



Delphi: un gioco veloce



Dimostriamo la facilità d'uso
di Delphi creando il vecchio gioco
del tennis con pochissime righe
di codice ♦ **Riccardo Bianco-Mengotti**

Fin dalla sua apparizione Delphi si è proposto come strumento di sviluppo rapido (Rad), un ambiente che permette di realizzare programmi rapidamente, con il minimo sforzo di scrittura di codice e con largo uso di tecniche visuali e grafiche. In particolare il programmatore si trova a dover scrivere codice solo dove necessario, dove cioè i comportamenti standard

devono essere rimpiazzati da azioni specifiche del programma. All'avvio di Delphi per esempio si parte subito con un programma "bianco" che, una volta compilato, è già un normale applicativo per Windows: può essere ridotto a icona, ridimensionato, spostato e così via. Tutto questo senza scrivere nemmeno una riga di codice, giacché i comportamenti standard delle finestre Windows vengono incorporati automaticamente.

A questa filosofia di programmazione si unisce la modularità dei componenti: mattoncini standard che coprono diverse esigenze, sia di interfaccia sia per esempio di gestione dati. L'utente costruirà il suo programma in modo visuale, aggiungendo graficamente componenti alle finestre e scrivendo qua e là il codice strettamente indispensabile, mentre la maggior parte dei comportamenti di routine verranno generati automaticamente.

R.A.D. cronometro alla mano

L'esperimento di sviluppo rapido consiste nel realizzare con Delphi un vecchio gioco che i lettori "non più ventenni" forse ricordano: il

◆ PARTITA A TENNIS CON NEWTON

Nell'esempio presentato in queste pagine la palla ha un comportamento molto semplice: va per la sua strada finché non rimbalza su un muro o una racchetta. Volendo però essere più precisi si possono provare a scrivere in Delphi le leggi fisiche in modo meno approssimato. La palla in volo si trova nella condizione indicata in figura, ha una velocità e una forza (resistenza) opposta alla velocità che la fa rallentare, cioè decelerare. In situazioni come questa, Newton riemerge dai ricordi della scuola e dà un suo aiuto dicendo che la forza è uguale alla massa per l'accelerazione. La velocità a ogni passo si ottiene dall'accelerazione e analogamente la posizione si ricava dalla velocità. Nel programma si usano vx e vy per rappresentare le velocità orizzontale e verticale; vanno quindi aggiunte altre variabili per la forza Fx ed Fy e per l'accelerazione ax e ay, tutte di tipo reale come vx e vy.



Ora si può scrivere il movimento nel modo corretto. A ogni passo (nella procedura Timer1Timer) si avrà

```
Fx:=-0.01*vx*vx;  
Fy:=-0.01*vy*vy;  
ax:=Fx/0.15;  
ay:=Fy/0.15;  
vx:=vx+ax*0.05;  
vy:=vy+ay*0.05;
```

e infine il codice già presente:

```
panel5.Left:=round(panel5.Left+vx);  
panel5.Top:=round(panel5.Top+vy);
```

Provando a cambiare il valore della massa (qui 0.15) e del fattore di resistenza (qui 0.01) si potranno sperimentare gli effetti sulla palla da tennis. Magari facendo rivoltare Newton nella tomba.

tennis, uno dei primi videogiochi, che girava su console collegata alla televisione (in molti casi ancora in bianco e nero).

Una volta creata in Delphi una nuova applicazione (File\New\Application) si può procedere al disegno del programma.

Innanzitutto occorre creare un pannello da usare come terreno di gioco, selezionarlo nella standard palette e disporlo sulla finestra. Sarà anche possibile spostarlo e ridimensionarlo graficamente usando le maniglie (quadretti neri ai vertici e sui lati). Quando il pannello o un qualsiasi oggetto è selezionato, la finestra dell'Object Inspector (a sinistra) ne mostra le proprietà; tra queste va modificato il colore, scegliendo per esempio il verde, che nel videogioco originale era effettivamente il colore del campo.

Si procede nello stesso modo per creare, nell'ordine, le due racchette, la rete a metà campo e la palla: tutti pannelli (TPanel) di colore bianco; è importante crearli in quest'ordine per poter seguire il codice riportato nell'articolo, che si riferisce a Panel1 (il campo), Panel2/Panel3 (le racchette sinistra/destra), Panel4 (la rete) e Panel5 (la palla). Naturalmente si può anche cambiare il nome dei componenti agendo sulla voce Name nell'Object Inspector, ma in generale occorre rispettare alcune regole, visto che i nomi sono elementi del linguaggio: non sono ammessi numeri come caratteri iniziali, non si possono usare spazi bianchi, interpunzioni, segni di operazioni matematiche ecc. Bisogna inoltre fare attenzione a creare tutti i pannelli (racchette, palla ecc.) in modo che giacciono su Panel1 (il campo) cliccando - alla creazione - sul campo stesso in un punto libero da altri componenti; in caso contrario potrebbe per esempio capitare che la palla venga creata come pannello all'interno di una racchetta.

Se si sbaglia comunque si possono cancellare i componenti selezionandoli e scegliendo la voce di menu Edit; attenzione però a cancellare senza pietà: si può annullare solo l'ultima azione.

A questo punto si impostano i comandi delle racchette, collegando il movimento della sinistra ai tasti A e Z e della destra ai tasti J e M. La pressione dei tasti è un evento della finestra principale; selezionando la finestra (non un suo componente), fare doppio click sulla seconda pagina dell'Object Inspector alla voce

OnClick: verrà generato il codice per la risposta all'evento, che dovrà essere completato come indicato qui di seguito:

```
procedure TForm1.FormKeyPress_  
(Sender: TObject; var Key: Char);  
begin  
  case key of  
    'A', 'a': panel2.Top:=panel2.Top-8;  
    'Z', 'z': panel2.Top:=panel2.Top+8;  
    'J', 'j': panel3.Top:=panel3.Top-8;  
    'M', 'm': panel3.Top:=panel3.Top+8;  
  end;  
end;
```

Per fare in modo che queste istruzioni vengano sempre richiamate alla pressione dei tasti bisogna impostare la proprietà KeyPreview della finestra a TRUE: in questo modo la finestra riceve tutto l'input da tastiera e, provando a eseguire il programma (Project Run), è possibile muovere le racchette.

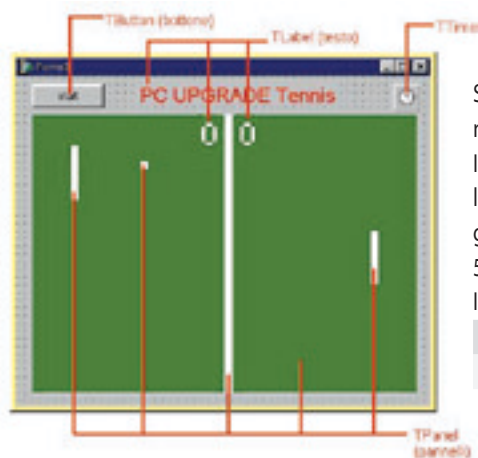
Non resta che programmare la palla. Il suo movimento nella realtà è continuo ma in un programma va immaginato come composto da una serie di passi, almeno uno ogni decimo di secondo per dare l'impressione di un

movimento fluido. Il componente Delphi che permette di eseguire codice a scadenze temporali si chiama TTimer e si trova nella pagina System della component Palette. Una volta inserito nella finestra del programma, impostare la proprietà Interval nell'Object Inspector a un valore di 50 (millisecondi); con un doppio click sul componente si genera il codice che verrà eseguito appunto ogni 50 millisecondi.

Il movimento della palla sarà quindi del tipo:

```
panel5.Left:=round(panel5.Left+vx);  
panel5.Top:=round(panel5.Top+vy);
```

dove vx e vy sono due variabili che rappresentano le velocità nelle due direzioni: è necessario usare delle variabili per poter simulare il rimbalzo invertendone i valori; le proprietà Left e Top sono numeri interi mentre vx e vy sono numeri reali che vengono arrotondati dalla >>



↑ Campo di battaglia
Gli elementi grafici dell'area di gioco sono pochi e le righe di codice necessarie a generarli sono immediatamente comprensibili.

» funzione round. Le variabili vanno dichiarate all'interno del blocco di definizione del Form, prima dell'ultima riga di chiusura "end;" come indicato di seguito:

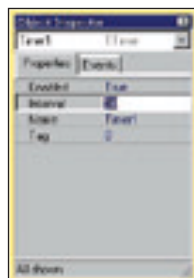
```
TForm1 = class(TForm)
. . . . .
private
{ Private declarations }
public
{ Public declarations }
vx,vy:real;
p1,p2:integer;
end;
```

Analogamente sono state dichiarate due variabili per tenere conto del punteggio e che verranno utilizzate in seguito. Notare che la definizione di TForm è stata generata automaticamente da Delphi; le uniche righe che l'utente deve aggiungere sono quelle che seguono la riga "public".

Le condizioni di rimbalzo per la palla vanno verificate a ogni passo e sono l'urto con le racchette e il raggiungimento dei bordi superiore e inferiore. Il superamento dei due bordi laterali determina invece il "punto" per un giocatore, che vedrà incrementata la variabile corrispondente al suo punteggio. Ecco come appare il codice che, istante per istante, definisce il comportamento della palla:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
// movimento normale
panel5.Left:=round(panel5.Left+vx);
panel5.top :=round(panel5.top +vy);
// rimbalzo sponde
if panel5.top<0 then vy:=-vy;
if panel5.top>panel1.Height-8 then vy:=-vy;
// urto racchetta
if abs(panel5.left-panel2.Left)<4 then
if (panel5.top>panel2.top) _
and (panel5.top<panel2.top+panel2.Height) then
vx:=-vx;
if abs(panel5.left-panel3.Left)<4 then
if (panel5.top>panel3.top) _
and (panel5.top<panel3.top+panel3.Height) then
vx:=-vx;
// punto
if panel5.left<8 then
begin
p2:=p2+1;
label2.caption:=inttostr(p2);
battuta;
end
else
if panel5.left>panel1.width-8 then
```

Misurare il tempo ➔
Attraverso l'Object Inspector
è possibile impostare gli intervalli di tempo
del componente TTimer.



Le toolbar

I bottoni dell'interfaccia di Delphi sono veri e propri strumenti di programmazione e aiutano a sviluppare più rapidamente qualsiasi applicazione.

```
begin
p1:=p1+1;
label1.caption:=inttostr(p1);
battuta;
end;
```

Si aggiunge ora una procedura per la battuta: dovrà essere eseguita ogni volta che un giocatore segna un punto e comunque all'inizio della partita. La procedura va dichiarata di seguito alle variabili, mentre il codice vero e proprio va scritto in un qualsiasi punto dopo la riga "implementation" nel modo seguente:

```
procedure TForm1.battuta;
begin
vx:=3;
vy:=random(4)-2;
panel5.Top:=random(panel1.Height);
panel5.left:=panel4.Left;
end;
```

Per ultima cosa si inserisce un bottone per avviare una nuova partita, azzerando i punteggi e richiamando la procedura di battuta:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
p1:=0;
p2:=0;
label1.caption:=inttostr(p1);
label2.caption:=inttostr(p2);
battuta;
end;
```

Il programma è ora pronto: compilando ed eseguendo si può provare questo tennis d'epoca. Con un po' di fantasia si possono immaginare diverse variazioni sul tema come per esempio un aumento della velocità della palla a ogni punto, colpi a effetto con rimbalzi casuali e altro ancora.

Esperimento riuscito

Cronometro a parte, l'esperimento di sviluppo rapido può dirsi riuscito; certo, tutto è relativo e chi non ha esperienza con Delphi dovrà riguardare questo esempio con più calma. Non si deve però perdere di vista il notevole risultato ottenuto in poco tempo: un programma semplice ma completo.

In conclusione, usando i diversi mattoncini è davvero possibile sviluppare un'applicazione in modo efficiente e rapido, come promesso; e questo copre la maggior parte delle normali esigenze in ambito aziendale o tecnico.

Se però si devono costruire i propri mattoncini o si deve agire sui meccanismi alla loro base - per esigenze comunque speciali - le cose si complicano un po' e si ritorna, in fin dei conti, alla tecnica di sviluppo tradizionale, nella quale il linguaggio la fa da padrone.