HyperCard II v 2.2

di Raffaello De Masi

Non credo di essere un fatalista, ma talvolta sono portato a ricredermi vedendo come certe persone, libri, prodotti, opere d'arte non riescono ad avere la giusta collocazione meritoria che spetta loro per la qualità che li contraddistingue. Vero è che spesso, almeno per le persone, il genio è riconosciuto solo dopo la morte (e per questo spero che il mio, sconfinato come non poteva non essere, sia riconosciuto nei secoli di là da venire), ma ogni tanto devo rilevare che anche nel software esistono gioielli di valore, frutto di un pesante e certosino lavoro programmatorio, che sono relegati nel limbo della semindifferenza o della semplice conoscenza superficiale. Un caso eclatante è Hypercard, che pare proprio «La figlia della sora Camilla, che tutti la vogliono e nessuno se la piglia»; tutti ne dicono, anche nella più cruda ignoranza, un gran bene, tutti lo conoscono, tutti, almeno una volta hanno aperto uno stack, o l'hanno utilizzato per tenerci la rubrica telefonica (già pronta), ma da questo, a metterci un poco più d'impegno e andare a vedere che cosa effettivamente fa o può fare, ci corre un bel pezzo di autostrada.

E così HyperCard, con la sua stoffa da campione, è stato costretto a fare sempre da gregario. Salutato all'inizio come una vera rivoluzione (comparve nel 1987 e portava, allora la firma di Bill Atkinson, gran guru della mela) rappresentò davvero una pietra nello stagno dei programmi fino ad allora presenti sul mercato. Ciononostante presentava alcune caratteristiche immediatamente non gradite dall'utenza: assenza di colore, ma gestito attraverso un XCMD, grafica in bitmap, lentezza intrinseca nella manipolazione degli stack, formato delle schede ridotto al solo schermo; si aggiungeva a questo la gratuità del pacchetto, inserito di serie nel sistema operativo che, per una perversa logica che colpisce tutto ciò che non costa nulla, lo trasformava di colpo in non più di un giocattolino, una utility, cui probabilmente prestare appena un pizzico di attenzione. E Valter Di Dio ne sa qualcosa, quando vede i suoi figliocci, ancorché pregevoli, degnati appena di uno sguardo di sufficienza solo perché, per il fatto che costano solo qualche migliaio di lire o addirittura una cartolina illustrata, devono per forza valere poco o niente.

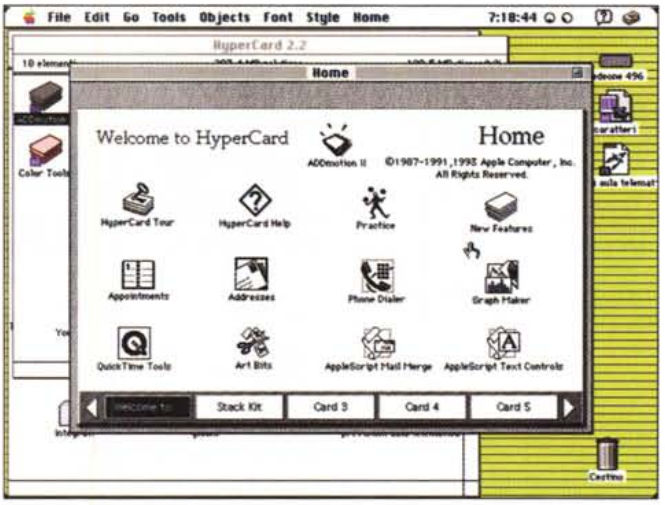
E a poco valse anche la scappatoia di scegliere la via di un pacchetto «enhanced», che apriva la porta a HyperTalk, linguaggio di programmazione raffinatissimo, intuitivo, più vicino alla lingua parlata dello stesso Basic e più divertente del PostScript. Tant'è che Apple abbandonò formalmente il package e ne affidò le sorti a Claris (sua sorellastra) per svilupparne una commercializzazione che, per la verità, non mi pare sia stata mai perseguita con qualche tenacia. Atkinson abbandonò il progetto che passò nelle mani di Ed Rosenzweig, autore già di pregevoli cose; questi riunì uno staff formidabile e ricostruì completamente il package, riscrivendone com-

HyperCard II v 2.2

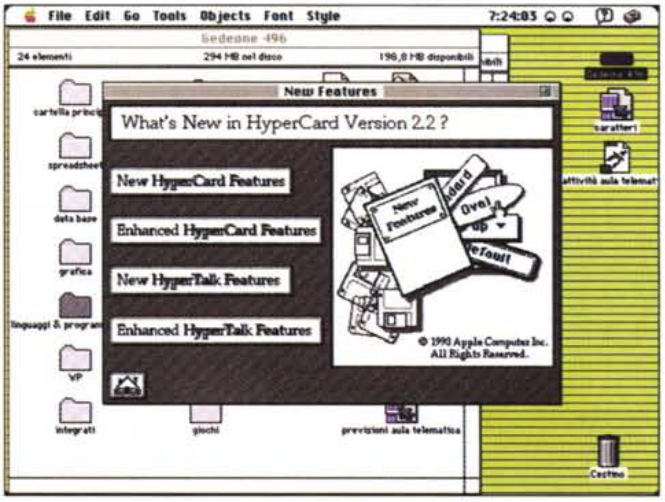
Apple Computer, Inc.
20525, Mariani Avenue
Cupertino, CA 95014-6299
Distributore:
Apple Computer Italia
Via Milano, 150
Cologno Monzese (MI)
Prezzo orientativo *(IVA esclusa):* *Lit. 210.000*

pletamente il codice (in MacApp) pur conservando quasi immutata l'interfaccia. Niente da fare; HyperCard rimase un aliante pur avendo il cuore di un F14, e lo dimostra il fatto che pur essendo un attrezzo di sviluppo formidabile anche per professionisti ebbe il suo massimo sviluppo nell'area dello shareware, dove stack finalizzati a scopi tanto diversi da non essere neppure qui accennabili prolificarono in fretta e finirono sulle raccolte in CD-ROM o nelle viscere labirintiche di Genie, Compuserve e del nostro MC-link.

Quali i motivi di tutto ciò? Ne parleremo, nel caso più in là. Adesso è il momento di presentare la versione 2.2, ritornata di nuovo sotto l'egida Apple, che la commercializza dietro il suo marchio. E la cosa è tanto più interessante in quanto questo aggiornamento vale davvero una nuova versione, visto quello che mette a disposizione.



La classica finestra principale di HyperCard; in alto, accanto all'intestazione, lo stack AddMotion, che va installato con una procedura separata.



Lo stack evidenziante le nuove caratteristiche del prodotto; lo stack è anche interessante perché rappresenta un buon esempio di presentazione.

Il pacchetto

HyperCard nella sua completa versione 2.2 si presenta come un poderoso pacchetto dotato di una monumentale documentazione (circa 1500 pagine) e di una altrettanto grossa mole di software (ben 11 dischetti, in parte anche compressi). Proprio per rimarcare ancora di più il fatto che il programma gira su macchine anche minime, i dischetti sono del tipo DSDD, da 800K, e possono essere riversati anche su macchine della classe Plus, a patto, ovviamente, che abbiano una memoria di massa sufficiente.

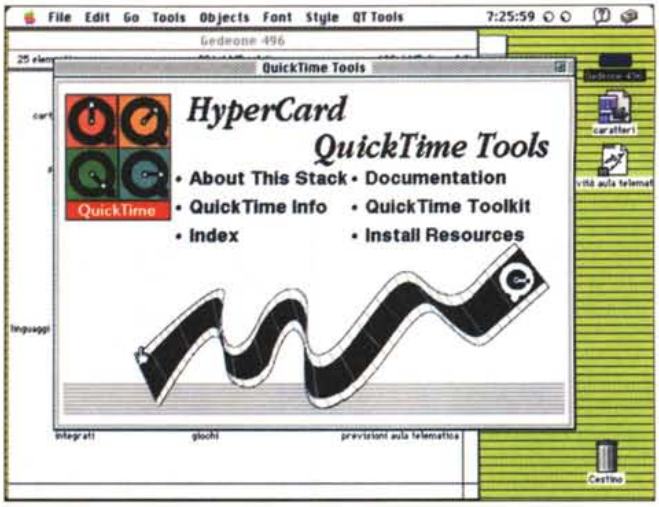
Una volta sistemato il pacchetto (con tutti gli add-in) con il solito Installer sull'HD viene creata una cartella-mostro della grandezza di ben 8.2 Mb, senza contare le estensioni piazzate automaticamente nella cartella sistema. La ben nota icona HyperCard (una manina appoggiata su uno stack) padroneggia un ben nutrito campo di cartelle e di stack allegati.

Lanciando il programma si passa ancora alla altrettanto familiare scrivania di Home, il capolinea-semaforo da cui raggiungere tutte le informazioni contenute nel programma e negli stack già pronti o da costruire.

Ma cosa è HyperCard, per chi non l'avesse (e sono molti) ancora capito? Per dirla in breve lo si può utilizzare come una semplice collezione di informazioni, come anche il più complesso mezzo di programmazione o applicativo che non solo esiste già sul mercato, ma che possiamo immaginarci. Tecnicamente HC è un tool di costruzione di applicazioni software; in altri termini è un mezzo per creare una propria strada per eseguire cose di proprio gusto e piacimento sul nostro computer. Si dirà che la stessa cosa la può fare un linguaggio di programmazione, ma anche il più familiare e potente tool di tal tipo, come ad esempio Think C o Future Basic, sta ad HC, almeno per quanto riguarda la facilità di ottenere risultati rapidi e funzionali, come un trattore sta a una Volvo. E questo, nella maggior parte dei casi, senza conoscere una sola parola di linguaggio di programmazione. Se poi ci aggiungiamo HyperTalk, il linguaggio built-in fornito col pacchetto, lascio immaginare solo a chi legge il nuovo rapporto.

Conoscere HyperCard

L'uso di HyperCard è fin troppo noto perché abbisogni di essere descritto in questo articolo; accenneremo quindi solo di passaggio alle tecniche più usuali del maneggio degli stack (cosa che poi è talmente facile da non abbisognare certo della mia limitata spiegazione). La stessa considerazione debbono aver fatto, ovviamente, anche i redattori del manuale se hanno pensato di relegare l'illustrazione delle tecniche di maneggio di quest'area in soli due capitoli del manuale di riferimento. La cosa è importante, in quanto HC2 è ben più di un semplice manipolatore di cataste, anche se questa caratteristica è la sua punta di diamante.



HyperCard e i suoi rapporti con QuickTime; le nuove tecniche sono illustrate in uno stack, chiaro e molto essenziale.

Credo che organizzare uno stack di base non eccezionalmente elaborato sia alla portata di tutti; per avere idea dell'abisso che c'è tra questa caratteristica e quello che il package completo mette a disposizione basterebbe solo ricordare che la versione non programmabile fornita con i package del System era dotata solo di un manualetto di una cinquantina di pagine. HC è ben altro, e ci si rende conto di ciò già al terzo capitolo del manuale di riferimento, «A Taste of HyperTalk», quando si viene introdotti all'uso dell'HyperCard Script Language, HyperTalk, appunto, per gli amici, che poi rimanda al grosso manuale del linguaggio, che è il vero pezzo forte del programma.

All'apertura, dicevamo, ci troviamo in Home, la scrivania propria di HC. Poco è cambiato rispetto alle versioni precedenti (anche la 1) e ritroviamo la classica struttura ad icone, dell'indirizzario degli stack piacevole e intuitiva da usare, vero salotto buono del pacchetto. Spulciando nel capitolo dedicato vediamo tra l'altro come usare HC su un file server o su un CD-ROM (che, ahimè, penalizza ancora di più la intrinseca lentezza del programma). Abbandoniamo subito questo livello (l'user level: di livelli ne abbiamo cinque: lettura, scrittura, disegno, programmazione di base, programmazione con script), per passare ad approfondire, dal livello più semplice, la possibilità di programmazione delle schede HyperCard. Ogni scheda è essenzialmente formata da due parti: una di sfondo e una di superficie, d'uso per così dire. Vediamone un momento la differenza: il foglio di superficie è quello destinato all'utente finale, dove verranno editati i dati; se guardiamo ad HyperCard come ad un database, l'insieme dei fogli di superficie rappresenterà la base dati vera e propria. Viceversa la scheda di background è, mantenendo la stessa similitudine, la maschera di input che riceverà i dati, e accoglierà tutto quello che non cambia da

Cosa offre di nuovo la versione 2.2?

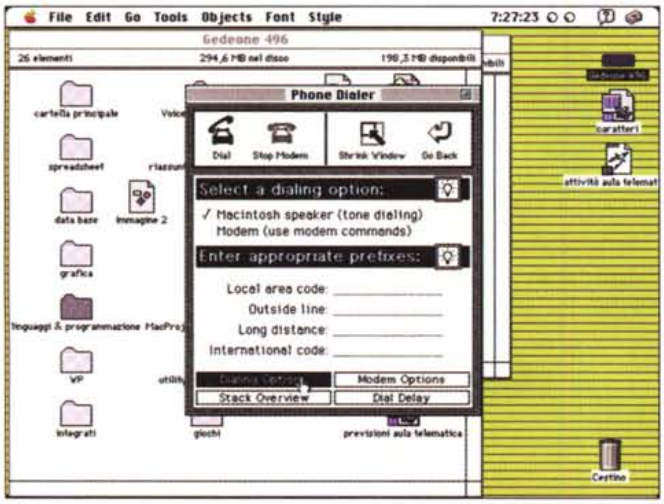
Sebbene l'ambiente di base sia stato oggetto di una rimanipolazione generale, a una prima occhiata la nuova versione di HC, sempre ovviamente confrontandola con le versioni complete che l'hanno preceduta, si presenta quasi la stessa. Invece molte sono le differenze e i miglioramenti apportati: vediamone alcuni.

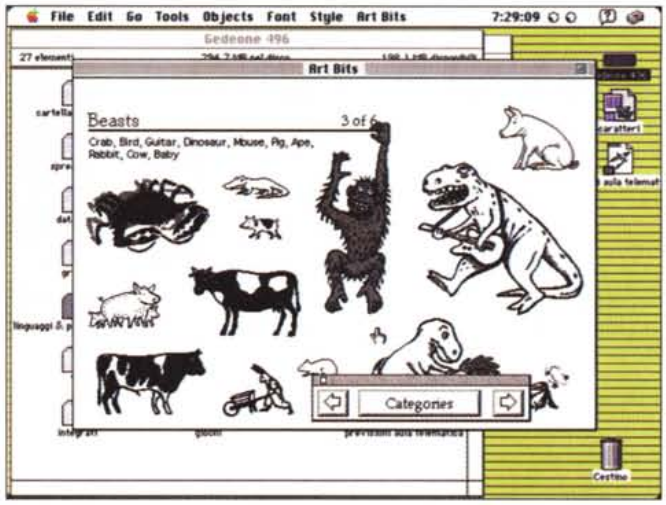
Migliore è l'ambiente di scripting. Oggi l'Apple Open Scripting Architecture (OSA), estensione di HC, può adottare appositi script per integrare HyperCard con altre applicazioni in modo da utilizzare caratteristiche di altri pacchetti negli stack prodotti, scambiando nel contempo dati con altre applicazioni. Fondamentale a questo proposito il fatto che OSA sia adottato da numerose software house, cosa che consentirà col tempo di creare ambienti perfettamente integrati cui concorreranno diverse applicazioni. Non a caso, quindi, HyperTalk possiede comandi del tutto equivalenti a quelli di AppleScript e, in ogni caso supporta istruzioni di questo macrolinguaggio inserite nei suoi sorgenti. Ma anche senza andare nello specifico linguaggio di programmazione, appaiono quasi subito evidenti le nuove caratteristiche dei bottoni, che possono oggi essere preselezionati in default, possiedono una nuova grafica, e possono essere trasparenti e ovali, così da essere adattati a icone e figure. È questo solo un particolare delle nuove regole inserite nel Macintosh Human Interface Guidelines (MHIG), proposto da Apple come nuovo, più avanzato, modello di ambiente operativo Mac; tanto per citare un altro esempio i campi possono essere organizzati in liste, ed è consentito un più ampio e articolato controllo delle funzioni attraverso shortcut, anche personalizzabili dall'utente.

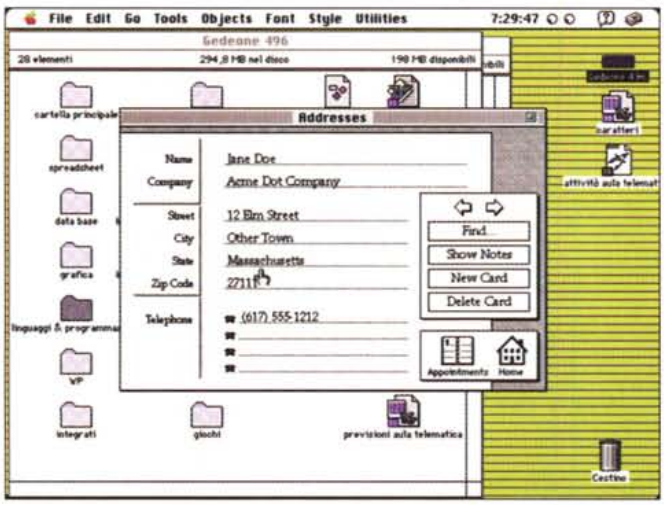
Nuovi, più efficienti protocolli di conversione, sovente trasparenti, permettono di superare le immancabili difficoltà derivanti dallo scambio di stack costruiti sotto sistemi operativi di diversi paesi. La standardizzazione dell'OSA ha inoltre consentito una più semplice portabilità da applicazioni esterne a HC, e questo non solo in termini di dati, ma anche di macroistruzioni. Ad esempio, HyperCard oggi si integra perfettamente in ambienti utilizzanti macroistruzioni dotati di altri standard (come FileMaker Pro 2, Excel 4 e 5, WordPerfect 3). Alla base c'è, comunque, la completa utilizzabilità di AppleScript, componente e possente motore principale di OSA, che, con buona probabilità farà in breve tempo piazza pulita di tutti i macrolinguaggi oggi in circolazione. Molti sono i vantaggi di AppleScript: completa riferibilità a oggetti comunque individuabili (ad esempio lettere, righe, paragrafi, o addirittura sezioni in un wp), utilizzabilità comune su programmi diversi (sebbene molti pacchetti abbiano i loro linguaggi di script, questi sono confinati al linguaggio stesso e non possono servire in un altro programma), possibilità di scrivere macroistruzioni che controllano applicazioni su più di un computer (un singolo script può controllare qualunque numero di applicazioni, e queste possono essere su ogni computer presente nel network). Inoltre AppleScript ha la caratteristica originale di supportare dialetti multipli; essendo il principio di AppleScript (e di HyperTalk) basato sulla somiglianza con la lingua parlata, è possibile inserire nel linguaggio parole di un altro idioma umano, tanto per fare un esempio, il giapponese, senza che il linguaggio stesso ne accusi disturbo.

Si è detto da qualche parte che AppleScript sia figlio di HyperTalk; l'affermazione è per lo meno riduttiva e non fa giustizia delle enormi possibilità del nuovo macrolinguaggio di Apple. È più giusto dire, invece che i due idiomi adottano lo stesso standard, risultano completamente compatibili (è possibile adottare, in un listato HT, comandi AppleScript senza alcuna preoccupazione) e sono solo i primi due tasselli di un mosaico che, pena l'incomunicabilità, dovrà essere rappresentato dall'intero mondo software Mac.

Non a caso pacchetti di grande prestigio, come MiniCad, QuarkXpress, l'appena comparso Excel 5 (per non parlare di tutto il pubblicato da Claris, ovviamente), tanto per citarne solo alcuni, sono già perfettamente aderenti e si può dire non passa giorno che altri si aggiungano alla lista. È il trionfo di AppleEvents, la struttura d'ambiente di System 7 che, al suo apparire, passò quasi inosservata, e che oggi comincia a mostrare i muscoli.







Alcune applicazioni create con HyperCard, in questo caso un combinatore telefonico, una collezione di disegni (si noti il navigatore), una rubrica di indirizzi.

scheda a scheda (ad esempio, bottoni, menu personalizzati, frecce di spostamento, la stessa grafica inerte di sfondo, e così via). Con questo criterio ogni stack ha una scheda solo di sfondo e un numero imprecisato di schede di superficie, corrispondenti al numero di record impostati.

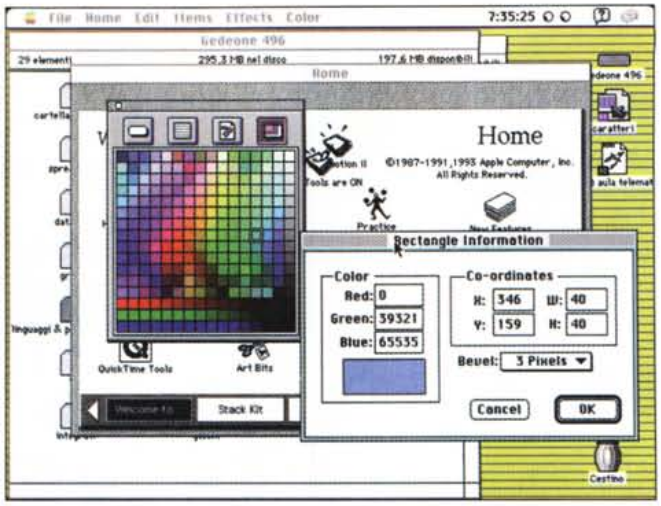
La creazione di un'interfaccia Mac-like

Chi ha cercato di programmare l'interfaccia Mac con linguaggi più o meno evoluti sa bene quanta fatica costi creare proprio quegli oggetti e quelle caratteristiche che rendono Mac tanto piacevole da usare. Molto spesso il programmatore, specie se impegnato in un grosso lavoro di definizione delle routine specifiche per il suo scopo, finisce per trascurare l'interfaccia, che diviene indesiderabilmente keyboard-driven, con buona pace degli utenti DOS. In HC la creazione dei gioielli di famiglia Mac è tanto semplice da divenire automatica. Si sceglie il tipo d'oggetto da creare e se ne draga l'estensione sullo schermo; se necessario si apre automaticamente una finestra di definizione dei relativi parametri e il gioco è fatto. Così bastano trenta secondi per definire un bottone, un paio di minuti per un menu, magari anche gerarchico, e poco più per organizzare un help contestuale di grande potenza. Si possono costruire facilmente bottoni interconnessi, anche con prescelta, e radiobutton di due o più elementi spendendo solo poche energie per redigere le relative didascalie. La nuova versione aggiunge diverse ciliegine alla torta; poiché anche l'occhio vuole la sua parte ecco comparire effetti speciali nella dissolvenza nel passaggio tra scheda e scheda; o magari ecco cambiare la forma del cursore o apparire una barra di progresso durante una ricerca o un ordinamento. Interessante la possibilità di inserire, nella programmazione, la utilizzabilità dei tasti di comando ed opzione, per consentire operazioni solo leggermente diversificate di un singolo comando, e la disponibilità di numerose funzioni preprogrammate per usi standardizzati (es.: ora e giorno, numero delle schede nello stack, nome del compilatore della scheda, definizione di un numero progressivo, e così via).

HyperTalk & AppleScript

E ci avviamo a grandi passi verso l'altro grande cuore di HyperCard, HyperTalk. Cosa sia HyperTalk è facile a dirsi e difficile a mostrarsi in poche righe; tanto per non ricorrere a un banale luogo comune, probabilmente non basterebbe metà di questo fascicolo per evidenziare tutte le caratteristiche di questo linguaggio. HT è, tout-court, il linguaggio di scripting dell'ambiente di sviluppo HyperCard. Esso permette di piegare il programma ai propri desideri e scopi, eseguendo operazioni e azioni su tutta la messe di oggetti propri del programma: bottoni, campi, menu, schede, stack, sfondi, finestre. HyperTalk serve, inoltre a mandare messaggi e a scambiare, trasparentemente o non, informazioni tra oggetti, dalla semplice operazione di schiacciare bottoni fino all'aprire schede o stack diversi, attivare messaggi di attenzione o errore, creare, modificare o distruggere file, e, sempre più difficile, inviare comunicazioni od ordini, attraverso Apple Event, da un processo a file o programmi esterni ad HyperCard stesso. HyperTalk permette di programmare completamente l'ambiente HC, sebbene la versione 2.2, che stiamo provando, abbia pieno accesso ad AppleScript, che può essere totalmente usato come separata opzione di scripting.

In ordine gerarchico gli oggetti manipolabili da HT sono cinque: sfondo, campi, bottoni, e, salendo nella gerarchia, la scheda e la catasta (stack). Inuti-



Le fasi di installazione di ColorTools, una combinazione di uno stack e di una serie di routine HyperTalk che rende accessibile il colore ad HyperCard.

le discutere di questi, c'è però da dire che ogni oggetto di HC può interagire con gli altri, con l'utente, con HyperCard stesso, e addirittura con l'intero ambiente Macintosh. Tutto questo avviene attraverso lo scambio di messaggi, che possono essere di tipo completamente diverso.

Ad esempio alcuni messaggi sono descrizione di cose che avvengono nell'ambiente; nel caso in cui si schiacci il tasto del mouse, viene generato un opportuno messaggio del tipo «mousedown» («mouseup» all'inverso), che possono essere opportunamente catturati da un'apposita routine di trap. In questo caso abbiamo un messaggio di sistema, vero e proprio in quanto, ovviamente, lo schiacciamento del tasto mouse è di gerarchia superiore all'ambiente HC stesso.

I messaggi sono inviati sempre a un oggetto, parola dai molti significati, dal reale al virtuale, e possono seguire una via molto articolata. Restiamo nel caso dello schiacciamento del tasto mouse. Immediatamente il messaggio viene inviato al tasto o al campo sottostante l'hot dot del puntatore sullo schermo. Esso viene immediatamente passato alla scheda, quindi allo sfondo, e ancora allo stack. Di qui passa all'Home e infine al programma vero e proprio (un esauriente approccio a questo meccanismo è fornito in un apposito capitolo della guida al linguaggio, «Handling Messages»). Ancora, oltre a fornire vere e proprie comunicazioni tra parti diverse, un messaggio può rappresentare un vero e proprio comando (come addizionare due numeri, stampare o passare a uno stack diverso).

Tutto ciò avviene attraverso uno script, che non è altro che una macro, una piéce di programma che esegue determinate funzioni.

Ogni oggetto HyperCard, anche il più semplice possiede uno script (anche se spesso esso può essere vuoto). Uno script è una collezione di «handler», metacomandi scritti in linguaggio molto simile all'inglese disposti uno per linea e completati da un Return (è possibile, come in ogni idioma che si rispetti, inserire commenti dove si vuole). Gli handler possono essere di due tipi: di messaggio e di funzione. Nel primo caso iniziano sempre con «on» e sono del tipo:

```
on mouseUp
go to next card
end mouseUp
```

dal significato abbastanza intuitivo. Nel secondo caso possono essere ben paragonabili ad DEF FN del Basic. La funzione:

```
function giorno
return first item of the long date
end giorno
```

seguita, quando necessario dalla chiamata

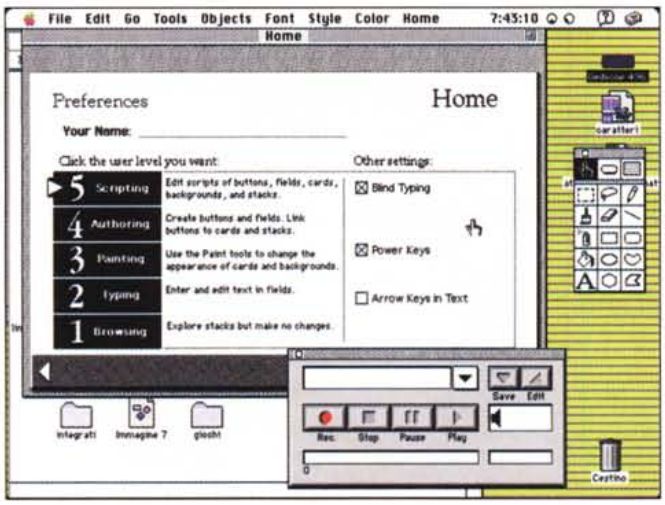
```
put giorno() into message box
```

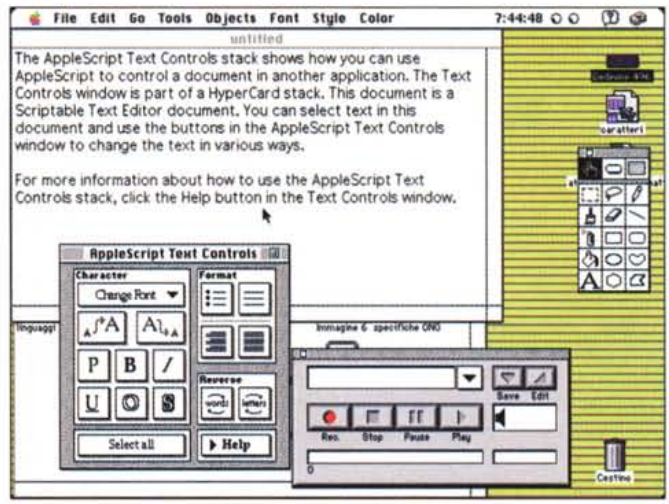
svolge un compito altrettanto intuitivo.

La difficoltà del maneggio delle finestre è ben nota ai programmatori di linguaggi convenzionali. Le finestre, per HyperCard, sono solo un altro tipo di elemento maneggiabile semplicemente attraverso HyperTalk. Ma si intenda: per finestra si intende non solo quella dello stack, generata dal programmatore, ma anche quelle di sistema, come quella di palette, di tool, di scroll, di messaggio e così via. La cosa è fondamentale se si immagina che, con una semplice chiamata, l'utente finale ha a disposizione, ad esempio, tutti i mezzi per disegnare nella scheda attiva. Non solo, ma HT mette a disposizione mezzi per creare finestre di supporto e di aiuto a quella principale definibili dall'utente e di accedere, direttamente in runtime, a quelle, di più basso livello, proprie del programma, come lo screen delle variabili (non mi pare che un altro linguaggio possa fare altrettanto con un semplice comando di un rigo). Per quanto riguarda ancora i menu HT permette diverse cosette interessanti; creare nuovi menu, aggiungere elementi a quelli già esistenti, giocare completamente sugli stili e le posizioni dei diversi elementi, riassegnare o assegnare nuovi shortcut da tastiera, attivare e disattivare automaticamente checkmark (i segni di spunto che compaiono quando si abilita una funzione da menu), abilitare, disabilitare, cancellare o resettare comandi e barre complete. E questo senza preoccuparsi della maggior parte dei problemi che operazioni di tal fatta comportano (come ad esempio il refreshing dello schermo o la gestione dei comandi «dimmed»).

Per le persone avanti nell'ars programmatoria, e per quelli che hanno buoni muscoli, è possibile accedere direttamente allo script attraverso il suo editor. Sebbene la maggior parte degli script si generi automaticamente in maniera abbastanza simile a quella che avviene quando si registra una macro in Excel o in WordPerfect, la cosa può essere utile in fase di debug o di semplice modifica, ad esempio, di messaggi. Occorre precisare, se non lo si fosse già capito, che lavorare su un sorgente HT è cosa ben più facile del più facile linguaggio. Il listato si presenta molto simile a una serie di ordini redatti in una lingua inglese molto più discorsiva del più piacevole Basic. L'editor, come ogni buon wp che si rispetti, gode, oltre dei soliti Taglia-Copia-Incolla di funzioni di ricerca e sostituzione raffinate, ma manca un controllore, in linea di sintassi (peraltro quasi inutile); ogni comando non può superare i 255 caratteri e ogni script le 30.000 linee; quest'ultimo mi sembra un traguardo difficilmente raggiungibile, ma se proprio uno ci si è messo di buzzo buono e l'ha raggiunto, nessun problema; basta spezzare lo script in due monconi e farli invocare tra di loro.

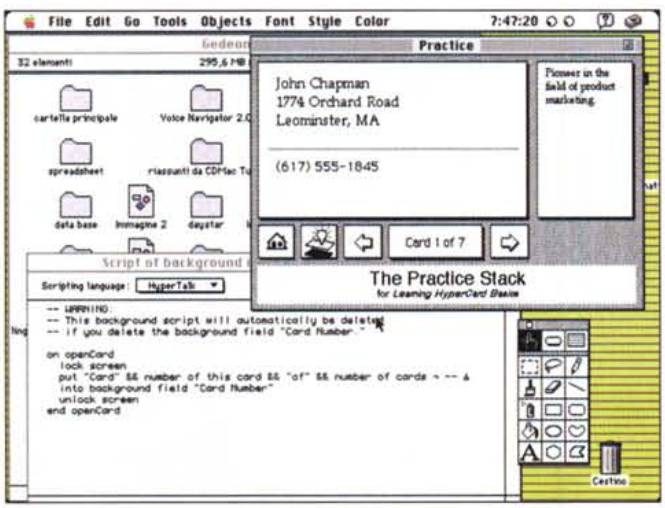
L'assenza del controllore di sintassi di linea è, come dicevamo, quasi irrilevante, anche perché HT mette a disposizione un debugger estremamente potente e funzionale; opzioni di STEP e di TRACE sono presenti (raffinate da collegate opzioni di «INTO»), è possibile settare checkpoint (permanenti e temporanei) fino a un massimo di 32 per ogni script, e assegnare addirittura ritardi alle fasi di TRACE. È ancora possibile avere sempre aperte due finestre aggiuntive, il Variable e il Message Watcher, per monitorare la consistenza di questi due elementi. I checkpoint, poi, divengono





▲ *9*

I livelli di utilizzazione di HyperCard, dalla semplice consultazione fino al più completo scripting; nella figura 9 una fase di utilizzazione dell'editor, nella 10 un esempio di script collegato ad una scheda.



◀ *10*

completamente trasparenti quando si passa alla fase utente, per cui non è neppure necessario preoccuparsi di cancellarli, alla fine.

Oltre 650 pagine dedicate al solo linguaggio la dicono lunga sulla potenza dello stesso, specie se si tiene conto che molti di essi sono vere e proprie macro. In questa ottica HT distingue, giustamente tra comandi veri e propri e proprietà, supercomandi specifici riferiti a parti dell'ambiente controllabili attraverso la trasmissione di parametri. Le property hanno una potenza da linguaggio superevoluto; tanto per fare qualche esempio andando a braccio, è possibile creare una proprietà che scambia uno script con un altro, o impedisce la cancellazione di uno stack, o assegna la grandezza massima di questo in termini di schede o di byte. O, ancora, impedisce certi tipi di ricerca, cambia a richiesta il background di una scheda, permette marcature di schede obbedienti a ceri parametri, abilita o no il comando Tab per passare tra i vari campi, determina o cambia la locazione di un bottone.

Pensate sia finita? Manco per sogno! Una delle caratteristiche spesso misconosciute di HyperCard è la possibilità di interfacciarsi con comandi (XCMD) e funzioni (XFCN) esterni; brevemente (tanto il manuale è estremamente chiaro in proposito) diremo che si tratta di risorse Mac direttamente eseguibili, scritte in linguaggi come Pascal, C, FutureBasic, o direttamente in assembly 68000. Essi possono essere chiamati direttamente da HT nello stesso modo dei comandi built-in e rispettano la stessa gerarchia di passaggio dei messaggi.

Gli add-in di HyperCard 2.2

Dicevamo all'inizio che questo aggiornamento del pacchetto avrebbe meritato gli onori di una versione 3. Questo è dovuto anche alla presenza di due pacchetti esterni aggiuntivi che ne esaltano le già superbe caratteristiche. Il primo è il già famoso AppleScript, primo vero grande linguaggio di script prodotto dalla Apple; purtroppo la manualistica ne ignora completamente la presenza e il tutorial è rappresentato solo da una serie di stack inseriti nel dischetto stesso. Il secondo package, AddMotion, prodotto dalla Motion Works, è invece un completo programma multimediale (specificamente mirato all'animazione, comunque) che crea documenti completamente integrabili nell'ambiente HyperCard. Dotato di un buon manuale, offre la possibilità di comprare, a prezzo ridotto, il più completo package capace di creare multimediali stand-alone.

Conclusioni

HyperCard è un mezzo potente e funzionale per creare applicazioni self-tailored in maniera semplice, accurata e sicura, con molto minore fatica di quanto richiederebbe un convenzionale linguaggio di programmazione. Dotato di potenzialità quasi illimitate, fornito di un linguaggio di scripting complesso ma dalla grande facilità d'uso (per non contare la presenza di AppleScript), gratificato dall'aggiunta del colore (anche se solo attraverso un XCMD, e dalla presenza di AddMotion, rappresenta un ambiente di sviluppo elastico, potente e alla portata anche di chi si avvicina senza sapere neppure una parola di programmazione.

Questa notevole potenza a disposizione fa perdonare i difetti più o meno piccoli evidenziati all'inizio (grafica in bitmap, ampiezza limitata delle schede, velocità non proprio entusiasmante) ma non bisogna dimenticare che tutto questo viene fornito a un prezzo modesto. Sarebbe chiedere troppo un runtime più veloce? Chissà. Claris-Apple ci hanno da tempo abituati alle lunghe distanze tra una versione e l'altra dei loro pacchetti. Che una (chissà quanto) prossima versione non ci possa accontentare? MC ▶