L'evoluzione della computer-graphics
Nuove frontiere per l'interazione uomo-computer

Il software grafico

Introduzione

Scopo di un linguaggio grafico è di mettere a disposizione dell'utente delle funzioni che gli permettano di eseguire un certo numero di operazioni, come tracciare linee, cerchi, quadrati, ingrandire parti del disegno, cancellare, colorare.

Queste funzioni devono avere cioè una stretta analogia con le principali operazioni eseguite da un disegnatore. La differenza con gli usuali linguaggi di programmazione sta quindi nel fatto che l'elaborazione dipende strettamente dal tipo di periferica grafica utilizzata. Può accadere che uno stesso programma, se utilizzato con differenti periferiche, ottenga diversi risultati.

Alla base di ciò esistono dei concetti la cui comprensione è necessaria per chiunque desideri «disegnare» col computer.

In questo articolo ci proponiamo perciò di analizzare questi concetti, con lo scopo non tanto di sviscerare tecnicamente le componenti di un linguaggio grafico, ma piuttosto di dare una visione complessiva dei suoi elementi costitutivi. In questo ambito un po' di formalismo matematico è necessario.

Vedremo poi negli articoli successivi alcuni esempi di linguaggi grafici, in modo da poterne valutare differenze e analogie.

Il primo concetto da affrontare è il Sistema di Riferimento. Quando noi vogliamo rappresentare un oggetto su di un foglio di carta, rispondiamo implicitamente ad una serie di domande, come per esempio «in che scala raffigurare l'oggetto»?, oppure «in che punto del foglio disegnarlo?». Questi quesiti rimandano al concetto di Sistema di Riferimento. Esso ci permette di scegliere dove (sulla superficie di visualizzazione) raffigurare l'oggetto. Questo discorso si estende quindi naturalmente alle periferiche grafiche. Esse sono dotate di un sistema di riferimento che permette all'utilizzatore di scegliere in maniera esatta dove collocare l'oggetto da raffigurare. Non solo, ma se si decidesse, una volta effettuato il disegno, di «spostarlo», l'operazione verrebbe condotta in maniera molto semplice, senza dover «manualmente» rigenerare il disegno.

Ciò dipende dal fatto che ogni parte del disegno è stata espressa sotto forma di entità geometriche (punto, linea, superficie, solido) e perciò dotata di coordinate che indicano la posizione di ciascuna entità in un particolare spazio geometrico.

Window e Viewport

Nei linguaggi grafici più diffusi, sono presenti due funzioni, il cui scopo è quello di semplificare al massimo le operazioni di scelta del sistema di riferimento e di posizionamento del disegno sulla superficie di visualizzazione:

Set Viewport (x1, y1, x2, y2)
Set Window (wx1, wy1, wx2, wy2)

Per Window si intende quella parte del modello che è stata scelta per la rappresentazione. Se per esempio si è scelto di disegnare una sinusoide, si deve stabilire che parte raffigurare, essendo questa una funzione periodica definita su tutto l'asse reale. La window definirà che intervallo della sinusoide dovrà essere rappresentato (figura 1). La window ha anche un'altra importante applicazione: lo zoom. Se vogliamo ingrandire un particolare del disegno appena rappresentato, non dobbiamo modificare il programma che lo genera, dovremo semplicemente modificare i parametri della window, rimpicciolendola. Poiché il disegno occuperà sempre la stessa porzione di schermo, il disegno risulterà ingrandito. La Viewport è invece quella parte della superficie di visualizzazione dove apparirà il disegno. Normalmente il disegno occupa

Figura 1 - *La funzione y = sin (x) è periodica e si estende lungo tutto l'asse reale. Se vogliamo rappresentarla su di una periferica grafica, dovremo stabilire che parte della funzione rappresentare. Questa operazione è detta Windowing. Dopodiché bisogna stabilire in che parte della superficie di visualizzazione rappresentarla (scelta del Viewport). Scelta una Window, si possono avere Viewport differenti.*

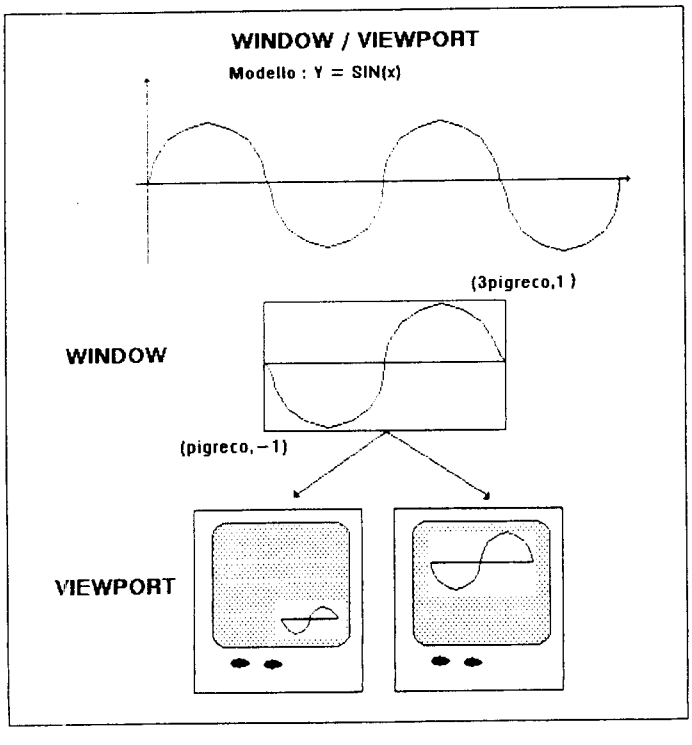
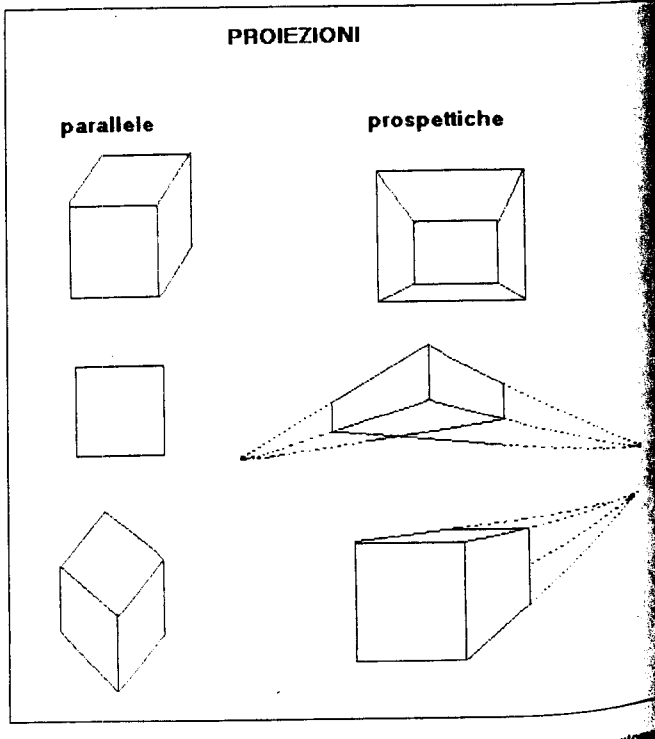


Figura 2 - *Esempio di proiezioni: nella prima (assonometria) vengono conservate alcune proprietà metriche, mentre nella seconda (centrale) alcune rette (parallele prima della proiezione), convergono in un punto, detto fuga.*



tutto lo schermo, ma tramite il concetto di viewport è possibile fare coesistere sulla stessa superficie più disegni non sovrapposti.
Entrambe queste regioni sono rettangolari e verranno perciò definite da quattro numeri. Le funzioni «Set Window» e «Set Viewport» sono il modo con cui l'utente stabilisce la window e la viewport. Entrambe queste funzioni avranno come parametri quattro numeri; nel caso della window saranno quattro numeri reali, espressi nel sistema di riferimento del modello, mentre nel caso della viewport saranno quattro numeri interi non negativi, il cui valore massimo non potrà eccedere la risoluzione del terminale utilizzato.
La possibilità di stabilire che parte del disegno raffigurare e in che punto dello schermo collocarla è comunque legata ad un'altra importante funzione di molti sistemi grafici e cioè il cosiddetto Clipping. Si tratta della capacità di non rappresentare parti del disegno esterne alla window stabilita.

Trasformazioni geometriche

Un altro strumento fondamentale nello sviluppo di applicazioni grafiche sono le cosiddette trasformazioni, che permettono di trasformare un insieme di punti con certe caratteristiche in un insieme con diverse caratteristiche.
Un esempio lo abbiamo appena visto. Viene detta «window to viewport mapping» e trasforma un punto appartenente al modello da rappresentare (per esempio 0,3.1415 nel caso della sinusoide) in un punto appartenente alla superficie di visualizzazione (per esempio 10,400). Altre trasformazioni, dette geometriche perché realizzano operazioni di tipo geometrico, permettono invece di fare eseguire movimenti o modificare l'aspetto di immagini già raffigurate sul terminale. Possiamo raggrupparle nel seguente modo:

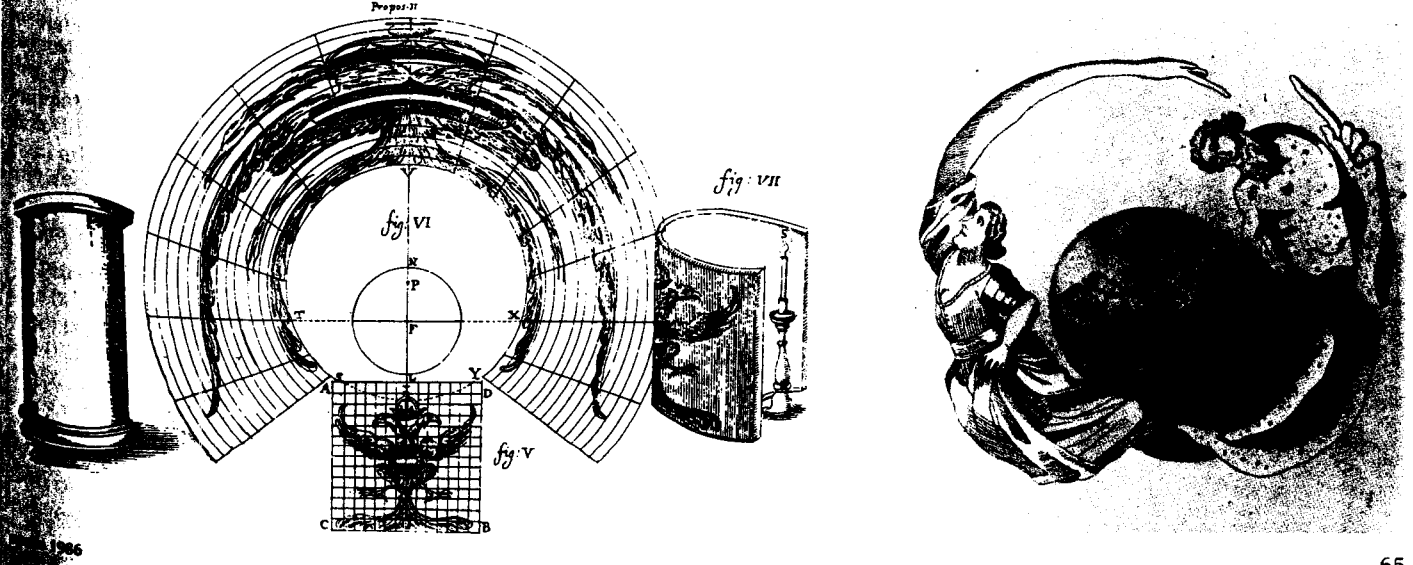
— tradizionali (traslazione, rotazione, riflessione)
— proiezioni
— deformazioni (scaling, torsioni, anamorfosi).

Le trasformazioni tradizionali permettono di spostare l'oggetto rappresentato lungo la superficie di visualizzazione, oppure di variarne l'orientamento, senza però variarne l'aspetto o la dimensione. Le più classiche sono la traslazione e la rotazione, alle quali se ne aggiungono altre come per esempio la riflessione speculare.
Un secondo gruppo di trasformazioni geometriche è costituito dalle proiezioni. Il loro scopo è di passare da una rappresentazione a tre dimensioni, ad una bidimensionale. Ciò deve essere fatto in quanto la superficie di visualizzazione è tipicamente a due dimensioni, mentre la realtà è a tre dimensioni. Vi sono molti tipi di proiezioni e sarebbe noioso elencarle. Basti dire che si dividono principalmente in due gruppi:

— proiezioni parallele
— proiezioni prospettiche.

Nelle prime (ortografiche, assonometriche) non esistono i punti di fuga e vengono rispettate alcune proprietà metriche, mentre nelle seconde (centrali) il parallelismo non viene sempre rispettato. Infatti alcune linee che normalmente sarebbero parallele, dopo questa trasformazione diventano convergenti verso uno o più punti, detti punti di fuga (vanish point) (figura 2).
Le deformazioni appartengono invece al terzo gruppo. Il loro scopo è modificare l'aspetto dell'oggetto raffigurato. La più semplice è certamente lo «scaling» che permette di variarne la dimensione applicando un fattore di scala. Deformazioni più sofisticate utilizzano invece le tecniche delle trasformazioni tradizionali, ma non vengono applicate ad un corpo rigido. Sono cioè operazioni incrementali in cui l'entità del movimento varia col variare dei punti. Per esempio nel caso del twisting, si applica una rotazione a tutti i punti dell'oggetto, ma l'angolo di rotazione non è costante, ma varia. In questo modo l'oggetto risultate sembra attorcigliato. Altre trasformazioni, molto usate nel '600, sono le anamorfosi. Il termine *anamorfosi* è stato coniato nel '600, per designare una certa specie di «depravazione ottica» basata sui giochi della prospettiva e della riflessione. Queste immagini sono distorte, spesso mostruose ed indecifrabili ma, se viste da un certo punto dello spazio o riflesse su specchi particolari (per esempio conici o cilindrici), si ricompongono, si rettificano, svelando figure a prima vista non percepibili (figura 3).
Tutte queste trasformazioni sono caratterizzate da delle formule matematiche che permettono di calcolare, dato un punto, il suo trasformato. Queste formule non sono generalmente uniche e la forma più utilizzata è quella matriciale, che permette di gestire in maniera omogenea tutti i tipi di trasformazione (box 1).

Figura 3 - *Esempio di Anamorfosi. Nel disegno a sinistra, risalente al 1646, viene messa in rilievo una tecnica per la costruzione di una anamorfosi cilindrica, mentre a destra viene mostrata la ricostruzione di un'immagine anamorfica su di uno specchio conico.*



Box 1

Matrici e trasformazioni geometriche

Le trasformazioni geometriche sono delle entità matematiche che permettono di variare alcune proprietà — per esempio posizione, orientamento o dimensione — di oggetti rappresentati geometricamente. Matematicamente possono essere espresse come sistemi di equazioni del tipo:

$$X' = f1\ (x)$$
$$Y' = f2\ (y)$$
$$Z' = f3\ (z)$$

dove si mette in risalto che il nuovo punto (X', Y', Z') è stato calcolato dal precedente (x, y, z) applicando in maniera opportuna una serie di funzioni. Per esempio la traslazione (spostamento lungo una retta) soddisferà le seguenti relazioni:

$$X' = x + SpostX$$
$$Y' = y + SpostY$$
$$Z' = z + SpostZ$$

mentre la variazione della dimensione dell'oggetto utilizzerà le seguenti equazioni:

$$X' = x \cdot \text{Fattore-di-scala}$$
$$Y' = y \cdot \text{Fattore-di-scala}$$
$$Z' = z \cdot \text{Fattore-di-scala}$$

Esiste comunque una maniera matematicamente più compatta per rappresentare le trasformazioni: la Rappresentazione Matriciale. Con questo strumento possiamo facilmente rappresentare in un'unica matrice più tipi di trasformazioni da applicare contemporaneamente all'oggetto. Se poi vogliamo applicare delle trasformazioni in successione, la rappresentazione matriciale lo permette in maniera molto naturale per mezzo della concatenazione di matrici.
Il loro grande vantaggio, rispetto al metodo classico, sta comunque nella loro generalità, per cui non si deve avere una serie di equazioni per ogni tipo di rappresentazione (figura 9). L'applicazione di una trasformazione verrà sempre realizzata moltiplicando il punto di trasformare (espresso sotto forma di vettore) per una opportuna matrice di trasformazione, nel seguente modo:

$$(X', Y', Z') = (x, y, z,) \cdot$$
$$\cdot \text{ Matrice Trasformazioni}$$

Per comporre due trasformazioni, sarà sufficiente applicare il prodotto matriciale:

Nuova Matrice = Matrice 1 · Matrice 2

In questo modo è possibile realizzare trasformazioni complesse semplicemente concatenando trasformazioni più semplici.

ROTAZIONE DI UN ANGOLO a

$$X' = X\cos(a) + Y\sin(a)$$
$$Y' = -X\sin(a) + Y\cos(a)$$

TRASLAZIONE

$$X' = X + Tx$$
$$Y' = Y + Ty$$

RAPPRESENTAZIONE MATRICIALE

TRASLAZIONE

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

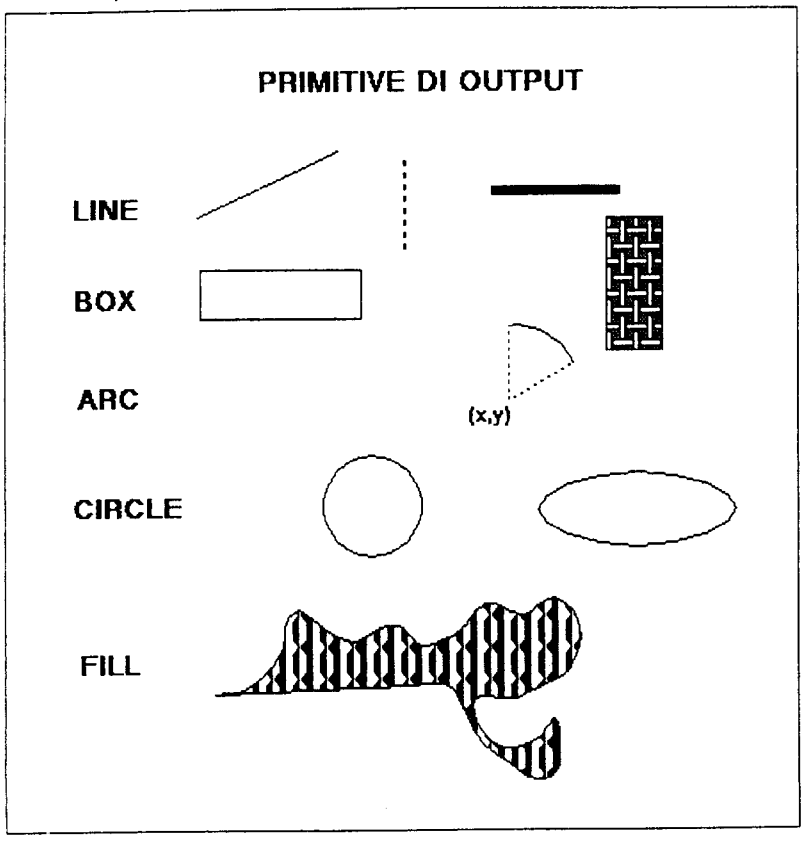
ROTAZIONE

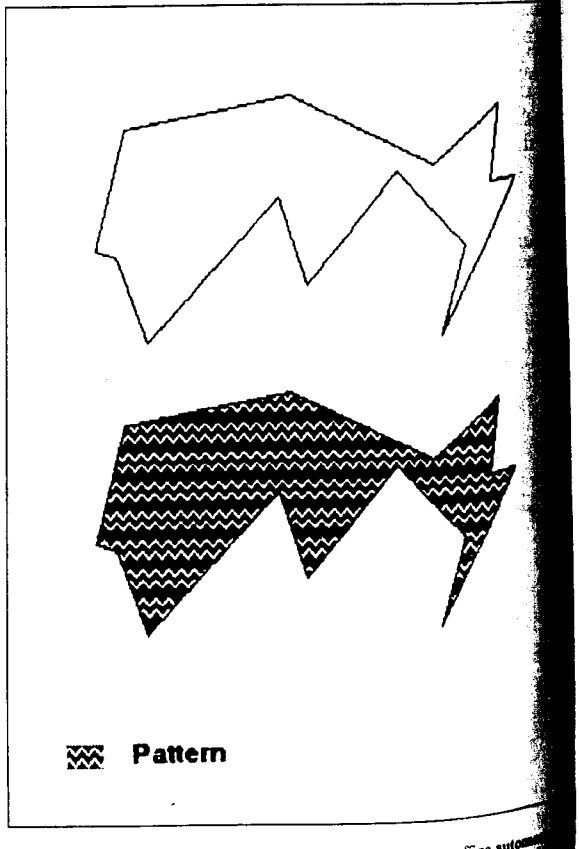
$$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[X'\ Y'\ 1] = [X\ Y\ 1] * MAT$$

Figura 9 - *Esempio di rappresentazione matriciale di una trasformazione.*

Figura 4 - *Esempio di primitive grafiche di output.* — **Figura 5** - *Applicazione della primitiva Fill ad un'area racchiusa da una poligonale (border). Il pattern viene replicato verticalmente e orizzontalmente fino a riempire l'immagine.*





Le primitive grafiche

Fino ad adesso abbiamo visto i fondamenti matematici per realizzare un disegno su di una periferica grafica. Non abbiamo però ancora parlato delle funzioni che ci permettono effettivamente di disegnare. Queste funzioni vengono generalmente chiamate primitive grafiche, in quanto permettono all'utente di eseguire semplici disegni (punti, linee, quadrati, cerchi ...). La costruzione di disegni più complessi dovrà quindi essere fatta utilizzando opportunamente e combinando queste primitive.

In questo articolo parleremo soltanto delle primitive di output, mentre rimanderemo il discorso sulle primitive di input in un prossimo articolo, quando si parlerà di interfaccia uomo-macchina.

La più semplice funzione di output per una periferica grafica è senz'altro la fuzione:

Set Pixel (x, y, colore)

che permette di assegnare un particolare colore al punto di coordiante (x, y). Pur nella sua semplicità, questa funzione ci può dare subito dei problemi in quanto non è una funzione del tutto generale. Infatti il valore del parametro colore dipenderà dal tipo di periferica utilizzata. In un terminale monocromatico semplice, potrà avere soltanto due valori: acceso/spento, mentre in terminali più complessi sarà l'indice nella tabella di colori. Inoltre nei terminali a vetore, non esiste il Frame Buffer (vedi articolo sui Sistemi Grafici), perciò il concetto di pixel (punto grafico) non esiste.

Questa considerazione non vale solo per la funzione Set Pixel, ma si applica a tutte le primitive grafiche e permette di evidenziare uno dei principali problemi della computer graphics: la dipendenza dalle periferiche utilizzate. In uno dei prossimi articoli questo problema verrà affrontato più in dettaglio, e si parlerà dei tentativi di risolverlo mediante gli standard grafici.

Nonostante ciò è possibile parlare di primitive grafiche logiche. Si possono cioè raggruppare in classi logiche (prescindendo dalle implementazioni particolari) le funzioni grafiche presenti sulle più diffuse periferiche grafiche, utilizzando come fattore aggregante le loro funzionalità principali (figura 4):

- Pixel (x, y, colore) o Pset (x, y, colore)
- Line (x1, y1, x2, y2) o Draw (x, y)
- Box (x, y)
- Fill (x1, y1, x2, y2)
- Arc (x, y, raggio, angolo)
- Circle (x, y, raggio, rapporto)
- Paint (x, y, colore, bordo)
- Point (x, y, colore, bordo)
- Clear o Cls

La prima, Set Pixel, l'abbiamo appena vista.

In questo elenco sono riportati dei nomi alternativi, spesso utilizzati nei linguaggi grafici per rappresentare lo stesso tipo di funzione.

Line permette di tracciare una linea retta congiungente il punto (x1, y1) con il punto (x2, y2). Spesso questa funzione si trova con una sola coppia di parametri (per esempio Draw [x, y]). Si tratta sempre della medesima funzione, l'unica differenza e che viene utilizzata la posizione corrente: ogni volta che viene eseguita una primitiva di output, il sistema ricorda l'ultima posizione in cui ha eseguito un tracciamento. Quindi Draw (x, y) vuol dire «traccia una linea retta che congiunge la posizione corrente con il punto (x, y) e fai diventare (x, y) la nuova posizione corrente». Questa convenzione non si applica solamente alla funzione logica Line, ma a tutte le primitive che eseguono dei tracciamenti.

La funzione Box traccia un rettangolo con lo spigolo in basso a sinistra coincidente con il punto (x1, y1) e quello in alto a destra col punto (x2, y2). La tecnica per specificare il rettangolo è analoga a quella utilizzata dalle funzioni Set Window e Set Viewport; viene cioè specifica la posizione dello spigolo in basso a sinistra e una diagonale. Talvolta è presente un parametro addizionale che dà la possibilità di creare un rettangolo colorato (non solo il contorno).

Arc permette invece di tracciare un arco, che sottende un certo angolo. La distanza dell'arco dal punto (x, y) (detto centro) viene regolata dal parametro «raggio». Ovviamente se l'angolo è giro, non verrà generato un arco, bensì una circonferenza. Si può quindi utilizzare questa funzione per tracciare circonferenze.

Spesso sono però disponibili funzioni ad hoc per rappresentare circonferenze. La loro differenza con le funzioni di tipo Arc è che forniscono un parametro addizionale (nell'elenco indicato come rapporto) che permette di controllare, durante la generazione della figura, il rapporto tra gli incrementi verso x e verso y. Variando questo parametro è quindi possibile generare, oltre che circonferenze, anche ellissi.

La funzione Fill permette il riempimento di aree con un certo colore o con un certo disegno, detto Pattern, che viene duplicato orizzontalmente e verticalmente, sino a riempire completamente l'area specificata (figura 5). Generalmente questa funzione ha bisogno di un punto interno (x, y) all'area da riempire e del colore di riempimento. È anche necessario specificare il colore del contorno in quanto la funzione non presuppone l'esistenza di una particolare area. Semplicemente partendo dal punto specificato (che deve essere interno alla figura e non sul contorno) incomincia a colorare in una direzione, finché non trova un pixel del colore del contorno, a questo punto cambia direzione e continua il processo. Per questo motivo l'area da riempire deve essere completamente circondata dal contorno. Se manca anche un solo puntino, il riempimento verrà esteso a tutta la superficie di visualizzazione. Clear, infine, permette di cancellare la superficie di visualizzazione. Ha quindi particolare importanza in applicazioni interattive.

La grafica tridimensionale ed il problema del realismo

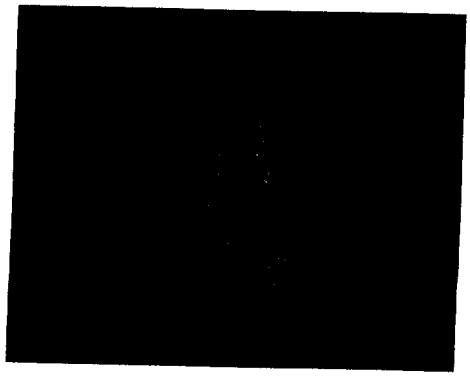
Concludiamo questa breve rassegna dando un cenno ad un gruppo di tecniche sviluppate recentemente per dare maggiore realismo alle immagini generate. Vengono utilizzate prevalentemente su terminali raster in quanto si basano sulla possibilità di colorare il singolo punto dell'immagine. Tre sono principalmente le tecniche utilizzate: Hidden Surface Removal (rimozione delle superfici nascoste); Shading (ombratura); Texturing (tessitura).

Ciascuna di esse cerca di realizzare, spesso con metodi matematici molto complessi e «time-consuming» alcune caratteristiche del processo della visione.

Le tecniche di rimozione delle linee nascoste sono forse le più diffuse. Si propongono di eliminare dal disegno quelle parti che, in una scena reale, non verrebbero viste dall'osservatore in quanto nascoste da oggetti non trasparenti. Potremmo dire che simulano una proprietà di molti oggetti reali, e cioè l'opacità ai raggi di luce (figura 6).

Tutte si basano su di una sorta di ordinamento che permette di stabilire, dato un gruppo di oggetti presenti sulla stessa linea visiva, quale è il più vicino all'osservatore (e quindi visibile).

Figura 6 - *Rappresentazione di un oggetto sia mediante la tecnica wire-frames, che con la rimozione delle superfici nascoste.*



Box 2

Lo Shading

La tecnica dello Shading (ombratura) contribuisce sensibilmente alla formazione di immagini realistiche su un dispositivo grafico di tipo raster. Una volta che le parti visibili del disegno sono state identificate da un algoritmo per la rimozione delle superfici nascoste, si deve usare uno «shading model» per calcolare l'intensità e i colori da utilizzare per rappresentarle sul terminale video.

Lo shading non simula esattamente il comportamento di luce e superfici nel mondo reale, ma ne approssima le condizioni; il disegno di un modello è sempre un compromesso tra precisione e costo computazionale. In questo caso il trade-off è dato in particolare dalla proprietà del sistema visivo umano, che tende ad influenzare la percezione del realismo.

Il modello di shading ha due ingredienti principali: le proprietà della superficie e quelle della luce che cade sulla superficie.

La principale proprietà della superficie è il Potere Riflettente, che determina quanto della luce incidente viene riflesso. In particolare se una superficie ha diversi poteri riflettenti per luci di differenti lunghezze d'onda, essa apparirà colorata. Un'altra importante proprietà, legata alla superficie in esame, è la trasparenza (figura 10).

Nel caso della luce, invece, sono particolarmente importanti l'intensità e il tipo di sorgente luminosa (puntiforme oppure illuminazione diffusa).

Vediamo ora un semplice modello di shading con illuminazione diffusa. Esso deve permettere di determinare lo «shade» di un punto sulla superficie in termini di un certo numero di attributi. Solitamente si usa la seguente formula:

$$E(p) = R(p) \cdot I$$

Figura 10 - *Esempio di immagine in cui vengono simulati gli effetti di trasparenza.*

che indica che l'energia luminosa proveniente dal punto p (e cioè E (p)) e dovuta all'illuminazione diffusa, è data dal prodotto di due fattori: l'illuminazione diffusa che cade su tutta la scena (I) e il potere riflettente in p (R (p)).

Se si vogliono modellare superfici colorate, bisogna valutare separatamente le tre componenti principali del sistema di colore:

$$E(p, \text{rosso}) = R(p, \text{rosso}) \cdot I(\text{rosso})$$

$$E(p, \text{verde}) = R(p, \text{verde}) \cdot I(\text{verde})$$
$$E(p, \text{blu}) = R(p, \text{blu}) \cdot I(\text{blu})$$

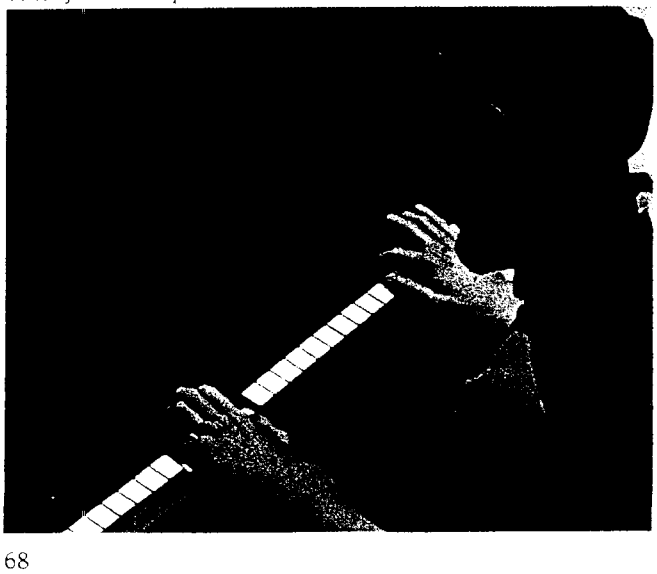
Queste tre relazioni possono ovviamente essere riassunte da una più composta notazione vettoriale.

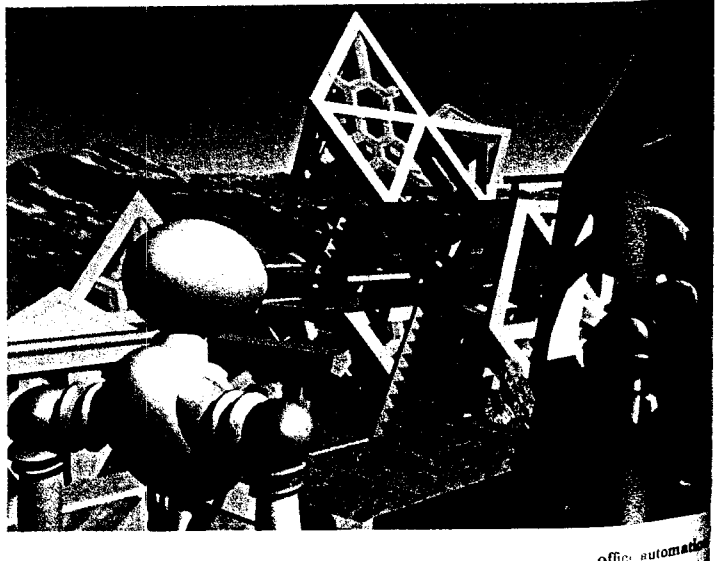
Questo modello può essere sofisticato. Se si vuole introdurre, per esempio, una sorgente luminosa puntiforme, bisognerà utilizzare la legge di Lambert, che afferma che «l'energia sprigionata da una sorgente puntiforme che cade su una superficie varia con il coseno dell'angolo di incidenza». In questo caso la relazione sarà (per il caso monocromatico):

$$E(p) = [R(p) \cdot \cos(ang)] \cdot I(p)$$

dove «ang» è l'angolo di incidenza tra la normale alla superficie ed il raggio luminoso.

Figura 7 - *Esempio di ombratura di una scena.* — Figura 8 - *Esempio di texturing. Utilizzando la tecnica di texturing è stato possibile dare alle montagne dello sfondo un aspetto reale.*





primo di questi algoritmi, sviluppato agli degli anni '60, funzionava esclusivamente su terminali a vettore. Su questi terminali gli oggetti venivano raffigurati come ...lie di linee (mesh), per cui il compito ...cipale di questo algoritmo era la individuazione delle linee nascoste. Quando i ...minali raster incominciarono ad essere ...ponibili a costi ragionevoli, l'attenzione ...spostò invece sulla rimozione delle superfici nascoste. Qualunque terminale si ...lizzi, non esiste però un unico algoritmo ...e risolva il problema della eliminazione ...maniera efficiente. Le principali differenze tra i metodi correntemente utilizzati nascono dalle diverse applicazioni in cui vengono impiegati.

...tecniche di shading cercano invece di ...ppresentare un modello di illuminazione ...quindi di calcolare zone luminose e zone ...mbreggiate nel disegno. Utilizzando delle ...genti luminose, si riesce a dare un notevole realismo al disegno (figura 7) in quanto si toglie quella caratteristica che hanno molte immagini sintetiche generate col computer, e cioè l'illuminazione diffusa su tutta la scena rappresentata, senza quindi ombre. Nel box 2 vengono affrontati brevemente due modelli di illuminazione.

...exturing, infine cerca di dare una parvenza di realtà alle superfici generate associando una tessitura che, «incollata» alla superficie, permetta di rappresentare fenomeni come onde del mare, nuvole, montagne (figura 8). Matematicamente la tessitura (o trama) di una superficie può essere definita come una funzione intensità (x, y), che associa ad ogni punto (x, y) appartenente alla superficie un valore di intensità. Il problema è quindi trovare particolari funzioni che rappresentino in maniera realistica situazioni naturali.

...ono altre tecniche, alcune molto recenti, che aggiungono contributi alla rappresentazione del realismo. Tra queste si può citare il Ray Tracing, che permette di realizzare immagini altamente realistiche modellando la riflessione, la rifrazione e la trasparenza negli oggetti. Le tecniche stocastiche (Frattali, Moto Browniano) che simulano movimenti complessi (un prato mosso dal vento) o realizzano immagini complesse (una costa molto frastagliata). Le Strutture Grafiche Gerarchiche, che permettono di modellare il movimento umano realizzando regole come «se si muove il polso, allora si muove anche la mano, ma non viceversa». Infine l'Antialiasing, che permette di ridurre le «scalettature» generate dai terminali raster. ■

Andrea Granelli

(3 - continua)

BIBLIOGRAFIA RAGIONATA

[BALT69] J. Baltrusaitis: *Anamorfosi o magia artificiale degli effetti meravigliosi,* Adelphi, 1969, Milano.

[BALT81] J. Baltrusaitis: *Lo specchio: rivelazioni, inganni e science-fiction,* Adelphi, 1981, Milano.

[FALC85] B. Falcidieno, S. Ansaldi: Software grafico, Franco Angeli, 1985, Milano.

[BARR84] A.H. Barr: *Global and Local Deformations of Solid Primitives,* Computer Graphics, vol. 18, n. 3, Luglio 1984.

[BUIT75] Bui-Tuong, Phong: *Illumination for Computer-Generated Pictures,* Communications of ACM, vol. 18, n. 6, Giugno 1975.

[CARL78] I. Carlbom, J. Paciorek: *Geometric Projection and Viewing Transformations:* Computing Survey, vol. 10, n. 4, 1978.

[CHIL74] R.L. Childress: *Sets, Matrices and Linear Programming,* Prentice-Hall, 1974, New Jersey.

[EARN85] R.A. Earnshaw Editor: *Fundamental Algorithms for Computer Graphics,* Springer-Verlag NATO ASI Series, 1985, Berlin.

[NEWM79] W.N. Newman, R.F. Sproull: *Principles of Interactive Computer Graphics,* McGraw-Hill, New York 1979.

[NORT82] A. Norton, A.P. Rockwood, P.T. Skolmoski: *Clamping: A method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space,* Computer Graphics, vol. 16. n. 3, Luglio 1982.

[OLIV] Olivetti: *MS GW-Basic Interpreter: Guida utente,* Codice 4013270 X (1), Olivetti.

[SPRO68] R.F. Sproull, I.E. Sutherland: *A Clipping Divider,* FJCC 1968, Thompson Books, Washington DC, 1968.

[SUTH74] I.E. Sutherland, R.F. Sproull, R.A. Schumaker: *A Characterization of Ten Hidden-Surface Algorithms,* Computer Survey, vol. 6. n. 1, Marzo 1974.

[NEWM79] contiene un po' tutti gli argomenti trattati nel presente articolo, in particolare tratta delle primitive di output, dando anche indicazione su come realizzarle. Per chi volesse invece vedere una implementazione delle primitive grafiche, può consultare [OLIV].
Sul concetto di window/viewport, oltre all'appena citato [NEW79] si può anche vedere per esempio una introduzione ad un linguaggio grafico come il GKS, dove il problema dei differenti sistemi di coordinate e relative trasformazioni di window e viewport è analizzato in dettaglio [FALC85], oppure [SPRO68].
Per la parte matematica sulle trasformazioni geometriche, si può vedere [CHIL74]. Chi è invece più interessato ad una realizzazione, può vedere [CARL78] per la parte più tradizionale, mentre per le deformazioni [BARR84]. Per chi volesse una trattazione più storico-filosofica sulle trasformazioni geometriche, con particolare riferimento alle riflessioni speculari e alle anamorfosi, può vedere [BALT81] e [BALT69].
Sulla parte del realismo tridimensionale, è difficile consigliare un testo, data la grande varietà di tecniche impiegate. Comunque, per la parte di rimozione delle linee nascoste, [SUTH74] contiene una interessante rassegna sugli algoritmi più utilizzati, mentre [BUIT75] parla dei modelli di shading. Per quanto riguarda invece il texturing, si può vedere [NORT82]. Chi volesse vedere una rassegna recente sugli algoritmi utilizzati nella computer graphics, può vedere [EARN85].