ANNA Animation Language

Per Apple II e MC-Tablet

di Roberto Angeletti

Torniamo a parlare dell'animazione. La volta scorsa (n. 35) abbiamo introdotto i problemi relativi all'immagazzinamento di una serie di immagini ed abbiamo presentato un programma in grado di costruire un archivio di disegni che potevano essere rivisti in sequenza, come con un proiettore di diapositive; per questo, le immagini venivano chiamate SLIDE. Chi si è cimentato nell'impresa "ippica" dell'esempio pubblicato, avrà trovato però il trotto del cavallo più simile a quello d'un brocco che a quello di un purosangue.

Come noto, il problema della velocità è quello principale nell'animazione; per questo, l'uso del Basic può riuscire "fatale". Un compilatore, come il TASC, può rendere molto più veloce il programma, eliminando i salti all'interprete ed il tempo, nel migliore dei casi, può ridursi ad un sesto. Tuttavia, il programma compilato ha un notevole spreco di memoria ed, inoltre, il comando della macchina risulta talmente limitato, da poter essere ristabilito solo da un RESET.

Questa mancanza di praticità, l'impossibilità di poter rimanipolare istruzioni di programma, cosa essenziale nel caso si volessero creare "in corso d'opera" altri effetti non previsti, e l'esiguità di memoria rimanente per immagazzinare le immagini, fanno sì che l'uso del compilatore, innegabilmente utile per altre applicazioni, debba essere scartato.

La cosa migliore da fare è quella di usare direttamente l'ASSEMBLER per scrivere le stesse "primitive" routine.

Come promesso, pubblichiamo appunto la "primitiva" principale del linguaggio ANNA, che abbiamo sviluppato, tralasciando, per questa volta, la trattazione di ogni singolo passo di programma, cosa che faremo in seguito.

La routine, nonostante vada ad inserirsi in memoria al di sotto della prima pagina grafica, in zona Basic, può convivere con un buon numero di programmi, a meno che questi non siano eccessivamente lunghi. Essenzialmente, essa può stare insieme al programma pubblicato la volta scorsa ed, anzi, quest'ultimo andrà modificato per giovare delle nuove possibilità. Vediamo le operazioni da compiere. Per prima cosa occorre andare in MONITOR con CALL-151 e bisogna inserire, a partire da $1E00, il dump di memoria della figura. Dopo di ciò, si inserisce da $300 la seconda routine, che è di pulizia delle pagine grafiche, più veloce di quella in ROM. Tornati, quindi, in BASIC con CTRL-C, si salva il tutto con "BSAVE ANNA. 0,A$1E00,L $14F" e con "BSAVE SPEED-CLEAR.0, A$300,L$D0".

Si carica, ora, il programma "SLIDE-CODER" e vi si apportano le seguenti modifiche:

```
DEL 10,100
DEL 190,280
180 PRINT : PRINT D$ "BLOAD SPEED-CLEAR.0"
190 CALL 7680
300 PRINT : PRINT D$ "BLOAD PADDLE.CODE" : GOTO 1080
2340 PRINT D$ "BLOAD ANNA.0"
```

e registriamo anche quest'altra versione con "SAVE SLIDE-CODER.0".

Ciò fatto, facciamo girare il programma e, dopo aver caricato la Shape ZOOTROPE e la SEQ creata la volta scorsa, eseguiamo l'opzione 5 SHOW DELLA SLIDE-TABLE, che si avvale della nuova routine. Se tutto è stato battuto correttamente, vedremo, finalmente, il cavallo correre.

Anche se l'argomento di cui trattiamo è l'animazione, vi stimoliamo, a questo punto, ad un utilizzo del sistema anche per la codifica di singole immagini, che non necessariamente debbano essere viste in una sequenza. Visto, infatti, il notevole risparmio di memoria che si raggiunge con la codifica nel vettore, il programma SLIDE-CODER può essere usato, ad esempio, per il progetto di schemi elettrici, usando come elementi grafici i simboli di resistenze, condensatori e altro, oppure anche per provare varie soluzioni d'arredamento, usando simboli di sedia, tavolo, letto e così via; in questo modo, possiamo crearci una documentazione delle varie trasformazioni della nostra idea di partenza fino alla soluzione finale, evitando l'abituale spreco di carta che contraddistingue il lavoro creativo.

Le immagini "automatiche"

Tutte le applicazioni di cui abbiamo trattato fanno un uso del computer possiamo dire quasi "accademico"; infatti, tutto ciò che abbiamo finora proposto è realizzabile anche con strumenti tradizionali. Abbiamo, anzi, riproposto cose vecchie d'un secolo.

Per "automatica" intendiamo qualunque immagine noi riusciamo ad ottenere senza il nostro intervento manuale nel disegno, cosa che, invece, abbiamo fatto fino ad ora. Una prerogativa importante del computer, per ciò che ci riguarda, è quella di fornirci immagini derivanti da formule matematiche. Anche questo tipo di immagini può venire codificato con il sistema del vettore d'elementi grafici, della cui struttura abbiamo parlato la volta scorsa. Questa volta approfondiamo più specificamente il problema reale della codifica, vedendo come funziona la subroutine in BASIC e come possiamo riutilizzarla per qualsiasi altro programma.

Bisogna innanzitutto chiarire quella che può essere considerata una limitazione: il numero degli elementi per ogni SLIDE non può superare il fatidico tetto di 255. Questo per tre ragioni tra di loro interagenti e cioè che nella maggior parte dei casi questo numero è largamente sufficiente; che superdimensionare il vettore avrebbe significato rallentare, per problemi di riporto tra due byte, la routine in ASSEMBLER; infine, che, d'altro canto, il tempo di costruzione di ogni immagine con un alto numero di elementi diventa non più sostenibile per una efficace animazione. Per il momento, quindi, accontentiamoci.

Questo programma è disponibile su disco presso la redazione. Vedere l'elenco dei programmi disponibili e le istruzioni per l'acquisto a pag. 162.

```
*300.3CF

0300- A9 00 A2 00 9D 00 20 9D
0308- 00 21 9D 00 22 9D 00 23
0310- 9D 00 24 9D 00 25 9D 00
0318- 26 9D 00 27 9D 00 28 9D
0320- 00 29 9D 00 2A 9D 00 2B
0328- 9D 00 2C 9D 00 2D 9D 00
0330- 2E 9D 00 2F 9D 00 30 9D
0338- 00 31 9D 00 32 9D 00 33
0340- 9D 00 34 9D 00 35 9D 00
0348- 36 9D 00 37 9D 00 38 9D
0350- 00 39 9D 00 3A 9D 00 3B
0358- 9D 00 3C 9D 00 3D 9D 00
0360- 3E 9D 00 3F CA D0 9D 60
0368- A9 00 A2 00 9D 00 40 9D
0370- 00 41 9D 00 42 9D 00 43
0378- 9D 00 44 9D 00 45 9D 00
0380- 46 9D 00 47 9D 00 48 9D
0388- 00 49 9D 00 4A 9D 00 4B
0390- 9D 00 4C 9D 00 4D 9D 00
0398- 4E 9D 00 4F 9D 00 50 9D
03A0- 00 51 9D 00 52 9D 00 53
03A8- 9D 00 54 9D 00 55 9D 00
03B0- 56 9D 00 57 9D 00 58 9D
03B8- 00 59 9D 00 5A 9D 00 5B
03C0- 9D 00 5C 9D 00 5D 9D 00
03C8- 5E 9D 00 5F CA D0 9D 60
```

Routine di Spead-Clear.

```
*1E00.1F4D

1E00- 4C 0D 1E A9 40 85 E6 20
1E08- 68 03 4C 36 1E 20 E2 F3
1E10- 2C 52 C0 A2 03 20 EC F6
1E18- A9 64 85 FA A9 60 85 FB
1E20- 85 FD A9 00 85 FC A0 00
1E28- C8 A5 E6 C9 20 F0 D4 A9
1E30- 20 85 E6 20 00 03 98 48
1E38- B1 FC AA CA 8A 48 A0 03
1E40- B1 FA 48 88 B1 FA AA 29
1E48- 07 48 8A 29 40 F0 0A 8A
1E50- 29 BF 4A 4A 4A AA 20 EC
1E58- F6 68 C9 02 F0 39 C9 03
1E60- F0 3A C9 04 F0 4E C9 05
1E68- F0 51 C9 06 F0 56 C9 07
1E70- F0 5B 48 88 B1 FA AA 68
1E78- A8 68 20 57 F4 20 3F 1F
1E80- 68 AA E0 00 D0 B5 A5 E6
1E88- C9 20 F0 22 2C 55 C0 68
1E90- A8 CC 00 60 D0 92 60 A9
1E98- 00 4C 9E 1E A9 01 AA 88
1EA0- B1 FA 85 FE 68 A8 A5 FE
1EA8- 20 3A F5 4C 7D 1E 2C 54
1EB0- C0 4C 8F 1E A9 00 85 FF
1EB8- 4C D1 1E A9 00 85 FF A9
1EC0- 01 4C D1 1E A9 01 85 FF
1EC8- A9 00 4C D1 1E A9 01 85
1ED0- FF 48 88 B1 FA AA 68 A8
1ED8- 68 20 11 F4 20 3F 1F A0
1EE0- 01 B1 FA 85 F9 C8 B1 FA
1EE8- 85 E7 C8 B1 FA AA 20 30
1EF0- F7 48 A6 1A A4 1B A5 FF
1EF8- D0 07 68 20 01 F6 4C 7D
1F00- 1E 68 20 5D F6 4C 7D 1E
1F08- 00 00 00 00 00 00 00 00
1F10- 00 00 00 00 00 00 00 00
1F18- 00 00 00 00 00 00 00 00
1F20- 00 00 00 00 00 00 00 00
1F28- 00 00 00 00 00 00 00 00
1F30- 00 00 00 00 00 00 00 00
1F38- 00 00 00 00 00 00 00 D8
1F40- 18 A9 03 65 FA 85 FA A9
1F48- 00 65 FB 85 FB 60
```

Routine principale del linguaggio Anna.

Dunque, il vettore è dimensionato in maniera standard per un massimo di 100 SLIDE composte di fino a 255 elementi ognuna; è chiaro che non avremmo, nel caso limite, la memoria sufficiente per contenere le 76601 "parole" del vettore, quindi è necessario valutare le risorse. Lo spazio di memoria disponibile viene organizzato nel seguente modo:

$6000 : indicatore numero totale SLIDE contenute

$6001-$6064 : indicatori numero elementi per SLIDE

$6065 : elementi grafici

$: shape-table

Come si vede, nel caso venga usata una shape-table, questa viene messa da $7000 in poi, riducendo lo spazio per le SLIDE; nel caso, però che le shape siano poco ingombranti, si potranno spostare più in alto, e si dovrà cambiare opportunamente il puntatore con i soliti POKE 232, POKE 233.

Sospendiamo, comunque, il discorso sulle shape e consideriamole, per questa volta, una possibilità che non utilizzeremo; in questo modo, abbiamo a disposizione per il vettore da $6000 a $9600.

La "routine di coding" pubblicata è, in sostanza, la chiave d'ingresso al sistema ed è quella che si occupa di scrivere il vettore; è importante comprendere il suo funzionamento per poterla utilizzare correttamente per altri programmi grafici.

Innanzitutto, ecco la lista delle variabili usate, con il relativo significato:

DT : è la locazione d'inizio del vettore

SD% : costituisce un puntatore che viene continuamente aggiornato e che può essere considerato la "testina di scrittura" del vettore

NF% : numero totale delle SLIDE contenute

E% : numero elementi per ogni SLIDE

X% Y% : coordinate; devono essere comprese nei limiti dell'alta risoluz.

IS% : è l'istruzione per quel determinato elemento e va definita:
= 0 nel caso di HPLOT
= 2 nel caso di HPLOT TO
= 4 nel caso di DRAW
= 6 nel caso di XDRAW

RZ : rotazione,

SL : scala,

N : numero della shape

altre variabili locali, che sarà bene non usare nel resto del programma, sono:

H% L% : parti alta e bassa delle cifre intere

La linea 0 del programma va utilizzata per un salto incondizionato alle varie inizializzazioni, che devono contenere anche quelle delle variabili DT e SD%.

Per ogni nuova immagine bisognerà eseguire un GOSUB 10, che aggiorna il puntatore in $6000.

Per la codifica dei vari elementi grafici andrà eseguita una sequenza simile a questa:

```
X% = coord.x : Y% = coord.y : IS% = codice : GOSUB 20
```

Nel caso si tratti di una Shape, oltre alla riga precedente, bisognerà aggiungere anche:

```
RZ = rot. : SL = scale : N = n. shape : GOSUB 40
```

Questo è tutto e le routine si occuperanno del lavoro di codifica.

Nel listato segnaliamo la forse non del tutto trasparente istruzione:

```
ON E% > 255 GOSUB 10
```

contenuta nella riga 20; per chiarire ogni dubbio in proposito, pubblichiamo il riquadro a parte. Essa permette la codifica di un numero di elementi superiore a 255, usando la tecnica del "volta pagina"; in questo modo un'immagine complessa viene tagliata in diverse trance e, per riottenerla completa, si dovrà disattivare il page-flipping per continuare a sommare le diverse SLIDE una sull'altra. Questo è un argomento che affronteremo in seguito.

Seguendo ora la ben nota logica didattica per la quale è sempre meglio non appesantire troppo il discorso, per non avere un repentino calo di interesse, fermiamoci qui, fornendo una applicazione concreta di quanto detto.

I due programmi che pubblichiamo ottengono praticamente lo stesso risultato; l'uno, quello più breve, esegue in un tempo minore rispetto all'altro le due lettere del marchio della rivista, generate da semplici formule. Il secondo programma permette di codificare il disegno in diverse SLIDE. la cosa più interessante di questo programma è il fatto che in esso viene implementata la cosiddetta funzione in CLIPPING; che permette la visualizzazione parziale dell'intero disegno, evitando ILLEGAL QUANTITY ERROR. È utile parlare più diffusamente di questa funzione, dato che riveste grandissima importanza nella computer grafica.

Il Clipping

È generalmente chiamato Clipping la procedura di troncamento di segmenti che giacciono su una superficie parzialmente visualizzata. Un primo metodo potrebbe essere quello di calcolare il segmento per punti ed eseguire il seguente test:

```
100 IF X > 0 OR X > 279 OR Y < 0 OR Y > 191 THEN 120
110 HPLOT X,Y
120...
```

Tuttavia, tale sistema è estremamente lento, dato che esegue il test punto per punto anche con segmenti del tutto esterni al campo visualizzato. Il procedimento "per segmenti" è molto complesso, ma è quello adottato nella subroutine. Come noto, le quattro linee costituenti i confini dello schermo dividono il piano in nove regioni, su una qualunque delle quali ognuno degli estremi del segmento puo stare. Se entrambi sono nella regione centrale, non ci sono problemi ed il segmento può essere senz'altro disegnato; negli altri

casi, dei test devono determinare quale parte del segmento è visibile. La routine di clipping deve determinare sistematicamente ogni punto di intersezione tra la linea del segmento e tutti i confini; essa viene cosi divisa in più piccole porzioni di cui va disegnata solo quella, naturalmente, che sta all'interno dello schermo, mentre le altre parti vanno scartate.

Nella riga 2 della subroutine pubblicata, gli estremi del segmento vengono prima classificati tramite quattro variabili booleane, poste a 0 se il punto sta nella parte relativa allo schermo rispetto al confine considerato. D() rappresenta il confine sinistro; E() quello destro; F() quello superiore e B() quello inferiore. Inoltre, A() è la x, mentre C() è la y degli estremi del segmento.

Se entrambi gli estremi cadono al di fuori dello stesso confine, la somma dei prodotti tra i corrispondenti indicatori booleani sarà uguale ad 1 ed il segmento potrà essere scartato, essendo impossibile che esso attraversi lo schermo. Questo test viene eseguito dalla riga 4, mentre la riga 5 si occupa di vedere se entrambi gli estremi sono interni, eseguendo la somma degli indicatori.

Nel caso siano tutti zeri, viene eseguito il plottaggio. Nelle righe dalla 6 alla 9 viene calcolato il punto di intersezione nel caso di un estremo al di fuori del confine e, una volta trovato quest'ultimo, il segmento viene immesso ricorsivamente nel processo di verifica. Alla fine, il segmento può essere scartato del tutto oppure disegnato in parte.

Per utilizzare la subroutine basta porre le coordinate dei due estremi come:

A(1) = x1 : C(1) = y1 : A(2) = x2 : C(2) = y2

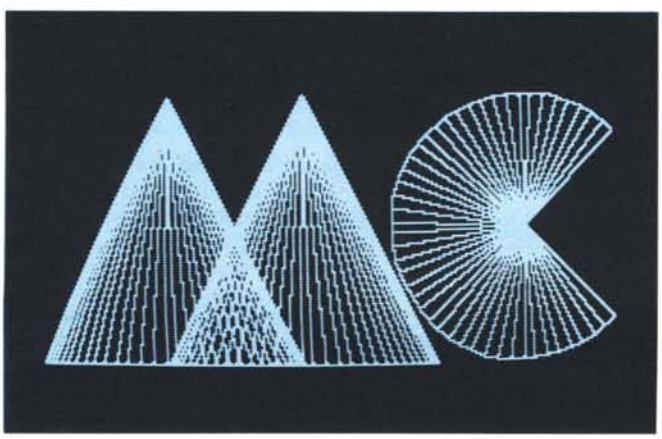
ed eseguire un GOSUB 1, che si occuperà del plottaggio o meno dell'intero o di una parte del segmento. Alla fine della routine c'è un miniprogramma di prova per verificare, inserendo coordinate, il corretto funzionamento. Cambiando i valori dei confini contenuti alla riga 0, si modifica il campo di visualizzazione che può, tuttavia, essere solo rimpicciolito e spostato all'interno dello schermo.

```
0  GOTO 1000
100 X =  COS (AN) * R + XO:Y =  SIN
    (AN) * R + YO: RETURN
200 XP = XO:YP = YO - 120:X = XO +
    60:Y = YO: RETURN
300  FOR AN = 0 TO PI STEP .1: HPLOT
    XP,YP TO X,YO: GOSUB 100: HPLOT
     TO X,YO: NEXT : HPLOT  TO X
    P,YP: RETURN
1000 PI = 3.1415:R = 60
1005  HGR2 : HCOLOR= 3
1010 XO = 220:YO = 100:AN = PI /
    4: GOSUB 100
1020  FOR AN = AN + .1 TO 7 / 4 *
    PI STEP .1
1030  HPLOT XO,YO TO X,Y
1040  GOSUB 100
1050  HPLOT  TO X,Y
1060  NEXT
1070  HPLOT  TO XO,YO
1080 XO = 120:YO = 160: GOSUB 200
    : GOSUB 300
1090 XO = XO - 60: GOSUB 200: GOSUB
    300
2000 :
2001 :
2002 :
2003  REM  PROGRAMMA UNO
```

Programma 1

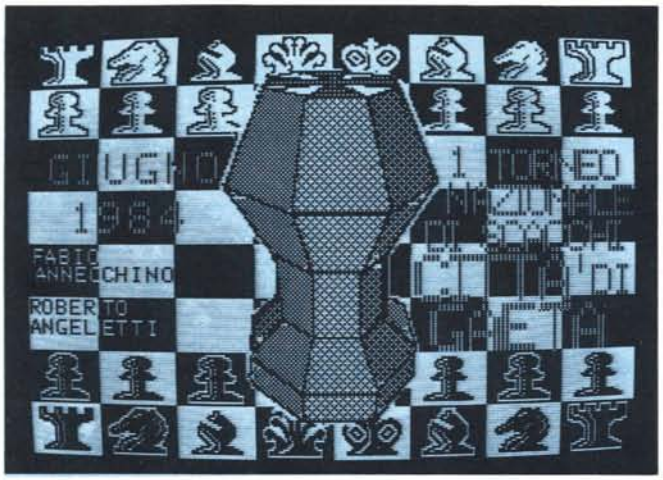
```
0  GOTO 10000
1 E = 1: GOSUB 2:E = 2: GOSUB 2: GOSUB 4: RETURN
2 D(E) = A(E) < B:E(E) = A(E) > A:F(E) = C(E) < C:B(E) = C(E) > D: RETURN

3  GOSUB 2
4  IF D(1) * D(2) + E(1) * E(2) + F(1) * F(2) + B(1) * B(2) <  > 0 THEN  RETURN

5 E = 1: IF D(E) + E(E) + F(E) + B(E) = 0 THEN E = 2: IF D(E) + E(E) + F(
    E) + B(E) = 0 THEN 100
6  IF D(E) THEN C(E) = C(1) + (C(2) - C(1)) * (B - A(1)) / (A(2) - A(1)):
    A(E) = B: GOTO 3
7  IF E(E) THEN C(E) = C(1) + (C(2) - C(1)) * (A - A(1)) / (A(2) - A(1)):
    A(E) = A: GOTO 3
8  IF F(E) THEN A(E) = A(1) + (A(2) - A(1)) * (C - C(1)) / (C(2) - C(1)):
    C(E) = C: GOTO 3
9  IF B(E) THEN A(E) = A(1) + (A(2) - A(1)) * (D - C(1)) / (C(2) - C(1)):
    C(E) = D: GOTO 3
10 E% = 0:NF% = NF% + 1: POKE DT,NF%: RETURN
20 E% = E% + 1: ON E% > 255 GOSUB 10: POKE DT + NF%,E%:SD% = SD% + 1:H% =
    X% / 256:L% = X% - H% * 256: POKE SD%,L%:SD% = SD% + 1: POKE SD%,H% +
    IS%:SD% = SD% + 1: POKE SD%,Y%: RETURN
40 SD% = SD% + 1: POKE SD%,RZ:SD% = SD% + 1: POKE SD%,SL:SD% = SD% + 1: POKE
    SD%,N: RETURN
100 IS% = 0:X% = A(1):Y% = C(1): HPLOT X%,Y%: GOSUB 20:IS% = 2:X% = A(2):
    Y% = C(2): HPLOT  TO X%,Y%: GOSUB 20: RETURN
200 A(2) = X:C(2) = Y: GOSUB 1:A(1) = X:C(1) = Y: RETURN
300 A(2) = X:C(2) = YO: GOSUB 1:A(1) = X:C(1) = YO: RETURN
1010 X =  COS (AN) * R + XO:Y =  SIN (AN) * R + YO: RETURN
1020 XP = XO:YP = YO - 120:X = XO + 60:Y = YO: RETURN
1030  FOR AN = 0 TO PI STEP .1:A(1) = XP:C(1) = YP: GOSUB 300: GOSUB 1010
    : GOSUB 300: NEXT :A(2) = XP:C(2) = YP: GOSUB 1: RETURN
1040 A = 140:D = 95: GOSUB 1050:A = 279:B = 140: GOSUB 1050:C = 95:D = 19
    1: GOSUB 1050:A = 140:B = 0: GOSUB 1050: END
1050  HGR2 : HCOLOR= 3: GOSUB 10
1060 XO = 230:YO = 100:AN = PI / 4: GOSUB 1010
1070  FOR AN = AN + .1 TO 7 / 4 * PI STEP .1
1080 A(1) = XO:C(1) = YO: GOSUB 200
1090  GOSUB 1010
1100  GOSUB 200
1110  NEXT
1120 A(2) = XO:C(2) = YO: GOSUB 1
1130 XO = XO - 100:YO = 160: GOSUB 1020: GOSUB 1030
1140 XO = XO - 60: GOSUB 1020: GOSUB 1030
1150  RETURN
10000 A = 279:B = 0:C = 0:D = 191:DT = 24576:SD% = DT + 100:PI = 3.1415:R
    = 60: GOTO 1040
20000 :
20001 :
20002 :
20003  REM  PROGRAMMA DUE
```

Programma 2

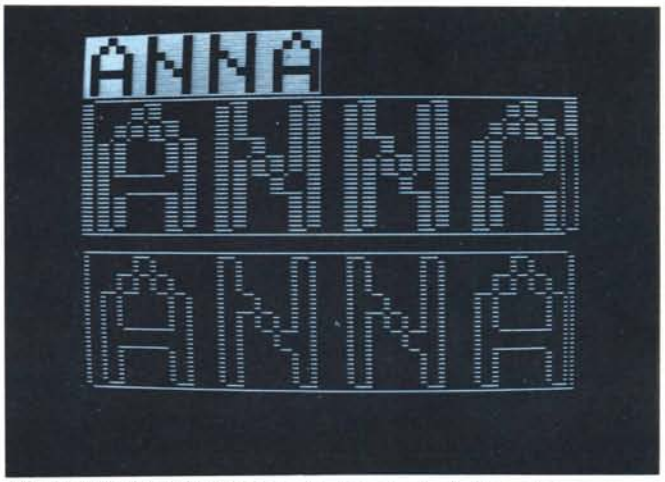


Il marchio della rivista, realizzato tramite semplici formule matematiche, viene codificato in SLIDE.

Un disegno alla M.C. ESCHER ottenuto con la ripetizione di una sola SHAPE.



Il LOGO di un programma per gli accoppiamenti di un torneo di scacchi. Sul video, il "pezzo" al centro ruota su se stesso. È da notare la scacchiera truccata.



Effettuando l'XDRAW d'una shape a matrice e spostando di poco la sua origine, è possibile creare un effetto dinamico di tipo "Broadway" sulle scritte.

Il "disegno misterioso"

Come in un famoso gioco, il programma numero due taglia il disegno in quattro pezzi e fa vedere solo la finestra di turno. Al termine, si possono registrare su disco le SLIDE corrispondenti con "BSAVE SEQ-MC/4,A$6000,L$821" per poter rivedere altre volte la sequenza, utilizzando le subroutine ANNA che abbiamo visto all'inizio.

Un esercizio che possiamo consigliarvi è quello di cercare di ottenere, con gli opportuni cambiamenti della riga 1040, altri tipi di visualizzazione del disegno, come per esempio un restringimento progressivo dell'immagine intorno ad un punto. Le prossime volte vedremo come sia possibile comandare una singola SLIDE e come sommarle le une sulle altre. Affronteremo, poi, l'interessante quanto affascinante argomento dell'animazione tridimensionale, che ci permetterà di far ruotare degli oggetti sullo schermo. Spiegheremo come si possano "agganciare" le varie routine in linguaggio macchina al BASIC Applesoft, con l'aiuto di un interprete appositamente realizzato.

```
0 A = 279:B = 0:C = 0:D = 191: GOTO
    10
1 E = 1: GOSUB 2:E = 2: GOSUB 2: GOSUB
    4: RETURN
2 D(E) = A(E) < B:E(E) = A(E) > A
    :F(E) = C(E) < C:B(E) = C(E)
    > D: RETURN
3  GOSUB 2
4  IF D(1) * D(2) + E(1) * E(2) +
    F(1) * F(2) + B(1) * B(2) <
    > 0 THEN  RETURN
5 E = 1: IF D(E) + E(E) + F(E) +
    B(E) = 0 THEN E = 2: IF D(E)
     + E(E) + F(E) + B(E) = 0 THEN
     HPLOT A(1),C(1) TO A(2),C(2
    ): RETURN
6  IF D(E) THEN C(E) = C(1) + (C(
    2) - C(1)) * (B - A(1)) / (A
    (2) - A(1)):A(E) = B: GOTO 3

7  IF E(E) THEN C(E) = C(1) + (C(
    2) - C(1)) * (A - A(1)) / (A
    (2) - A(1)):A(E) = A: GOTO 3

8  IF F(E) THEN A(E) = A(1) + (A(
    2) - A(1)) * (C - C(1)) / (C
    (2) - C(1)):C(E) = C: GOTO 3

9  IF B(E) THEN A(E) = A(1) + (A(
    2) - A(1)) * (D - C(1)) / (C
    (2) - C(1)):C(E) = D: GOTO 3

10  HGR : HCOLOR= 3
20  INPUT "COORD. (X1,Y1,X2,Y2) =
     ";A(1),C(1),A(2),C(2)
30  GOSUB 1
40  GOTO 20
41 :
42 :
43 :
44 :
45 :
50  REM Routine di clipping
51 :
52  REM ROBERTO ANGELETTI 1984
53  REM for MC-microcomputer
54 :
```

Routine di clipping.

```
0  GOTO 200
10 E% = 0:NF% = NF% + 1: POKE DT,
    NF%: RETURN
20 E% = E% + 1: ON E% > 255 GOSUB
    10: POKE DT + NF%,E%:SD% = S
    D% + 1:H% = X% / 256:L% = X%
     - H% * 256: POKE SD%,L%:SD%
    = SD% + 1: POKE SD%,H% + IS
    %:SD% = SD% + 1: POKE SD%,Y%
    : RETURN
40 SD% = SD% + 1: POKE SD%,RZ:SD%
    = SD% + 1: POKE SD%,SL:SD% =
    SD% + 1: POKE SD%,N: RETURN

50 :
60 :
70 :
80 :
90 :
110 :
120  REM  Routine di coding
130 :
140  REM  ROBERTO ANGELETTI 1984
150  REM  for MC-microcomputer
160 :
170 :
180 :
190 :
200 DT = 24576:::: REM   $6000
300 SD% = DT + 100: REM   $6065
```

Routine di coding.

Bibliografia

F. Petroni "Disegnare entro i margini", MCmicrocomputer n. 18
A. Da Messina "Un package grafico", M&P computer n. 27
M. Waite "Computer graphics primer", Howard W. Sams & Co. Inc.
V. Di Dio "Hi Speed HGR Clear", MC-microcomputer n. 23

Uso inusuale di "ON... GOTO-SUB"

Un comando del Basic dalle innumerevoli possibilità e che, invece, è conosciuto in maniera limitata è l'ON...GOTO....

La sua sintassi è chiara e viene liquidata in poche righe sui vari manuali: ON V1 GOTO 100,200,300 esegue un salto a 100 nel caso che V1 valga 1, un salto a 200 nel caso valga 2, a 300 se 3. Insomma; può essere considerato come il raggruppamento di vari IF...THEN:

```
IF V1 = 1 THEN 100
IF V1 = 2 THEN 200
IF V1 = 3 THEN 300
```

La differenza principale tra le due soluzioni, oltre all'ovvio risparmio di memoria, sta nel fatto che l'ON può stare su una sola riga, cosa impraticabile per la sequenza di IF.

Una cosa importante da notare è che con:

```
1000 IF X > 12 THEN GOSUB 100 : X = 45 : GOSUB 300
```

nel caso che la condizione non si verifichi, la seconda e la terza parte della riga non vengono eseguite. Invece, usando:

```
1000 ON X >12 GOSUB 100 : X = 45 : GOSUB 300
```

la sostituzione di X ed il salto a 300 vengono comunque eseguiti, rimanendo invariato il funzionamento della prima parte. La sintassi ora vista è perfettamente leggittima e non comporta rischi di nessun genere, sempreché le "intenzioni" siano ben espresse. Un oculato ed attento uso dell'ON può, oltre che compattare il programma, supplire a lacune del Basic, quali il CASE e l'ELSE. Ad esempio:

```
1000 ON X = 12 GOTO 100 : X = 45 : GOSUB 300
```

esegue solo la sostituzione di X ed un salto a 300 nel caso che X sia comunque diverso da 12. Nel caso che X sia uguale a 12 si esegue il salto a 100; inoltre, se al termine della riga 300 si esegue un salto incondizionato a quella seguente alla 1000, abbiamo realizzato il meccanismo completo di "altrimenti".