

Addentriamoci nella grafica
in Windows con Delphi, proviamo
le funzioni grafiche e vediamo
con esempi pratici i grafici di funzioni
e le principali funzioni grafiche

◆ Riccardo Bianco-Mengotti

Grafica con Delphi

A differenza di sistemi come Dos o Unix l'interfaccia di default per Windows è completamente grafica. Questo significa che i programmi scritti per Windows utilizzano linee, forme, tratteggi, colori, caratteri eleganti e così via, anziché apparire con il classico schermo nero e cursore lampeggiante.

Onori e oneri, è il caso di dirlo.

Onori: chi scrive programmi per Windows non si deve più preoccupare di "aprire" la grafica, "ingraziarsi" la scheda video, verificare la risoluzione, il numero di colori e così via.

Oneri: qualsiasi programma, anche piccolo, dovrà considerare concetti come pixel, font, colori e certi programmatori vecchio stile sembrano avere una specie di allergia a queste cose.

In realtà, almeno inizialmente, per la maggior parte delle situazioni sono sufficienti i pochi elementi basilari che si è cercato di raccogliere in queste pagine.

La grafica in Windows con Delphi

Innanzitutto occorre tenere presente che lo schermo è composto da una griglia di punti (pixel). Ogni punto ha un colore scelto, in genere, tra 16 milioni circa;



Grafica Bitmap ➔
L'oggetto TCanvas ha un apposito metodo per la generazione di ellissi.



tanti risultano combinando tra loro 256 sfumature di blu, verde e rosso. Per ottenere la definizione numerica di un colore basta accostare le due cifre (in esadecimale, da 00 a FF) di ciascun colore base nell'ordine indicato sopra, precedendo tutta la cifra con il simbolo \$ (che in Delphi indica i numeri esadecimali). Perciò per esempio il blu chiaro sarà \$FF0000, il blu scuro \$800000, il nero \$000000, il bianco \$FFFFFF o il giallo (rosso+verde) \$00FFFF.

L'angolo dello schermo in alto a sinistra è il primo punto e viene indicato con le coordinate [0,0]. La prima coordinata rappresenta la colonna della griglia-schermo o, per chi è abituato alle coordinate cartesiane, l'ascissa; la seconda coordinata indica la riga (ordinata). Si tratta, per inciso, degli stessi numeri riportati sui depliant di monitor e schede video, tipicamente 640x480, 800x600, 1.024x768, 1.280x1.024, 1.600x1.200 e così via.

Queste sono le coordinate dell'intero schermo, mentre in Windows ciascun programma avrà in generale a che fare solamente con le coordinate della propria finestra. O meglio, avrà a che fare con la parte utilizzabile della propria finestra, escludendo infatti la barra del titolo, eventuali barre di menu e il bordo, elementi dei quali si occupa automaticamente il sistema operativo.

Windows è un ambiente a eventi e perciò anche il disegno della finestra di un programma (refresh) deve

essere avviato in conseguenza di un evento. In pratica un metodo di refresh (OnPaint) viene richiamato ogni volta che una finestra di un programma viene creata oppure viene ripristinata dallo stato di icona o, ancora, un'altra finestra sovrapposta viene spostata. Si può sperimentare facilmente la faccenda creando un nuovo programma, cliccando due volte sulla voce OnPaint della finestra principale nell'Object Inspector e quindi inserendo l'istruzione MessageBeep(0); In questo modo eseguendo il programma si sentirà il beep di default tutte le volte in cui viene richiamato il metodo OnPaint.

Il concetto di Canvas

L'elemento che in Delphi è incaricato di gestire la grafica bitmap è la classe TCanvas. Come suggerito dal nome, si tratta della tela sulla quale dipingere la grafica; l'analogia con la pittura è dovuta al meccanismo per cui il nuovo disegno (la nuova pennellata) viene sovrapposto al colore originale presente sullo schermo (la tela).

L'oggetto TCanvas contiene tutte le funzioni grafiche di base come la manipolazione diretta dei pixel, il disegno di forme geometriche, la gestione dei colori e via dicendo.

Le proprietà del Canvas più importanti sono Pen e Brush, ovvero rispettivamente la penna con cui vengono tracciate linee e contorni e il pennello con cui vengono riempite le forme piene.



➔ **Tracciare linee**
 Per generare una linea da una coppia di coordinate a un'altra si usa il comando *LineTo*.

◆ LE PRINCIPALI FUNZIONI GRAFICHE

I metodi dell'oggetto Tcanvas permettono di gestire le più comuni operazioni grafiche. In particolare per disegnare le forme base ecco le principali funzioni a disposizione con una rapida descrizione. Si tratta di funzioni semplici ma indispensabili da conoscere.

```
procedure TextOut _
(X, Y: Integer; _
const Text: string);
```

Consente di scrivere il testo text nella posizione [x,y]

```
procedure LineTo _
(X, Y: Integer);
```

Permette di disegnare una linea dalla posizione corrente alla posizione [x,y] usando la penna corrente

```
procedure Rectangle _
(X1, Y1, X2, Y2: _
Integer);
```

Consente di disegnare un rettangolo con vertici [x1,y1] e [x2,y2] riempito con il pennello corrente, usando per il contorno la penna corrente

```
procedure Ellipse _
(X1, Y1, X2, Y2: _
Integer);
```

Consente di disegnare un'ellisse inscritta nel rettangolo con vertici [x1,y1] e [x2,y2], come si può osservare nell'immagine in basso, riempito con il pennello corrente, usando per il contorno la penna corrente.

Se invece di un rettangolo si specifica un quadrato, la figura disegnata sarà, come è facile aspettarsi, un cerchio. Combinando opportunamente queste funzioni è possibile ottenere forme complesse e articolate, come per esempio frecce, griglie, quote, simboli tecnici e così via.



>>



↑ Ancora con TCanvas
Un altro metodo della classe TCanvas consente la generazione di rettangoli.

Per impostare il colore si utilizzeranno le proprietà Canvas.Pen.Color e Canvas.Brush.Color mentre per cambiare stile della linea o tratteggio si agirà sulle proprietà Pen.Style e Brush.Style.

Per modificare i singoli punti della finestra si può agire sulla proprietà Pixel, che è una matrice che contiene appunto la griglia di pixel della parte di schermo occupata dalla finestra. Per esempio per disegnare un punto verde alle coordinate [100,100] si scriverà Pixel[100,100]:= \$00FF00.

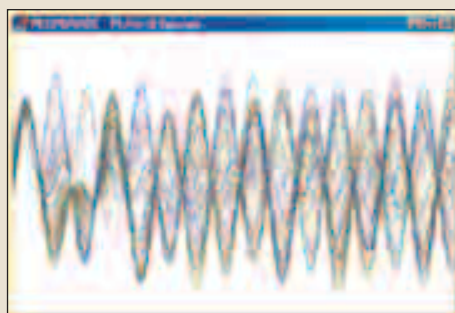
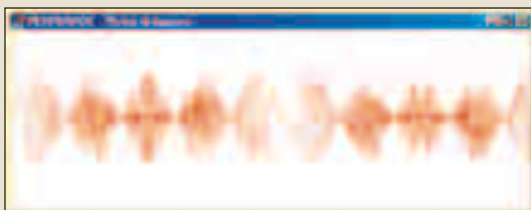
Ogni Canvas prevede una posizione corrente, che può essere considerata la posizione in cui si trova sospesa la penna di un disegnatore; per cambiare la posizione si usa il metodo Canvas.MoveTo(x,y).

La posizione corrente può essere sempre ottenuta usando la proprietà Canvas.PenPos.

◆ UN ESEMPIO: GRAFICI DI FUNZIONI

Questo semplice esempio permette di provare le funzioni grafiche di Delphi realizzando un semplice plotter di funzioni x-y.

Per prima cosa occorre creare una nuova applicazione (File/New/Application) e disporre sulla finestra un oggetto TPaintbox che verrà utilizzato per disegnare i grafici. Impostare quindi la proprietà Align dell'oggetto PaintBox a AlClient, in modo che esso occupi tutto lo spazio disponibile e scegliere un colore (tranne il rosso) per lo sfondo della finestra (proprietà Color del Form). Le istruzioni grafiche andranno inserite nell'evento di ridisegno dell'oggetto Paintbox: a questo scopo fare doppio clic sulla voce OnPaint (pagina degli eventi nell'Object Inspector, con il PaintBox selezionato).



DELPHI, UN PO' DI STORIA

Delphi fa la sua comparsa a metà degli anni '90 ed è una evoluzione del noto Turbo Pascal, ovvero la versione Borland del linguaggio Pascal. Con l'arrivo di Delphi, Borland passa alla programmazione visuale Rad (Rapid application development), unendo un ambiente di sviluppo comodo e integrato (davvero simile a Visual Basic) alle migliori prestazioni offerte da un compilatore puro ed efficiente. L'immagine un po' scolastica e appannata del Pascal viene quindi sostituita con una nuova visione più propriamente orientata ai database, sottolineata oltretutto dalla scelta di un nuovo nome. Considerato il momento d'oro per i database l'operazione è riuscita, con una vittima illustre però, stiamo riferendoci alla visione del Pascal come linguaggio nel senso più ampio del termine. L'unico aspetto negativo è l'equazione Delphi=database. Nel bene e nel male.

Per disegnare una linea si parte tipicamente dalla posizione corrente e si usa il metodo Canvas.LineTo(x,y). In generale si useranno MoveTo e LineTo per disegnare un segmento, come mostrato nell'esempio riportato di seguito:

```
Canvas.MoveTo(20,10);
```

```
Canvas.LineTo(150,30);
```

Le più importanti istruzioni per disegnare forme geometriche sono comunemente riportate e spiegate brevemente nel box dedicato alle principali funzioni grafiche.

Quasi tutti gli elementi visibili dell'interfaccia contengono un oggetto TCanvas. Tuttavia l'oggetto che è preposto alla grafica è TPaintBox (nella pagina System della Component Palette), che contiene un oggetto Canvas sul quale è possibile disegnare la grafica. □

Il codice da inserire per disegnare, per esempio, il grafico di un'onda (sinusoide) di colore rosso sarà il seguente:

```
procedure TForm1.PaintBox1Paint(Sender: TObject);
var i:longint;
begin
  with PaintBox1 do
  begin
    Canvas.Pen.Color:=$0000FF;
    Canvas.MoveTo(0,100);
    for i:=1 to Width do
      Canvas.LineTo(i,100-round(sin(i/10)*50));
    end;
  end;
end;
```

Si possono provare varie funzioni per ottenere grafici diversi, sostituendo per esempio la linea che contiene LineTo con una delle linee seguenti:

```
Canvas.LineTo _
(i,100-round(sin(i/10)*50*cos(i/20)));
```

```
Canvas.LineTo _
(i,100-round(sin(i*i/100)*50*sin(i/20)));
```