TECNICHE

Help Compiler 4.0

Diamo uno sguardo alle nuove potenzialità offerte dall'Help Compiler Workshop fornito a corredo di Visual Basic 4.0

di Roberto Scano

Con Windows 95 è finalmente arrivata una nuova e più potente versione di Win Help; quasi contemporaneamente, è stato rilasciato Visual Basic 4.0 che nella versione Professional e Enterprise include il file HCW.EXE nella directory \TOOLS\HCW. Si tratta dell'Help Compiler Workshop, un programma che semplifica notevolmente la vita del programmatore nella creazione dei file HLP, permettendogli la modifica del file di progetto HPJ e dei file "contents" CNT, il test del progetto e il pieno controllo sul processo di compilazione. Help Compiler Workshop raccoglie le informazioni dal file HPJ e combina i file RTF, le bitmap ecc., in un unico file HLP. Il programma è composto dai seguenti file:

HCW.EXE, HCRTF.EXE	Help Compiler Workshop
SHED.EXE	Hotspot Editor ver. 2.0
MRBC.EXE	Multi-Resolution Bitmap Compiler ver. 1.1

Ciò che colpisce è Help Workshop che ha una nuova interfaccia grafica, grazie alla quale è possibile modificare facilmente le impostazioni. Help Workshop può leggere i file di progetto HPJ creati con la versione 3.x di Winhelp, ma i vecchi progetti, una volta salvati nel nuovo formato, non potranno essere processati dalle precedenti versioni di HC. Analizziamo ora le principali caratteristiche del nuovo Winhelp.

MACRO

Nel precedente numero di VBJ era presente un mio articolo che descriveva l'utilizzo delle macro più comuni dell'Help Compiler. Questa nuova versione ha potenziato la raccolta di macro esistenti offrendo, ad esempio, la possibilità di richiamare una proprietà dal pannello di controllo o di controllare se un file esiste. Con il nuovo Help Compiler non vi sono praticamente limiti al numero delle macro utilizzabili e le macro negli hotspot possono utilizzare più di 4096 caratteri. Il programma Help Workshop modifica automaticamente i nomi delle macro trasformandoli nella loro forma breve: ad esempio, se utilizziamo la macro *CreateButton*, Help Workshop la tradurrà automaticamente in *CB*. Un'altra novità è la possibilità di omettere i parametri nella macro se questi sono zero oppure una stringa nulla. Un esempio: per aggiungere un pulsante che richiami la calcolatrice di Windows, prima era necessario utilizzare l'istruzione

```
CreateButton(btn_calc,'&Calcolatrice',
    "ExecProgram(`calc', 0)')
```

mentre ora si può scrivere

```
CreateButton('btn_calc','&Calcolatrice',
    ExecFile(calc.exe ))
```

dove si utilizza inoltre la macro ExecFile in luogo della vecchia ExecProgram. Analizziamo ora le nuove macro suddividendole per categorie di utilizzo.

MACRO DEI PULSANTI

Per quanto riguarda i pulsanti, sono disponibili tre nuove macro. ChangeEnable (button-ID, button-macro) permette di assegnare una macro ad un pulsante inattivo e lo rende attivo. L'utilizzo di questa macro è equivalente all'utilizzo di due macro esistenti nella vecchia versione, ChangeButtonBinding ed EnableButton. Il parametro *button-ID* rappresenta il nome che Winhelp utilizza per identificare il pulsante, definito dalla macro CreateButton; *button-macro* è il nome della macro definita dall'utente o la macro di Winhelp che viene eseguita alla pressione del pulsante. Se nel nostro file di guida abbiamo un pulsante inattivo identificato con BTN_PROVA, e desideriamo renderlo attivo e collegarlo ad una macro About() per visionare la finestra di informazioni, dobbiamo introdurre la seguente riga nella sezione [CONFIG] del file HPJ:

```
ChangeEnable (BTN_PROVA, About())
```

Le altre nuove macro sono Finder() , che visualizza la finestra di ricerca testo con l'ultima ricerca effettuata e Menu() che invece mostra il menu con le funzioni di copia, taglia, ecc. che compare solitamente alla pressione del tasto destro del mouse.

MACRO PER ACCEDERE A PROGRAMMI ESTERNI

Questo è un nuovo tipo di macro, che ha lo scopo di aiutare l'utente nella personalizzazione della guida attraverso il richiamo ad altri programmi (come i componenti del pannello di controllo, MS Paint, ecc.) o tutorial creati con un qualsiasi linguaggio di programmazione. A questa categoria appartengono tre nuove macro.

ControlPanel (CPL_name, panel_ name, tabnum) avvia un componente del pannello di controllo, visualizzando una cartella prescelta. *CPL_name* specifica il nome del programma che contiene il componente del pannello di controllo, *panel_name* è il nome del componente del pannello di controllo e deve essere lo stesso testo che appare sotto l'icona del componente prescelto del pannello di controllo; *tabnum* è il numero della cartella da visualizzare (la prima cartella ha indice zero). Per trovare i nomi dei componenti installati, è sufficiente posizionarsi sulla directory \WINDOWS\SYSTEM e visionare i file CPL. Se volessimo offrire la possibilità all'utente di modificare il fuso orario, tramite un normale pulsante sarà necessario inserire la seguente macro:

```
CreateButton ('btn_app', '&Fuso
    Orario...', 'ControlPanel
    (TIMEDATE.CPL,Data/Ora,1)')
```

La macro ControlPanel può essere utile nel caso l'utente debba apportare delle modifiche alla configurazione del sistema per utilizzare un determinato programma, ad esempio modificando le impostazioni della scheda audio o aggiungendo qualche componente di sistema.

ExecFile (program, arguments, display-state, topic-id) permette di eseguire un programma o un file associato ad un programma; *program* indica il nome dell'eseguibile o del file da visualizzare collegato ad un programma registrato, *arguments* è l'eventuale riga di comando da passare al programma; *display-state* è un valore indicante il modo di visualizzazione della finestra (se non viene specificato nessun valore, la finestra viene attivata nella posizione e nelle dimensioni correnti); *topic-id* specifica l'ID dell'argomento che viene visualizzato se il programma non viene eseguito. Questa macro sostituisce la macro ExecProgram, ormai obsoleta.

Può capitare di voler offrire all'utente la possibilità di compilare un modulo di registrazione di un programma shareware, dopo averne letto la licenza d'uso. Sarà quindi possibile inserire un file in formato Winword contenente il modello di ordine del programma (ad esempio, registra.doc), oppure eseguire un applicativo appositamente creato (ad esempio, registra.exe). Nel primo caso, tramite la macro CreateButton si creerà un pulsante e per il parametro *macro* si utilizzerà la stringa "ExecFile(registra.doc)"; nel secondo caso si userà invece la stringa "ExecFile(registra.exe)"

FileExist(filename) controlla se il programma contraddistinto dal nome indicato dal parametro *filename* esiste o meno. Questa macro va utilizzata assieme a macro del tipo *IfThenElse* (già viste nell'articolo precedente). Ad esempio, si potrebbe una macro potrebbe controllare se è installato nel sistema il programma MSPAINT.EXE: in caso affermativo il programma verrà eseguito, altrimenti si dovrà visualizzare una finestra con topic "uninstall":

```
IfThenElse (FileExist(C:\WINDOWS\
    MSPAINT.EXE), ExecFile(MSPAINT),
    JumpId(uninstall))
```

ShellExecute (filename, options, show-flag, operation, path, topic-id) è una macro che esegue o stampa un determinato file, indicato dal parametro *filename*. Il parametro *options* specifica i parametri passati al programma (soltanto in caso di file eseguibile: se si tratta di un documento, il parametro va omesso) mentre *show-flag* specifica il modo in cui verrà visualizzato il programma una volta eseguito; *operation* indica l'operazione che si intende effettuare tra quelle offerte: "open" (default), "opencpl" e "print"; *path* indica la cartella di default, o una stringa nulla se non si desidera specificare una cartella di default, mentre *topic-id* rappresenta l'ID dell'argomento da visualizzare se l'esecuzione della macro fallisce.

NUOVA MACRO PER I BOOKMARK

Con la vecchia versione dell'Help Compiler, per accertare se un segnalibro non esisteva bisognava anteporre NOT alla macro; la nuova macro IsNotMark ("marker-text") riduce il numero delle parentesi necessarie e rende più semplice la vita all'autore della guida:

```
IfThen(IsNotMark('Prova Segnalibro'),
    "DisableButton(`btn_app')')
```

MACRO DI COLLEGAMENTO

Alle macro di collegamento Jump e Popup descritte nel precedente articolo sono state aggiunte tre nuove macro.

ALink (keyword; keyword, type, topic-id, window-name) ricerca le parole chiave specificate dalle note a piè di pagina contrassegnate dalla lettera A anziché dalla solita K. Il primo parametro include una o più parole chiave (separate da un punto e virgola), se una parola chiave contiene una virgola l'intero argomento deve essere racchiuso tra virgolette. Il parametro *type* specifica l'azione da compiere se viene trovata una o più parole chiave: i valori disponibili sono JUMP (1) per indicare che se anche viene trovato una sola parola chiave, Winhelp deve saltare direttamente a quell'argomento, TITLE (2, il default) per specificare che se la parola chiave viene trovata in più di un file HLP, Winhelp visualizza il titolo del file di guida che contiene la parola chiave, TEST (4) che invece indica che la macro deve restituire un valore indicando se è stata trovata o meno una delle parole chiave ricercate. Il significato degli altri parametri dovrebbe essere noto: *topic-id* specifica l'ID dell'argomento che viene visualizzato se non viene trovata nessuna parola chiave, *window-name* è il tipo di finestra in cui visualizzare l'argomento. Ad esempio, se desideriamo avere una lista degli argomenti che contengono la parola chiave "Visual Basic" come nota a piè di pagina di tipo A, sarà necessario utilizzare la macro:

```
ALink (Visual Basic)
```

La macro KLink (keyword; keyword, type, topic-id, window-name) è molto simile alla precedente e utilizza gli stessi parametri: la differenza sta nel fatto che KLink utilizza le note a piè di pagina di tipo K, quelle cioè che contraddistinguono il capitolo dell'argomento (vedi il mio articolo in VBJ n. 6 del dicembre '95).

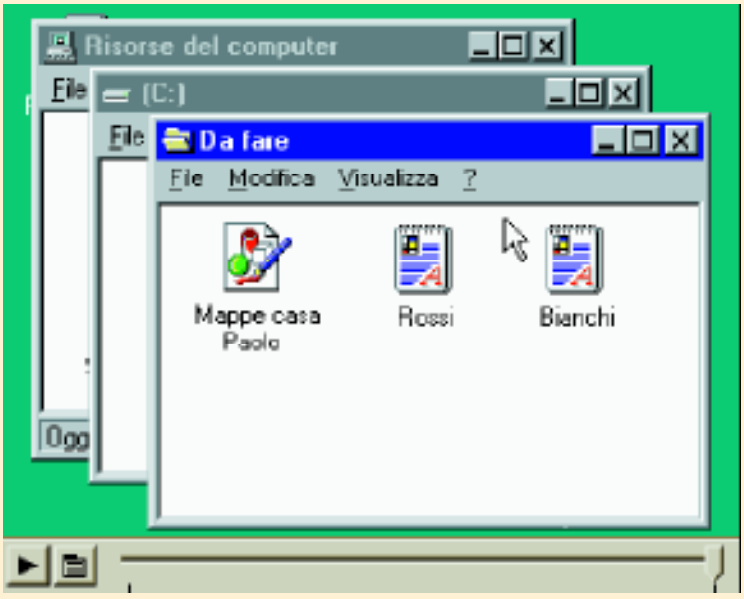

Figura 1

L'esecuzione di un file AVI con in basso gli strumenti per il controllo dell'esecuzione

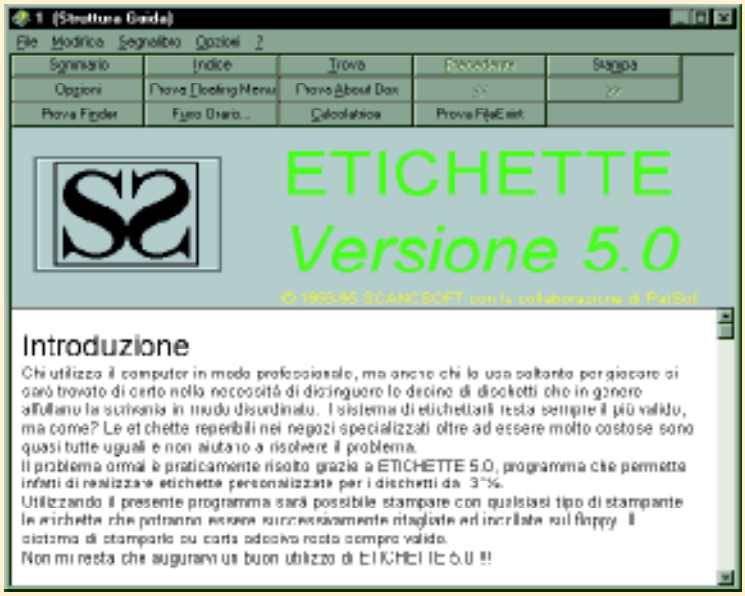

Figura 2

Una delle schermate del progetto di help fornito su dischetto

L'ultima delle nuove macro di collegamento è UpdateWindow (filename>window-name, topic-id) che permette di saltare all'argomento *topic-id* dela finestra specificata dal parametro *window-name*, restituendo però come finestra attiva quella che ha chiamato la macro. Il parametro *filename* viene utilizzato soltanto se l'ID dell'argomento è contenuto in un file HLP diverso da quello in cui è contenuta la macro: in questo caso il nome della finestra deve essere separato dal nome del file tramite il carattere ">". Ad esempio, se dobbiamo richiamare la parola chiave "Visual Basic" nella finestra principale di un file di help esterno sarà necessario utilizzare la macro in questo modo:

```
UpdateWindow (c:\prova\prova.hlp>main,
    Visual Basic)
```

MACRO PER LA GESTIONE DEI MENU

La nuova macro FloatingMenu () che si differenzia dalla vecchia Menu() per il fatto che quest'ultima viene eseguita al momento in cui viene incontrata e, se è contenuta in un gruppo di macro, viene eseguita prima di qualsiasi altra macro. Per la gestione delle singole voci di menu sono state aggiunte alcune nuove macro che vanno a potenziare le precedenti.

ExtInsertItem (menu-ID, item-ID, item-name, macro, position, display-state) ha le stesse funzioni della macro InsertItem esistente, con il parametro aggiuntivo *display-state* che può assumere i seguenti valori: GRAYED (disabilita il menu), CHECKED (inserisce un check mark vicino alla voce di menu) e SEPARATOR (aggiunge una barra separatrice al menu). Quindi, se desideriamo aggiungere elemento al menu File della guida (es. "&Esecuzione di MS-Paint"), inserendolo in prima posizione, sarà necessaria la seguente macro:

```
ExtInsertItem (mnu_file, item_paint,
    &Esecuzione di MS-Paint,
    ExecFile(mspaint.exe), 0)
```

ExtInsertMenu (parent-ID, menu-ID, menu-name, menu-position, display-state) inserisce un sottomenu ad un menu precedentemente definito: ad esempio, per aggiungere una voce di sottomenu "&Primo sottomenu" non attiva all'inizio del menu File della Guida, si potrà usare la macro

```
ExtInsertMenu (mnu_file, sub_prova,
    &Prova sottomenu, 0, GRAYED)
```

ExtAbleItem (item-id, display-state) abilita o disabilita una voce di menu, avente per *item-id* il valore assegnato dalle macro AppendItem, ExtInsertItem o InsertItem, e per *display-state* i valori GRAYED (se disabilita la voce di menu) e CHECKED (se inserisce un check mark alla voce di menu)

MACRO PER LA STAMPA DI ARGOMENTI DELLA GUIDA

Le precedenti versioni di Winhelp non permettevano la gestione personalizzata della stampa degli argomenti della guida: l'unica possibilità offerta all'utente era la stampa attraverso la macro Print che operava l'argomento visualizzato correntemente. Con la nuova versione dell'Help Compiler è possibile stampare più di un argomento alla volta, permettendo anche all'utente meno esperto di avere una copia su carta degli argomenti più interessanti della guida. Le nuove macro sono tre.

IntMPrint () prepara Winhelp alla stampa di più argomenti. La macro visualizza la finestra di impostazione della stampante: se l'utente clicca su OK, la macro IntMprint restituisce il valore TRUE, altrimenti restituisce il valore FALSE. È bene che quasta macro sia utilizzata con la macro di controllo IfThen, in modo da effettuare la stampa degli argomenti soltanto se l'utente clicca su OK nella finestra di impostazione della stampante.

MPrintId (topic-id) stampa un determinato argomento identificato da *topic-id*.

EndMPrint () nasconde la finestra di dialogo "Stampa in corso di ..." e termina l'intera operazione di stampa.

Ecco un breve esempio concreto: se si desidera fornire agli utenti la stampa di una serie preordinata di più argomenti, senza costringerli a visualizzarli singolarmente, sarà sufficiente creare una macro (e associarla ad un pulsante) che contenga questo susseguirsi di comandi

```
IfThen(IntMPrint(), MPrintId(Intro):
    MPrintId(Imposta_lbl):
    MPrintId(Registra): EndMPrint())
```

Per maggiori dettagli si veda il file MACRO.ZIP sul floppy disk allegato.

TRAINING CARD

Le training card sono finestre secondarie che permettono di inviare o ricevere istruzioni da altri programmi. Questa funzione è particolarmente per guidare l'utente passo dopo passo nell'apprendimento di una applicazione. Quando un utente utilizza un passo della guida, è possibile visualizzare subito il passo successivo oppure mostrargli le informazioni sull'errore commesso. Per sapere come agisce l'utente ed impostare tutte le soluzioni possibili si utilizza la macro *Tcard (command)* . Si tratta di una macro molto potente e versatile, la cui descrizione è troppo complessa per essere completata in queste pagine: probabilmente ne riparleremo in un altro articolo su VBJ.

NUOVI COMANDI WINHELP API

Parallelamente all'introduzione dei nuovi comandi, Winhelp 4.0 supporta

Tabella 1

Caratteristica	Limite
Lunghezza file di help	2 gigabyte
Topic per file (rtf)	praticamente nessun limite
Topic per Help file	praticamente nessun limite
Topic per keyword	64.000
Lunghezza footnote	16.383 caratteri
Lunghezza keyword	255 caratteri
Testo hotspot nascosti	4.096 caratteri
Titolo dell'help	127 caratteri
Titolo del topic	127 caratteri
Titolo delle window custom	50 caratteri
Nome delle window custom	8 caratteri
Stringa di Copyright	255 caratteri
Stringa di Browse	50 caratteri
Bitmap referenziate	65.535 bitmap per Help file
Nome di file	259 caratteri
Nomi di font	31 caratteri
Font range	max 20 range
file di log degli errori	senza limiti (Help Workshop può visualizzare 64K di testo in Windows 95 e 1 MB di testo in Windows NT, ma il log file stesso non ha limite di lunghezza)
Stringhe di citazione	2.000 caratteri
Definizioni di window	255 per progetto
Caption delle Window	50 caratteri
Elementi nel file Contents	praticamente nessun limite
Titoli dei Contents	9 livelli
Topic dei Contents	255 caratteri
Lunghezza dei titoli di Contents	praticamente nessun limite

I limiti del sistema di help

anche un certo numero di nuove funzioni richiamabili direttamente dai propri programmi:

HELP_CONTEXTMENU
visualizza la guida "context-sensitive" quando viene cliccato il pulsante destro del mouse
HELP_FINDER
visualizza la finestra di ricerca
HELP_SETPOPUP_POS
imposta la posizione per la prossima finestra di pop-up
HELP_TCARD
indica che un comando è per un'istanza "training-card" del programma Winhelp
HELP_WM_HELP
visualizza la guida "context-sensitive" alla pressione del tasto F1

Supponiamo che il nostro utente desideri visionare la finestra di ricerca, semplicemente cliccando una voce di menu del nostro programma Visual Basic: sarà sufficiente inserire le seguenti righe di codice nell'evento Click() di quel determinato menu:

```
Sub mnuHelpFinder_Click ()
    DummyVal$ = ' '
    Temp% = Winhelp(myForm.hWnd,
    App.Path + '\NOMEFILE.HLP,
    HELP_FINDER, DubbyVal$)
End Sub
```

FINESTRE SECONDARIE

Le finestre secondarie possono contenere una barra di pulsanti (con massimo 22 pulsanti) che è completamente configurabile nel file HPJ; questa azione può essere programmata facilmente nell'Help Compiler Workshop ed è possibile utilizzare sia i pulsanti standard che quelli personalizzati. Con il nuovo Winhelp è possibile utilizzare oltre 255 finestre secondarie, ma se ne possono visualizzare contemporaneamente "soltanto" nove (e vorrei proprio vedere un utente alle prese con nove finestre di help aperte contemporaneamente!). Le finestre secondarie possono essere controllate con due nuove macro.

CloseSecondarys () chiude tutte le finestre secondarie tranne quella attiva. Questa macro può essere utile se l'utente ha aperto troppe finestre secondarie e potrebbe esser associata ad un pulsante per aiutare l'utente ad eliminare la confusione e lasciare aperta soltanto la finestra visualizzata. Per far ciò basta utilizzare la solita macro CreateButton nel modo seguente:

```
CreateButton(button_CLOSE, '&Chiudi tutte
    le finestre', CloseSecondarys()).
```

SetPopupColor (r, g, b) permette di impostare il colore dello sfondo per tutte le finestre di popup. I valori del rosso, verde e blue sono interi nell'intervallo 0-255. Se, ad esempio, desideriamo che tutte le finestre di popup abbiano sfondo verde, sarà necessario inserire la seguente macro nella sezione [CONFIG] del file HPJ:

```
SetPopupColor (0, 255, 0)
```

L'altra nuova macro del gruppo è Compare ("filename.hlp") , che permette di aprire una seconda istanza di Winhelp ed affiancarla a quella in esecuzione. Questa macro può essere utile per paragonare due versioni della stessa guida, il che può risultare utile in caso di aggiornamento del file oppure di una sua "traduzione-on-line".

GESTIONE DELLE IMMAGINI

Winhelp ora può visualizzare immagini a qualsiasi risoluzione (16 o 256 colori, 24 bit). Quando la risoluzione di un bitmap è diversa dalla risoluzione corrente, Winhelp cerca di compensare. Se la risoluzione corrente prevede più colori di quelli contenuti nella bitmap non vi sono problemi, ma in caso contrario Winhelp deve operare una riduzione dei colori, che spesso porta a un peggiornamento notevole della qualità dell'immagine. Per ovviare a questo problema è possibile inserire nel file della guida la stessa immagine salvata a risoluzioni differenti (e con nomi differenti). Fatto ciò, le immagini si possono caricare a runtime usando il comando {bmx lista_file} , con i nomi dei file separati dal carattere ";": Winhelp determinerà la risoluzione sul sistema dell'utente e provvederà ad utilizzare l'ìmmagine con la risoluzione più appropriata. Ad esempio, se creiamo un immagine di prova DEMO_256.BMP a 256 colori e DEMO_016.BMP a 16 colori, sarà possibile richiamarle con la seguente riga:

```
{bmx DEMO_016.BMP; DEMO_256.BMP}
```

Un altro frequente problema è il seguente: se l'*aspect ratio* di una bitmap contenuta nel file HLP è differente da quello del monitor dell'utente, Winhelp distorce l'immagine e l'immagine potrebbe diventare di difficile comprensione, soprattutto se contiene del

Tabella 2

SW_HIDE	nasconde la finestra
SW_MINIMIZE	minimizza la finestra
SW_RESTORE	ripristina le dimensioni e la posizione iniziale
SW_SHOW	visualizza la finestra alla posizione corrente
SW_SHOWMAXIMIZED	attiva la finestra e la massimizza
SW_SHOWMINIMIZED	attiva la finestra e la minimizza
SW_SHOWNA	visualizza la finestra allo stato corrente ma Winhelp abbandona il focus, rendendola inattiva
SW_SHOWNOACTIVATE	visualizza la finestra nelle dimensioni e alla posizione più recenti ma non attiva la finestra
SW_SHOWNORMAL	attiva la finestra e la visualizza

I possibili valori del parametro display-state della macro ExecFile

testo. Se si desidera che Winhelp non distorca le immagini è necessario utilizzare il programma MRBC.EXE: una volta creato il file con estensione MRB sarà possibile inserirlo nel file RTF con il comando bmx appena descritto. È comunque consigliabile, se l'immagine contiene del testo, utilizzare il formato metafile WMF.

Una novità di Winhelp 4.0 nella gestione delle immagini è la possibilità di operare in modo che lo sfondo bianco di una immagine diventi trasparente, così da "fondersi" con il background della finestra. Per fare ciò è necessario utilizzare il comando per caricare le immagini seguito dalla lettera "t". Ad esempio, se desideriamo caricare l'immagine DEMO.BMP con lo sfondo trasparente ed allinearla a sinistra, sarà necessario inserire nel file RTF la seguente riga:

```
{bmlt DEMO.BMP}
```

GESTIONE DEI FILMATI AVI

Winhelp supporta direttamente l'inclusione di filmati AVI, con il comando

```
{mci[_left | _right] [options,] filename}
```

Left e *right* rappresentano l'allineamento della finestra di riproduzione, mentre *options* specifica le opzioni per il controllo multimediale appena creato. Per utilizzare più di una opzione è necessario uno spazio (non vanno utilizzate virgole o punti e virgola). Le opzioni possibili sono EXTERNAL (mantiene il file *filename* esterno al file HLP), NOPLAYBAR (non visualizza la barra di riproduzione e non permette all'utente di agire sul filmato), NOMENU (non visualizza il pulsante di menu se c'è la barra di riproduzione), REPEAT (ripete il file una volta terminata la sua l'esecuzione), PLAY (avvia l'esecuzione del filmato). In caso non vengano specificate le opzioni, il file appare con la barra di riproduzione, il pulsante di menu e non viene eseguito finché l'utente non lo avvia.

Se volessimo avviare il file WHATSON.AVI (contenuto nella directory \WINDOWS\ HELP) senza la barra di riproduzione ed inglobandolo nel file di help sarà necessario inserire l'istruzione

```
{mci NOPLAYBAR PLAY, C:\WINDOWS\HELP\
   WHATSON.AVI}
```

si veda anche il sorgente su dischetto AVI1.ZIP. Come si potrà notare eseguendo il file HLP, se si effettua un aggiornamento della finestra (spostamento, ingrandimento, riduzione, ecc.) il filmato viene rieseguito. Per lasciare la possibilità all'utente di rieseguire il firmato a suo piacimento, inter-rompendolo quando desidera, è necessario sostituire l'istruzione precedente con:

```
{mci PLAY, C:\WINDOWS\HELP\WHATSON.AVI}
```

il sorgente è nel file AVI2.ZIP su dischetto. Come si può notare, in questo caso il filmato si riavvia anche quando l'utente clicca sul pulsante di esecuzione. Correntemente Winhelp 4.0 visualizza soltanto il formato AVI, ma Microsoft ha annunciato che le prossime versioni potranno utilizzare anche altri formati multimediali.

COME AGGIUNGERE UN HOTSPOT CHE ESEGUE UNA MACRO

Una cosa che non ho chiarito nei precedenti articoli è la seguente: come richiamare una macro di quelle descritte direttamente da un hotspot. Per poter far ciò è necessario digitare il testo che si desidera utilizzare come hotspot facendogli seguire, senza spazi, un punto esclamativo e la macro che si intende eseguire. Nella nuova versione è necessario evidenziare con una doppia sottolineatura l'hotspot ed assegnare la proprietà di testo nascosto al rimanente testo (compreso il punto esclamativo). Se, ad esempio, si vuole che al click dell'utente sulla frase "Avvia la stampa del modello" venga avviata la macro di stampa *Print()*, sarà necessario scrivere il testo nel file RTF quanto segue:

Avvia la stampa del modello!Print()

Questa volta la nostra analisi dell'Help Compiler è veramente arrivata alla fine. Nel floppy disk allegato alla rivista vi sono tre esempi di file HLP, due contenenti filmati multimediali ed uno contenente il sorgente della guida di una mia applicazione che sfrutta alcune delle macro qui descritte.

Tabella 3

IDABORT (3)	l'utente ha cliccato il pulsante "Abbandona"
IDCANCEL (2)	l'utente ha cliccato il pulsante "Annulla"
IDCLOSE (8)	l'utente ha chiuso la guida passo a passo
IDHELP (9)	l'utente ha cliccato il pulsante "Guida"
IDIGNORE (5)	l'utente ha cliccato il pulsante "Ignora"
IDOK (1)	l'utente ha cliccato il pulsante "Ok"
IDNO (7)	l'utente ha cliccato il pulsante "No"
IDRETRY (4)	l'utente ha cliccato il pulsante "Riprova"
IDYES (6)	l'utente ha cliccato il pulsante "Si"

I possibili valori dell'argomento Command della macro Tcard

Roberto Scano ha conseguito il Diploma di Ragioniere e Perito Commerciale lo scorso anno. Ha esperienza di produzione e vendita software sia nel mercato shareware che commerciale, collabora spesso con riviste informatiche e da due anni si occupa di programmazione Windows in Visual Basic. Può essere raggiunto tramite telefono o fax al numero 041-5263540