Primi esercizi di Visual Basic

di Francesco Petroni

Questo articolo ha due finalità. La prima è di costituire un complemento della prova del Visual Basic, presentata in questo stesso numero, complemento che serve ad esemplificare praticamente alcuni concetti esposti teoricamente nella prova stessa.
La seconda finalità è quella di fare da introduzione pratica alla programmazione con il Visual Basic.
In una apposita sezione parliamo anche del ToolBook versione 1.5, l'interessante prodotto della Asymetrix, visto (si trattava della versione 1.0) nei numeri 104 e 106 di MC, che per molte caratteristiche assomiglia al Visual Basic, al punto da presentare una certa sovrapposizione di aree di applicabilità

A chi fosse veramente interessato ad avvicinarsi alla programmazione sotto Windows, ovviamente non ci stiamo rivolgendo agli specialisti che già padroneggiano il C e l'SDK, ma agli utenti normali, che non si accontentano di essere solo utilizzatori passivi dei vari pacchetti, consigliamo vivamente di non limitarsi a vedere un solo prodotto (e comunque il primo da studiare potrebbe essere proprio il Visual Basic), ma anche di spingersi su altri prodotti con i quali sia possibile realizzare applicativi sotto Windows.

Uno dei più interessanti visti recentemente è proprio il ToolBook, che, essendo giunto alla versione 1.5, già mostra degli affinamenti significativi. Lo trattiamo in uno specifico riquadro.

Tornando al Visual Basic, si tratta sicuramente di un prodotto molto importante, soprattutto per la sua collocazione, descritta nella prova, intermedia tra prodotti evoluti di programmazione, che richiedono all'utilizzatore numerose e consolidate preconoscenze, e la programmazione eseguita con le funzioni Macro dei prodotti applicativi, troppo condizionata e limitata dal prodotto di partenza.

Il Visual Basic può essere affrontato anche da un utente non molto evoluto e può essere utilizzato per realizzare applicazioni di qualsiasi tipo, a patto, è bene chiarirlo subito, che non siano troppo specializzate o troppo complesse.

Il «taglio» che daremo a questo articolo è assolutamente introduttivo. Ai nostri lettori non richiederemo preconoscenza del Quick Basic, ma solo una preconoscenza, come semplici utilizzatori, di Windows. Pretendiamo però che abbiano letto la prova del Visual Basic presentata qualche pagina fa.

Gli esercizi proposti sono semplicissimi ed ognuno riguarda un solo argomento. Il tempo richiesto per rieseguirli è inferiore ai quindici minuti. Al contrario gli esercizi presenti nel Tutorial del prodotto sono più complessi in quanto trattano più funzionalità insieme, e richiedono conseguentemente un notevole impegno di tempo.

In una generica applicazione Visual Basic, per quanto semplice possa essere, esistono Form, che possono essere viste lato Programmatore e lato Utilizzatore, esistono Control, e ognuno di questi oggetti può comportare uno o più pezzi di programma, che appaiono ciascuno in una finestra. Necessariamente quindi la documentazione di una applicazione realizzata con il Visual Basic appare frammentata.

Ogni esercizio proposto quindi fa riferimento ad una o più illustrazioni, in genere costituite da «Hard Copy» di un «collage» di pezzi di videate, che hanno lo scopo di far comprendere l'esercizio stesso, più di quanto riusciremmo descrivendolo.

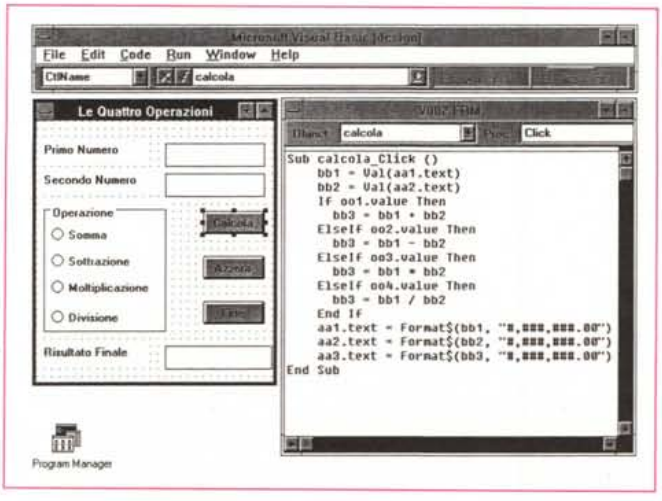


Figura 1 - Esercizi MS Visual Basic - Piccola Calcolatrice.
La prima cosa da fare è quella di impadronirsi delle due filosofie «Object Oriented» e «Event Driven», dopodiché la realizzazione di una qualunque applicazione diventa sicuramente più rapida. Se poi si tratta di un'applicazione semplice, come questa del nostro primo esercizio, il grosso della sua realizzazione consiste nel disegno, vero e proprio, della Form.

I nostri esercizi

Una piccola calcolatrice

Questo primo esercizio consiste in una sola Form, cui abbiamo assegnato il nome «Le Quattro Operazioni», che contiene alcune Label (le scritte fisse), una Frame, che contiene a sua volta quattro Option Button (mutuamente esclusivi) per scegliere quale operazione eseguire, tre Text Box, le prime due per ricevere in input i due operatori e la terza per ricevere in output il risultato, ed infine tre Command Button per mandare in esecuzione rispettivamente il calcolo, l'azzeramento delle due Text Box e l'uscita dal programma.

A fianco alla Form (fig. 1) possiamo vedere il «codice» sottostante il Command Button «Calcola», in cui abbiamo

inserito le istruzioni che interpretano i due valori di Input inseriti nelle Text Box, poi quelle che, a seconda dell' Option Button scelto, eseguono l'operazione e infine quelle per visualizzare sia i due dati iniziali che il risultato finale nelle tre Text Box relative.

Non occorre spiegare ulteriormente il funzionamento dell'applicazione. Ci limitiamo solo ad alcune considerazioni.

L'«Event» che manda in esecuzione l'applicazione è il «Click» semplice sull'«Object» costituito dal «Button», cui abbiamo dato il nome «Calcola».

Le Text Box accettano solo stringhe, per cui sia in fase di loro lettura (bb1=Val(aa1.text)), sia in fase di loro scrittura (aa1.text=..), vanno eseguite delle conversioni da Stringa a Numero e viceversa.

Occorre abituarsi subito alla gestione degli Object, ad esempio una Text Box ha tra le numerose Properties una Property «CntName» che va usata come suo identificativo nei programmi (es. aa1) e una Property «Text» che contiene il valore della Box.

Le istruzioni di conversione da numero a stringa sono due: Str$ e Format$ che, come vedremo meglio dopo, permette di definire tutti i formati (per numeri, date e ore) standard in tutti i prodotti Windows.

La Frame serve per suddividere in gruppi gli Option Buttons. Nella nostra applicazione, in cui c'è un solo Gruppo, la Frame poteva essere omessa.

Il Command Button, da noi non documentato, «Azzera», contiene solo istruzioni del tipo aa1.text=«», che servono per svuotare le Text Box del loro contenuto, mentre quello «Fine», contiene solo l'istruzione End.

Poiché, a meno di differenti specifiche, ogni Frame dispone per default del Bottone per l'attivazione del Menu di Controllo, possiamo in ogni caso chiudere l'applicazione attraverso l'opzione Chiudi sempre presente in tale menu.

Se «compiliamo» la nostra applicazione, con l'istruzione Make Exe del menu File, un file eseguibile risulta di circa 6 kbyte (comprensivo della Icona di 766 byte, facilmente assegnabile in quanto è una property della Form). L'eseguibile può essere utilizzato anche su altre macchine a patto che su queste si porti anche il file VBRUN100.DLL.

Variante Scroll Bar

Si tratta dell'applicazione di prima leggermente modificata.

Vogliamo che l'input dei due operatori sia eseguito agendo su due Scroll Bar orizzontali (fig. 2).

Anche la definizione di una Scroll Bar, cui collegare la variazione di una qualsiasi variabile, diventa una operazione semplicissima. Basta tracciarla, definire il valore minimo, quello massimo e quello di step e la Barra è pronta per l'uso. Il valore impostato sulla barra è subito disponibile, per qualsiasi utilizzo, nella property nn2.value.

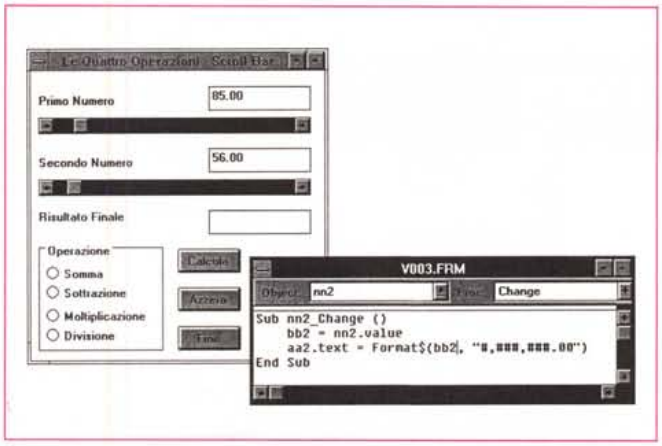


Figura 2 - Esercizi MS Visual Basic - Utilizzo di Scroll Bar.
Anche la definizione di una Scroll Bar, cui collegare la variazione di una qualsiasi variabile, diventa una operazione semplicissima. Basta tracciarla, definire il valore minimo, quello massimo e quello di step e la Barra è pronta per l'uso. Nel nostro esercizio deleghiamo alla Barra l'input dei due valori coinvolti nel calcolo.

Come gestire una List Box

Anche la gestione di una List Box, è semplicissima. L'unico inconveniente, se così lo vogliamo definire, è che il contenuto della List Box, non è (come ci si poteva aspettare) una sua «Property».

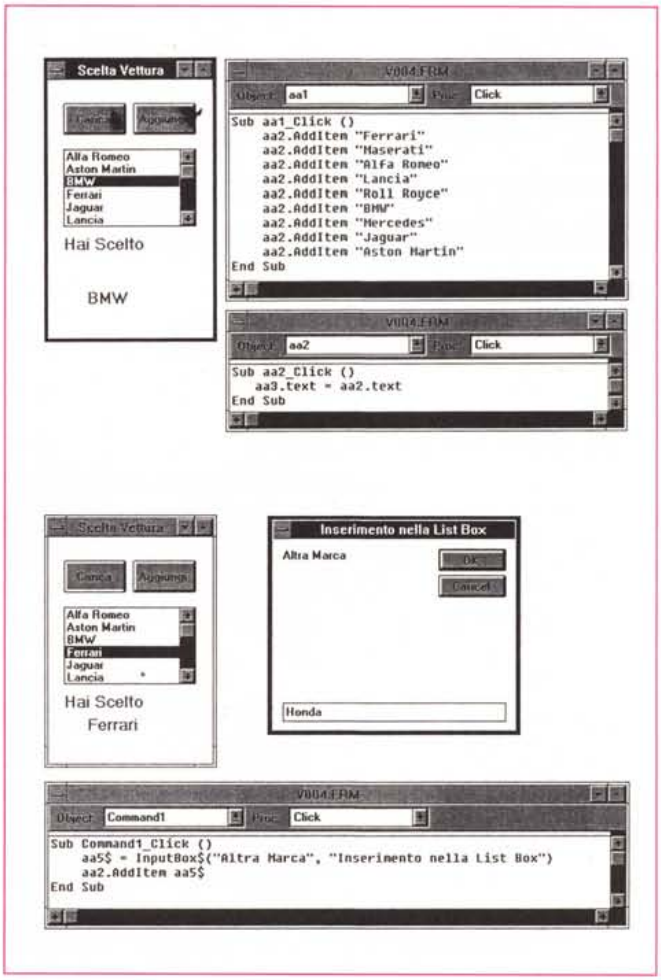


Figure 3, 4 - Esercizi MS Visual Basic - Gestione di un List Box.
Il contenuto di una List Box si definisce da programma, con una semplice ed apposita istruzione. La sua alimentazione può essere eseguita una tantum, ad esempio nel «pezzo» di programma mandato in esecuzione al momento del caricamento della Form, oppure al volo, quando un evento lo richieda. Nella seconda illustrazione mostriamo anche l'istruzione InputBox$ che manda in esecuzione una semplicissima Dialog Box, che serve per digitare un solo valore. Includiamo poi tale valore nella List Box.

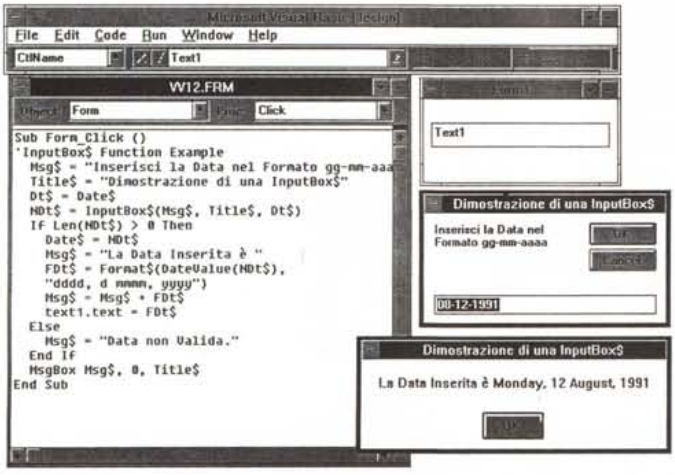


Figura 5 - Esercizi MS Visual Basic - Uso di una funzione InputBox$.
Nel vecchio Basic la principale istruzione per gestire la digitazione dei dati era la Input. Nel Visual Basic la stessa si è trasformata nella funzione InputBox$, che produce «gratuitamente» una Dialog Box, per la quale si possono definire una serie di parametri e nella quale l'utilizzatore digita direttamente la stringa di risposta.

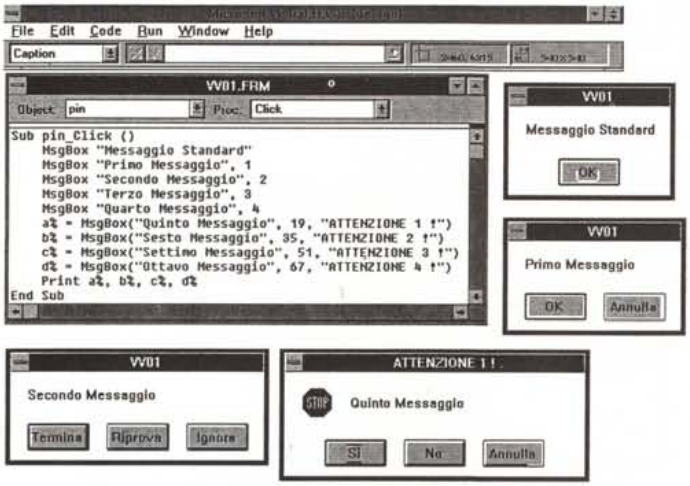


Figura 6 - Esercizi MS Visual Basic - Campionario di MsgBox$.
Altra vecchia conoscenza Windows è la Message Box, la finestra più semplice, che può essere utilizzata sia per lanciare un Messaggio di tipo Warning, sia per ricevere una risposta di tipo OK / Annulla, sia per una risposta del tipo OK/Annulla/Cancella. In pratica la funzione restituisce un valore numerico dipendente dal tasto che è stato cliccato.

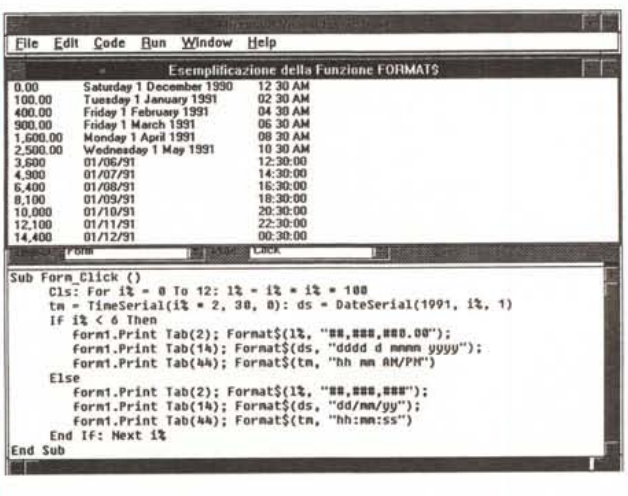


Figura 7 - Esercizi MS Visual Basic - Campionario di Formati.
In questo caso utilizziamo una Form «vuota». L'Evento che manda in esecuzione il programma può essere il Form_Load. In altre parole il programma, che mostra una serie di varianti del comando Format$, che utilizza ovviamente i formati standard Windows (quelli anche di Excel e di WinWord, per intenderci), parte direttamente al caricamento del programma.

In altre parole la List Box si può alimentare solo da programma utilizzando l'apposita istruzione «aa2.AddItem», in cui ovviamente «aa2» è il CtlName assegnato alla List Box stessa.

Se quindi la List Box va alimentata da programma occorre decidere come e quando tale operazione va fatta. Se si tratta di una Lista «statica», può essere caricata al lancio dell'applicazione, ad esempio al verificarsi dell'evento Form.Load. Se si tratta di una lista lunga, il suo contenuto può essere «piazzato» in un file, letto e trasferito con un'istruzione di assegnazione nella lista, ecc.

Come al solito è il programmatore che deve decidere

Nel nostro esercizio (fig. 3), anche in questo caso autocommentato, abbiamo inserito due Command Button, il primo (aa1) che carica la List Box e il secondo (aa2) che legge il valore scelto nella List Box e lo piazza in una Text Box (aa3).

Nella figura 4 abbiamo inserito un'altra funzionalità che permette, attraverso una funzione InputBox$, che vedremo tra un può, di aggiungere un ulteriore elemento alla Lista (aa2.AddItem aa5$).

La List Box dispone di varie altre Properties, ad esempio quella per stabilirne l'ordine (può essere alfabetico, oppure imposto da programma, ecc.), quella per leggere il valore scelto, sia come Text che come numero progressivo.

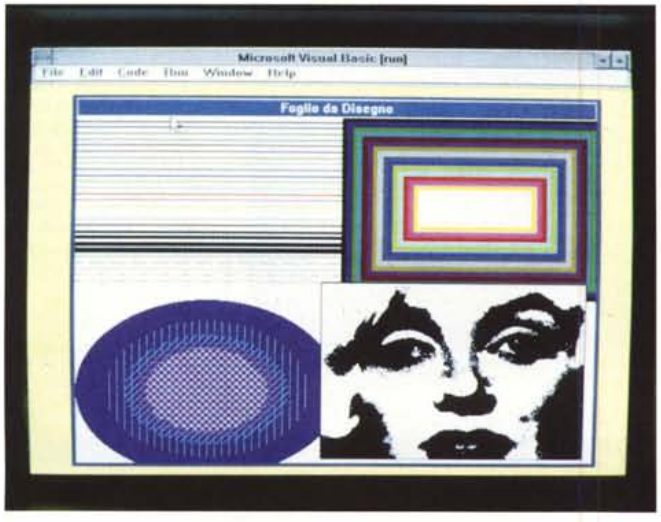
Approfittiamo dell'occasione per citare la presenza di un altro tipo di List Box, la Combo Box, che combina le funzioni di List Box, con quelle di Text Box. È insomma possibile sia scrivere un elemento, sia sceglierlo tra quelli presenti nella lista, con tutte le varianti del caso. La più immediata è quella che l'elemento digitato a mano, perché non presente nella Lista, gli si può aggiungere.

Uso delle funzioni InputBox$ e MsgBox$

Nel vecchio Basic la principale istruzione per gestire la digitazione dei dati è la rudimentale Input.

Nel Visual Basic la stessa si è trasformata nella funzione a$=InputBox$ che produce «gratuitamente» una Dialog Box nella quale l'utilizzatore digita direttamente la stringa di risposta.

I parametri definibili per tale Dialog Box sono il Titolo, che appare sul bordo superiore della Window, il Messaggio, che appare al suo interno, l'even-



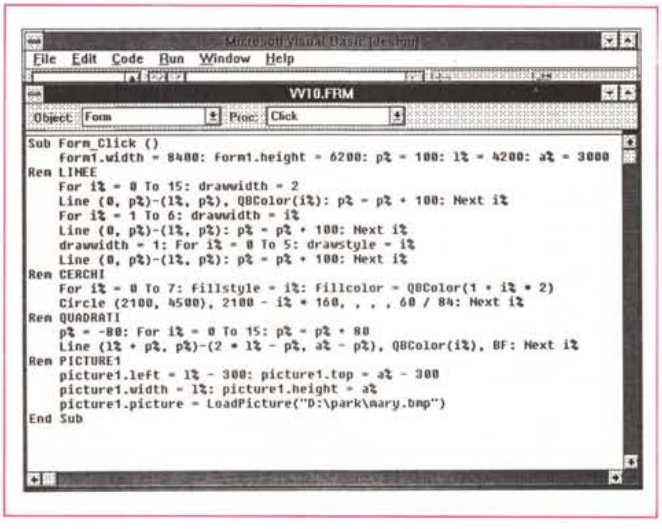


Figure 8, 9 - MS Visual Basic - La Form come Foglio di disegno. Il Visual Basic dispone di una serie di istruzioni grafiche di tracciamento analoghe a quelle presenti nel Quick Basic. Variano invece quelle che servono per definire il tipo di videata, che sotto Windows è unico, definito in fase di Setup, e quelle per la gestione delle Windows (vecchia istruzione Quick Basic) che ora sotto Windows non ha più senso. Le istruzioni di tracciamento operano o all'interno della Form Attiva o all'interno di un Control di tipo Picture.

tuale valore di Default che la variabile da digitare deve assumere inizialmente, e infine la posizione che la Box deve prendere sul video.

I tasti OK e Cancel e le Icone sul bordo della Box sono di Default e non sono modificabili.

Tutto il resto, ad esempio l'eventuale controllo da eseguire sul dato immesso, va risolto dal programmatore nel programma che richiama la InputBox$.

Altra vecchia conoscenza Windows è la Message Box (fig. 6). In pratica una InputBox$ di tipo più semplice che non dispone di una Text Box in cui digitare ma solo di Bottoni.

Può essere utilizzata sia per lanciare un Messaggio di tipo Warning (un bottone), sia per ricevere una risposta di tipo OK / Annulla (due bottoni), sia per una risposta del tipo OK/Annulla/Cancella (tre bottoni).

In pratica la funzione restituisce un valore numerico dipendente dal tasto che è stato cliccato. Questo valore può quindi essere usato per attivare o meno specifiche routine del programma.

Piccolo campionario di formati

In questo caso utilizziamo una Form «vuota».

L'Evento che manda in esecuzione il programma può essere il Form_Load oppure il Form_Click (un Click sulla Form vuota).

Il nostro scopo è quello di verificare l'istruzione Format$ che converte un numero, oppure una data, oppure un orario, in una stringa secondo una «picture» che segue le regole già vigenti sotto Windows (ben note a chi usa Excel, WinWord, ecc.).

Per visualizzare i dati abbiamo utilizzato la rudimentale istruzione Print che non permette di definire direttamente l'allineamento a destra o l'allineamento numerico. Se la visualizzazione avvenisse, come più naturale, in una Text Box, basterebbe definirne la property «Alignment».

L'istruzione Print ha numerose altre

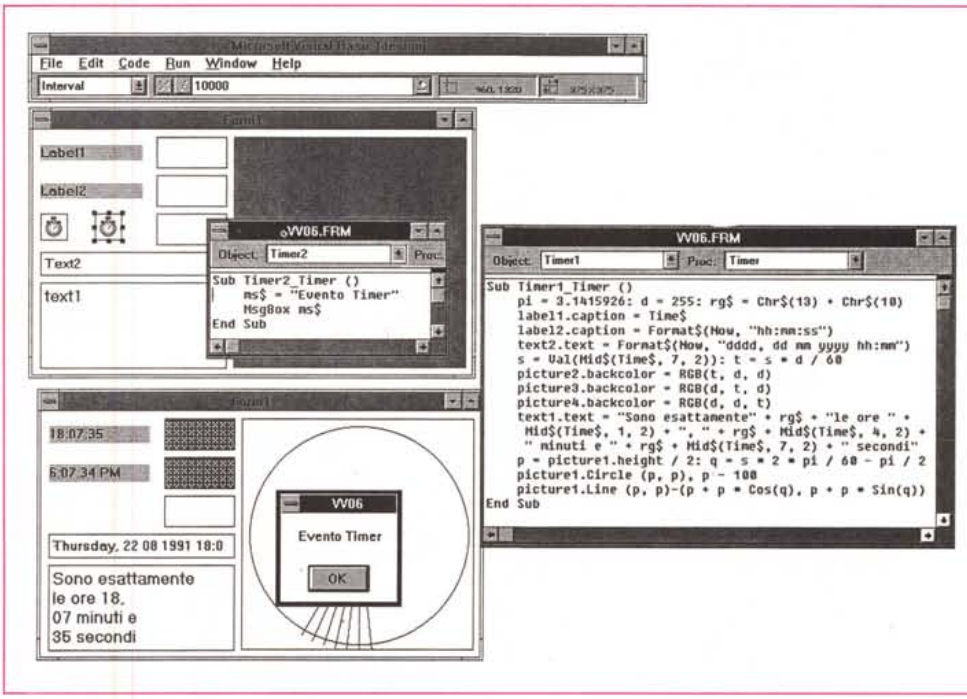


Figura 10 - MS Visual Basic - Utilizzo del Timer.
L'Icona Timer permette di definire un Evento regolato da un orologio. Una delle property del Timer è l'intervallo che serve ad indicare ogni quanto tempo si verifica l'Evento Timer. Nel programma va poi ovviamente definita una routine che viene eseguita al verificarsi di tale evento.

«varianti» per rispondere anche alla necessità di stampare direttamente Form o Object di contenuto Grafico.

Le istruzioni grafiche sul Foglio di disegno costituito dalla Form

Rispetto al Quick il Visual Basic «perde» tutte le istruzioni di definizione della Window, ora sostituita in tutto e per tutto dalla Form, e «acquista» le istruzioni che permettono di inserire, in un Control di tipo Picture, una immagine già disponibile su file (ne abbiamo parlato nella prova).

Del tutto nuovo è il comando (in realtà è una property dell'Object) che permette di scegliere l'unità di misura, scelta tra le 8 disponibili. Quella usata nell'esempio è quella standard, che «ragiona» in twip (1440 twip per pollice). Si può però lavorare in punti, pollici, centimetri, millimetri, pixel.

I comandi di tracciamento rimangono invece pressoché invariati e permettono di tracciare poche «primitive»: punti, linee, rettangoli, archi e cerchi. Altre istruzioni permettono di definire colore, spessore e tratteggio delle linee, oppure colore e retinatura delle figure piene.

La gestione dei colori è alla Windows, in cui il colore è definito tramite le sue tre componenti (l'istruzione è RGB(rosso,verde,blue)). È anche possibile definire i colori «alla QBasic» ed in questo caso l'istruzione diventa QBColor(c), in cui C va da 0 a 15.

Nella figura vediamo una videata grafica con quattro componenti e il listato, che non commentiamo, con cui è stata generata.

Il Timer

Una property del Control Timer è l'intervallo che serve ad indicare ogni quanto tempo si verifica l'Evento Timer. Nel programma sottostante va poi ovviamente definita una routine che viene eseguita al verificarsi dell'Evento.

Il Timer, la cui Icona appare solo in fase di realizzazione della Form, mentre in fase di esecuzione scompare, può servire sia per creare eventi periodici, ad esempio per creare un Orologio personalizzato, oppure per gestire un Evento «una tantum» (fig. 10).

Cosa collegare all'evento Timer dipende dal programmatore. Noi lo abbiamo collegato ad una serie di orologi digitali, ad un orologio analogico (creando graficamente una specie di lancetta), ad un orologio in cui con il passare dei secondi cambiano tre colori (che però nell'Hard Copy non vedete) di tre Control di tipo Picture Box.

Nel nostro esempio abbiamo usato le istruzioni di sistema Time$ (dà l'ora in formato stringa) e Now (dà la data e l'ora in formato numero progressivo) e l'istruzione Format$, ecc. Tali istruzioni vengon mandate in esecuzione ogni volta che si verifica l'evento Timer1_Timer.

La principale caratteristica dell'evento Timer è quella di essere eseguito indipendentemente da cosa succede nel resto del programma, e nel resto di Windows.

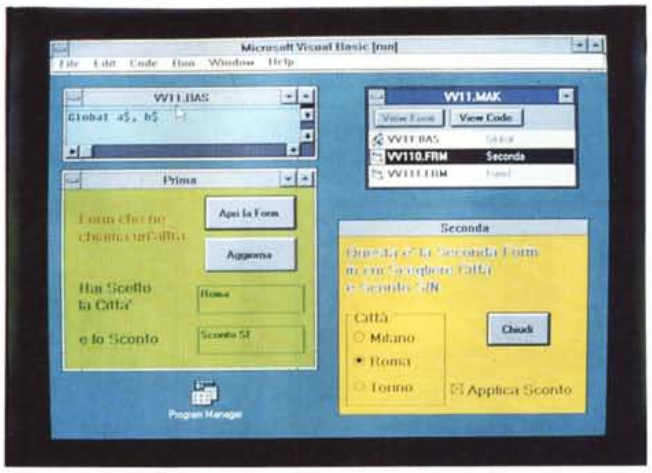


Figura 11 - MS Visual Basic - Generazione «al volo» di una Form. Questo esercizio serve per sperimentare le istruzioni che permettono di attivare al volo una Form, nella quale ad esempio eseguire una serie di digitazioni, e i cui risultati possano esser usati nella Form di partenza.

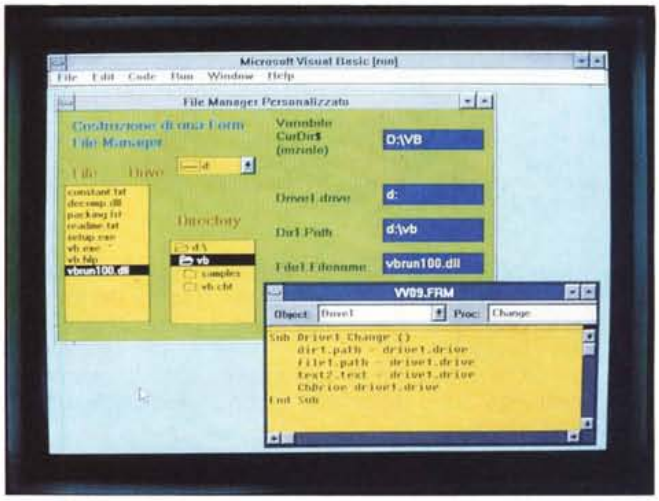


Figura 12 - MS Visual Basic - Utilizzo delle Icone di File Manager. Tre delle quattordici Icone della ToolBox servono per costruire una Form con funzioni di File Manager, utile in un'applicazione in cui l'utilizzatore possa scegliere Drive, Directory e File su cui lavorare. Le funzionalità attivabili sono le stesse presenti in un File Manager standard, ad esempio è possibile impostare le estensioni di default, è possibile gestire gli errori di lettura, è possibile gestire il doppio Click, il Drag e così via.

Generazione «al volo» di una Form

Una applicazione può svilupparsi su più Form. Ogni Form ha una serie di property che permettono di definirne il tipo di cornice, la posizione e la dimensione. È possibile scegliere se dotarla o meno delle barre di scorrimento oppure delle Icone di massimizzazione, di iconizzazione e di ripristino, e di quella che attiva il Menu di Controllo. Nella figura 11 si può notare come la nostra seconda Form sia stata molto alleggerita.

Tutte le property (non solo quelle relative alle Form) possono comunque essere definite lavorando sulla Property Bar, oppure utilizzando istruzioni di programmazione. In altre parole sarebbe possibile anche realizzare una Form che si muove «da sola» sul video o che si restringe.

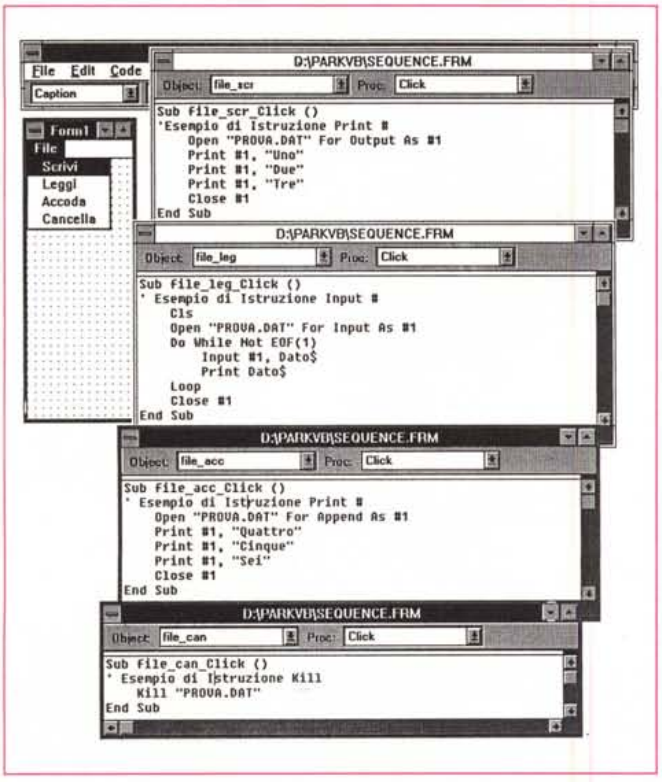
Esistono inoltre istruzioni per caricare, attivare, nascondere, scaricare (il tutto però via programma) le varie Form.

Come forse ricorderà chi ha letto la Prova di qualche pagina le variabili possono essere definite a livello di Form. Se le variabili però vanno passate da una Form all'altra occorre definirle (come Globali nella sezione Global) a livello di applicazione.

L'esercizio di figura 11 serve proprio per sperimentare le istruzioni che permettono di attivare al volo una Form, nella quale ad esempio eseguire una serie di digitazioni e i cui risultati possano ritornare nella Form di partenza.

Le icone per costruire un proprio File Manager

Tre delle quattordici Icone del ToolBox possono servire per gestire il rap-




```
Type rt                         ' modulo Globale
    codice As String * 4        ' definizione della rt
    nome As String * 12
    importo As Currency
End Type

Sub Command1_Click ()           ' bottone Inserisci
    Dim rv As rt                ' definizione della rv
    rv.codice = text1.text
    rv.nome = text2.text
    rv.importo = Val(text3.text)
    Open "PROVA.DAT" For Random As 1 Len = 24
        rl = Len(rv)            ' lunghezza del record
        lr = LOF(1)             ' lunghezza del file
        nr = lr / rl            ' numero del record
        Put 1, nr + 1, rv
    Close 1
    text1.text = "": text2.text = "": text3.text = ""
    text4.text = Str$(nr + 1)
End Sub

Sub Command2_Click ()           ' bottone Indica il Numero
    Dim rv As rt
    nv$ = InputBox$("Indica il Numero di Record Voluto")
    nn = Val(nv$)
    Open "PROVA.DAT" For Random As 1 Len = 24
        Get 1, nn, rv
    Close 1
    text1.text = rv.codice
    text2.text = rv.nome
    text3.text = Str$(rv.importo)
    text4.text = nv$
End Sub

Sub Command3_Click ()           ' bottone Cerca la Sigla
    Dim rv As rt
    ns$ = InputBox$("Indica il Codice del Record Voluto")
    Open "PROVA.DAT" For Random As 1 Len = 24
    rl = Len(rv): lr = LOF(1): nr = LOF(1) / rl
    For i% = 1 To nr
        Get 1, i%, rv
        If rv.codice = ns$ Then
            Exit For
        End If
    Next i%
    Close 1
    text1.text = rv.codice
    text2.text = rv.nome
    text3.text = Str$(rv.importo)
    text4.text = Str$(i%)
End Sub
```

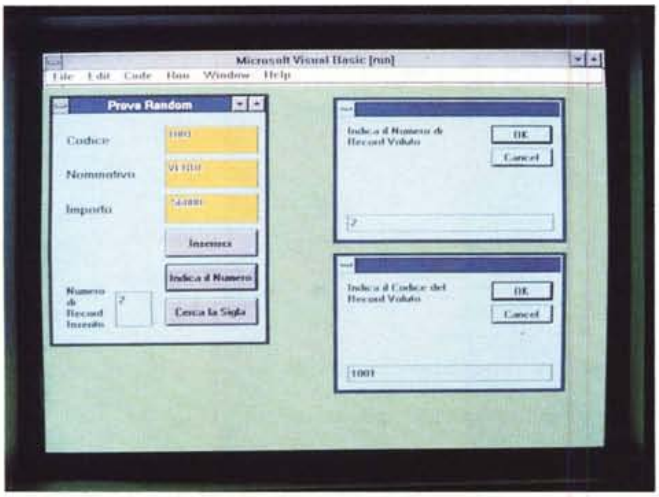


Figure 13, 14, 15 - MS Visual Basic - Funzioni di I/O.
Il Visual Basic dispone delle stesse funzioni di I/O già presenti nei vecchi Basic. È possibile gestire file Sequenziali e Random. È chiaro che l'applicazione più immediata è quella che consiste nel riempire delle Text Box con i dati letti dal File (o viceversa, ovviamente).

porto tra l'applicazione e le unità di memoria di massa (fig. 12). Risultano quindi molto utili quando l'applicazione possa permettere la scelta, da parte dell'utente, di un file esterno.

Nel realizzare le proprie funzioni di accesso ai file si può raggiungere un livello di raffinatezza pari a quello di un File Manager evoluto, come ad esempio quello in dotazione in Windows stesso.

Il nostro esercizio si limita però a gestire la modifica della Path iniziale, disponibile nella funzione di sistema CurDir$.

La lettura delle scelte eseguite (ad esempio se si clicca su drive A: quando il drive iniziale era C:) si effettua tramite le funzioni «drive1.drive» in cui drive1 è il nome dell'Object e drive è la sua attuale property. Per modificare conseguentemente la Path corrente si usano istruzioni di assegnazione del tipo «dir1.path=».

Una delle property del File List Box è la Pattern, con la quale è possibile stabilire l'estensione voluta per i file.

Utilizzi più evoluti, che rimandiamo ad una prossima occasione, sono quelli che permettono di gestire il Doppio Click (ad esempio per attivare il caricamento diretto del file) o il Drag del file, ad esempio per trasferirlo direttamente all'interno di un Control di tipo Grafico. Si tratta di eventi un po' più complessi che riguardano due Control.

La funzione di Input Output

Per quanto riguarda le funzioni per la gestione di files esterni contenenti dati vanno dette innanzitutto due cose. La prima è che queste non hanno nulla a che vedere con gli aspetti grafici del Visual Basic. La seconda è che le istruzioni disponibili sono del tutto analoghe a quelle già presenti nel Quick Basic.

Sono quindi gestibili file con accesso Sequenziale o file con accesso Random.

Nella figura 13 vediamo un esercizietto in cui, tramite un semplicissimo Menu (che non spieghiamo), vengono attivate solo quattro voci che eseguono le quattro operazioni possibili su un file Sequenziale. La prima opzione Scrive nel file «Uno», «Due», «Tre».

La struttura dei comandi è semplicissima:

```
OPEN <file> FOR OUTPUT AS #1
PRINT #1, <record>
CLOSE #1
```

Se eseguita più volte riscrive lo stesso file. #1 rappresenta è il numero progressivo del file dati aperto.

La seconda opzione Legge il File. Con l'istruzione:

```
OPEN <file> FOR INPUT AS #1
WHILE NOT EOF(1)
```

```
INPUT #1, <record>
LOOP
CLOSE #1
```

si genera un ciclo che legge tutti i record, supposto che non se ne conosca il numero e che non si voglia incorrere in errori alla fine dei dati.

La terza opzione Accoda scrive, alla fine del file che già contiene i tre dati, i tre nuovi record «Quattro»,«Cinque» e «Sei». Se eseguita più volte allunga il file. Le istruzioni diventano:

```
OPEN <file> FOR APPEND AS #1
PRINT #1, <record>
CLOSE #1
```

L'ultima opzione Cancella serve per cancellare file e suo contenuto.

Il limite del file sequenziali è ben noto, risale ai tempi... del nastro (quando questo veniva usato come unità di memoria di massa). Non si può trattare un Record per volta, ma vanno letti tutti i Record insieme. Nei file di tipo Random invece ad ogni record viene assegnato un numero progressivo che può essere utilizzato per richiamare solo quel record, senza «scomodare» tutti gli altri.

Il tutto si basa su una precisa lunghezza di Record, la cui composizione (struttura) va dichiarata addirittura nel modulo Globale, per cui si da modo al Visual Basic di poter determinare la posizione precisa del Record voluto.

In pratica nel modulo Globale si definisce una unica variabile, nel nostro ca-

Asymetrix ToolBook 1.5

Negli stessi giorni in cui è arrivato in Redazione il Visual Basic è arrivata anche la nuova versione del ToolBook, prodotto dell'Asymetrix, casa molto «vicina» alla Microsoft, già presentato diffusamente su MC in due articoli apparsi sui numeri 104 e 106.

Approfittiamo dell'articolo sul Visual Basic per informarvi rapidamente, in questo riquadro, anche sulle novità presenti nel ToolBook 1.5.

Lo facciamo perché riteniamo che il lettore interessato alle problematiche di programmazione sotto Windows non debba limitarsi alla studio di un solo linguaggio, ma debba allargare le sue conoscenze anche a prodotti che, magari attraverso altre tecniche, permettano di fare pressapoco le stesse cose.

Tra l'altro i due prodotti, che sfruttano ambedue l'accattivante «estetica» Windows, si assomigliano tremendamente (fig. 1).

Anche per il ToolBook, come del resto per il Visual Basic, esiste o esisterà una versione OS/2.

Precisiamo però per chiarezza quattro differenze fondamentali tra i due prodotti, differenze che ne delineano un po' più chiaramente i rispettivi ambiti applicativi.

Con Visual Basic si realizzano delle applicazioni Windows vere e proprie (desinenza EXE), che possono girare su qualsiasi macchina in cui sia stata semplicemente installata una specifica libreria DLL. Con il ToolBook si realizzano invece applicazioni ToolBook (desinenza TBK) eseguibili solo in macchine in cui sia presente o il ToolBook stesso o il suo «runtime».

ToolBook è un prodotto «flat-file», ovvero tutta l'applicazione e i suoi dati risiedono in un unico file, quello citato che ha desinenza TBK. Visual Basic, in quanto linguaggio a tutti gli effetti, dispone anche di istruzioni I/O, che gli permettono di dialogare con files dati esterni, anche di enormi dimensioni.

La filosofia del ToolBook è orientata evidentemente al Book, e alle sue Pages, mentre nel Visual Basic la struttura principale è la Form. Il primo quindi è più adatto per applicazioni «a pagine», come ad esempio quelle Ipertestuali o Ipermediali (fig. 2), vero «cavallo di battaglia» del ToolBook, mentre il secondo non presenta particolari preferenze applicative.

La programmazione vera e propria nel ToolBook è delegata ad un linguaggio di tipo Script, linguaggio che somiglia più alla lingua (inglese) parlata che ad un liguaggio tradizionale di programmazione, quale invece è il Visual Basic.

Passando alle analogie, la prima è l'approccio Screen Painter, che fa ormai assomigliare qualsiasi prodotto sotto Windows ad un prodotto Grafico, in cui sostanzialmente si disegna (ma in pratica si programma) sul video.

La seconda analogia è la disponibilità di

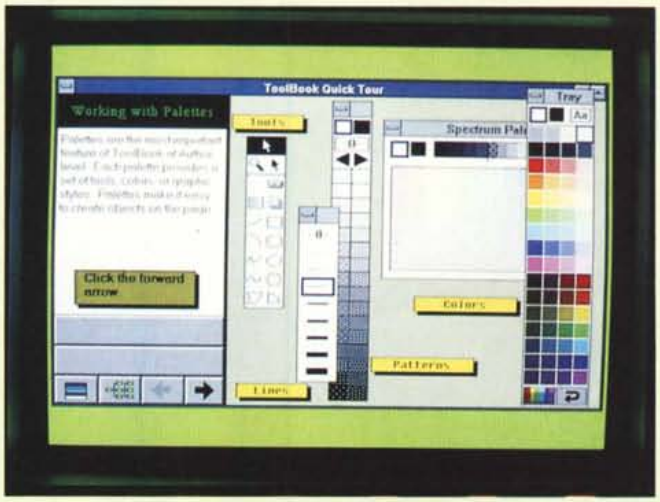


Figura 1 - Asymetrix ToolBook 1.5 - Approccio Screen Painter. Nei numeri 104 e 106 di MC abbiamo pubblicato due articoli dedicati al ToolBook, l'innovativo prodotto dell'Asymetrix, che presenta numerosissime analogie con il Visual Basic, soprattutto per il suo approccio di tipo «Screen Painter». Del ToolBook abbiamo in questi giorni ricevuto la versione 1.5 (numero insolito, più di 1.1 ma meno di 2.0).

Figura 2 - Asymetrix ToolBook 1.5 - HyperText and Hypermedia. La filosofia del ToolBook, legata al concetto di Book, di Pages e di Object, lo rende adatto soprattutto ad applicazioni IperTestuali e/o IperMediali. Tra il materiale messo a disposizione con la versione 1.5 c'è anche un catalogo di applicazioni sviluppate, in USA e Canada, da varie case Software.

so si chiama «rt», che comprende tutti i Campi in cui è strutturato il Record.

Il nostro esercizio (di cui vediamo l'aspetto esteriore in figura 14 e quattro listati in figura 15) consiste in un semplice schedarietto, in pratica una Form con tre Text Box, contenenti i tre campi del Record (Codice, Nome e Importo), e tre bottoni:

Inserisci (Command1) che trasferisce nelle variabili rv.* il contenuto delle Text Box e gli assegna un numero progressivo, gestito tramite la funzione Len(rv) che fornisce la lunghezza del Record e Lof(1) che fornisce la lunghezza del file.

Indica il Numero, che, dato un numero richiesto tramite una InputBox$, legge direttamente il relativo Record.

Abbiamo poi inserito il Command Button Indica il Sigla, che cerca, leggendo uno per uno tutti i Record, quello che corrisponde ad un Codice voluto.

Chiariamo, per chi avesse per caso visto prodotti tipo dBASE o Paradox e non conoscesse invece i linguaggi, che il concetto di Campo Chiave, vero e proprio pilastro nei prodotti DBMS, nei linguaggi non esiste.

Se volete che uno dei campi del vostro archivio sia un campo chiave dovete costruire voi un algoritmo che metta in relazione il contenuto del campo con il numero del Record. Questo lo dovete fare voi, con la programmazione.

MC

svariati Tool, con i quali anche l'impostazione delle proprietà, e delle caratteristiche dei vari elementi disegnati, diventa un'attività grafica. In figura 3 vediamo un collage di Finestre di Dialogo relative al nuovo strumento Dialog Box Editor.

In figura 4 possiamo invece vedere una videata presa da un applicativo ToolBook, anch'esso disponibile con il materiale, il cui scopo è di mettere a confronto, anche eseguendo dei Test di velocità, differenti tecniche di programmazione. Va da sé che i suggerimenti forniti hanno valore assoluto, valgono sia per il ToolBook, che per il Visual Basic, in cui sono praticabili le stesse tecniche.

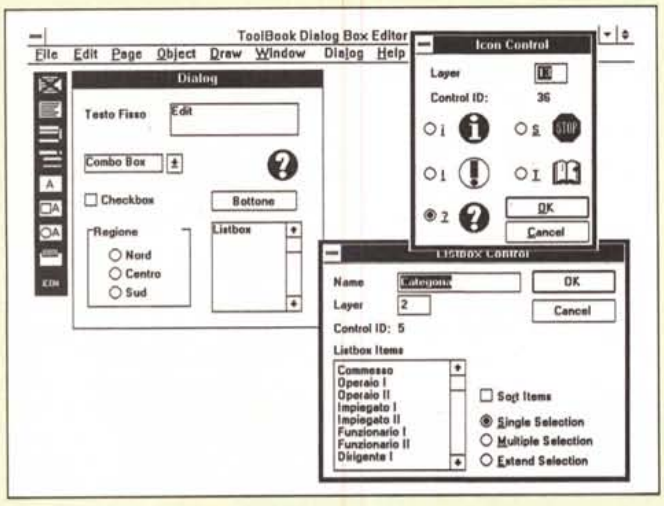


Figura 3 - Asymetrix ToolBook 1.5 - Dialog Box Editor.
Detto dell'approccio Screen Painter, che nel ToolBook serve per disegnare la Page del Book e nel Visual Basic per disegnare la singola Form, va citata un'altra analogia, di tipo eminentemente operativo, costituita dalle modalità di definizione degli Object, presenti rispettivametne nella Page e nella Form. Si tratta ovviamente dei classici Oggetti Windows.

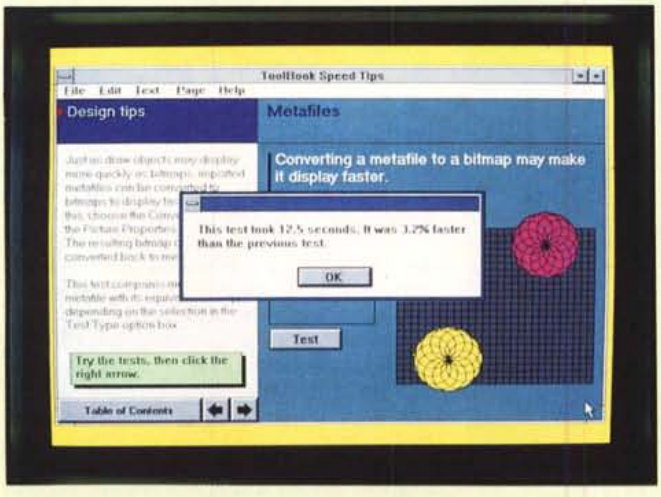


Figura 4 - Asymetrix ToolBook 1.5 - Prove di Velocità.
Interessantissima è l'applicazione ToolBook, inserita nel materiale, che serve a verificare praticamente la differenza, in termini di prestazioni velocistiche, tra varie soluzioni alternative dello stesso problema applicativo. le indicazioni che ne derivano hanno un interesse generale, valgono insomma non solo per le applicazioni ToolBook, ma per tutte le applicazioni Windows, Visual Basic incluso.

Le novità del ToolBook 1.5

La versione si chiama 1.5, ed è un numero più grande di 1.1, in quanto le novità sono numerose, ma è più piccolo di 2.0, in quanto non è stata aggiornata la manualistica, che presenta semplicemente una «Release Notes», opusculetto con le cose in più. Le quali sono :

— miglioramento delle prestazioni velocistiche, aumento della dimensione massima dei programmi, incremento della qualità delle stampe,
— nuove tipologie di «object» definibili, in particolare Push Buttons e List Box,
— nuove istruzioni nel linguaggio Open Script,
— inserimento di funzioni di NetWorking, a protezione dei files ToolBook condivisi,
— possibilità di importare nuove tipologie di formati grafici, (letti attraverso alcuni convertitori) anche di tipo vettoriale,
— implementazione dei comandi per la gestione di canali DDE,
— «traduzione» della messagistica Windows per poterla incorporare nell'applicazione ToolBook,
— utility Dialog Box Editor con la quale si confeziona una Dialog Box, incorporabile nell'applicazione TBK.

Oltre a questi miglioramenti interni vanno citate le nuove ToolBook Ideas, applicazioni da saccheggiare non solo come Idee, ma anche come Books già funzionanti, da convertire a seconda delle proprie esigenze, e tra queste una applicazione dBASE Reader, che serve per dimostrare come dal ToolBook si possono anche gestire, utilizzando una specifica libreria DLL, archivi in formato DBF.

Questo ad ulteriore dimostrazione del fatto che un applicativo Windows, che permetta lo sfruttamento delle tecniche DDE, e che disponga di istruzioni in grado di attivare ed utilizzare funzioni presenti in librerie esterne di tipo DLL, è in ogni caso un prodotto «aperto» i cui confini in termini di aree di applicabilità sono assolutamente indefiniti, anzi si possono estendere nelle più imprevedibili direzioni.

MC