L'ARTE DI USARE I CICLI FOR ... NEXT

Un ciclo FOR ... NEXT serve per far contare il computer fino a un numero prefissato: questa funzione, apparentemente semplice, ha un'infinità di applicazioni

I cicli FOR ... NEXT sollevano il programmatore da molte fastidiose ripetizioni, all'interno dei programmi.

Quando, ad esempio, vogliamo che il computer ripeta una serie di operazioni un numero prefissato di volte, si usa un ciclo FOR ... NEXT.

Come vedremo, i cicli FOR ... NEXT si possono anche usare per 'dipingere' col computer, ma sono anche utili nei giochi e nei programmi applicativi.

COS'È UN CICLO FOR ... NEXT

In BASIC, un ciclo FOR ... NEXT è una particolare struttura per far ripetere al computer più volte una stessa operazione.

Supponiamo, per esempio, di voler trovare le radici quadrate di tutti i numeri compresi tra 1 e 100. Potremmo scrivere:

```
PRINT SQR(1)
PRINT SQR(2)
PRINT SQR(3)
```

e via dicendo. Il computer, a ogni richiesta così formulata, visualizzerebbe il risultato. Un simile sistema, però, è quantomai faticoso.

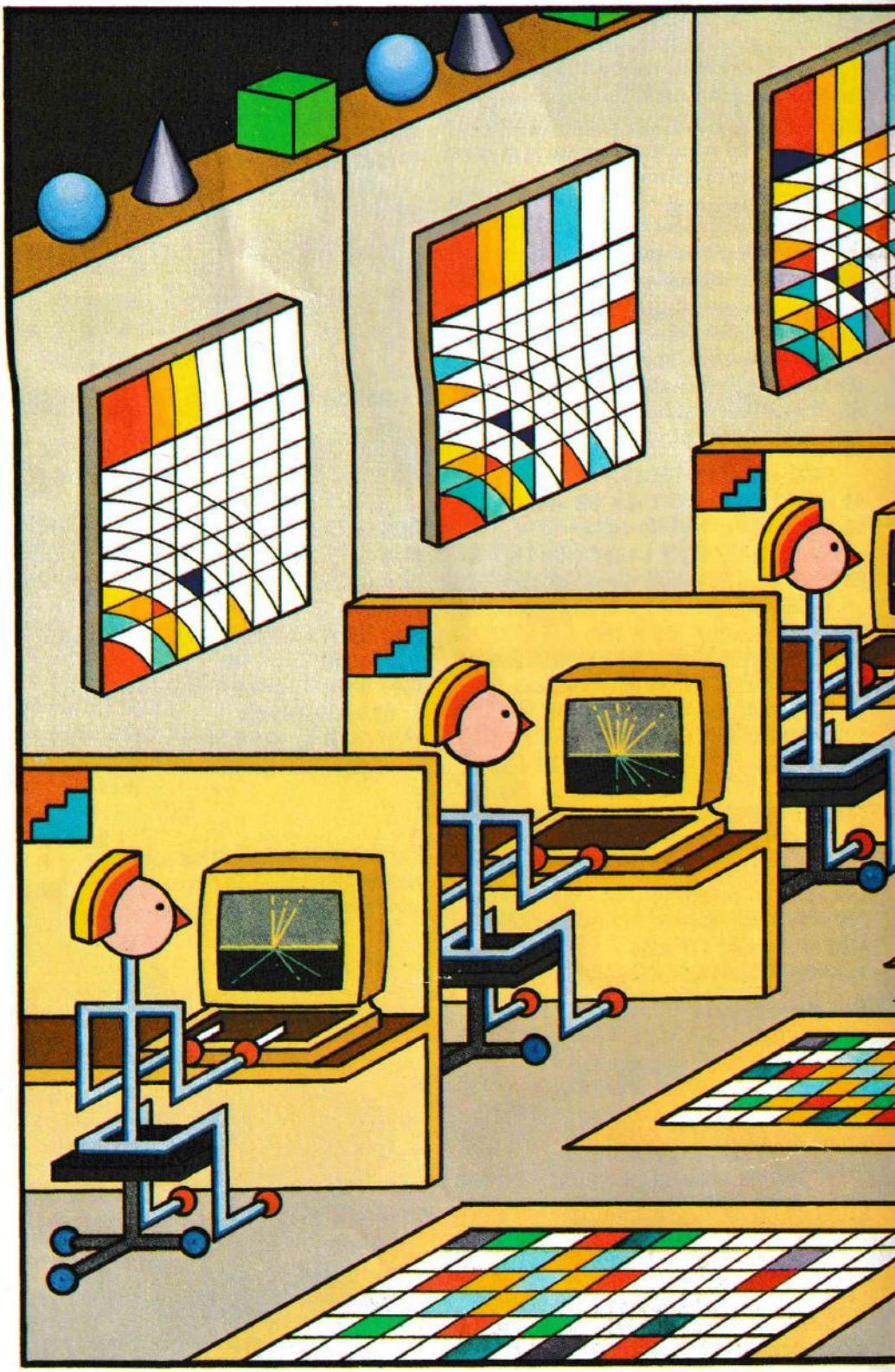
Per un uso più efficiente delle capacità del computer si scriva invece:

```
10 FOR n=1 TO 100
20 PRINT n, SQR n
30 NEXT n
40 PRINT "ecco fatto!"
```

```
10 FOR N=1 TO 100
20 PRINT N;"□";SQR(N)
30 NEXT N
40 PRINT "ECCO FATTO!"
```

```
10 FOR N=1 TO 100
20 PRINT N, SQR (N)
30 NEXT N
40 PRINT "ECCO FATTO!"
```

Il programma fa sì che il computer visualizzi, automaticamente, il numero 1 e la sua radice quadrata, poi il numero 2 e la



- IL COMPUTER CONTA DA 1 a 1ØØ
- I CICLI **FOR ... NEXT** PER CREARE PAUSE
- LA CREAZIONE DI EFFETTI AUDIO
- DIPINGERE COI NUMERI
- UN'AURORA MULTICOLORE COI CICLI **FOR ... NEXT**
- PATCHWORK E RICAMI COMPUTERIZZATI

sua radice e così via, fino ad arrivare al numero 1ØØ.

Come ciò possa accadere è presto detto. Quando il BASIC incontra l'istruzione FOR, sa che le successive linee di programma (comprese tra la FOR e la NEXT più 'vicina') devono essere ripetute un certo numero di volte (nel nostro esempio devono essere ripetute 1ØØ volte).

Cosicché, la linea 2Ø viene eseguita una prima volta, calcolando la radice quadrata di 1, poi una seconda volta, calcolando la radice quadrata di 2, poi una terza volta calcolando la radice quadrata di 3 e così via di seguito.

Quando il programma raggiunge il massimo valore (specificato dopo la FOR), l'esecuzione prosegue dalla linea 4Ø.

I VALORI FRAZIONARI

Un ciclo FOR ... NEXT può anche procedere per 'passi' diversi da 1.

Si provi ad esempio:

```
1Ø FOR n=1 TO 3Ø STEP 2.7
2Ø PRINT n, SQR n
3Ø NEXT n
```

```
1Ø FOR N=1 TO 3Ø STEP 2.7
2Ø PRINT N;"□";SQR(N)
3Ø NEXT N
```

```
1Ø FOR N=1 TO 3Ø STEP 2.7
2Ø PRINT N, SQR(N)
3Ø NEXT N
```

Si noterà che il computer non si preoccupa del fatto che 3Ø non sia divisibile per il passo richiesto, ma che, raggiunto il valore più vicino, il programma si arresta.

Un'altra cosa, alla quale il computer non presta attenzione, è il numero di linee comprese tra FOR e NEXT: le esegue tutte, quante esse siano.

AZIONE DI RITARDO

I cicli FOR ... NEXT vengono utilizzati per moltissimi scopi.

Uno di questi è semplicemente quello di... perdere tempo!

Si veda, a questo proposito, il programma delle tabelline a pagina 7.

In quel caso, il ciclo FOR ... NEXT non fa altro che contare ed il risultato più appariscente è una pausa nell'esecuzione. I computer contano molto rapidamente e bastano poche frazioni di secondo (il preciso intervallo di tempo varia da computer a computer) per fare una semplice somma.

Per ottenere una pausa che sia apprezzabile, è necessario fargli contare almeno fino a 1ØØØ.

Se però eseguiamo il ciclo riportato di seguito:

```
1Ø FOR N=1 to 1ØØØØØ
2Ø NEXT N
```

con tutta probabilità avremo il tempo di uscire comodamente per berci un buon caffè!

D+R

Come fare per tenere il conto delle variabili impiegate in un programma ed evitare doppioni?

Come regola generale, un listato di programma è più chiaro se alle variabili vengono assegnati nomi composti da più lettere, facili da ricordare e di senso compiuto. Ciò è particolarmente vero nei programmi più lunghi. Scrivere FOR riga ... o FOR colonna ... è più facilmente interpretabile di FOR x ... o FOR y. Alcuni BASIC, però, sono in grado di riconoscere soltanto nomi composti da non più di due lettere: in questo caso, può esser utile annotarsi, sopra un foglietto a parte, il nome di ciascuna variabile ed una breve descrizione del suo impiego.

RALLEGRIAMO LE PAUSE

Talvolta, i programmatori, per rendere meno noiose le pause, le 'allietano' con qualche sottofondo musicale. Si provi:

```
10 FOR n=29 TO 10 STEP-1
20 BEEP .015, n
30 NEXT n
```

```
10 FOR N=160 TO 100 STEP -4
20 SOUND 1,-15, N, 1
30 NEXT N
```

```
10 FOR N=12 TO 1 STEP -1
20 PLAY "T40;04;"+STR$ (N)
30 NEXT N
```

Suoni di questo tipo vengono spesso usati per avvertire di un errore, oppure, nei giochi, per segnalare l'atterraggio di una navetta spaziale. Da notare che, quando il conto avviene alla rovescia, il passo STEP deve essere negativo (nel nostro esempio è −1).

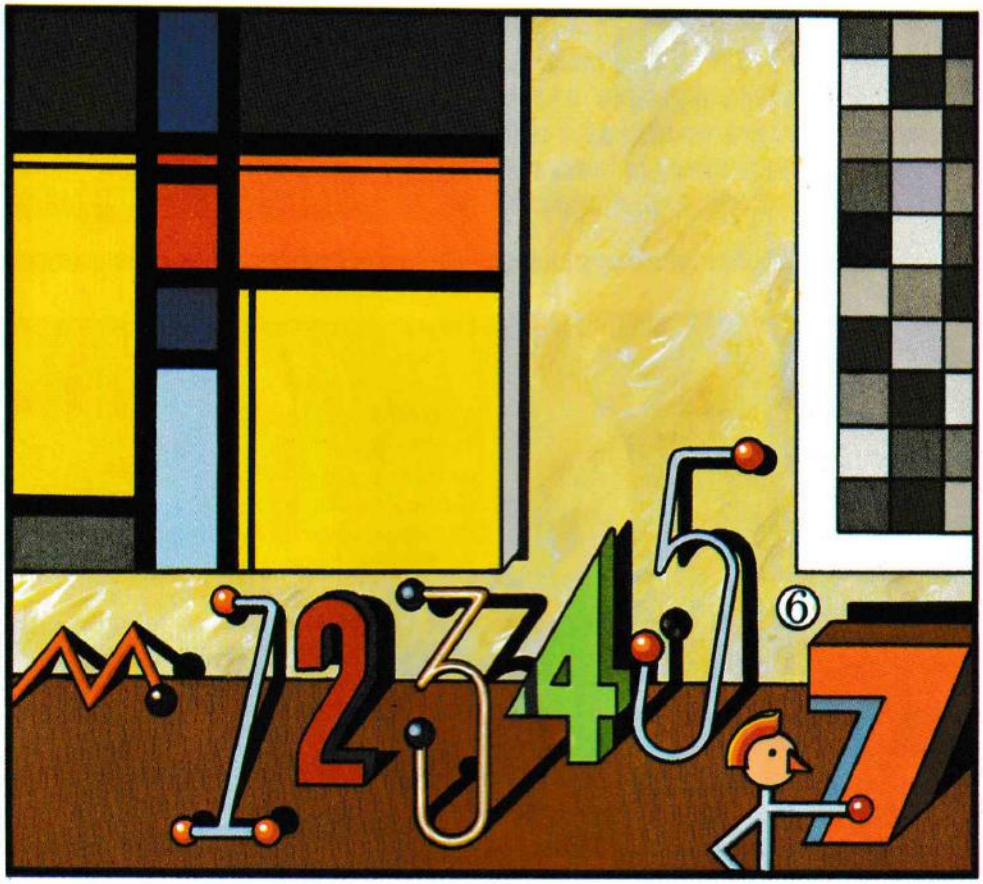
(Il Commodore 64 non possiede comandi elementari per i suoni e occorrerebbe scrivere un apposito sottoprogramma, richiamato mediante una GOSUB inserita alla linea 20).

Microtip

Usiamo il ciclo giusto

L'uso dei cicli FOR ... NEXT è assai frequente, ma altrettanto frequenti sono i casi nei quali è bene evitarne l'uso. La regola generale è:
quando si desidera far ripetere un gruppo di istruzioni un numero prefissato di volte, senza interruzioni di sorta, allora deve essere impiegato un ciclo FOR ... NEXT.

Quando, invece, si desidera far ripetere una certa sequenza fino al verificarsi o meno di una condizione, è preferibile utilizzare strutture del tipo WHILE ... WEND oppure REPEAT ... UNTIL (se il BASIC non consente queste strutture, è necessario ricorrere all'istruzione GOTO, che è posta all'interno del ciclo).



DIPINGERE COI NUMERI

Un po' per gioco e un po' per imparare qualcosa ancora sui cicli, vediamo come ottenere degli effetti grafici utilizzando i FOR ... NEXT. Ecco un esempio:

```
10 FOR n=0 TO 21
20 LET m=RND*31
30 INK RND*7+1
40 PRINT AT n,m; "■"
50 NEXT n
60 GOTO 10
```

```
8 MODE 2
9 VDU23;8202;0;0;0;
10 FOR riga=0 TO 30
20 LET colonna=RND(19)
30 COLOUR 128+RND(7)
40 PRINT TAB(colonna,riga)"□";
50 NEXT riga
60 GOTO 10
```

```
7 CLS0
10 FOR N=0 TO 63
20 LET M=RND(32)-1
30 LET C=RND(9)-1
40 SET (N,M,C)
50 NEXT N
60 GOTO 10
```

In questo caso, la linea 10 fissa la profondità dell'immagine che verrà creata sullo schermo e, al tempo stesso, avverte il computer che verrà visualizzata una riga alla volta.

La linea 20 fissa invece la larghezza dell'immagine e, assieme alla linea 40, comanda al computer di visualizzare a caso dei quadratini colorati.

La linea 30 genera una colorazione casuale dei quadratini.

```
10 PRINT "■"
15 FOR N=0 TO 24
20 M=INT(RND(1)*18)
25 C=INT(RND(1)*40)
30 POKE 1024+(40*N)+C,160
40 POKE 55296+(40*N)+C,M
50 NEXT
60 GOTO 15
```

Il programma per il Commodore funziona in modo leggermente diverso: la linea 15 'conta' le righe dello schermo, la linea 20 sceglie (a caso) la quantità di quadratini da visualizzare su ciascuna riga, mentre

la linea 30 seleziona, sempre a caso, la colorazione dei quadratini.

VARIAZIONE SUL TEMA

Per familiarizzarsi sia con i cicli FOR ... NEXT che con la funzione RND, si può tentare di modificare i programmi appena visti. Ecco, tra le tante possibili, due variazioni (non si dimentichi d'impartire il comando NEW quando s'imposta un nuovo programma):

```
10 FOR n=0 TO21
20 FOR m=0 TO 31
30 INK RND*7+1
40 PRINT AT n,m;"■"
45 NEXT m
50 NEXT n
60 GOTO 10
```

```
10 LET n=RND*21
20 FOR m=0 TO 31
30 INK RND*7+1
40 PRINT AT n,m; "■"
45 NEXT m
60 GOTO 10
```

```
8 MODE 2
9 VDU 23;8202;0;0;0;
10 FOR riga=0 TO 30
20 FOR colonna=0 TO 19
30 COLOUR 128+RND(7)
40 PRINT TAB(colonna,riga)"□";
45 NEXT colonna
50 NEXT riga
60 GOTO 10
```

```
8 MODE 2
9 VDU 23;8202;0;0;0;
10 LET riga=RND(31)-1
20 FOR colonna=0 TO 19
30 COLOUR 128+RND(7)
40 PRINT TAB(colonna,riga)"□"
50 NEXT colonna
60 GOTO 10
```

```
8CLS0
10 FOR N=1 TO 60
20 FOR M=0 TO 31
30 LET C=RND(9) -1
40 SET (N.M.C)
45 NEXT M
50 NEXT N
60 GOTO 10
```

```
8CLS0
10 LET N=RND(60)
20 FOR M=0 TO 31
30 LET C=RND(9) -1
40 SET (N.M.C)
45 NEXT M
60 GOTO 10
```

```
10 PRINT "□"
20 FOR M=0 TO 999
30 LET C=INT(RND(1)*16)
40 POKE 1024+M,160
50 POKE 55296+M,C
60 NEXT M
70 GOTO 20
```

```
10 PRINT "□"
20 LET N=INT(RND(1)*25)*40
30 FOR M=N TON+39
40 LET C=INT(RND(1)*16)
50 POKE 1024+M,160
60 POKE 55296+M,C
70 NEXT M
80 GOTO 20
```

D+R

Le istruzioni del BASIC, quali ad esempio PRINT, devono essere sempre digitate maiuscole?

In genere sì, tranne nel Commodore, dove si usano le maiuscole se si è in 'modo maiuscolo', le minuscole se si è in 'modo text'.

Prima di proseguire, ecco un piccolo esperimento che andrebbe fatto sullo Spectrum, sugli Acorn e sul Dragon: si cancelli la linea 45 dal primo dei due programmi e si esegua di nuovo il programma dopo le seguenti modifiche:

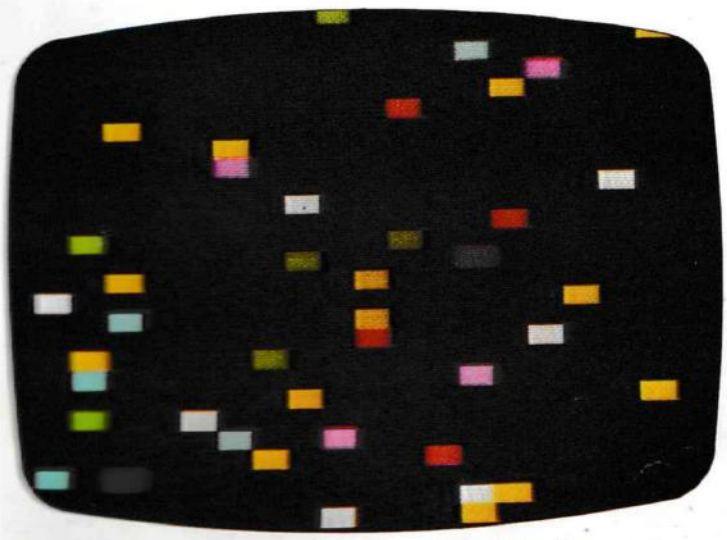
```
55 NEXT m
```

```
55 NEXT colonna
```

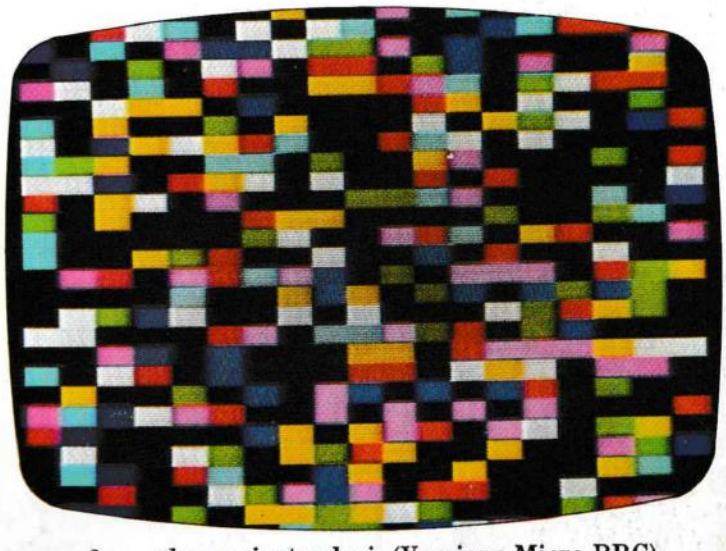
```
55 NEXT M
```

Questo serve a scoprire l'ultima importante caratteristica dei cicli FOR ... NEXT: se ne esiste più d'uno, nel medesimo programma, ciascun ciclo deve essere *completamente contenuto* (il termine esatto è *nidificato*) all'interno di un altro. Se due cicli si sovrappongono, il programma non funziona.

Sul Commodore, nel caso specifico di questo programma, un simile accorgimento è superfluo, ma la regola generale è comunque valida.



1. Prima fase del 'patchwork' sullo schermo...

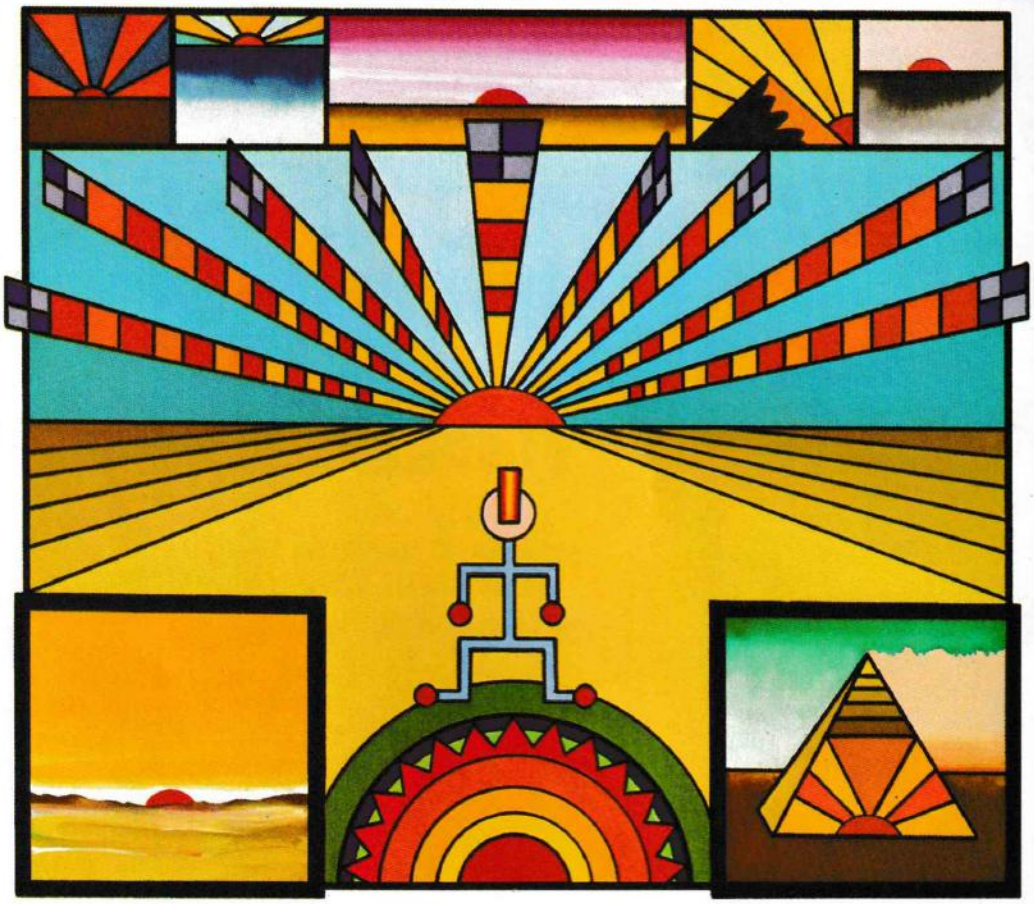


2. ... che va ripetendosi. (Versione Micro BBC)

UN'AURORA MULTICOLORE

Questo programma utilizza un ciclo FOR ... NEXT per generare l'immagine di un'aurora.

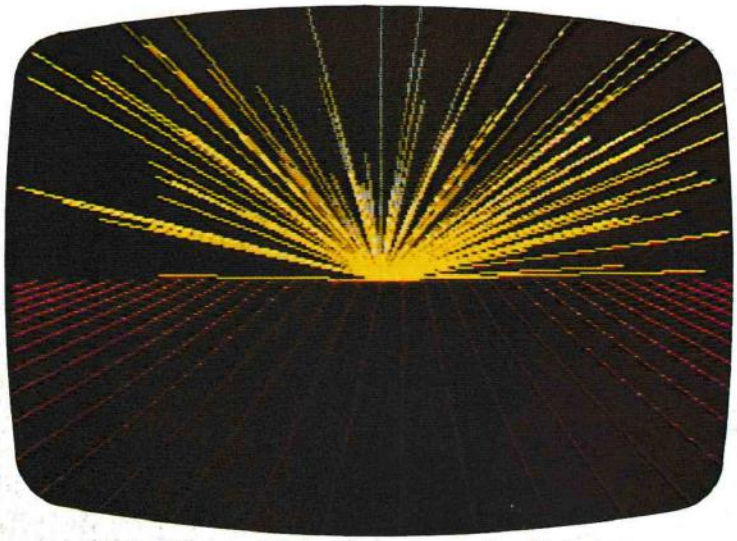
Nella prima parte viene fissata l'origine, sullo schermo, dalla quale partono i 'raggi solari', nella seconda parte del programma, invece, vengono tracciate le linee prospettiche di sottofondo.



```
5 BORDER 0:PAPER 0:INK 6:CLS
10 FOR n=1 TO 80
20 PLOT 127,75
30 DRAW INT (RND*250)-125,INT(RND*97)
40 NEXT n
45 INK 5
50 FOR n=75 TO 0 STEP-15
60 PLOT 127,75
70 DRAW-127,-n: PLOT 127, 75: DRAW
   127,-n
80 NEXT n
100 FOR n=- 127 TO 127 STEP 20
110 PLOT 127,75
120 DRAW n,-75
130 NEXT n
```

```
10 MODE1
15 GCOL0, 2
20 FOR S=1 TO 80
30 LET X=RND(1280)
40 LET Y=512+RND(512)
50 MOVE 640,512
60 DRAW X,Y
70 NEXT S
75 GCOL0, 1
80 FOR L=0 TO 1280 STEP 40
90 MOVE L,512
100 DRAW(L-512)*4,0
110 NEXT L
```

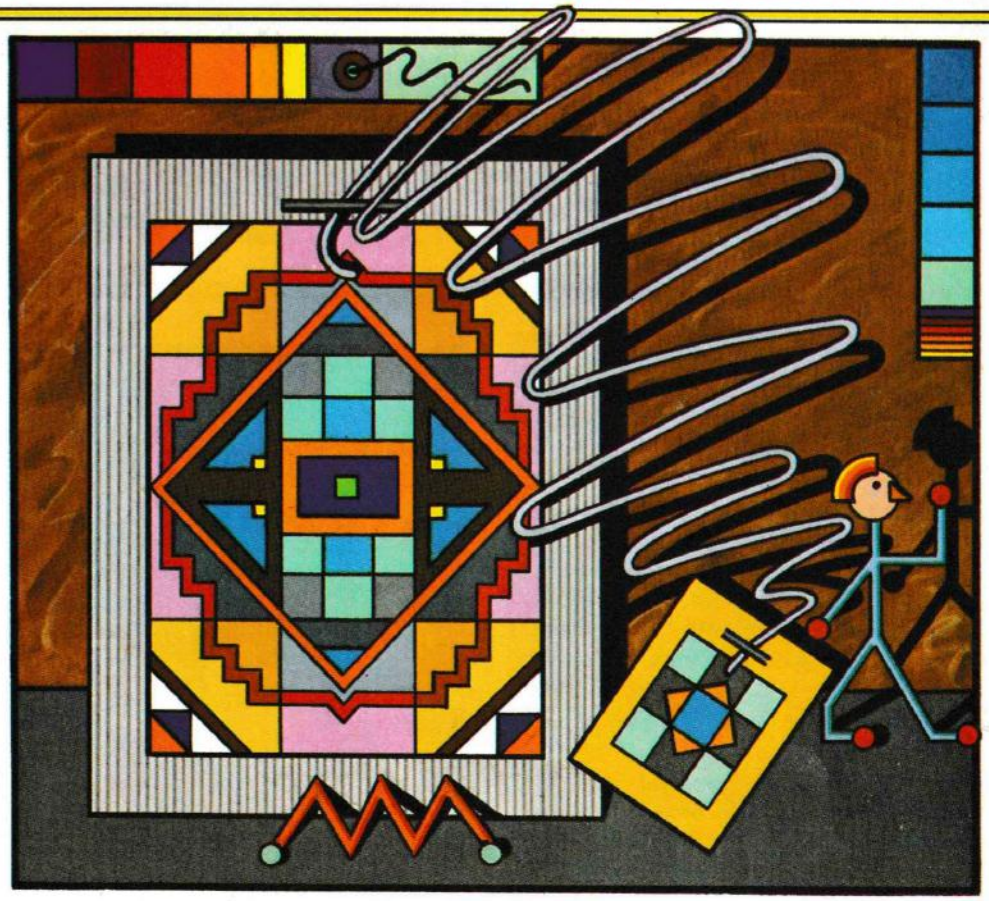
```
20 PMODE 3,1
30 PCLS 3
40 COLOR 2
50 SCREEN 1,0
60 FOR N=1 TO 80
70 LINE(127,95)-(256-RND(256),96
   -RND(96)),PSET
80 NEXT N
90 COLOR 4
100 FOR N=95 TO 191 STEP 12
110 LINE(127,95)-(0,N),PSET
120 LINE(127,95)-(255,N),PSET
130 NEXT N
140 FOR N=0 TO 255 STEP 10
150 LINE (127,95)-(N,191), PSET
160 NEXT N
170 GOTO 170
```



3. Il levarsi del sole ... secondo gli Acorn

4. Un bel ricamo eseguito al ... BBC!

RICAMI... ISTANTANEI



Infine, ecco un programma veramente spettacolare per tre dei nostri computer:

```
10 FOR n=0 TO 255 STEP 2
15 INK RND*8
20 PLOT 0,0: DRAW n,175
30 PLOT 255,0: DRAW -n,175
40 PLOT 0,175: DRAW n, -175
50 PLOT 255,175: DRAW -n,-175
60 NEXT n
70 GOTO 10
```

```
8 MODE 2
10 FOR N=0 TO 1279 STEP 10
15 GCOL 0,RND(7)
20 MOVE 0,0: DRAW N,1023
30 MOVE 1279,0: DRAW 1279-N,1023
40 MOVE 0,1023: DRAW N,0
50 MOVE 1279,1023: DRAW 1279-N,0
60 NEXT N
70 GOTO 10
```

```
3 PMODE 3,1
6 PCLS
9 SCREEN1,0
10 FOR L=0 TO 255 STEP 2
15 COLOR RND(4)
20 LINE (0,0)-(L, 191),PSET
30 LINE (255,0)-(255-L, 191),PSET
40 LINE(0,191)-(L,0),PSET
50 LINE(255,191),-(255-L,0),PSET
60 NEXT L
70 GOTO 10
```

Ciascuno dei quattro segmenti che compongono l'immagine inizia con un sol punto, posto in un angolo dello schermo. Il ciclo FOR ... NEXT, in questo caso, serve per costruire le linee colorate che attraversano lo schermo. Come funzioni esattamente l'impostazione grafica di questo programma sarà chiaro più avanti nel corso, ma si provi a eliminare le linee 30 e 40 ...

Altri esperimenti possono essere i seguenti: eliminare il tracciamento di alcune delle righe, variare i colori impiegati o il valore dello STEP nella linea 10, omettere la GOTO alla fine del programma. In pochi minuti si possono ottenere centinaia di 'motivi' originali degni delle più note case di moda! Questi esperimenti, oltre a esser più o meno divertenti, insegnano anche ad abituarci all'uso delle varie istruzioni grafiche.

5. La versione su Spectrum è più stilizzata...

6. ... ma le variazioni sono infinite!