di Francesco Petroni

Ricominciamo da capo Elementi fondamentali

Continuiamo la serie di articoli intitolati 'ricominciamo da capo' trattando altri problemi fondamentali in Computer Grafica destinati principalmente ai principianti.

Un primo gruppo di problemi, oltre ad essere propedeutico a quasi tutti i temi di Computer Grafica, è anche elementare al punto che gli esempi che pubblichiamo a corredo dell'articolo, sono dei programmi lunghi una sola riga. Questo primo gruppo è costituito da:

— programmi che si basano su uno/due loop, oppure non si basano su loop;

— programmi in cui occorre stabilire un punto iniziale, in pratica il punto di partenza del disegno che costituisce l'output, oppure pragrammi in cui il punto di partenza non è identificabile;

— programmi grafici che contengono condizioni logiche (salti condizionati o espressioni logico-matematiche).

Vedremo come risolvere, se è possibile farlo, il problema della eliminazione dei salti.

Analizzeremo inoltre, nel corso dell'articolo, un'altra problematica classica della tecnica grafica e di conseguenza della Computer Grafica, la scelta della Scala di Rappresentazione di un output.

La possibilità, offerta dal Basic interprete di quasi tutti i microcomputer, di poter scrivere più istruzioni sulla stessa riga trova in genere due limiti: il limite fisico della lunghezza massima della riga (al massimo 254 caratteri) e un limite logico dovuto al fatto che non è possibile utilizzare istruzioni di salto dirette all'interno di una riga.

Ciascun programmatore ha un suo stile di lavoro. C'è chi cerca di compattare il programma al massimo, evitando REM, utilizzando tabelle, inzeppando le righe di istruzioni con l'obiettivo di risparmiare memoria e quindi di velocizzare l'esecuzione, ma raggiungendo anche lo scopo di rendere il listato poco 'leggibile' agli altri e a se stesso.

C'è, dal lato opposto, chi diluisce il listato al massimo, inserendo REM che commentano non solo ogni subroutine, ma ogni singola istruzione, o che addirittura hanno scopo decorativo. Il risultato è quello di avere listati chilometrici dove però le righe interessanti sono pochissime.

L'atteggiamento intermedio è, come al solito, il più logico e consiste nel cercare il giusto equilibrio fra l'economia di memoria, che è una delle poche regole fondamentali del programmatore, e la leggibilità dei listati. E leggibilità significa facile individuazione delle varie routine e comprensione del loro svolgimento.

L'idea di fare programmi lunghi una sola riga non aggiunge nulla a quanto detto, ma dati gli illustri precedenti (vedi articolo di B.A. sul numero 3 di MCmicrocomputer) mi sento stimolato a provarci finalizzando questi tentativi a programmi di argomento grafico.

Vediamo quindi di elencare e descrivere le varie fasi di un programma generico (grafico o no) e di specificare quali siano le problematiche connesse alle varie fasi, in funzione del voler realizzare un programma di una sola riga e del fatto che tale programma deve essere grafico.

Un qualsiasi programma ha una fase iniziale, nella quale vanno specificate, una volta per tutte, le costanti e definiti i valori delle variabili. C'è poi la fase centrale (che definiremo MAIN Program) in cui vengono svolte le elaborazioni principali. C'è poi la fase finale, che può comprendere l'esposizione del risultato, oppure semplicemente la fine dell'eleborazione, che avviene a causa del verificarsi di una data condizione. Sono considerazioni molto ovvie, ma che è bene aver chiare quando si programma.

Anche un programma di disegno, per quanto semplice, ha questa struttura. Se si disegna una circonferenza, ad esempio, occorre predefinire centro e raggio, ma poi occorre stabilire il punto dal quale partire, il "senso di marcia" e definire la condizione di fine, corrispondente al tracciamento di un intero angolo giro.

Se il Basic possiede l'istruzione CIRCLE il problema per il programmatore non esiste, in quanto la soluzione è insita nell'istruzione CIRCLE. Ma in generale occorre stabilire le varie fasi del disegno (inizio, svolgimento, fine) così come quando usiamo la matita cominciamo da un punto e finiamo con un altro punto.

Anche le istruzioni tipiche del Basic grafico seguono questa logica.

Vi sono istruzioni di inizio linea:

PSET (X,Y) per il Basic IBM/Olivetti
HPLOT (X,Y) per l'Applesoft.

Istruzioni di continuazione o di fine linea:

LINE -(X,Y) e HPLOT -(X,Y)

Istruzioni che iniziano e finiscono una linea:

LINE (X,Y)-(X1,Y1) e HPLOT (X,Y)-(X1,Y1)

Nel programma occorrerà individuare una o più condizioni di inizio e una o più condizioni di fine, e conseguentemente occorrerà utilizzare questa o quella istruzione.

In pratica però la scelta di una soluzione rispetto alle altre rimane compito di chi fa il programma.

Per fare un esempio supponiamo di dover tracciare la traiettoria di un oggetto che si muova con una certa legge matematica. Abbiamo almeno tre soluzioni.

Innanzitutto dobbiamo calcolare varie posizioni assunte dall'oggetto e cioé quella iniziale (corrispondente al momento del lancio), quella finale (corrispondente al raggiungimento del bersaglio o all'uscita dal formato video) e un certo numero di posizioni intermedie. Il numero di calcoli intermedi dipende dalla precisione, che vogliamo nel tracciato con l'ovvia considerazione che una maggior precisione costa maggiori tempi di elaborazione.

Le tre soluzioni sono:

— precalcolo delle posizioni (in formato video), loro memorizzazione in una tabella e loro successiva visualizzazione. Il risultato consiste nel disegnare una serie di segmentini, ciscuno dei quali ha in comune con il precedente e il successivo il punto iniziale e il punto finale. La condizione di inizio è individuata dal primo elemento della tabella e quella di fine dall'ultimo.

— Calcolo e visualizzazione immediata di ciascuna posizione. In tal caso occorre individuare il punto iniziale (ad es. se la funzione è legata al tempo occorre individuare la posizione al punto zero) e il punto finale (ad es. quando le coordinate correnti corrispondono a quelle del bersaglio). L'inconveniente è che se le due precedenti condizioni sono individuate a mezzo istruzione IF, occorre fare i due test anche per tutte le posizioni intermedie, con grave rallentamento dell'esecuzione. Il test per il punto iniziale si può evitare facilmente isolandone il calcolo e la visualizzazione.

— Calcolo di due posizioni successive e visualizzazione del segmentino che le unisce. Questo metodo ha il vantaggio di utilizzare una sola istruzione grafica e cioé DA... A... Lo svantaggio consiste nel fatto che ciascun punto deve essere calcolato due volte oppure i valori calcolati dovendo

essere usati due volte debbono essere passati da una variabile (quella di fine segmento) a un'altra (quella di inizio segmento).

Va poi considerata la differente logica delle due condizioni iniziale e finale. Mentre la prima è sicuramente identificabile e quindi isolabile, la seconda si può presentare in varie forme, ad es.:

— Programma che non finisce mai. Quando, volutamente o meno, si creano delle iterazioni interrompibili solo con un Break.

— Programma che finisce in seguito al verificarsi di una condizione. Ovvero all'interno della elaborazione principale esistono uno o più test logici per verificare il raggiungimento di una condizione finale (es. il raggiungimento del bersaglio).

```
print 3>3,3=3,3<3
 0             -1            0
print 3*(3=3),3*(3=3+1)
-3             0
print 3*((3=3)+1),3*(3=(3+1))
 0             0
a=4
b=5
print a=a,a=a=a
-1             0
print a>(a=a),(a>a)=a
-1             0
print b*(b>a)+a*(b<a)
-5
```

Figura 1 - Esercizio sugli Operatori Logici. È un hard copy di alcune operazioni eseguite in IBM Basic usato in maniera diretta (cioè senza programma).

— Programma che finisce con la fine dell'esecuzione del Main (es. fine di un loop).

Ciascuna di queste forme ha un suo campo di applicabilità.

In definitiva si può dire che non esiste una regola di comportamento migliore delle altre e quindi la scelta della soluzione da adottare tra le varie possibili è affidata alla sensibilità del programmatore.

Istruzioni di salto e di salto condizionato

Le istruzioni di confronto e quelle analoghe di salto condizionato sono di importanza fondamentale nella programmazione in qualsiasi linguaggio e su qualsiasi argomento, a tal punto che non esisterebbero né programmi né probabilmente computer se non si potesse eseguire queste funzioni.

E come tutte le istruzioni, permettono sia un uso standard, quello citato nei manuali e che deve essere ben noto a chi programma, sia un uso "avanzato" che sfrutta al massimo anche le possibilità "nascoste" dell'istruzione, e che quindi consente soluzioni brillanti per problemi complessi. Questo vale anche per i programmi grafici dove, come abbiamo visto, esistono sempre problemi di individuazione e trattamento di condizioni particolari.

Le istruzioni sono:

IF >condizione< THEN >istruzioni< A<ELSE>istruzioni B<ON>condiz. numerica<GOTO/GOSUB>riga di programma <WHILE>condizione numerica< >Istruzioni<WEND ecc.

Quando la condizione è riducibile a una semplice espressione logica (es. M > 10 oppure A$="FINE") si possono usare espressioni matematiche contenenti espressioni logiche.

Infatti un confronto del tipo citato produce un risultato numerico (1 se la condizione è vera -1 (oppure 0 per altri computer) se la condizione è falsa) e tale risultato può essere inserito come elemento di una espressione matematica complessa.

In figura 1 vediamo l'esecuzione di una serie di esperimenti su tale problematica. Data la semplicità con la quale si possono eseguire esercizi usando il Basic in modo diretto, è opportuno oltre allo studio dei manuali, da ritenere "obbligatorio", anche molto esercizio, specialmente su questi argomenti che vanno assolutamente padroneggiati da chi programma.

Programma dodici settori

La modalità SCREEN 3 disponibile nel GWBASIC dell'Olivetti M24 offre uno schermo di lavoro di ben 256.000 pixel. E poiché questa modalità è presente anche sul monitor B&N si può dire che questa,

```
100 K=K+S:PRINT TAB(K)"■":IF K>10 THEN S=-1:GOTO 100:ELSE IF K<1 THEN S=1:GOTO 1
00: ELSE GOTO 100
101 REM
199 REM
200 K=K+S:PRINT TAB(K);:IF K>10 THEN S=-1:PRINT " ":GOTO 200:ELSE IF K<1.1 THEN
S=1:PRINT "<":GOTO 200: ELSE IF S=-1 THEN PRINT "/":GOTO 200:ELSE PRINT "\":GOTO
 200
201 REM
299 REM
300 CLS:S=1:Y=1:FOR X=1 TO 79:Y=Y+S:T=2*(Y=1)+1+(Y>1 AND Y<24)+(Y=24):S=S-T:LOCA
TE Y,X:PRINT "#";:NEXT X
```

Figura 2 - Tre programmi Monoriga Alfanumerici su IBM. La sequenza di caratteri in output viene gestita dai vari statement inseriti in ciascun programma.

```
100  PRINT  CHR$ (7)  REM   GRIGLIA
110  HGR2   HCOLOR= 3 P = 9 A = 0 B = 279 C = 0 D = 191 S = B / D  FOR
     I = A TO B STEP P  HPLOT I,C TO I,D  HPLOT A,I / S TO B,I / S  NEXT
     I
200  PRINT  CHR$ (7)  REM   RAGGIERA
210  HGR2   HCOLOR= 3 P = 9 A = 0 B = 279 C = 0 D = 191 S = B / D  FOR
     I = A TO B STEP P  HPLOT I,C TO B - I,D  HPLOT A,I / S TO B,D -
     I / S  NEXT I
300  PRINT  CHR$ (7)  REM   VOLUTA
310  HGR2   HCOLOR= 3 P = 9 A = 0 B = 279 C = 0 D = 191 S = B / D  FOR
     I = A TO B STEP P  HPLOT I,I / S TO B - I,D  HPLOT B - I,I / S TO
     B,D - I / S  NEXT I
400  PRINT  CHR$ (7)  REM   SPIRALE
410  HGR2   HCOLOR= 3 P = 3.14159  HPLOT 140,96  FOR A = 0 TO 8 * P STEP
     P / 15 R = R + 1 X = 140.5 + R *  COS (A) Y = 96.5 +  7 * R *  SIN
     (A)  HPLOT  TO X,Y  NEXT A
500  PRINT  CHR$ (7)  REM  DIAMANTE
510  HGR2   HCOLOR= 3 P = 3.14159 R = 129  FOR A = 0 TO 2 * P STEP P
     / 15 X = 140.5 + R *  COS (A) Y = 96.5 +  684 * R *  SIN (A)  HPLOT
     0,0 TO X,Y  HPLOT  TO 279,191  HPLOT 279,0 TO X,Y  HPLOT  TO 0,
     191  NEXT A
600  PRINT  CHR$ (7)  REM  TRATTEGGIO
610  HGR2   HCOLOR= 3 P = 9 B = 279 D = 191 S = B / D  HPLOT 0,0 TO
     B,0 TO B,D TO 0,D TO 0,0  FOR I = 0 TO 279 L =  INT (I / 31) *
     31  HPLOT L,D - I / S  NEXT I
700  PRINT  CHR$ (7)  REM   SCALETTA
710  HGR2   HCOLOR= 3 P = 9 B = 279 D = 191 S = B / D  HPLOT 0,0 TO
     B,0 TO B,D TO 0,D TO 0,0  HPLOT 0,D  FOR I = 0 TO 279 L =  INT
     (I / 31) * 31  HPLOT  TO L,D - I / S  NEXT I
800  PRINT  CHR$ (7)  REM  POLIGONI
810  HGR2   HCOLOR= 3 P = 3.1416 S = 2 * P -  001  FOR L = 3 TO 8 R =
     L * 11  HPLOT 140.5 + R,95.5  FOR A = 0 TO 2 * P STEP S / L X =
     140.5 + R *  COS (A) Y = 95.5 + R *  SIN (A)  HPLOT  TO X,Y  NEXT
     A  NEXT L
900  PRINT  CHR$ (7)  REM  CICLOIDE
910  HGR2   HCOLOR= 3 P = 3.1416 S = P / 20 R = 2.111  HPLOT 170,95
      FOR A = 0 TO 18 * P STEP S:K = R * A X = 140.5 + 57 *  COS (A)
      - 27 *  COS (K) Y = 95.5 + 57 *  SIN (A) - 27 *  SIN (K)  HPLOT
      TO X,Y  NEXT A
1000 PRINT  CHR$ (7)  REM  MERLETTO
1010 HGR2   HCOLOR= 3  DIM X(50),Y(50) P = 6.283 / 19  FOR I = 1 TO
     19 A = P * I X(I) = 130 *  COS (A) + 140 Y(I) = 90 *  SIN (A) +
     96  NEXT I  FOR L = 1 TO 18  FOR I = L TO 19  HPLOT X(I),Y(I) TO
     X(L),Y(L)  NEXT I,L
```

Figura 3 - Dieci programmi Monoriga Grafici su Apple. Alcuni programmi sono una versione "spartana" dei vari programmi apparsi più volte in questa rubrica.

insieme alla velocità dell'8086, doppia rispetto a quella dell'8088, sono due caratteristiche con le quali si supera un bel po' lo standard IBM.

Il programma, listato in figura 7 e il cui output è riportato in figura 8, divide lo schermo in 12 settori in ciascuno dei quali disegna una certa curva bidimensionale, che viene anche descritta con un titolo.

Poiché tutte le curve sono del tipo Y = FUN Z(X), è stato possibile inserire ciascuna curva in una sua riga, nella quale è stata inserita anche la scritta sotto forma di DATA.

L'esecuzione vera e propria viene affidata ad un loop sulla X, eseguito dodici volte per ognuna delle quali viene richiamata la routine della relativa funzione.

Non è un programma in una sola riga, ma viene utilizzato un metodo drastico di riduzione spazio, mettendo in evidenza (come direbbe un matematico) tutte le fasi comuni ai dodici segni. Inoltre viene ottenuta la leggibilità del listato in quanto, come detto, ciascuna funzione ha una sua riga e la sua descrizione.

In realtà la vera caratteristica del disegno è rappresentata dall'uso delle istruzioni GET e PUT, con le quali viene realizzata una "animazione" in fase di costruzione del disegno, animazione ovviamente non visibile in una foto. Infatti ogni singolo settore viene disegnato nel quadrante in alto a sinistra, poi viene immagazzinato in un ARRAY, con la istruzione GET (X,Y) - (X1,Y1),D%, e poi viene spostato nel quadrante di destinazione, tramite la istruzione, complementare alla precedente PUT (X,Y),D%.

Con questa modalità, che tra l'altro ha un effetto "spettacolare" si semplifica il problema del riferimento in quanto ciascun disegno viene realizzato nel quadrante, di 156 per 128 pixel in alto a sinistra, senza dover effettuare costose operazioni di scaling.

Un'ultima notazione riguarda l'uso della Funzione VIEW di riga 140. Con questa funzione si "isola" il quadrante e le successive operazioni grafiche riguardano solo l'interno della VIEW. In questo modo viene pulito il quadrante prima di un nuovo disegno. Il successivo VIEW, di riga 220, serve per "resettare" la finestra e per permettere quindi l'animazione.

Programmi di una sola riga

I programmi lunghi una sola riga sono 16. I primi tre realizzati in Basic-A IBM con uscita sul video monocromatico, facilmente adattabili a tutte le altre situazioni (listati contemporaneamente in fig.2). Poi ce ne sono 10 in Applesoft, con uscita su pagina grafica HGR2, listati in figura 3. Infine gli ultimi 3 sono realizzati in GWBASIC con Olivetti M24, che permette in modalità SCREEN 3, una definizione di 640 per 400 pixel. I listati sono in figura 4. Di questi ultimi programmi sono riportati due output in figure 5 e 6.

```
100 SCREEN 3:CLS:LINE (0,0)-(639,399),,B:X=10:Y=10:PSET (X,Y):FOR X=10 TO 630:Y=
X-310*((X<320)+1):LINE-(X,Y):NEXT X
101 REM
199 REM
200 SCREEN 3:CLS:LINE (0,0)-(639,399),,B:X=10:Y=10:PSET (X,Y):FOR X=10 TO 630:X1
=INT(X/40):Y=X-(X1*40-10)*((X<(X1*40))+1):LINE-(X,Y):NEXT X
201 REM
299 REM
300 SCREEN 3:CLS:LINE (0,0)-(639,399),,B:FOR K=20 TO 320 STEP 50:X=10:Y=10+K:PSE
T (X,Y):FOR X=10 TO 630:X1=INT(X/40):Y=K+X-(X1*40-10)*((X<(X1*40))+1):LINE-(X,Y)
:NEXT X:NEXT K
```

Figura 4 - Tre programmi Grafici Monoriga su Olivetti M24. Mischiando i concetti di loop, operatore logico, ecc., si ottengono risultati anche complessi, pur nel limite monoriga.

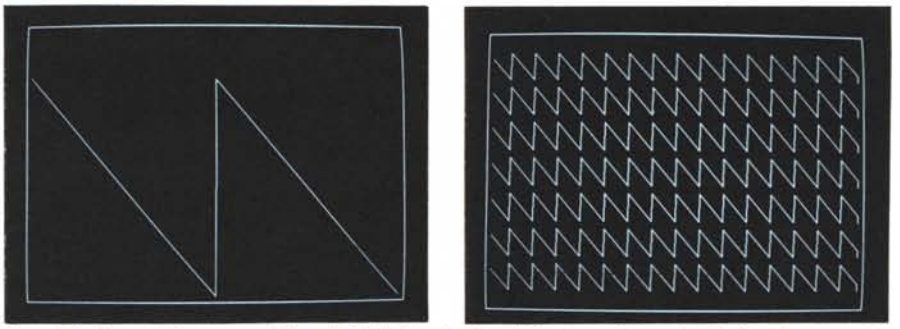


Figure 5/6 - Output del programma Olivetti n. 3. L'effetto "seghetta" è realizzato inserendo nella formula matematica un fattore logico, che come visto nel testo, può assumere solo due valori.

I primi tre hanno una uscita alfanumerica. In quello di riga 100 il carattere "■" viene printato a zig zag sul video. L'inversione del senso è realizzata invertendo il segno della variabile S, che rappresenta l'incremento della posizione orizzontale del cursore. L'inversione è gestita tramite due IF THEN ELSE nidificati.

Il programma di riga 200 è una evoluzione del precedente. In pratica viene riconosciuto il senso corrente o il momento della inversione, e di conseguenza viene stampato un differente carattere. In questo caso la nidificazione degli IF/THEN/ELSE è più complessa.

L'ultimo programma della prima serie (riga 300) utilizza invece espressioni matematiche che contengono fattori logici. In pratica viene stampato il carattere "#". C'è un loop sulla X, che rappresenta la posizione orizzontale del cursore. La posizione verticale Y è calcolata con una espressione che comunque la riporta entro il formato video.

Il secondo gruppo di programmi, quelli in Applesoft, sono grafici e quindi realizzano ciascuno un disegno descritto nel titolo del programma stesso. Senza commentarli uno per uno diremo che si basano tutti su uno o due loop, con i quali vengono fatti

```
100 REM dodici curve sul piano
110 REM inizializzazione generale
120 P=3.14159:DIM D%(2800):SCREEN 3:CLS
130 FOR V=2 TO 0 STEP -1:FOR O=3 TO 0 STEP -1:J=(V*4+O+1):K=13-J
140 VIEW (0,0)-(156,128):CLS
150 O1=O*160:O2=O1+156:V1=V*132:V2=V1+128:O3=78:V3=64
160 READ K$:LINE (0,0)-(156,128),,B:LINE (1,1)-(155,127),3,,F
170 LINE (0,64)-(156,64):LINE(78,0)-(78,128):LOCATE 8,2:PRINT J;" "K$;
180 FOR X=-75 TO 75 STEP 2
190 ON K GOSUB 300,310,320,330,340,350,360,370,380,390,400,410
200 X1=X+O3:Y1=Y+V3:IF X=-75 THEN GOSUB 250 :ELSE GOSUB 260
210 NEXT X:IF K= 12 THEN PRINT CHR$(7):GOTO 240
220 VIEW (0,0)-(639,399):GET (0,0)-(156,128),D%:PUT (O1,V1),D%
230 PRINT CHR$(7):NEXT O:NEXT V
240 I$=INKEY$:IF I$="" THEN 240 ELSE SCREEN 0:CLS:END
250 PSET (X1,Y1):RETURN
260 LINE -(X1,Y1):RETURN
300 Y=50*SIN(X/10)/(X/10+.0001):RETURN:DATA "Y=SIN(X)/X"
310 Y=50/X:RETURN:DATA                      "IPERBOLE"
320 Y=.2*X^2-X+2-30:RETURN:DATA             "PARABOLA 2"
330 Y=45*SIN(X/30):RETURN:DATA              "SINUSOIDE"
340 Y=.6*X+12:RETURN:DATA                   "RETTA"
350 Y=20*SIN(X/20)-30*COS(X/24):RETURN:DATA "TRIGON. 2"
360 Y=10*LOG(ABS(X/80)):RETURN:DATA         "LOGARITMO"
370 Y=20*SIN(X/2)-16*COS(X/3):RETURN:DATA   "TRIGON. 1"
380 Y=10*LOG((X+76)/10):RETURN:DATA         "LOGARITMO"
390 Y=20/SIN(X/10):RETURN:DATA              "SIN-INVERSA"
400 Y=(ABS(X/8))^1.2:RETURN:DATA            "ESPONENZIALE"
410 Y=.1*(X/10)^3-.1*(X/10)^2+2:RETURN:DATA "PARABOLA 3"
```

Figura 7 - Programma grafico in 12 settori su Olivetti. La definizione disponibile (in modo SCREEN 3) è di ben 640 per 400 pixel, e la velocità è quella dell'8086. E queste sono condizioni che si sentono.

```
100 CLS:REM operazioni manuali di scaling
110 REM x0,y0 vertice in alto a sinistra finestra video
120 REM x1,y1 vertice in basso a destra  finestra video
130 REM X8,Y8 vertice in alto a sinistra del soggetto
140 REM X9,Y9 vertice in basso a destra  del soggetto
150 INPUT "Immetti x0,y0  ",X0,Y0
160 INPUT "Immetti x1,y1  ",X1,Y1:PRINT
170 INPUT "Immetti x8,y8  ",X8,Y8
180 INPUT "Immetti x9,y9  ",X9,Y9:PRINT
190 VX=X1-X0:VY=Y1-Y0
200 SX=X9-X8:SY=Y9-Y8
210 FX=VX/SX:FY=VY/SY
220 TX=X0-X8*FX:TY=Y0-Y8*FY
230 PRINT "Form. Video      ";VX;" in Orizz. ";VY;" in Vert."
240 PRINT "Form. Soggetto   ";SX;" in Orizz. ";SY;" in Vert."
250 PRINT "Fatt. Scala      ";FX;" in Orizz. ";FY;" in Vert."
260 PRINT "Fatt. Traslaz.   ";TX;" in Orizz. ";TY;" in Vert."
270 DEF FNXX(X)=X*FX+TX:DEF FNYY(Y)=Y*FY+TX
280 PRINT:INPUT "Punti Soggetto    ",X,Y
290 PRINT "Punti Video    ",FNXX(X),FNYY(Y):GOTO 280
```

```
Immetti x0,y0  100,100
Immetti x1,y1  200,200

Immetti x8,y8  -1,-1
Immetti x9,y9  1,1

Form. Video      100  in Orizz.  100  in Vert.
Form. Soggetto   2  in Orizz.  2  in Vert.
Fatt. Scala      50  in Orizz.  50  in Vert.
Fatt. Traslaz.   150  in Orizz.  150  in Vert.

Punti Soggetto   0,0
Punti Video                150            150

Punti Soggetto   -1,1
Punti Video                100            200

Punti Soggetto   1,-1
Punti Video                200            100

Punti Soggetto
```

Figura 9 - Esercitazioni di scaling manuale. Listato e output. La conoscenza dei "sistemi di riferimento" e delle funzioni di "scaling" è un prerequisito fondamentale per chi vuole realizzare programmi grafici.

variare o coordinate (orizzontali e/o verticali) oppure angoli.

Nel programma "SCALETTA" l'effetto scala è generato utilizzando la funzione INT, che permette di tenere fisso il valore Y entro un certo intervallo della X.

Il Basic dell'Apple II non contiene l'IF THEN ELSE. Quindi non è possibile in certi casi lavorare su una sola linea, in quanto la situazione ELSE va riportata alla linea successiva.

L'ultimo gruppo di programmi, scritti in GWBASIC, comprende espressioni logico matematiche abbastanza complesse con le quali si possono ottenere svariati effetti "a basso costo". Ne riportiamo due esempi, anche questa volta senza descrivere il programma nel dettaglio.

È evidente che tali programmi hanno solo uno scopo didattico in quanto praticamente non è mai necessario porsi vincoli così drastici in sede di programmazione. E inoltre lo stesso risultato si può raggiungere con routine più facili da realizzare e più leggibili.

Lo scaling

Un altro elemento base in tutti i problemi di computer grafica è lo "scaling", ovvero quel complesso di operazioni logico-matematiche che vanno eseguite sui dati relativi all'oggetto da visualizzare per tradurli in dati compatibili con i formati in uscita.

È una problematica che entra in tutte le applicazioni e che quindi deve essere ben conosciuta da chi vuol fare in proprio computer grafica senza limitarsi ad usare programmi altrui.

Ai neofiti della computer grafica suggeriamo quindi di studiare il problema e di fare su di esso esercizi per arrivare a padroneggiarlo, cosa che gli renderà più semplici i passi successivi.

Ovviamente il problema è stato più volte trattato in questa rubrica (vedi per esempio MCmicrocomputer n. 3 pag. 52).

Lo "scaling" può essere paragonato alla scelta delle apparecchiature e delle condizioni con le quali fare una fotografia di un certo soggetto. La fotografia può andare dalla microfotografia (apparecchio collegato ad un microscopio), macrofotografia (foto di soggetti molto piccoli e utilizzo di obiettivi particolari) e così via fino alla foto fatta applicando un dorso fotografico ad un telescopio.

Ma anche fotografia di qualche cosa di astratto come ad esempio di una superficie geometrica nello spazio derivante da una formula matematica, soggetto che nessuna macchina fotografica può riprendere.

I passi logici che bisogna percorrere per riprodurre un soggetto con un computer sono:

— esprimere il soggetto in termini di dati matematici; (scelta e preparazione del soggetto);

— sottoporre i dati matematici all'operazione di scaling; (scelta della lunghezza focale dell'obiettivo, per mettere in primo piano il soggetto e puntamento verso il soggetto, che non è assolutamente detto che stia davanti all'obiettivo);

— visualizzare i dati così trattati; (equivalente allo scatto).

Detto in altri termini ciascun dato relativo al soggetto del disegno, può essere traslato, ruotato, ingrandito, rimpicciolito, prima di poter essere visualizzato. E queste operazioni è bene farle subito prima della visualizzazione in quanto riguardano questa fase della procedura (strettamente dipendente dai formati in uscita) e non ad esempio la definizione dei dati del soggetto.

La suddivisione in tre fasi riguarda i tipici programmi di computer grafica. Esiste però la possibilità di evitare una delle fasi. I programmi di una sola riga pubblicati nella prima parte dell'articolo, ad esempio, riguardano dati già inizialmente compatibili con il formato in uscita in modo da rendere inutile lo scaling.

È quindi evidente che tali programmi non "girano" in computer che hanno diffe-

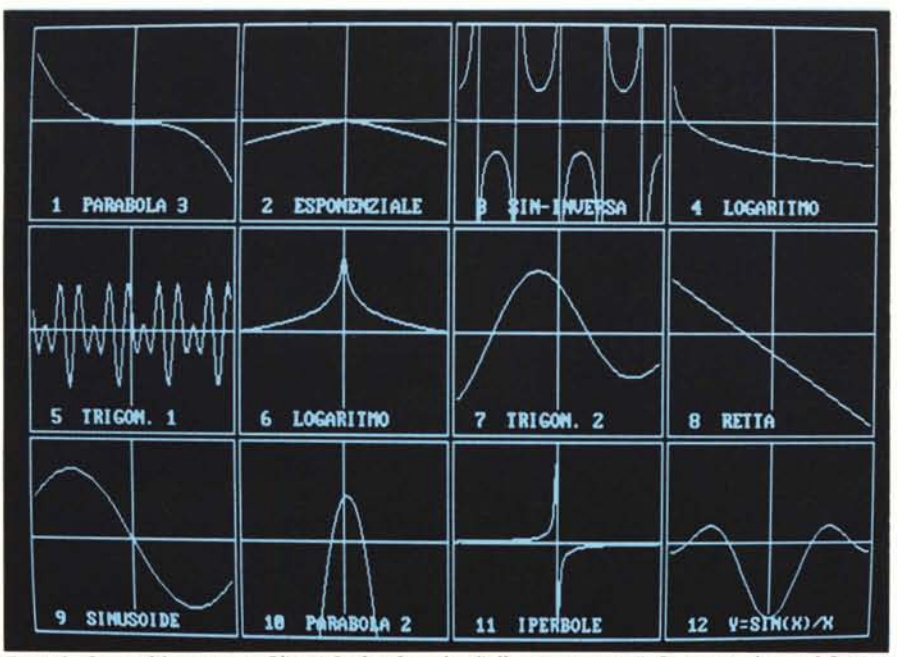
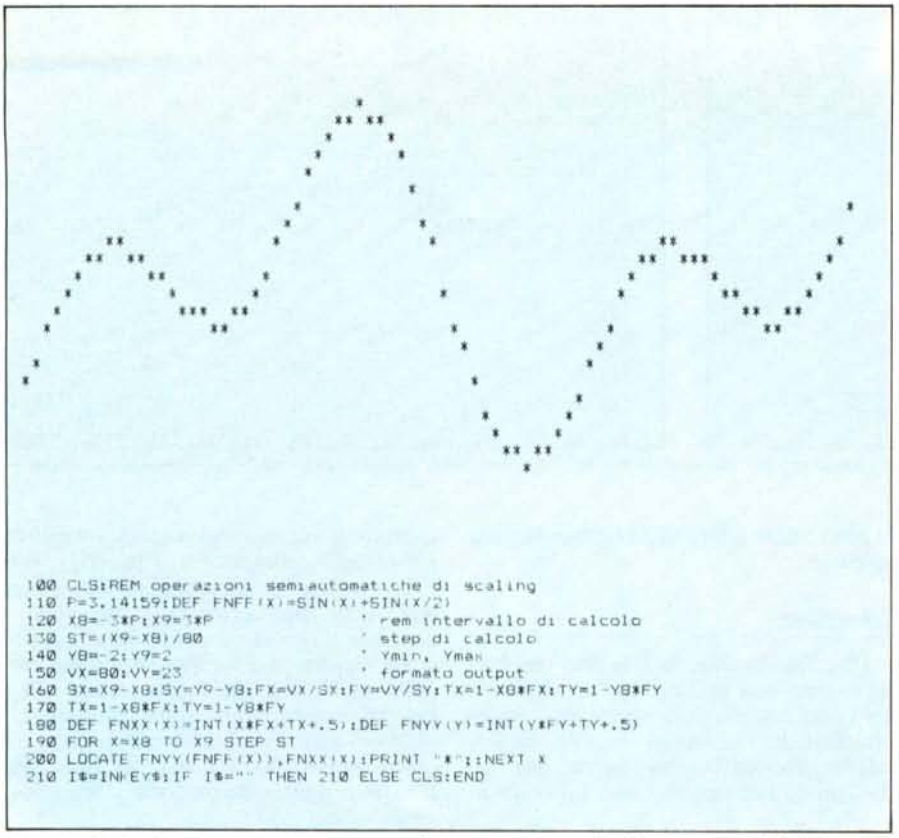


Figura 8 - Output del programma Olivetti. La foto fa perdere l'effetto "animazione". Ogni settore ha una definizione 213 per 133 pixel, più che sufficienti per disegni bidimensionali di una certa complessità.



```
100 CLS:REM operazioni semiautomatiche di scaling
110 P=3.14159:DEF FNFF(X)=SIN(X)+SIN(X/2)
120 X8=-3*P:X9=3*P                    ' rem intervallo di calcolo
130 ST=(X9-X8)/80                     ' step di calcolo
140 Y8=-2:Y9=2                        ' Ymin, Ymax
150 VX=80:VY=23                       ' formato output
160 SX=X9-X8:SY=Y9-Y8:FX=VX/SX:FY=VY/SY:TX=1-X8*FX:TY=1-Y8*FY
170 TX=1-X8*FX:TY=1-Y8*FY
180 DEF FNXX(X)=INT(X*FX+TX+.5):DEF FNYY(Y)=INT(Y*FY+TY+.5)
190 FOR X=X8 TO X9 STEP ST
200 LOCATE FNYY(FNFF(X)),FNXX(X):PRINT "*";:NEXT X
210 I$=INKEY$:IF I$="" THEN 210 ELSE CLS:END
```

Figura 10 - Scaling Semiautomatico. Il programma utilizza come output un video alfanumerico standard. Il formato in uscita è quindi di 80 per 24 punti.

renti formati in uscita, in tal caso si fa lo scaling oppure si cambiano i dati iniziali.

Come si fa lo scaling?

Bisogna considerare tre elementi:

1 - si vuole/non si vuole fare lo scaling automatico.

Cioè vogliamo imporre dei margini esterni rispetto al soggetto oppure deve essere il programma che calcola i margini del disegno automaticamente in modo da riempire il più possibile il formato di uscita.

Ad esempio se di alcuni valori statistici vogliamo costruire un istogramma, uno scaling automatico porrà il massimo della scala di riferimento un funzione del massimo valore da visualizzare. Uno scaling manuale, viceversa, impone una scala di riferimento indipendentemente dai valori delle colonnine.

2 - Esistono/non esistono funzioni di scaling nel linguaggio.

Alcuni Basic più avanzati comprendono istruzioni che eseguono uno scaling automatico consistente nel poter imporre un certo formato di uscita dipendente dal soggetto. Ad esempio se vogliamo visualizzare una funzione in un intervallo (—1, +1) della X, entro il quale i valori Y oscillano tra -2 e 2, basta usare l'istruzione WINDOW (-1, -2) - (1, 2) e automaticamente viene eseguito lo scaling che traduce i valori interni a tali intervalli nei valori del formato massimo della macchina.

3 - Si conoscono/non si conoscono subito tutti i dati da visualizzare.

Se il soggetto è il risultato di un calcolo complesso è probabile che i dati da visualizzare non si conoscano se non dopo la elaborazione principale e poiché lo scaling si fa, come vedremo, considerando i valori massimi e minimi di tali dati occorreranno tre successive elaborazioni, la prima di predisposizione dei dati e di ricerca dei valori max, min, la seconda di definizione delle funzioni di scaling, la terza di scaling di tutti i dati calcolati precedentemente.

Le funzioni classiche di scaling sono: nel caso non vi siano rotazioni:

$X2 = X1 \times S + TX$ X2, Y2 dati in formato out
$Y2 = Y1 \times S + TY$ X1, Y1 dati in formato iniziale
S fattore di scala
TX, TY valori di traslazione

nel caso invece vi sia una rotazione (angolo tra i due riferimenti A):

$X2 = (X1 \times COS(A) + Y1 \times SIN(A)) \times S + TX$
$Y2 = (-X1 \times SIN(A) + Y1 \times COS(A)) \times S + TY$

infatti per $A = 0$ si hanno le funzioni di prima.

Gli esercizi

Il primo programma di esercitazione non produce output grafico, ma un output alfanumerico dei risultati di una operazione di scaling. Vengono chiesti dapprima il formato output (attraverso i vertici della diagonale) e poi il formato massimo del soggetto (sempre attraverso la diagonale del rettangolo). Con questi valori si calcolano i range delle X e Y rispettivamente del formato dati e del formato out (righe 190 - 200) e i valori degli spostamenti TX, TY necessari per "centrare" il disegno (210). Con tutti questi valori vengono costruite due funzioni FNXX(X), FNYY(Y) da utilizzare nella routine di traduzione (riga 270), che richiede in INPUT coppie di punti in formato dati che vengono restituiti in formato output (righe 280-290).

Il formato di uscita dipende dalla definizione, in pixel, dell'uscita grafica del computer, ma di questa uscita se ne può utilizzare anche una porzione, ad esempio se si vuole fare un disegno all'interno di una finestra. Il formato massimo dei dati è equivalente al rettangolo che ha come vertici della diagonale i punti ideali (Xmin, Ymin) e (Xmax, Ymax), ideali perché non è detto che esista un punto Xmin, Ymin da visualizzare.

L'esempio pubblicato assieme al listato (in fig. 9) chiarisce un po' di più il funzionamento dell'operazione "scaling".

Il secondo programma viene definito "scaling semiautomatico" in quanto prevede un'uscita su video alfanumerico standard (di 80 per 24 caratteri), prevede la definizione a programma del range di calcolo della funzione (da -3 pigreco a +3 pigreco) e del range della Y in output (da -2 a +2).

In pratica le funzioni di scaling, definite in riga 180, traducono i dati della funzione da visualizzare in valori compresi nel formato video. E cioè per il valore di $X = -3$ pigreco della funzione il valore corrispondente da visualizzare sarà 1, e per il valore $X = 3$ pigreco deve essere 80.

Ma mentre lo scaling sulla X è controllato, poiché è sulla X che gira il loop principale, lo scaling della Y non è predeterminabile se non dopo aver calcolato tutti i valori Y da visualizzare. Nel nostro caso poiché la funzione definita in riga 110 non può superare i valori $Y = -1.5$ e $Y = 1.5$ si può prefissare in (-2, +2) il range della Y, senza pericolo di uscire fuori.

L'esercizio eseguito sul formato video alfanumerico nulla toglie all'efficacia dell'esercitazione.

Non abbiamo trattato, ma pensiamo di farlo prossimamente, il problema della tipologia delle scale. Poiché la scala è per definizione il rapporto numerico tra le grandezze misurate sull'output e quelle reali, esistono due differenti necessità:

— occorre conservare le proporzioni (il fattore di scala S è unico nelle due direzioni e le scale sono lineari).

— Non occorre oppure non si possono conservare le proporzioni (si possono usare Sx e Sy differenti, oppure i valori X e Y non sono omogenei (è il caso degli Istogrammi), oppure trattandosi di grafici di funzioni matematiche si usano scale non lineari (es. scale logaritmiche)).

Torneremo ancora sull'argomento. MC

Con Framework avrebbero fatto cose ancora piú grandi.

Leonardo (1452-1519)

Dante (1265-1321)

Galileo (1564-1642)

Cavour (1810-1861)

Dante, per esempio, avrebbe senz'altro sfruttato la procedura "profili" e i programmi di elaborazione testi di Framework: attraverso il video del suo Personal Computer avrebbe potuto buttare giù le prime idee (usando un riquadro per ognuna). E ritoccare lo scritto fino a dare forma completa ai sui capolavori.

Pensate: senza nemmeno un errore e in pochissimo tempo!

Oppure Leonardo: avrebbe potuto utilizzare Framework per sviluppare la sua immaginazione creativa e per archiviare i soggetti dei suoi disegni.

Il suo genio, forse, sarebbe volato più in alto.

E Galileo? Avrebbe potuto usare tutta la potenza di Framework per effettuare calcoli, collegamenti e prove senza perdere mai di vista le stelle.

D'altra parte, Framework sarebbe stato utile anche a Cavour. Per analizzare gli avvenimenti, per visualizzare con grafici le mosse degli avversari e per prevedere gli effetti di una decisione.

Avrebbe potuto realizzare anche ottime statistiche.

ASHTON·TATE

I programmi della Ashton-Tate sono tradotti in Italiano e supportati dalla Editrice Italiana Software.

Editrice Italiana Software S.p.A.

Editrice Italiana Software S.p.A.
Foro Bonaparte, 48 - 20121 - Milano
Tel. (02) 877312 - 877983

E oggi, a chi serve Framework? Praticamente a tutti. Anche a voi, perché è in grado di aiutarvi a risolvere tutti i problemi. E' un fantastico programma Ashton-Tate, capace di svolgere, da solo, una grande mole di lavoro: analisi, calcoli, previsioni, testi, comunicazioni, grafica, gestione dati.....

Vi permette di ridurre i tempi delle decisioni e di aumentare i margini di sicurezza.

Framework. Un grande programma che tutti capiscono, perché parla italiano.

Framework è distribuito dall'Editrice Italiana Software.