TO 7
9060 READ a: POKE USR CHR$ (i+14
3)+n,a: NEXT n: NEXT i
9070 CLS : PRINT AT 9,7;"PREMI P
ARLA";AT 11,5;"SUL TUO REGISTRATO
RE"
9080 PRINT AT 13,0;"(Batti LOAD"
;"
9090 LOAD "C"
9199 REM Data for letters and
symbols:
9200 DATA 0,24,36,36,126,102,102
,102
9210 DATA 0,124,68,68,126,70,70,
```

```
20  DATA  0,124,64,64,96,96,96,1
30  DATA  0,126,66,66,98,98,98,1
40  DATA  0,126,64,64,120,96,96,
550 DATA  255,129,189,165,165,18
9,255
660 DATA  0,126,64,64,96,102,96,
70  DATA  0,68,68,68,124,76,76,7
930 DATA  0,16,16,16,24,24,24,24
900 DATA  204,204,51,51,204,204,
51,51
3000 DATA 0,0,0,0,0,0,0,0
3100 DATA 0,30,30,30,48,48,48,62
3200 DATA 0,126,66,66,66,66,66,6
330 DATA  0,124,76,76,76,76,76,7
340 DATA  0,126,66,66,98,98,98,1
350 DATA  0,124,68,68,124,64,64,
3600 DATA 0,0,0,0,0,0,0,0
3700 DATA 0,124,68,68,120,70,70,
3800 DATA 0,124,64,64,124,12,12,
390 DATA  0,126,16,16,24,24,24,2
400 DATA  0,68,68,68,68,68,68,12
4100 DATA 0,126,66,2,30,16,0,16
4200 DATA 0,66,66,66,66,66,66,12
4300 DATA 0,24,24,0,0,24,24,0
4400 DATA 0,68,68,124,16,16,16,1
499 REM Data for "cars":
500 DATA  8,73,73,93,127,93,65,6
5510 DATA 33,66,188,60,60,29,2,4
5200 DATA 254,16,68,63,56,16,254
5300 DATA 4,2,29,60,60,188,66,33
5400 DATA 65,65,93,127,93,73,73,
5550 DATA 32,64,184,60,60,61,66,
5600 DATA 127,8,28,252,28,8,127,
5570 DATA 132,66,61,60,60,184,64
9999 REM Data for Numbers:
6000 DATA 0,60,66,66,98,98,98,60
6100 DATA 0,8,8,8,12,12,12,12
6200 DATA 0,124,32,32,60,96,96,62
6300 DATA 0,124,32,32,60,0,0,126
```

```
9640 DATA 0,64,96,100,100,126,4,
4
9650 DATA 0,62,62,64,60,6,6,126
9660 DATA 0,62,64,64,124,70,70,1
26
9670 DATA 0,120,8,8,12,12,12,12
9680 DATA 0,60,66,66,60,70,70,60
9690 DATA 0,60,66,66,60,6,6,126
9999 POKE 23606,0: POKE 23607,60
: RETURN
```



Ammirate la copia dello schermo, a colori poi e' ancora meglio.

Ed ora vediamo la seconda parte del programma che crea il game vero e proprio.

Elenco delle variabili

h	Secondi impiegati e record
b$(2,5)b$(1)	Contiene i caratteri del circuito in corso, b$ (2) contiene gli spazi del circuito
n	Si alterna tra i valori 0 e 1 durante il disegno del circuito da parte dell' array b$(.)
c	Numero delle auto rimaste
m	Minuti
s	Secondi
x	Coordinata verticale dell' auto
y	Coordinata orizzontale dell' auto

dr — Direzione dell' auto da 0 a 7. (0-su,2-destra ecc...)

g — Marcia. Da 1 (piu' lenta) a 4 (piu' veloce)

a1 — Legge la tastiera con IN per vedere se e' stato premuto il tasto 0 per l' accelerazione

d1 — Come il precedente ma stavolta riferito al tasto A per la decelerazione

mv — Verifica se sia stato battuto uno dei tasti per spostare l' auto (O oppure P)

a,
b — Coordinate che la subroutine usa per disegnare qualsiasi cosa. Tale subroutine e' richiamata da due diversi posti, "a" e "b" vengono assegnati prima di chiamare la subroutine in modo che vengano PRINTate le coordinate esatte

t — Tempo (in secondi) impiegato per completare il circuito. Viene comparato con "h"

z$ — Usata come stringa d' ingresso

LOOP,
ALTER,
NORM,
TIME,
START,
NEXT CAR,
END,
CRASH,
FINISH,
BYE — Sono tutti "tokens" per la partenza delle subroutines : ALTER e' la subroutine usata per inserire il set alternativo di caratteri; NORM (normale) e' per cambiare di nuovo; TIME presenta tempo; START inizializza la partenza di una nuova gara; NEXT CAR inizializza una nuova auto; END e' il punto in cui finisce il programma quando l' auto urta contro qualcosa; CRASH e' per quando l' auto urta contro il circuito e FINISH e' per quando l' auto raggiunge la griglia di partenza.

Nel listing sono presenti degli user-graphic alle linee 5300 (quattro I) e 5310 (lettera C).

Circuito (Parte 2)

```
   10 GO TO 5000: REM init/instr
  5300 REM Con lo Spectrum issue 2
  5301 REM variare 255 con 191,
  5302 REM 254 con 190 e 253 con 1
69
  100 LET a1=IN 64510-255: LET d1
=IN 65022-255: LET mv=IN 57342
```

```
  120 LET dr=dr+(mv=254)-(mv=253)
-(0 AND dr=7 AND mv=254)+(8 AND
dr=0 AND mv=253)
  140 LET x=x-(dr=0 OR dr=1 OR dr
=7)+(dr>2 AND dr<6): LET y=y-(dr
>4 AND dr<=7)+(dr>0 AND dr<4)
  150 PRINT AT x,y;CHR$ (144+dr)
  180 IF al OR dl THEN PLOT 4+g*8
,56: DRAW 0,-5: LET g=g-(al AND g
<4)+(dl AND g>1): PLOT 4+g*8,56:
 DRAW 0,-5
  200 FOR i=1 TO 5-g: BEEP .02,RN
D*2+(-4*g): NEXT i
  400 PRINT AT x,y;CHR$ (144+dr)
  420 GO TO LOOP+(400 AND ATTR (x
,y)<>56)
  900 GO TO END

 9000 REM Alternative:

10000 POKE 23606,136: POKE 23607,
11400 RETURN

14990 REM Normal:

15000 POKE 23606,0: POKE 23607,60
: RETURN

19990 REM PRINT TIMES

20000 GO SUB ALTER
20010 LET s$=("0" AND m<10)+STR$
 m+("0" AND s<10)+STR$ s
20020 PRINT AT a,b; INK 3;CHR$ (C
ODE s$(1)+104);CHR$ (CODE s$(2)+
104)
20030 PRINT AT a,b+3;CHR$ (CODE s
$(3)+104);CHR$ (CODE s$(4)+104)
20040 GO SUB NORM
20050 RETURN

49990 REM ******INITIALISE*******

50000 LET h=1E38
50010 LET LOOP=100
50020 LET ALTER=1000: LET NORM=15
00
50030 LET TIME=2000
50040 LET START=5300: LET NEXT CA
R=5400
50050 LET NEW GAME=9000: LET BYE=
0
50060 LET END=7000
50070 LET CRASH=7500: LET FINISH=
0
51000 POKE 23693,33: POKE 23659,1
: CLS : INPUT ;
51110 GO SUB ALTER
51120 DIM b$(2,5): LET b$(1)=CHR$
 16+CHR$ 2+CHR$ 17+CHR$ 6+CHR$ 1
```

```
02: LET b$(2)=CHR$ 16+CHR$ 1+CHR
$ 17+CHR$ 7+CHR$ 107
5130 BRIGHT 0: OVER 0: FLASH 0:
INVERSE 0: BORDER 5: PAPER 7: IN
K 1
5140 PRINT AT 0,0;: RESTORE 6500
: LET n=1
5150 FOR i=1 TO 189: READ a: LET
 n=n-n+NOT n: FOR v=1 TO a: PRIN
T b$(n+1);: NEXT v: NEXT i
5160 PRINT AT 19,10; INK 0;"ij";
AT 20,10;"ij";AT 1,2; INK 3;"bes
tkvvxvv";AT 6,6;"timex";AT 1,14;
 INK 1;"spectrumkspeedway";AT 13
,1;"gear"
5170 FOR i=1 TO 4: PLOT 4+1*8,62
: NEXT i: INK 8
5299 REM START INIZ
5300 GO SUB ALTER: PRINT AT 8,6;
 INK 3;"OOxOO"
5310 LET c=3: FOR i=1 TO c: PRIN
T AT 16+i,1;"E": NEXT i: PLOT 8,
63: DRAW 31,0
5320 PRINT #0; INK 2; PAPER 6;"f
ffffffspectrumkspeedwayffffff"
5330 LET m=0: LET s=0
5399 REM NEXT CAR INIT:
5400 LET x=20: LET y=10: LET dr=
6: LET g=1
5410 PLOT 12,66: DRAW 0,5
5420 OVER 1: GO SUB NORM
5430 POKE 23672,0: POKE 23673,0:
 BEEP 1,0
5490 GO TO LOOP
5499 REM data for screen:
6500 DATA 34,10,2,17,35,11,1,5,3
4,7,12
6510 DATA 1,2,2,5,1,2,6,4,7,2,1,
2,3,2,6,3,2,5,1,2,1
6520 DATA 2,2,5,2,2,5,2,9,2,1,2,
1,3,1,2,2,3,4,2,3,5
6530 DATA 1,2,1,2,1,3,1,2,3,2,4,
4,7,2,1,2,1,3,1,3,2,2,4
6540 DATA 8,3,2,1,2,2,2,2,2,3,2,
4,8,1
6550 DATA 2,3,2,1,3,1,3,2,2,8,4,
1,2,3,2,1,3,2,2
6560 DATA 2,2,3,4,3,2,1,3,1,3,2,
2,2,2,2,3
6570 DATA 4,2,3,1,6,2,3,2,2,2,3,
7,3,3,4,2,4
6580 DATA 1,3,2,3,6,3,9,4,2,3,3,
2,2,1,2,3,9
6590 DATA 3,3,3,4,2,2,1,2,3,7,4,
3,3,5,2,2,1,2,13,4,5,1,4,6,11,6,
8,34
6997
6998 REM END OF RUN
6999
7000 PLOT 4+g*8,56: DRAW 0,5
```

```
10 OVER 0: LET s=s+INT ((PEEK
23673*256+PEEK 23672)/50)
20 LET m=m+INT (s/60): LET s=s
-m*60: LET a=8: LET b=6: GO SUB
THEME
30 GO TO CRASH+(FINISH-CRASH A
ND ATTR (x,y)=56)
40 REM *********CRASH*********
50 FOR i=0 TO 25: LET dr=dr+1-
(8 AND dr=7): PRINT AT x,y;CHR$
(144+dr): BEEP .02,i: NEXT i
60 GO SUB ALTER: PRINT AT x,y;
" ": GO SUB NORM
70 PRINT AT 16+c,1;" ": LET c=
c-(1)
80 IF c>0 THEN GO TO NEXT CAR
90 GO TO NEW GAME
100 REM *********FINISH********
110 PRINT AT x,y;CHR$ (144+dr):
t=60*m+s
120 GO SUB ALTER
130 INPUT '
140 PRINT AT 1,0;#0;"jjjj bravo
ce l'hai fatta jjjjjj": PAUSE 2
00
150 INPUT '
160 IF t<h THEN LET h=t: LET a=
1: LET b=7: GO SUB TIME: GO SUB
ALTER: PRINT AT 1,0;#0;"e hai f
atto il miglior tempo ": GO TO
170
180 PRINT AT 1,0;#0;"but you di
dn't get the best time!"
190 PAUSE 200
200 PRINT AT x,y;"j"
210 GO SUB NORM
220 REM *******NEW GAME ?*******
230 INPUT "Another game ? (Y/N)
": LINE z$
240 IF (z$+" ")(1)="n" THEN GO
TO BYEE
250 INPUT "Press ENTER to play
": LINE z$
260 GO TO START
270 BORDER 1: PAPER 1: INK 7: C
LS
280 PRINT AT 10,13; FLASH 1; IN
K 6; PAPER 2;"BYEEE"
290 STOP
```

Note

10	Inizializzazione
100	Legge la tastiera
120	Cambia la direzione
140	Cambia le coordinate
160	Disegna l' auto
180	Cambia marcia
200	Suono
220	Cancella l' auto
240	Verifica se l' auto ha urtato qualcosa
1000	Set di caratteri alternativo
1500	Set di caratteri normale
2010	Il tempo viene inserito in una stringa
2030-2040	Scrive il tempo
5000	Miglior tempo inizializzato
5010-5070	"Tokens" per le subroutines inizializzate
5100	Disegna le strisce
5130	Colori
5150-5170	Disegna lo schermo
5300	Scrive il tempo
5310	Auto rimaste
5320	Scrive sulla 23esima riga
5430	Azzera i contatori del tempo
6500-6590	Dati usati alla linea 5150 per scrivere sullo schermo
7000	Cancella il marker delle marce
7010-7020	Calcola il tempo e lo presenta
7030	Salta a "CRASH" o a "FINISH"
7500	Ruota l' auto
7510	Riposiziona il circuito
7520-7540	Diminuisce il numero delle auto e salta di conseguenza
8000	Disegna l' auto
8030	Congratulazioni
8050	Inserisce il nuovo tempo nella variabile "best time"
8080	Riposiziona la griglia
9000	Verifica per un' altra gara
9020	Tutto pronto
9980-9999	Allegorie

Seguono due ottimi programmi sviluppati originalmente per tutt' altro. L' idea originale era di usare una racchetta o una superficie scorrevole per catturare i blocchi che piovono dal cielo. Ogni blocco puo' valere da uno o due punti. I blocchi non catturati piombano al suolo accatastandosi fino a formare un muro che, se raggiunge l' altezza della racchetta, fa finire la gara.
Il primo dei due games basati su tale principio e' stato chiamato "Chomper".
Dovete mangiare il cibo che cade dal cielo composto da pere, salsicce, birra, polli arrosto e hamburger. Cadono anche dei

fucili che dovrete recuperare per poter sparare ai mostri che crescono mangiando quanto non siete riusciti ad acchiapare. Le istruzioni sono contenute nel programma stesso. Le coordinate dei tre oggetti che cadono contemporaneamente (siano questi cibo o armi) sono memorizzate in array per poterli muovere tramite loop.

```
Graphic A     : linee 1000 e 3000 (doppio)
Graphic B     : linea 4000
Graphic J e K: linee 2010,4040 e 5410 (primo set della linea)
Graphic L e M: linea 2000 (primo set)
Graphic N e O: Linea 2000 (secondo set) e 5410 (secondo set)
Graphic P     : linea 7010 (primo)
Graphic Q     : linea 7010 (secondo)
Graphic R     : linee 7020 e 7050
Graphic S     : linea 7030
```

Elenco delle variabili

i	Variabile di controllo dei loop per far cadere gli oggetti dal cielo
a(.)	Array per le coordinate verticali degli oggetti
b(.)	Array per le coordinate orizzontali degli oggetti
t2	Altezza dell' ultimo oggetto mangiato dal mostro. Usato per vedere se il mostro ha raggiunto l' altezza del "Chomper"
c(.)	Memorizza il colore di PAPER di ogni oggetto in caduta
d(.)	Memorizza il colore di INK di ogni oggetto in caduta
c$(.)	Memorizza la forma dell' oggetto (ad esempio una pera)
m	Usato per alternare il movimento della bocca de "Chomper"
a$	Per leggere la tastiera
p	Coordinata orizzontale del giocatore
gun	Numero dei fucili recuperati
rnd	Usata per selezionare la specie di cibo da far cadere
n	Controllo per il loop di lettura dei colori del cibo
t1	Usata per ingrassare il mostro quando questo ha mangiato
z	Variabile di controllo per la cancellazione dei mostri quando siano stati colpiti
s	Punti
i, n, z	Variabili di controllo per gli altri loop

```
   10  GO TO 5000
  200  FOR i=1 TO 3
  300  IF a(i)>=2 THEN PRINT AT a(
i),b(i);"■"
  400  LET a(i)=a(i)+1
  550  IF ATTR (a(i),b(i))=31 THEN
 GO SUB 3000: GO SUB 1000: BEEP
.005,20
  600  IF ATTR (a(i),b(i))=29 THEN
 GO SUB 2000: IF t2=15 THEN GO T
O 7000
  700  REM Chomper divorato
  650  IF a(i)=22 THEN GO SUB 1000
  700  IF a(i)>=2 THEN PRINT PAPER
 c(i); INK d(i);AT a(i),b(i);c$(
i)
  800  NEXT i
  900  LET m=m+(NOT m)-m
 1000  LET a$=INKEY$: IF a$="5" OR
a$="8" THEN PRINT AT 15,p;" ";
LET p=p+(a$="8" AND p<31)-(a$="5
" AND p)
 1100  IF a$="0" AND gun>0 THEN GO
SUB 4000: REM Sparo
 1200  PRINT INK 7;AT 15,p;CHR$ (1
45+m)
 1300  GO TO 20
 3000  STOP
 9999  REM ancora cibo
10000  IF RND>.9 THEN LET c$(i)="■
": LET c(i)=3: LET d(i)=0: GO TO
 1020
10100  LET rnd=INT (RND*5): RESTOR
E 6000: FOR n=1 TO rnd+1: READ c
(i),d(i): NEXT n: LET c$(i)=CHR$
(144+rnd)
10300  LET a(i)=1: LET b(i)=INT (R
ND*32): RETURN
11999  STOP
12000  REM il mostro cresce
12001  LET t1=INT (b(i)/4)*4+1: PR
INT INK 5;AT a(i)-1,t1;"LM";AT a
(i),t1;"NO"
12010  BEEP .05,0: PRINT INK 6;AT
(i)-1,t1;"OK": BEEP .05,-12
12020  LET t2=a(i): GO SUB 1000
12030  RETURN
12999  STOP
13000  REM test del cibo mangiato
13001  IF c$(i)="■" THEN PRINT AT
 30,-gun; PAPER 4; INK 0;"A": LE
T gun=gun+1: GO TO 3100
13010  LET s=s+1: PRINT AT 0,0-(LE
N (STR$ s)); PAPER 4; INK 0;s
13020  RETURN
14000  REM sparo
14001  PRINT INK 0;AT 15,p;"B": FO
R z=16 TO 20
14010  IF ATTR (z,p)=29 THEN PRINT
 AT z,p;"■": PRINT AT z,p+(ATTR
(z,p+1)=29)-(ATTR (z,p-1)=29);"
■"
```

```
4020 NEXT z: INK 7: PLOT p*8+3,4
7: DRAW 0,-38: BEEP .1,40: BEEP
.15,30
4030 INK 3: PLOT p*8+3,47: DRAW
0,-38
4040 IF ATTR (21,p)=29 THEN PRIN
T INK 5;AT 21,INT (p/4)*4+1;"JK"
4050 PRINT AT 0,31-gun; PAPER 4;
INK 4;" ": LET gun=gun-1
4060 RETURN
5000 BORDER 1: PAPER 4: INK 0: C
LS
5010 PRINT TAB 12; FLASH 1;"Chom
per"
5020 PRINT '''"Sei il chomper nel
la meta' superiore dello schermo
e devi mangiare il cibo o cattu
rare i fucili che piovono dal ci
elo. Con il cibo incrementi "
5040 PRINT "i punti,coi fucili d
evi uccidere il mostro sottostan
te che cresce col cibo non racco
lto."
5050 PRINT "Muovi con i tasti:",
"5-sinistra","8-Destra","spara
con lo 0. Il numero dei fucili
(e degli spari) lo trovi "
5060 PRINT "nell'angolo in alto
a destra","i punti a sinistra."
5070 PRINT '''TAB 2; FLASH 1;"Pre
mi un tasto per continuare"
5080 IF INKEY$<>"" THEN GO TO 50
80
5090 IF INKEY$="" THEN GO TO 509
0
5099 REM Colori-Colori
5100 BORDER 0: PAPER 3: INK 3: C
LS
5110 GO SUB 9000: REM UDG
5120 LET s=0: LET p=15: LET m=0:
LET gun=0:
5130 DIM a(3): DIM b(3): DIM c$(
3): DIM c(3): DIM d(3): FOR i=1
TO 3: LET a(i)=-5*i: LET b(i)=IN
T (RND*32)
5140 LET rnd=INT (RND*5): RESTOR
E 6000: FOR n=1 TO rnd+1: READ c
(i),d(i): NEXT n
5150 LET c$(i)=CHR$ (146+rnd): N
EXT i
5400 PRINT PAPER 4; INK 0;"
Armi res.";TAB 0;"
"
5410 FOR i=0 TO 7: PRINT AT 20,i
*4+1; INK 5;"JK";AT 21,i*4+1;"NO
": NEXT i
5490 GO TO 20
6000 REM dati-colore del cibo
6000 DATA 3,6,2,5,3,5,3,0,3,4
6000 REM fine della gara
```

```
7000 FOR z=15 TO 21: PRINT AT I,
t1;"  ": NEXT z: INK 5
7005 IF t1=p OR t1+1=p THEN GO T
O 7050
7010 LET m=0: LET z$=("P" AND t1
>0)+("Q" AND t1-P<0)
7015 FOR z=t1 TO p STEP (SGN (p-
t1)*2)
7020 PRINT AT 15,z;(z$ AND m);("
" AND NOT m)
7030 LET m=m-m+(NOT m): BEEP .1,
INT (RND*30): PRINT AT 15,z;"S"
7040 NEXT z
7050 PRINT AT 15,p;"R": BEEP 1,5
-30: PRINT INVERSE 1; FLASH 1;
INK 7;AT 10,11;"GARA FINITA"
7060 PAUSE 100: BORDER 6: PAPER
INK 0: CLS
7070 PRINT AT 6,0;"Hai fatto ";s
o";"gu";" armi rimaste
7080 PRINT AT 8,0;"Vuoi fare un'
altra gara (y/n) ?"
7090 GO TO 7090+(INKEY$="y")+(29
AND INKEY$="n")
7100 IF INKEY$<>"" THEN GO TO 71
00
7110 PRINT AT 10,0;"Vuoi le istr
uzioni (y/n) ?"
7120 GO TO 7120-(2120 AND INKEY$
="n")+(2090 AND INKEY$="y")
9000 REM User graphics
9000 RESTORE 9010: FOR i=1 TO 19
: FOR n=0 TO 7: READ a: POKE USR
CHR$ (143+i)+n,a: NEXT n: NEXT
i
9010 DATA 0,1,253,63,19,31,3,3:
REM ancora armi
9020 DATA 31,63,52,52,60,48,32,3
REM armi giu'
9030 DATA 192,192,224,224,112,12
4,63,14: REM salsiccce
9040 DATA 128,255,255,0,0,255,25
5,128: REM hamburger
9050 DATA 7,15,15,14,16,32,192,1
28: REM leg
9060 DATA 252,121,127,121,121,12
1,252: REM birra
9070 DATA 16,24,60,60,255,255,12
6,60: REM pere
9080 DATA 66,195,231,255,255,126
,66,60: REM chompers apertos
9090 DATA 66,126,255,255,255,126
,66,60: REM chompers chiusos
9100 DATA 32,32,95,112,120,60,31
,15: REM mostro ↑<,apertO
9110 DATA 4,4,6,14,30,60,248,240
REM mostro ↑>,apertO
9120 DATA 15,31,63,127,127,63,31
,15: REM mostro ↑<,chiUso
9130 DATA 240,248,252,254,254,25
```

```
9120 REM mostro ↑>,chiuso
9130 DATA 4,9,18,36,36,18,9,4: R
EM corpo ↑ <
9150 DATA 32,144,72,36,36,72,144
,32: REM corpo ↑ >
9160 DATA 60,126,207,3,3,207,126
,60: REM mostro <
9170 DATA 60,126,243,192,192,243
,126,60: REM mostro >
9180 DATA 60,126,255,255,255,255
,126,60: REM mostro chiuso
9190 DATA 60,66,153,36,36,153,66
,60: REM corpo <,>
9200 RETURN
9900 BORDER 7: PAPER 6: INK 2: C
LS
9999 PRINT FLASH 1;AT 10,13;" CI
AO ": STOP
```

Note

10	Inizializza
20-80	Muove il cibo e rileva se sia stato mangiato
30	Cancella i vecchi oggetti
50	Rileva se e' stato mangiato dal "Chomper"
60	Rileva se e' stato mangiato dal mostro
70	Stampa gli oggetti
100	Muove il "Chomper"
120	Stampa il "Chomper"
1000	Decide se il prossimo oggetto da far cadere sia un fucile
2000-2010	Ristampa il mostro non appena si ingrossa
3000	Fucile mangiato
3010	Cibo mangiato
4000-4020	Cancella il mostro se viene colpito
4020-4030	Sparo del fucile
4040	Ristampa il mostro nella parte bassa dello schermo
4050	Diminuisce la fornitura d' armi
5020-5090	Scrive le istruzioni e prepara alla partenza
5100	Colori
5110	UDG
5120	Inizializza le variabili
5130	Arrays
5140-5150	Cibo
5400-5490	Disegna lo schermo
7000-7050	Il mostro divora il "Chomper"
7060-7120	Bit finale e nuova gara
9000	Definisce gli UDG
9010-9190	Dati per gli UDG
9990-9999	Fine della gara

Il secondo game mette a disposizione due uomini i quali devono riuscire a fermare le gocce di pioggia che cadono verso di essi

e nell' oceano. Anche questo programma usa gli array per muovere indipendentemente le gocce, ma pur avendo diverse tecniche in comune con il precedente, fornisce un risultato del tutto diverso e sicuramente migliore.
Il miglior risultato che io ricordi e' 48, valore che con un po' di pratica riuscirete sicuramente a battere.

Gli UDG del programma si trovano alle linee : 70-lettera A, 1000-lettera H e 8000-due lettere I.

Elenco delle variabili

i	Controllo del loop per muovere le gocce
a(.)	Coordinata verticale della goccia
b(.)	Coordinata orizzontale della goccia
temp	Memorizzazione temporanea di b(.) per cancellare la vecchia goccia dopo che sia stata inizializzata la sua sostituzione. Da' l' impressione della continuita'
s	Punti
a$	Usato per leggere la tastiera
l	Marker per spostare i giocatori a destra o a sinistra
m	Si alterna tra 0 e 1 facendo camminare gli uomini
p	Coordinata orizzontale del primo uomo
t$	Stringa composta da due elementi per memorizzare le due possibilita' di orientare verso destra o verso sinistra la meta' superiore dell' omino, "l" lo portera' poi nella giusta direzione
b$	Simile a t$, riguarda la parte inferiore dell' omino. La direzione la stabilisce "m"
wl	Livello dell' acqua
h	Record
i,n	Variabili di controllo dei vari loop

```
   10 GO TO 5000
   200 FOR i=1 TO 3
   300 IF a(i)>0 THEN PRINT AT a(i
),b(i);" "
   400 LET a(i)=a(i)+1
   500 IF a(i)=15 AND (b(i)=p OR b
(i)=p+13) THEN LET temp=b(i): GO
 SUB 1000: PRINT AT 15,temp;" "
   600 IF a(i)=18 THEN LET temp=b(
i): LET s=s-1: GO SUB 1000: GO S
UB 2000: PRINT AT 17,temp;" "
   700 IF a(i)>0 THEN PRINT INK 5;
AT a(i),b(i);"A"
   800 NEXT i
   900 LET a$=INKEY$: IF a$="5" OR
 a$="8" THEN PRINT AT 16,p;" ";T
AB p+13;" ";TAB p;" ";TAB p+13;"
": LET l=(a$="8"): LET m=m-m+(N
OT m)
```

```
  100 LET p=p+(a$="8" AND p<18)-(
a$="5" AND p>0)
  110 PRINT INK 2;AT 16,p;t$(l+1)
;TAB p+13;t$(l+1);TAB p;b$(l*2+m
+1);TAB p+13;b$(l*2+m+1)
  120 GO TO 20
  999 REM nuove gocce
1000 PRINT INK 5;AT a(i)-(a(i)=1
8),b(i);"H": LET a(i)=0: LET b(i
)=INT (RND*32): LET s=s+1
1010 BEEP .05,RND*20+20: RETURN
1999 REM livello del mare
2000 PLOT 0,wl: DRAW INK 1;255,0
2010 LET wl=wl+1: IF wl=31 THEN
GO TO 8000: REM uomini annegati
2020 RETURN
4999 REM inizializzazione
5000 LET h=0
5005 INK 0: PAPER 4: BORDER 3: C
LS
5010 PRINT AT 1,11; FLASH 1;"Tem
porale"; FLASH 0,,,"Dovete contr
ollare 2 uomini con l'ombrello,p
er mezzo dei tasti ",,"5-SIN ",,
"8-DEST "
5020 PRINT "per'catturare' le go
cce d'acqua che scendono dal cie
lo. Ognuna mancata,alza il livel
lo dell'oceano sottostante. Cerc
ate di affogare il piu' tardi po
ssibile."
5030 PRINT AT 14,3; FLASH 1;" Un
 tasto per continuare"
5040 IF INKEY$<>"" THEN GO TO 50
40
5050 IF INKEY$="" THEN GO TO 505
0
5060 CLS : GO SUB 9000: REM UDG
5070 LET p=15: LET m=0: LET l=0:
 LET wl=0: LET s=0
5080 DIM a(3): DIM b(3): FOR i=1
 TO 3: LET b(i)=INT (RND*32): LE
T a(i)=-i*5: NEXT i
5089 REM COLORI-COLORI
5090 BORDER 5: PAPER 7: INK 7: C
LS
5100 PLOT 0,31: DRAW INK 0;255,0
5110 PLOT 0,wl: DRAW INK 1;255,0
5120 GO TO 20
7999 REM acqua troppo alta
8000 PRINT AT 17,temp;" "; INK 2
;AT 16,p;" ";TAB p+13;" ";TAB p;
"I";TAB p+13;"I": RESTORE 9500:
FOR i=1 TO 11: READ v,x: BEEP v,
x: NEXT i
8010 BORDER 2: PAPER 1: INK 7: C
LS
8020 PRINT TAB 5;"Hai preso ";s
;" gocce";("s" AND s<>1)
8030 IF h>=s THEN PRINT AT 2,8;"
```

```
Il record e' ";h: GO TO 8050
8040 PRINT AT 2,7;"Nuovo record
":: LET h=s
8050 PRINT AT 4,0;"Vuoi giocare
ancora ? (s/n) ?"
8060 GO TO 8060+(1930 AND INKEY$
="n")+(INKEY$="s")
8070 IF INKEY$<>"" THEN GO TO 80
70
8080 PRINT AT 6,0;"Vuoi le istru
zioni ? (s/n) ?"
8090 GO TO 8090-(3085 AND INKEY$
="s")-(3030 AND INKEY$="n")
9000 REM User-grafica
9002 DIM t$(2): DIM b$(4)
9010 RESTORE 9050: FOR i=1 TO 9:
FOR n=0 TO 7: READ a: POKE USR
CHR$ (143+i)+n,a
9020 IF i>1 AND i<4 THEN LET t$(
i-1)=CHR$ (143+i)
9030 IF i>3 AND i<8 THEN LET b$(
i-3)=CHR$ (143+i)
9040 NEXT n: NEXT i
9050 DATA 0,16,16,56,56,124,56,0
9060 DATA 80,82,145,16,16,12,18,
0
9070 DATA 60,74,137,8,8,48,72,50
9080 DATA 68,60,12,12,12,84,36,1
0
9090 DATA 68,60,12,12,12,11,9,24
9100 DATA 34,60,48,48,48,42,36,4
0
9110 DATA 34,60,48,48,48,208,144
,24
9120 DATA 0,0,0,195,36,153,102,2
4
9130 DATA 0,0,0,24,165,153,90,60
9140 RETURN
9990 REM dati motivo
9500 DATA .5,0,.37,0,.12,-1,.5,0
,-.37,3,.12,2,.37,2,.12,0,.37,0,.
12,-1,1,0
9900 BORDER 2: PAPER 6: INK 2: C
LS: REM Programmazione
9990 PRINT AT 10,13; FLASH 1;" C
IAO ": STOP
```

Note

10	Inizializza
20-80	Loop per muovere le gocce e per verificare
30	Cancella la goccia
50	Rileva se ha colpito l' omino
60	Rileva se ha colpito il suolo
70	Disegna la goccia
90	Fornisce l' input e cancella la vecchia sagoma dell' omino

100	Cambia la posizione
110	Ristampa l' uomo
1000	Stampa lo splash, aggiorna il punteggio e inizializza una nuova goccia
2000	Disegna il nuovo livello
2010	Incrementa il livello dell' acqua e verifica se e' ad altezza d' uomo
5005-5030	Istruzioni
5040-5050	Tutto pronto
5060	UDG
5070	Inizializza le variabili
5080	Inizializza le gocce negli array
5090-5110	Disegna lo schermo
8000	Disegna l' omino carico e suona il motivetto
8010-8090	Bit finale (punti, nuova gara ecc...)
9000	Inizializza le variabili stringa per l' omino
9010-9040	Definisce gli user-graphic e li pone nelle variabili stringa
9050-9130	Dati per gli UDG
9990-9999	Fine del game

Capitolo 9

COME RENDERE PIU' INTERESSANTI I GAMES

L' attrattiva di un game molto dipende dal modo in cui viene presentato. Non cadete nella prolissita', eliminate le frange inutili senza fare piu' di quanto il listato non richieda. Badate anche a non introdurre troppe presentazioni o infioriture che, se da un lato migliorano l' aspetto del display, dall' altro rendono piu' lento il programma.
Eccovi alcuni preziosi suggerimenti :
-Stendete le istruzioni con una nota di humor rendendole piu' divertenti e fuori dalla norma apportando qua' e la' piccoli ritocchi tanto da far capire che i games servono a divertire e non vanno presi troppo seriamente.
-In casi particolari, il computer deve dare l' impressione di commentare l' evento. Ad esempio, in un game tipo "Mastermind", potete inserire prima e dopo ogni mossa, frasi del genere : "Siamo duri eh?..." seguito da una pausa e poi "finalmente ci sei arrivato!". Anche qui pero', occhio a non esagerare perche' cadreste nel noioso.
-Per dare forma agli oggetti usate gli UDG e non simboli come l' asterisco, il punto esclamativo o simili che non rendono assolutamente alcuna idea. La gente non ricorda se la bomba e' grossa la meta' dell' aereo che la sgancia, bensi' nota il realismo delle scene e del succedersi delle azioni.
-Sperimentate diversi colori usando le piu' svariate combinazioni, e magari randomizzando i colori di PAPER e di INK in funzione di certi eventi cosi' come avviene nel programma ICBM.
-Generate spesso le esplosioni, sono cariche di effetto. State pero' attenti che all' atto dell' esplosione,il resto del programma rimane inattivo per cui un abuso in questo senso porterebbe a interruzioni troppo frequenti che ostacolerebbero l' effetto prefissato.
-Dovendo inserire effetti musicali, non affidatevi a frequenze di note random, ma ricorrete a filastrocche piacevoli come spiegato nel contesto del volume.
Fate attenzione alla velocita' con la quale il programma gira. E' vero che piu' una azione e' veloce e meglio e', ma anche qui non bisogna esagerare sconfinando nell' impossibile. Se ritenete troppo rapido lo svolgimento di determinate fasi, rallentatelo frapponendo piacevoli effetti musicali. Ovviamente la velocita' di un game dipende da piu' fattori quali ad esempio : le tecniche usate, ivi compresa la compattezza del programma in codice, l' organizzazione dei componenti e quella delle variabili. Un programma con "x" linee comprendenti' ognuna uno statement gira piu' lentamente di "x" statement raggruppati nella stessa linea.
Il componente piu' importante di un game e' il loop principale, seguito dalle subroutines, prima quelle piu' usate poi quelle meno (come le istruzioni). Quando il programma incontra GO TO o

GO SUB, esplora in successione i numeri delle linee fino a che non raggiunge quella richiesta. Poiche' il computer parte dall' inizio del listato, piu' vicina si trova la linea alla quale va eseguito il salto e piu' rapidamente questo avviene. Stesso discorso vale quando il computer consulta una variabile. Quelle insite nel primo blocco considerato vengono inizializzate per prime e quindi rintracciate piu' velocemente.
Il BASIC viene spesso criticato per essere un linguaggio "non strutturato" a causa appunto del GO TO che, se usato impropriamente, dirotta fuori destinazione il flusso del programma. E' possibile a tale scopo, strutturare i programmi in modo da far loro assumere le caratteristiche del PASCAL il quale impiega dei loop formati da GO SUB. Questo procedimento si basa su di un "sottoprogramma", richiamato dal loop principale, in grado di fare le operazioni piu' svariate come causare un' esplosione, far sparare un laser, oppure muovere gli invaders.
Provate a dare un' occhiata a quanto segue, e tenetene presente come intelaiatura di games :

```
   10 GO SUB 5000: REM inizializz
azione
   100 REM LOOP PRINCIPALE
   200 GOSUB GET INPUT
   300 GOSUB MOVE BASE
   400 GOSUB MOVE INVADERS
   500 GOSUB CHECK END: REM IF en
d=1 THEN END
   600 GOSUB FIRE
   700 GOSUB CHECK HIT
   800 IF NOT end THEN GO TO 20
   801 REM FINE DEL LOOP
   900 GOSUB EXPLODE
  1000 STOP
  4000 REM ***INPUT***
  4000:
  4999 RETURN
  5000 REM ***MOVIMENTO BASE***
  5000:
  5999 RETURN
  6000 REM **MOVIMENTO INVADERS***
 10000:
 14999 RETURN
 15000 REM ***FINE***
 15000:
 15999 REM IF [hit] THEN end=1
 16000 RETURN
 19000 REM ***FUOCO***
 40000:
 44999 RETURN
 45000 REM **RILEVAMENTO URTI**
 55000:
 59999 RETURN
 60000 REM ***ESPLOSIONE***
 60000:
 64999 RETURN
 40000 REM ***INIZIALIZZAZIONE***
```

```
5000 REM istruzioni
5010:
5200 REM inizializzazione
5210 LET end=0
5220:
5300 LET GET INPUT=2000
5310 LET MOVE BASE=3000
5320 LET MOVE INVADERS=1000
5330 LET CHECK END=1500
5340 LET FIRE=2000
5350 LET CHECK HIT=2500
5360 LET EXPLODE=3000
5490 RETURN
```

Come potete constatare, questo metodo e' chiarissimo ma, porta a programmi di una certa lunghezza. Suo grosso vataggio, e' quello della tecnica modulare in base alla quale ogni blocco puo' essere registrato separatamente e aggiunto ad altri tramite MERGE.

La "standardizzazione" delle subroutines e delle variabili e' talvolta indispensabile per rendere piu' comoda la stesura del listing. Io, di solito, uso "i" e "n" per il controllo dei loop, "s" oppure "x" per determinare i punti e cosi' via. La ricorrenza di questi simboli mi permette di scegliere appropriatamente il nome delle altre variabili.
Ed eccoci gunti al termine. Con l' augurio che questo libro vi abbia fornito le informazioni indispensabili a realizzare games sempre piu' divertenti, mi congedo augurandovi "buona programmazione"!.

Il volume descrive dettagliatamente tutte le tecniche di stesura di giochi ARCADE, partendo dalla lettura della tastiera e toccando la definizione grafica, l'impiego del suono e l'uso degli operatori logici, per migliorare la qualità dei programmi. Altri argomenti esaminati in dettaglio sono l'animazione degli oggetti, lo scrolling dello schermo e l'impiego dei comandi PEEK e POKE per il loro uso più corretto.
Il tutto è accompagnato da 18 programmi la maggior parte dei quali o sono inediti, oppure riguardano versioni migliorate di games di grande successo come "INVADERS" e "BOMBER". I più interessanti sono stati registrati sulla **cassetta allegata al volume** al fine di farvi risparmiare ore di digitazione.

Cod. 9003

L. 25.000
prezzo comprensivo di cassetta

CREATE GIOCHI ARCADE COL VOSTRO SPECTRUM

di DANIEL HAYWOOD

edizioni JCE