COSTRUIRE IPERTESTI CON TOOLBOOK

Vasco Badii
Francesco Leonetti
Mario Rotta

PREMESSA: PERCHE' TOOLBOOK ?

Quando Bill Atkinson sviluppò il primo programma per la costruzione di ipertesti pensò ad una pila di schede collegabili tra loro in modo non necessariamente sequenziale. Il programma si chiamava Hypercard, e fu messo a disposizione dei comuni mortali intorno alla metà degli anni '80, per essere utilizzato su computer di tipo Apple Macintosh. Hypercard, giunto alla versione 2.2 , è ancora oggi uno dei migliori strumenti disponibili per la costruzione di ipertesti multimediali; negli ultimi anni, tuttavia, la concorrenza è diventata sempre più aggressiva e vivace, e molti altri programmi si sono imposti su questo particolare mercato. Uno di questi è proprio Asymetrix TM Toolbook, a cui è dedicato questo volume. Perchè Toolbook ? Toolbook può essere considerato l'omologo di Hypercard per chi lavora sui Personal Computer basati su MS-DOS, anzi, su Windows, l'interfaccia grafica dei PC IBM compatibili. Giunto alla versione 3.0, rappresenta un ambiente per lo sviluppo di ipertesti potente e relativamente semplice da usare. Se possedete un computer Apple Macintosh potete naturalmente continuare a costruire i vostri ipertesti con Hypercard, ma se lavorate su un Personal Computer e usate Windows, allora Toolbook, in questo momento almeno, è il programma che fa per voi. Per la verità esistono altri linguaggi ipertestuali in ambiente MS-DOS, come LinkWay della IBM, tanto per citarne uno; e sono già disponibili strumenti "multipiattaforma", che consentono cioé di costruire un ipertesto che possa essere usato su qualsiasi tipo di sistema (Macintosh, Windows, OS/2, ecc.). Ma dei primi è già stato scritto molto, dei secondi è forse prematuro parlare. Concentriamoci quindi sugli strumenti "monopiattaforma" per interfaccia grafica: pensando al grande pubblico, e soprattutto all'immediato futuro della tecnologia informatica, riteniamo che i modelli di sviluppo da tenere maggiormente presenti siano proprio quelli offerti dai due prodotti sopra citati, Hypercard e Toolbook. La scelta di dedicare la nostra attenzione a Toolbook e quindi allo sviluppo di ipertesti su Personal Computer IBM compatibili basati su Windows non è subordinata ad una risposta a domande del tipo: "qual è il più facile ?", "qual è il più potente ?", "qual è il più economico ?". Alcuni ritengono che Hypercard abbia dei vantaggi, altri si trovano benissimo con Toolbook. La domanda che ci siamo posti, piuttosto, è un'altra. Quale tipo di computer è più diffuso nelle scuole ? La risposta, in questo caso, non può che essere una: PC IBM compatibile con MS-DOS e WINDOWS. Ecco perchè gli autori, che si avvalgono delle esperienze avviate da alcuni anni nell'ambito del Corso di Perfezionamento in Informatica per la Didattica delle Discipline Umanistiche e per l'uso Educativo dei Beni Culturali promosso dal Laboratorio di Tecnologie dell'Educazione della Facoltà di Magistero dell'Università di Firenze, hanno scelto in questo volume di presentare Toolbook. Durante i corsi svolti dal Laboratorio, si insegna ai partecipanti a costruire ipertesti usando proprio Toolbook, con risultati molto interessanti: persone che non conoscevano affatto l'uso del computer, se non a livello molto elementare, imparano, usando questo strumento, a prendere confidenza con un linguaggio di programmazione, tanto da riuscire, in poche giornate, a realizzare degli ipertesti o dei piccoli multimedia. Probabilmente, ciò si deve anche alla fortunata metafora scelta dai professionisti della Asymetrix come elemento base della struttura operativa di Toolbook. La metafora, infatti è quella del libro. Non tutti sanno che cos'è e come funziona un programma, ma tutti sanno che cos'è e come è fatto un libro: l'idea di costruire un libro, anche se si tratta di un libro elettronico, è particolarmente stimolante per gli aspiranti autori di ipertesti, quasi tutti di formazione umanistica, intimoriti non solo dall'idea di dover affrontare un linguaggio informatico, ma spesso a disagio anche con metafore più lontane dalla loro cultura. Toolbook si sta rapidamente diffondendo nelle scuole, e anche molti programmatori professionisti lo usano. Questi ultimi non hanno bisogno di un manuale come questo per orientarsi all'interno del programma. Gli insegnanti, invece, e più in generale molti autori di ipertesti non professionisti, lamentano una certa difficoltà di orientamento all'interno dei segreti dell'ambiente di programmazione, anche perchè, nonostante siano disponibili versioni di Toolbook parzialmente in italiano, la manualistica allegata al pacchetto della Asymetrix è in lingua inglese, e ha caratteristiche molto spinte dal punto di vista tecnico, tanto da risultare piuttosto difficile a quanti non

sono già ben dentro l'argomento. Accade così che molti, pur comprendendo la facilità d'uso e la praticità del linguaggio, lamentino la mancanza di un adeguato supporto manualistico, semplice e in lingua italiana: lo scopo di questo volume è anche quello di colmare questa lacuna.

Antonio Calvani
Laboratorio di Tecnologie dell'Educazione
Università di Firenze

MACCHINE E PROGRAMMI

Hardware e software

Il computer si presenta a noi come un insieme di due componenti: Macchina e Programmi. In inglese: *Hardware* e *Software*. La Macchina è tutto ciò che noi vediamo e tocchiamo: la tastiera, il video, la scatola con le luci lampeggianti dove alle volte inseriamo i dischetti, ecc.
E i Programmi ? I Programmi costituiscono l' imponderabile, l' inafferrabile, potremmo dire l'anima del computer. Spesso sono confusi con i dischetti, che invece, più semplicemente, li contengono ma non sono essi stessi programmi.

I programmi definiscono il comportamento della macchina. Se sul video vediamo apparire la figura di una cartellina colorata di rosso, è perché c'é un programma che "dice" ad una parte della macchina (il video) di visualizzare quella figura e di colorarla in quel modo. Se il computer, quando lo accendo, mi dice "Benvenuto !", dandoci una sensazione di amichevole confidenza, è perché un programma gli ha detto di comportarsi così.

Le macchine le fanno le fabbriche. I programmi li fanno i Programmatori. Dopo aver letto questo libro e aver sperimentato gli esercizi e gli esempi proposti, anche voi diventerete dei Programmatori di Ipertesti. O almeno, lo speriamo.

Da quando, per la prima volta, il computer è entrato negli uffici o nelle case, da quando, cioè, esiste ciò che chiamiamo Personal Computer, relativamente alle macchine, all' *hardware*, non è cambiato molto. L'aspetto del computer è sempre lo stesso: una tastiera, una scatola, un monitor, una stampante. Inoltre la macchina continua a ragionare in sistema binario, con i suoi bravi 0 e 1. Magari sempre più velocemente e con sempre maggiore potenza potenza di calcolo, ma sempre di 0 e di 1 si tratta.

Per quel che riguarda i programmi, invece, si è assistito ad una vera e propria rivoluzione concettuale. Sono i programmi, il *software*, la chiave del successo definitivo del computer. Non a caso l'uomo più ricco d'America non è il Presidente della IBM, che è una fabbrica di computer, ma Bill Gates, il Presidente della Microsoft, che è un'azienda che produce *software*, quella che ha inventato il mitico MS-DOS, e che per fortuna non si è fermata lì. Qualcuno di voi ricorda il suo primo incontro con il sistema operativo MS-DOS ? Una serie insistente di "bip" e una sequenza di parole incomprensibili, fino a che, su un noioso schermo nero a caratteri verdastri, non si formava la lettera c seguita da due punti, una barra, e un trattino che lampeggiava senza sosta. Provate, oggi, ad accendere un computer con un sistema operativo a interfaccia grafica: colori, finestre, icone, suoni, è come se dai tempi del vecchio *prompt* (la letterina seguita dai due punti e dalla barra, tecnicamente, si chiamava così) fossero passati secoli, ere geologiche. E invece sono solo pochi anni. Non esistono termini di paragone che possano rendere l'idea di che cosa è stato lo sviluppo del *software* nell'ultimo decennio. Grazie ai nuovi programmi, il computer di oggi comunica in modo più simile al nostro, e noi riusciamo a comunicare con lui più facilmente. Cosa è successo ? Semplicemente, alla comunicazione puramente digitale di una volta, si è aggiunto il livello di comunicazione analogico che l'uomo normalmente usa per interagire.

Analogico e digitale

Questo concetto va chiarito meglio, perchè è molto importante per capire quante e quali siano le potenzialità ancora nascoste in quegli straordinari strumenti della comunicazione che sono gli ipertesti e gli ipermedia. Una comunicazione puramente digitale fa uso solo e soltanto di simboli, ai quali si è convenuto di attribuire un significato, formato e consolidato nei secoli. Il linguaggio scritto è un esempio di comunicazione digitale. Se dobbiamo dire che una cosa è grande, scriviamo "grande", cioé scriviamo in sequenza il simbolo "g" seguito da "r", "a", "n", "d", "e". Noi italiani, quando vediamo scritta quella parola, sappiamo cosa si intende. Magari un giapponese no. Così per dire che una cosa è piccola, scriviamo "piccolo", cioè il simbolo "p", seguito da "i", "c", ecc. Stiamo usando simboli, sequenze di simboli, alle quali il nostro linguaggio convenzionale ha attribuito un significato. La comunicazione, a questo livello, è quindi digitale. Quando parliamo, le cose cambiano. Inevitabilmente, e in particolare noi italiani, quando vogliamo dire che una cosa è grande pronunciamo sì la parola "grande", ma spesso inarchiamo le sopracciglia, allarghiamo le braccia, e quanto più vogliamo comunicare il senso della grandezza della cosa di cui stiamo parlando, tanto più le allarghiamo. Alla comunicazione digitale, in questo caso, si è aggiunta quella analogica. E ora, forse, anche il giapponese di prima riesce a capire qualcosa di ciò che vogliamo dire.

Nel codice analogico, la forma della comunicazione (significante) varia in modo *analogo*, *conforme* a ciò che vogliamo comunicare (significato). Il termometro al mercurio è un esempio di strumento che comunica la temperatura in modo analogico: la colonnina si alza o si abbassa in proporzione alla quantità di calore che sta misurando. I termometri digitali, invece, visualizzano un numero che potrebbe non dirci assolutamente niente, a meno di non conoscere la scala che quel termometro sta usando in quel momento: 32 gradi sono tanti se espressi in Celsius, ma se si usa la scala Fahrenheit, in realtà indicano un freddo polare ! Ebbene, l'uomo comunica non solo a livello digitale, sintattico, ma anche e soprattutto a livello analogico, emozionale, empatico. Potremmo dire che esistono quindi due livelli di comunicazione, di cui si deve tenere adeguatamente conto, perché solo nel loro insieme essi definiscono il significato reale del messaggio. Nella sola comunicazione digitale, infatti, si nasconde un grosso limite. Ad esempio, se l'editore ci chiede: "avete finito di scrivere il libro sugli ipertesti ?" noi possiamo rispondere: "Sì, domani". Letteralmente, prendendo cioè per buona la sola comunicazione digitale, il giorno dopo dovremmo aver consegnato il libro. In realtà, l'intonazione della voce (che è un elemento analogico, non digitale) lasciava sottintendere che la nostra era una risposta ironica, per cui il giorno dopo il libro non sarebbe stato pronto per niente. La componente analogica, talvolta, stravolge completamente il significato che l'elemento digitale sta trasportando sui suoi simboli. Nel linguaggio scritto si cerca, con la punteggiatura per esempio, di ovviare a questa carenza di efficacia comunicativa del codice digitale, ma non sempre ci si riesce, e, comunque, ascoltare direttamente il tono della voce è tutta un'altra cosa che riprodurne il senso con punti esclamativi, punti di sospensione, ecc. Il limite dei computer è stato a lungo quello di essere soltanto freddi strumenti digitali. Che fare, dunque, per renderli analogici ? Colpo di genio: aggiungere la grafica, i colori e i suoni. Non è più necessario imparare a memoria dei comandi criptici e delle sigle aberranti. Devi scrivere una lettera ? Eccoti una scrivania, un foglio, un cestino, delle cartelline dove raccogliere le tue lettere, penne e inchiostri a volontà. Sai farlo nella realtà ? Allora saprai farlo anche sul computer perché ora il computer mostra un mondo *analogo* alla realtà. Certo non si è ancora giunti all' Iperuranio di Platone, ci vuole ancora un pò di tempo per imparare ad usare autonomamente un computer, ma per fare quello che prima richiedeva settimane di studi, supporti di Tecnici, e nottate insonni, ora sono sufficienti pochi giorni di "affiatamento" accompagnati da un buon manuale, o meglio ancora da una guida ipertestuale.

Veniamo dunque agli ipertesti, questi sconosciuti di cui tutti parlano. Cerchiamo di darne una

definizione e capiamone la componente analogica e digitale. Immaginate ora di sentire in lontananza l' Inno alla Gioia della Nona Sinfonia di Beethoven. Qualcuno ha lo stereo acceso ? Ma quale stereo ! E' dal computer che proviene questa musica ! Sì, proprio da quel computer che dieci anni fa, quando si accendeva, faceva solo "bip". Ora il computer ha la "scheda audio" con il CD-ROM, usa Windows - che nasconde il buon vecchio MS-DOS - e con il CD Audio Player permette di ascoltare anche i Compact Disc musicali. Nel frattempo, mentre le note di Beethoven scorrono, immaginate di leggere sullo schermo una piccola biografia del grande Ludwig e di vederlo, letteralmente, passeggiare lungo le strade di Vienna, mentre il vostro mouse apre e chiude schede o un blocco per gli appunti, su cui potete scrivere, con la tastiera anzichè con la penna, le vostre impressioni su ciò che state ascoltando, vedendo, leggendo.

Non avete afferrato l'ultimo periodo ? Niente paura, tra un pò tutto vi sarà chiaro. Allacciate le cinture, stiamo per decollare.

CHE COSA SONO GLI IPERTESTI ?

Ipertesti e Ipermedia

Gli ipertesti vanno molto di moda di questi tempi, tanto che diverse case produttrici di programmi, pur di stare al passo, spacciano per ipertesti programmi che iper non sono e neanche fanno testo. La naturale applicazione degli ipertesti è nel campo didattico, ma non solo.

Per poter riconoscere un buon ipertesto, e per poterlo quindi realizzare, bisogna sapere con chiarezza che cosa è. Ebbene, si può dire che l'ipertesto è un *testo non lineare*. Notate che in questa breve definizione si fa uso di una negazione (non lineare), per cui, implicitamente, si sta dicendo cosa non è un ipertesto, piuttosto che definire che cosa è. Questo la dice lunga su quanto sia complicato riuscire a definire un concetto che ci è chiaro nella mente ma che non riusciamo ad esprimere. Provate ad esempio a dare una definizione del concetto di *insieme*. Senza usare suoi sinonimi, però. Si può solo cercare di definirlo con degli esempi.

Le pagine che avete letto finora costituiscono un testo lineare. Avete letto in stretta sequenza parole, righe e periodi. Avete cioé seguito linearmente il percorso che vi è stato proposto. La lettura lineare è monodimensionale e monodirezionale: si va da sinistra a destra, si va giù di un rigo e si va ancora da sinistra a destra, si legge così come scriverebbe una vecchia macchina da scrivere: un lettore lineare si comporta come una macchina da scrivere che invece che scrivere, legge: una macchina per leggere ! Ma voi non siete una macchina per leggere ! Magari, in questo momento, avete una matita tra le mani e la fate scorrere lungo le righe. Forse avete sottolineato qualche passaggio. E' probabile che ciò che alcuni di voi hanno sottolineato non coincida affatto con ciò che hanno sottolineato altri. Comportandovi così, facendo cioè delle pause nel corso della lettura, delle riflessioni, dei rimandi ad argomenti che associate a quelli su cui vi siete appena soffermati, facendo tutto questo e seguendo i vostri stimoli, le vostre associazioni di idee, avete, in qualche modo, trasformato un testo lineare in un ipertesto: in un testo *non lineare*. Il concetto di ipertesto, come vedete, non coinvolge direttamente le tecnologie informatiche. Si può fare un ipertesto anche stando in biblioteca e consultando un codice medievale: se ogni qualvolta che ne sentiremo la necessità andremo a prendere un altro libro, cercheremo il brano che ci interessa e poi riprenderemo la lettura precedente, trovandoci alla fine con il tavolo pieno di libri polverosi, e pile di altri volumi a destra e a sinistra, il nostro comportamento sarà stato ipertestuale. E la nostra lettura, certamente *non lineare*, sarà stata probabilmente proficua e interessante, un buon ipertesto, magari, anche se farlo ha comportato sudore e affanno. Certo, sarebbe tutto molto più semplice e comodo se mentre stiamo leggendo fosse il libro stesso che, ad un nostro gesto, si alza, va a prendere *quell'altro* volume, cerca da solo *quel* brano e ce lo propone. Quindi, sempre ad un nostro comando, ritornasse al testo che stavamo leggendo e magari, se è di cori medievali che il codice sta parlando, ci facesse ascoltare un esempio di gregoriano. Ebbene, il computer fa proprio questo quando esegue un programma ipertestuale. Tutto quello che dobbiamo fare è premere un pulsante.
L' ipertesto è dunque una rete di documenti, di testi, legati tra di loro come in una fantastica ragnatela. Consultarlo significa seguire quei legami e saltare da un punto all'altro della rete, ritornare indietro per seguire altri legami, e così via.

L'autore di ipertesti altri non è che il ragno tessitore di questa ragnatela. Solo che non basta che sappia costruire i legami: deve anche saper offrire gli opportuni strumenti per percorrerli, i "pulsanti" che il lettore deve premere, aiutandolo ad uscirne fuori quando prima o poi questi si sentirà perso in un punto senza ricordarsi perché e vorrebbe tanto ritornare a quel libro con cui aveva iniziato la lettura.

Gli elementi collegati di un ipertesto si chiamano NODI e i collegamenti si chiamano LEGAMI, in inglese: *link*. Ci sono diversi tipi di nodi e diversi tipi di legami. Quando i nodi sono soltanto dei testi, allora ci si trova di fronte ad un ipertesto puro. Oggi però gli ipertesti puri sono molto rari. Non c'è libro, per esempio, che non abbia una qualche illustrazione, una foto. Mentre stiamo leggendo una critica sulla Gioconda di Leonardo, vorremmo poter avere davanti l'immagine di quell'opera: in questo caso il legame ipertestuale andrà da un testo ad una immagine. Da un *nodo* di tipo *testo* ad un *nodo* di tipo *immagine*. Così, mentre stiamo leggendo la biografia di Mozart, ci piacerebbe ascoltare qualche brano di quel compositore geniale. Il legame, allora, andrà da un testo ad un suono, da un *nodo* di tipo *testo* ad un *nodo* di tipo *suono*. Infine, leggendo un commento sul peso che ha avuto Charlie Chaplin nell'evoluzione del cinema, la nostra mente vorrebbe andare a rivedere quella certa sequenza del film "Tempi Moderni". Il legame ci porterà dunque ad un *nodo* di tipo sequenza filmata, *video*. I nodi di un ipertesto, quindi, possono essere *multimediali*, ovvero condividere molteplici media, codici di comunicazione. L' ipertesto è diventato un IPERMEDIA.

Il computer, oggi, rappresenta l'unico strumento tecnologico con il quale è possibile costruire *ipermedia*: oggetti che si lasciano consultare proponendovi testi, suoni ed immagini, secondo la vostra curiosità ed interesse, lasciandovi esplorare e "navigare" attraverso questa rete di nodi, sempre pronti e recuperarvi se vi sentite persi.

Data questa definizione di ipertesto e di ipermedia, cercate voi stessi di individuare quali sono gli elementi di comunicazione analogica che tali oggetti consentono e quelli digitali di cui dispongono. Proviamo ad esempio a capire in che modo, in un contesto ipermediale, può essere affrontato e risolto il problema della comunicazione del senso ironico di un'affermazione. Supponiamo che si voglia dire, ironicamente: "che bella giornata !". Senza ipermedia, senza grafica, senza suoni, con un vecchio computer MS-DOS, avremmo potevo solo visualizzare sul monitor, con caratteri verdastri, la frase "che bella giornata !" Nonostante il punto esclamativo, o forse proprio per quello, tutti avrebbero capito solo il senso letterale di questa frase. In un ambiente ipermediale possiamo invece visualizzare sul computer la stessa frase, ma, nello stesso tempo, mettere sullo sfondo delle parole l' immagine di uno spaventoso temporale, magari accompagnata da suoni di lampi, vento, tuoni e rovesci d'acqua. Il senso diventa chiaro.

Un ipertesto, o meglio ancora un ipermedia, rivela un processo cognitivo della nostra mente, che integra in una rete, informazioni di diverso tipo e natura, riuscendo a rappresentare anche gli elementi emozionali della nostra conoscenza. Tanto più un ipertesto riesce a coincidere con la rete della nostra mente, tanto più riesce a comunicare informazioni ed a coinvolgerci intimamente. Ecco spiegato il successo e l'efficacia didattica di questi strumenti rispetto a quelli tradizionali.

Il proposito di questo libretto è quello di illustrarvi i concetti generali relativi alla costruzione di un ipermedia, da buoni ragni, e quindi di mostrarvi, con stile da "manuale", uno degli strumenti usati per tessere la ragnatela: Toolbook. A proposito: il nome è una contrazione inglese che letteralmente significa "strumento per fare libri". Libri elettronici e multimediali, naturalmente.

Nodi e link

Dicevamo che un ipertesto è una rete di documenti (*nodi*) collegati tra di loro dai cosidetti "legami ipertestuali" (*link*). Il tipo di nodo definisce il tipo di ipertesto: se i nodi sono tutti documenti testuali, allora l' ipertesto è puro; altrimenti, se coesistono immagini grafiche, suoni, o sequenze in movimento, allora l' ipertesto è un ipermedia.

Il nodo è perciò l' unità compositiva elementare di un ipertesto. Esso può essere costituito di poche righe visualizzate su una finestrella, oppure occupare un' intera videata e contenere sia testo che immagini. Le combinazioni sono molteplici e si è liberi di scegliere la forma che riteniamo più adatta per costruire il nodo.

Una volta costruiti i nodi, si dovranno poi creare i legami, i collegamenti che rendono vivace e dinamico l' ipertesto. E' seguendo questi *link* che l'utente esplora i nodi, ovvero *naviga* nelle informazioni dell' ipertesto seguendo di volta in volta i legami che in quel momento ritiene più interessanti o più coerenti con la sua ricerca. Un legame è caratterizzato dal fatto che unisce il nodo di partenza, quello che in questo momento stiamo consultando, con uno o addirittura più nodi (raggruppati) di arrivo. Il richiamo del nodo di arrivo avviene nel momento in cui attiviamo il *link*.

L'attivazione del *link* dipende dal modo in cui esso è rappresentato nel nodo di partenza. Innanzitutto in un nodo ci possono essere più legami che richiamano altri nodi. E' importante quindi evidenziarli e in qualche modo esplicitare che cosa essi farebbero (a che nodo ci manderebbero) se venissero attivati. Ci sono diverse tecniche per fare ciò. Tali tecniche definiscono implicitamente anche i tipi di *link*.

Una tecnica consiste nell' inserire nel nodo un "pulsante" che, se cliccato (neologismo, dall' inglese *to click*: premere il pulsante del mouse quando questi è posizionato su un oggetto visualizzato sullo schermo), attiva il legame e richiama il nodo collegato. Il pulsante ha in genere una "etichetta" che ne definisce il significato. Ad esempio, se il nodo contiene un testo il cui argomento è un' opera di Mozart, un pulsante con l' etichetta "Ascolta il brano" potrebbe richiamare un nodo di tipo "suono", che esegue il brano in questione. Un altro esempio: in un nodo, una "videata", se volete semplificare il concetto, potrebbe esserci l' immagine dell' enigmatica Monna Lisa di Leonardo. Ebbene, un pulsante potrebbe, se attivato, richiamare un altro nodo che mostra un testo descrittivo dell' opera. Questo secondo nodo potrebbe essere visualizzato sulla stessa videata, su una finestra a parte, o in un riquadro di dimensioni prefissate, o addirittura in un' altra videata. I pulsanti, naturalmente, possono anche essere "nascosti", ovvero non essere esplicitamente visibili e non avere etichette. Tornando all'esempio della videata con la Gioconda, si potrebbero posizionare pulsanti siffatti sovrapponendoli all'immagine: l'utente, così, avrebbe l'impressione di fare clic direttamente sull' immagine, anche se in realtà il suo clic attiva il pulsante nascosto sovrapposto ad essa. Non necessariamente un pulsante deve essere attivato cliccando: l'attivazione può avvenire in molti altri modi, ad esempio "sfiorandolo", ovvero passandoci sopra con il mouse, senza premere. Avremo modo, più avanti, di approfondire le tecniche di realizzazione dei *link* tramite i pulsanti e si vedrà quando conviene usare una tecnica piuttosto che un'altra.

Oltre ai "pulsanti", che rappresentano esplicitamente un legame, un'altra tecnica per costruire un legame è quella che usa le *hot-word* (espressione inglese che letteralmente significa: "parola calda"). L'espressione "hot" ci fa capire che si sta parlando di qualcosa di attivo, di qualcosa di simile ai pulsanti, qualcosa che ci permette di saltare da un nodo all'altro o di eseguire un'operazione. Alcune parole dei testi del nodo possono essere "hot". Facendo clic sulla parola calda, si genera un evento (il clic del mouse) che "sveglia" la parola facendola reagire. Se il nodo che abbiamo costruito contiene del testo, quindi, alcune delle sue parole possono essere rese calde. Per distinguerle dalle altre, da quelle "fredde", può essere sufficiente attribuire loro un diverso colore, o sottolinearle. Un clic sulla parola potrebbe visualizzare su un riquadro un altro nodo che spiega il significato di quella parola (come in un glossario) o saltare ad un altro nodo dell'ipertesto.

Più semplicemente, gli americani chiamano i pulsanti e tutto ciò che permette di interagire con l'ipertesto *hot-spot* ("punto caldo"). Come abbiamo già detto, il nodo di un ipertesto deve contenere, oltre alle informazioni che esso vuole comunicare (e che possono assumere qualsiasi forma), anche i "legami" che, una volta attivati, vanno a richiamare altri nodi. Tutto ciò che reagisce ad una nostra azione, di solito effettuata con il mouse, più raramente usando la tastiera, è detto "zona calda", in contrapposizione all'inerte e freddo nodo che si limita a mostrare le informazioni. Gli *hot-spot* sono dunque le zone calde dell'ipertesto, quelle che reagiscono agli eventi che noi provochiamo e ci permettono di interagire con l'ipertesto. I pulsanti e le "parole calde" di cui si è parlato sono un esempio di *hot spot.*

Ma in che senso un *hot spot* reagisce ad un evento da noi provocato ? E cosa sono in realtà questi "eventi" ? In che modo si può istruire un *hot spot* (un pulsante, una "parola calda", o qualche altro "oggetto") perchè reagisca ad un evento in un certo modo piuttosto che in un altro ? E come si può definire il comportamento delle zone calde ? Calma, calma. Tornate alle pagine iniziali: si diceva che ciò che definisce il comportamento del computer si chiama "programma". Quindi, a questo punto, le questioni che ci stiamo ponendo coinvolgono l'attività di programmazione: è la programmazione che permette di definire il comportamento di un ipertesto e di assegnare un comportamento a tutti i suoi *hot-spot*. Cominciamo dunque a entrare nel merito.

Costruire ipertesti

Abbiamo già detto che per costruire un ipertesto non è strettamente necessario un computer. Vogliamo provare ? Bene. Prendete dei fogli di carta, una penna, dello spago, delle etichette da "pacchi postali", colla e forbici. Supponiamo di voler fare un ipertesto su Leonardo da Vinci. Procuriamoci un'immagine di Leonardo, il classico autoritratto con il barbone (che si conserva nella Biblioteca Reale di Torino), ritagliamola e incolliamola sul primo foglio. Accanto all'immagine incolliamo anche un rettangolino di carta su cui scriveremo alcune notizie biografiche essenziali su Leonardo. Di fatto, abbiamo creato un nodo che contiene testi ed immagini. Di nodi come questo ne possiamo fare altri: una pagina che parla di Leonardo inventore, un'altra su Leonardo pittore, un'altra ancora su Leonardo scultore, una sul contesto storico in cui Leonardo è vissuto, ecc. Prepariamo anche altri nodi, altre pagine contenenti ciascuna un'opera di Leonardo (un dipinto, un affresco, una scultura, un progetto meccanico, ecc.) con relativa descrizione. Alla fine ci ritroveremo con una serie di fogli, appiccicosi e odoranti di colla, messi lì sul tavolo in ordine sparso. Abbiamo predisposto i nodi. Formiamo ora una pila mettendoli uno sopra l'altro. In testa alla pila mettiamoci il primo foglio che abbiamo preparato, quello con il barbone di Leonardo. Ma in realtà, poichè è un ipertesto quello che vogliamo fare, non è necessario che i nodi vengano disposti secondo una precisa sequenza.

Ora costruiamo i *link*, i collegamenti. Sul primo foglio, nel punto in cui si parla dell'eclettismo di Leonardo, facciamo un foro, in cui faremo passare un pezzo di spago, che poi annoderemo. Sullo spago applichiamo un'etichetta con su scritto "Leonardo Pittore". Andiamo poi ad annodare l'altra estremità dello spago proprio sul foglio che parla di Leonardo pittore. Ecco, abbiamo creato un *link*. Allo stesso modo, potremmo collegare al primo foglio altri fogli, quelli su Leonardo scultore, Leonardo scienziato, ecc. Da ognuno di questi fogli potrebbero poi partire altri "spaghi", che verranno collegati alle pagine su cui avevamo inserito le opere di Leonardo, e così via, fino a che non avremo creato quella "ragnatela" di cui si diceva, una rete in cui i collegamenti non siano semplicemente del tipo foglio precedente - foglio successivo, come in un normale libro, ma in cui, seguendo gli spaghi (come seguendo il "filo" di un discorso), si possa saltare da una pagina ad un'altra anche se quest'ultima è molto lontana da quella di partenza nella pila di fogli che abbiamo creato.

Ecco a voi un'ipertesto. Molto artigianale, certo, probabilmente di difficile consultazione. Ma di un vero e proprio ipertesto si tratta.

Note metologiche

Abbiamo ritenuto opportuno mostrare prima di tutto come si può costruire un ipertesto "cartaceo" perché le analogie tra l'esempio appena illustrato e le modalità di costruzione di un ipertesto "elettronico" sono molte. L'esempio può aiutarci quindi a comprendere meglio, almeno in una prima fase, i concetti che stanno alla base della progettazione e della realizzazione ipertestuale. Un ipertesto, evidentemente, deve essere progettato prima di essere realizzato sul computer. E il progetto, spesso, viene sviluppato su tradizionali supporti cartacei, viene "disegnato" sulla carta, riportato in una "mappa" sulla quale vengono evidenziati i nodi e tracciati i *link* che li collegano. Una buona "mappa" è uno dei segreti che stanno alla base della costruzione di un buon ipertesto, e sarebbe interessante sviluppare meglio, in questa stessa sede, le questioni metodologiche legate al problema delle fasi e delle modalità di sviluppo di un ipertesto. Si tratta, però, di un argomento così importante e complesso da non poter essere esaurito in poche righe. Il nostro scopo, del resto, è quello di affrontare solo l'ultima delle fasi di costruzione di un'ipertesto, la sua realizzazione attraverso l'uso di uno strumento informatico, nella fattispecie Toolbook. A livello metodologico, possiamo soltanto limitarci ad alcune note, riferendoci in modo particolare allo sviluppo di ipertesti con finalità educative e didattiche. Possiamo riassumere il problema in questi termini: progettare un ambiente di apprendimento ipermediale implica l'individuazione delle stesse componenti identificabili nel progetto di un sistema educativo. Ad esempio:

- si dovrà tener conto delle **caratteristiche del discente**: conoscenza pregressa, stile di apprendimento, motivazione;
- si dovranno aver chiari gli **obiettivi**, ovvero analizzare il contesto e individuare i risultati che si vogliono raggiungere;
- si dovrà mettere a punto un **modello pedagogico**: tutoriale, simulazione, esercitazione e pratica, ecc.;
- si dovrà stabilire quale sarà l' **interfaccia**: quali saranno le modalità di navigazione dell'ipertesto, la grafica e il linguaggio di comunicazione;
- si dovrà chiarire la **struttura** dell'applicazione: stabilire, ad esempio, se debba essere gerarchica, con argomenti e sottoargomenti, o associativa, con parole o icone come links, ecc.;
- si dovrà chiarire e definire il **contenuto**: quali sono gli argomenti che dovranno essere trattati ed esplorati dal discente;
- si dovrà scegliere un **formato**, ovvero stabilire quale mezzo utilizzare per presentare il contenuto (testo, suoni, grafici, animazione, ecc.).

Al termine della fase di progettazione, dunque, si dovrebbe giungere alla messa a punto di un prototipo dell'ipertesto, che, come tale, potrà essere rappresentato sulla carta, così da poter essere la guida per la fase successiva, quella di cui ci occuperemo più estesamente: l'assemblaggio, la realizzazione. Per quanti volessero approfondire il problema metodologico, che, ripetiamo, è di primaria importanza rispetto al problema dell'uso degli strumenti di sviluppo come Toolbook, ricordiamo che sull'argomento è ormai disponibile una vasta letteratura, un cenno alla quale è riportato nell'appendice bibliografica in coda a questo volume. Per il momento, pur non ritenendo queste note sufficienti a chiarire le implicazioni concettuali delle tecniche ipertestuali e dell'uso degli ipertesti nella pratica didattica, non possiamo fare altro che procedere sulla strada che abbiamo intrapreso, dando per acquisita, da parte degli aspiranti autori di ipertesti che stanno leggendo queste

parole, una riflessione attenta sugli argomenti qui accennati.

IL NOSTRO PRIMO IPERTESTO

Per cominciare...

Supponiamo quindi di sapere già, e di aver già capito che cos'è un ipertesto, e di aver chiaro non solo a che cosa serve in generale, ma che cosa, personalmente, vogliamo che sia il "nostro" ipertesto. Si tratta, ora, di realizzarlo, usando una tecnologia informatica e un linguaggio di programmazione. Sulla scia dell'esempio già utilizzato, supponiamo di voler rendere "elettronico" quell'ipertesto su Leonardo da Vinci cha avevamo costruito con i fogli di carta, la colla e lo spago. In fondo, se avete le mani già sporche per aver messo insieme quella pila di schede, l'idea di "sporcarsi le mani" accendendo il computer e utilizzando uno strumento di programmazione - quella stessa idea che spesso spaventa tutti coloro che informatici non sono - vi sembrerà meno sconvolgente.

Nell'esempio accennato, avevamo raccolto dei **materiali**, un po' come se fossero gli "ingredienti" dell'ipertesto:

Un'immagine di Leonardo da Vinci (l'autoritratto con la barba lunga)
Una serie di notizie biografiche
Un'immagine della Gioconda
Notizie e testi di approfondimento su "Leonardo Pittore"
Un'immagine tratta dagli appunti scientifici dell'autore (ad esempio lo studio sull'uomo)
Notizie e testi di approfondimento su "Leonardo Scienziato"

Fermiamoci qui, per il momento. L'esempio, seppure molto semplice, può già ritenersi significativo. Al limite, potremmo aggiungere a quanto già raccolto un breve scritto introduttivo e una serie di piccole schede dedicate ad alcuni personaggi contemporanei di Leonardo: nulla ci vieta, tuttavia, di integrare il materiale con altre informazioni o immagini, man mano che l'ipertesto prende forma.

Per poter cominciare, dobbiamo sederci davanti a un computer. Ciò di cui abbiamo bisogno è un Personal Computer IBM compatibile che abbia almeno queste caratteristiche:

- PC 80386 DX 33MHz o superiore (ma è raccomandabile almeno un 80486 DX2 a 66 Mhz)
- MS-DOS 3.1 o superiore
- Windows 3.1 o superiore
- 4 MB Ram (ma sono raccomandati 8 o più MB di RAM)
- 1 lettore per dischetti da 1.44 MB
- 1 Hard Disk con almeno 8 MB di memoria disponibile
  (ma possono essere necessari fino a 24 MB a seconda dell'installazione scelta)
- 1 Monitor VGA o, meglio ancora, SVGA, con scheda grafica fino a 16,8 milioni di colori
- Tastiera e mouse
- 1 lettore CD-ROM almeno a doppia velocità (ma non è indispensabile)
- Scheda audio a 16 bit compatibile con lo standard Sound Blaster (ma non è indispensabile)

Niente paura. Abbiamo semplicemente descritto tecnicamente la configurazione di un tipico computer su cui è possibile usare Toolbook e con il quale, quindi, si possono costruire ipertesti. Avremmo potuto anche risparmarvi questo elenco di caratteristiche tecniche dicendovi di accendere un computer sul quale è già stato installato Toolbook, e che quindi risponde alle caratteristiche minime sopra esposte, anche se a voi non dicono nulla. Tuttavia, ci sembra che un accenno alla configurazione

richiesta per lavorare con Toolbook non sia inopportuno: un computer come quello appena descritto non è nè una macchina fantascientifica nè un oggetto particolarmente costoso: è il tipo di computer che oggi viene considerato *entry-level*, ovvero un computer amatoriale e di costo medio-basso, facilmente reperibile. Se decidete di acquistare un computer, oggi, il minimo che possiate pretendere è che corrisponda a queste caratteristiche. Non accontentatevi di niente di meno.

"Lanciare" Toolbook...

Entriamo adesso in Toolbook e vediamo come è fatto. La versione di Toolbook a cui ci riferiamo è la 3.0, l'ultima disponibile alla data attuale. Della stessa versione, esiste un pacchetto con estensioni multimediali.

Una volta acceso il computer, potremmo trovarci in ambiente MS-DOS o direttamente in ambiente WINDOWS. Toolbook "gira" soltanto in ambiente Windows, per cui sarà bene entrare subito nella più amichevole e comprensibile interfaccia offerta da Windows.

MS-DOS, in genere, si presenta con il caratteristico *prompt*:

C:\>_

Il trattino lampeggia (si chiama "cursore") e indica che MS-DOS è pronto a ricevere i nostri "comandi". Chiunque abbia usato il computer almeno un paio di volte non ha certo bisogno di ulteriori spiegazioni. Ricordiamo, comunque, che per passare a Windows bisogna scrivere un'istruzione:

C:\>WIN

Quindi premete "**INVIO**". Se tutto è andato bene, e se Windows è correttamente installato, dovreste trovarvi di fronte ad una videata di questo tipo:

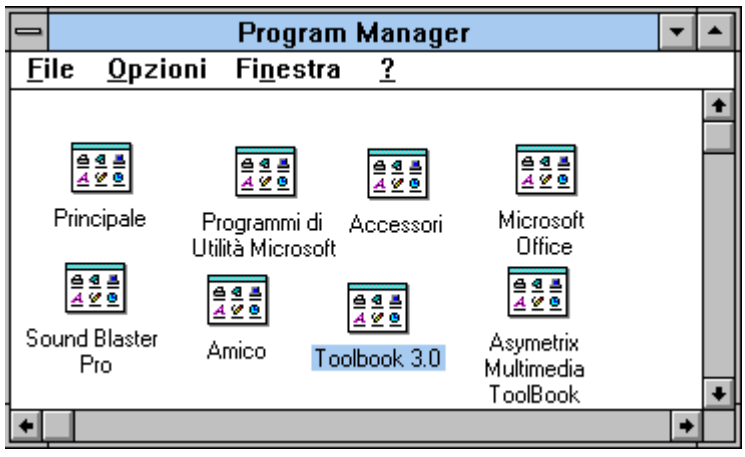

Figura 1: come si presenta il Program Manager di Windows

La figura che abbiamo allegato, naturalmente, non coinciderà esattamente con ciò che vedrete sullo schermo del vostro computer. Windows simula una "scrivania" su cui si possono accumulare cartelle

(o meglio, "finestre" e "icone"), e, si sa, ognuno mette sulla sua scrivania ciò che vuole e si orienta meglio nel *suo* ordine, o disordine.

Cercate ora, sulla vostra scrivania Windows, un'icona denominata "**Asymetrix Toolbook 3.0**", o "**Asymetrix Multimedia Toolbook**", o "**Toolbook**", o che comunque evochi la presenza del programma che state cercando. Con il mouse posizionate la freccia in corrispondenza dell'icona e fate "doppio clic" (oppure fate un clic e premete "invio"), in modo da aprire la finestra del gruppo Toolbook.

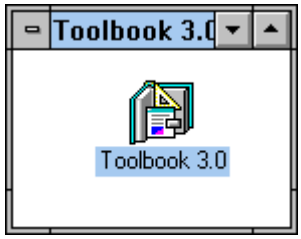

Figura 2: l'icona per lanciare Toolbook

Se non trovate la finestra e non c'è nulla che evochi il programma, è probabile che sul vostro computer Toolbook non sia stato installato. Dovrete dunque installarlo, utilizzando i dischetti, o il CD-ROM, allegati al pacchetto della Asymetrix. Rimandiamo agli approfondimenti e alle appendici per quel che riguarda le istruzioni e i consigli sull'installazione del programma. Se invece avete trovato la finestra e l'avete "aperta", cercate, in essa, l'icona corrispondente al programma Toolbook vero e proprio (le altre icone che troverete si riferiscono ad applicazioni allegate come esempi, alle guide di supporto che integrano il programma e ad alcuni strumenti utili per svolgere particolari operazioni). Individuata l'icona - in genere è la prima nel gruppo - ripetete la stessa operazione di cui sopra (doppio clic o clic e "invio"). Se non ci sono problemi, ovvero se Toolbook è correttamente installato, sarete accolti da un messaggio di presentazione e, subito dopo, si aprirà un banco di lavoro per la costruzione di ipertesti che si presenta come nella **figura 3**.

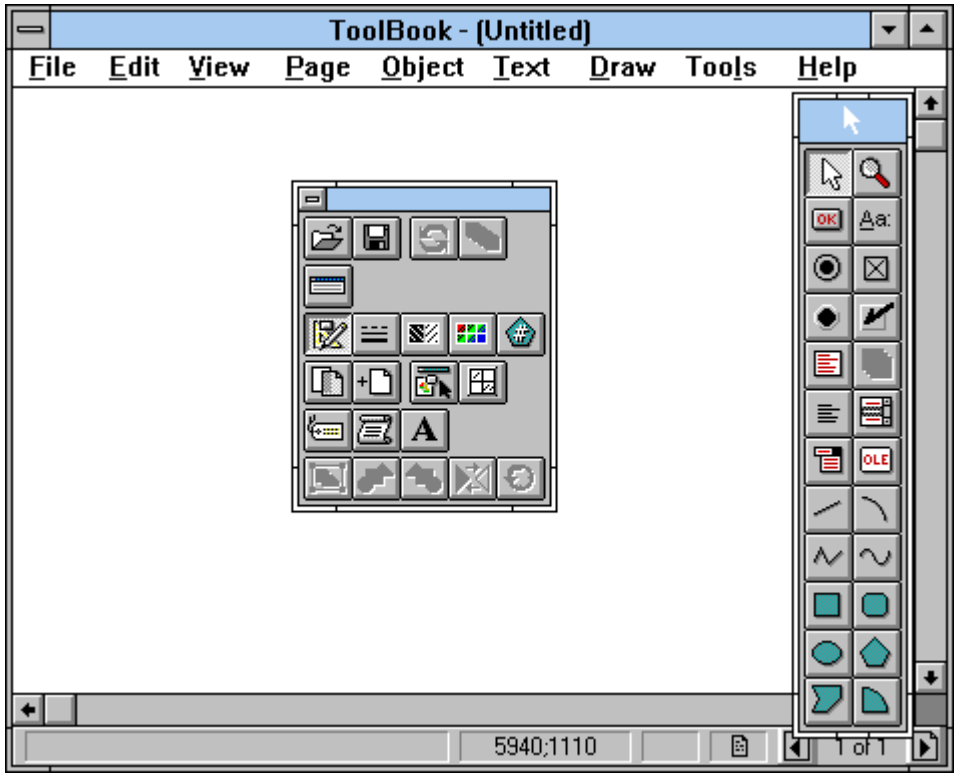

Figura 3: l'ambiente di lavoro di Toolbook

Avete "aperto" Toolbook. Alcuni dicono che lo avete "lanciato". Come tutti i programmi Windows, Toolbook può essere "aperto" o "lanciato" direttamente da MS-DOS, scrivendo questo comando:

C:\>WIN tb30 <INVIO>
(Se avete installato la versione 3.0 non multimediale di Toolbook)

oppure:

C:\>WIN mtb30 <INVIO>
(Se avete installato la versione 3.0 multimediale di Toolbook)

In qualunque modo lo abbiate aperto, ora il programma attivo è Toolbook, e possiamo finalmente iniziare a creare il nostro primo ipertesto.

Ricordate l'ipertesto cartaceo ? Com'era stato fatto ? Usando fogli, ritagli di giornali o libri per le immagini, ritagli di testi, spaghi, etichette, ecc. Era fatto, cioè, di "oggetti", ciascuno con delle caratteristiche fisiche (forma, contenuto, ecc.), successivamente organizzati ed integrati secondo uno schema logico ipertestuale. Con Toolbook si lavora in un modo analogo.

Una scrivania, un libro...

L'ambiente di lavoro Toolbook può essere paragonato ad una scrivania su cui sono disposti, più o meno ordinatamente, tutti gli strumenti dei quali lo scrittore di ipertesti avrà bisogno per costruire il suo programma. Lanciando Toolbook, ad esempio, sullo schermo apparirà sembre un piccolo box con i simboli di una serie di oggetti; inoltre, verrà automaticamente messo a disposizione dell'utente un altro box, con una serie di pulsanti che attivano le funzioni più comuni, come "apri", "salva", "taglia", "copia", "incolla", ben note agli utenti Windows e il cui significato è abbastanza facilmente intuitibile anche per i neofiti. Le stesse funzioni riportate sotto forma di pulsanti in quest'ultimo box sono richiamabili attraverso la barra dei menu, che si trova in alto, subito sotto la barra colorata dove di solito è scritto "Toolbook - (Untitled)". La barra di menu è una caratteristica dell'ambiente operativo Windows: si presenta come una striscia su cui sono scritti i nomi di alcune funzioni, o meglio, di alcuni gruppi di funzioni: cliccando su uno dei nomi con il mouse si aprirà una "tendina" con un preciso elenco di funzioni attive o, a seconda del momento, disattivate (il testo, in questo caso, è scritto in un colore diverso, in genere in grigio chiaro). L'intera finestra di Toolbook, così come i singoli box, possono essere spostati sullo schermo "trascinandoli" con il mouse, tenendo il tasto premuto sulla barra colorata della finestra o dei box. Non possono invece essere automamente trascinati nè i menu nè la barra grigia che vedrete in basso, e che viene detta *status box*, perchè in essa sono riportate alcune informazioni essenziali - e molto utili - che riguardano il programma che state costruendo. I box, i menu e la barra di stato, si diceva, sono un po' come degli utensili appoggiati su una scrivania. Quelli che abbiamo descritto, naturalmente, sono solo alcuni degli utensili disponibili, quelli che Toolbook ritiene indispensabili. Ce ne sono tuttavia molti altri (ad esempio la "tavolozza" dei colori), che potranno essere richiamati di volta in volta, quando e se saranno necessari nella costruzione dell'ipermedia, per essere magari richiusi subito dopo l'uso, anche perchè gli strumenti, spesso, impediscono di valutare come si deve lo spazio bianco che si trova all'interno della finestra principale, che è la nostra area effettiva di lavoro. Naturalmente, un utente smaliziato impara ben presto ad adoperare - così come fa già usando la sua videoscrittura preferita - i cosidetti tasti di scelta rapida, ovvero delle combinazioni attivabili con la tastiera che permettono di richiamare quasi tutte le funzioni del programma: tutto diventa più veloce e gratificante, e, se si è diventati bravi, ci si può permettere perfino di "chiudere" alcuni box o di eliminare la barra dei menu, per poter disporre meglio dell'area di lavoro e mantenere su di essa una migliore "visibilità". Tra i primi consigli che possiamo dare ai neofiti di Toolbook c'è anche questo: imparare a prendere confidenza con gli strumenti e i menu e, contemporaneamente, con le combinazioni di tasti che attivano le stesse funzioni richiamabili dai box o dai menu. Nelle schede di approfondimento allegate a questo volume verranno riportati in molti casi i tasti di scelta rapida collegati a determinate funzioni.

Torniamo ora alla videata di apertura di Toolbook. Lo spazio bianco è la nostra area di lavoro, si è detto. Ma in che cosa consiste ? Che cosa rappresenta ? La metafora scelta da Toolbook è semplice da capire: quello spazio bianco è la pagina di un libro. Una pagina ancora vuota di un libro un po'particolare, certo, ma del tutto analoga ai fogli di quella pila che, ricordate...

Creare le pagine, dare loro un nome...

Così come quello cartaceo, anche l'ipertesto elettronico è fatto di "oggetti". Il più evidente di questi oggetti è proprio la pagina, il foglio di carta. Quella che in Toolbook appare come uno spazio vuoto su cui "galleggiano" i box con gli strumenti è proprio una pagina bianca pronta per essere da noi riempita. Osservate ora la barra dello *status*, in basso: a sinistra c'è scritto "page 1" e a destra "1 of 1". Vuol dire che noi ci troviamo a pagina 1 di un libro elettronico che ha soltanto una pagina ("1 of 1"). Lavoriamo su questa, per il momento.

Cominciamo con il definire il nome e i contenuti della pagina. Il nome ? Certo, il nome. Le pagine di un qualsiasi libro sono generalmente identificate attraverso un numero ("vai alla pagina 4", "confronta a pag. 25", ecc.) e su quella base si può comunque trovare la pagina che ci interessa. Il numero è una proprietà implicita della pagina: dipende infatti dalla sua posizione "fisica" nella sequenza - di pagine - che nell'insieme costituisce il libro. Anche le pagine del libro elettronico che è possibile costruire con Toolbook hanno un numero - lo abbiamo già visto indicato in basso, sulla barra di *status* - che corrisponde alla loro sequenza "fisica". Ma mentre in un libro cartaceo le pagine possono essere sfogliate solo sequenzialmente - e quindi è sufficiente indicarle con il loro numero - in un libro elettronico, in un ipertesto, si può saltare da una pagina all'altra *non sequenzialmente*: il numero non basta, non sempre almeno. Così, in Toolbook, possiamo, anzi, *dobbiamo* dare esplicitamente un "nome" alla pagina, assegnare ad essa una sorta di etichetta (come avevamo fatto nell'ipertesto cartaceo), riferendoci alla quale potremo orientarci meglio tra i nodi dell'ipertesto. Dire "vai a pagina 4" o dire, invece, "vai alla pagina denominata 'LEONARDO PITTORE'", infatti, sono due cose ben diverse. Il riferimento al "nome" di una pagina è molto più significativo che non il riferimento ad un asettico numero. Inoltre, dare un nome alle pagine permetterà prima di tutto a noi, autori dell'ipertesto, di ritrovarle facilmente senza bisogno di "sfogliare" il nostro libro elettronico. Ricordate ? Le pagine sono dei nodi: che cosa pensate che sia più facile ? Attivare un *link* tra due pagine indicandole con un numero (che quasi sempre non ricordiamo) o indicandole con il nome che avremo attribuito loro su base "logica" ? E' buona norma, quindi, denominare *sempre* le pagine, facendo tuttavia attenzione, quando il nostro ipertesto comincerà a "crescere", a non dare mai lo stesso nome a due pagine diverse, perchè in tal caso Toolbook non sarebbe in grado di distinguerle.

La pagina, per Toolbook, è un *oggetto* (vedremo di chiarire meglio, più avanti, che cosa significa questo). Il nome di una pagina è quindi una delle "proprietà" o "attributi" con le quali Toolbook descrive e identifica gli "oggetti" di cui è costituita qualsiasi applicazione generata con il programma. Per indicare o modificare le proprietà di un oggetto si deve selezionare, nel menu **Object**, la voce corrispondente all'oggetto che a noi interessa. In questo caso la voce **Page Properties**. Cliccate quindi sulla voce **Object** della barra di menu di Toolbook e successivamente cliccate su **Page properties.**

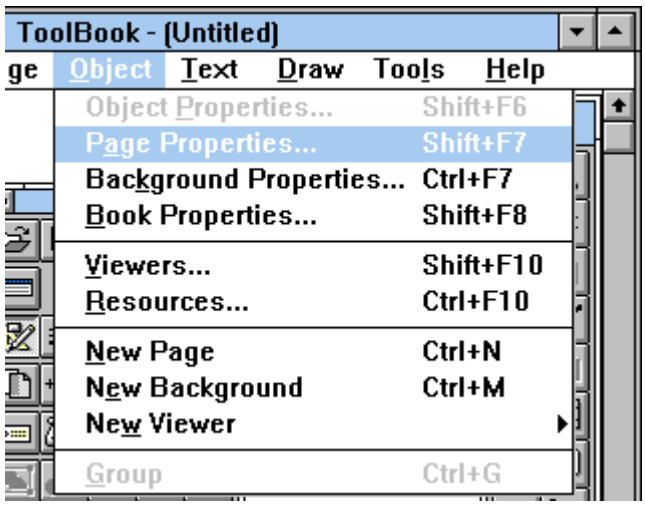

Figura 4: il menu "Object" e le voci corrispondenti

Apparirà una finestra di dialogo come quella della **figura 5**.

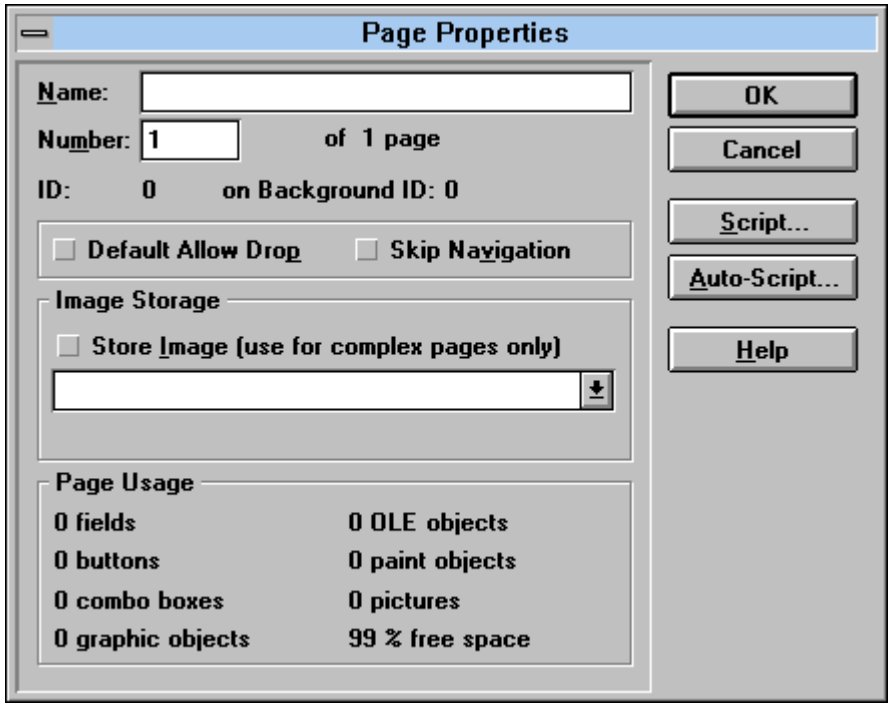

Figura 5: il "dialog box" che consente di definire e modifcare le caratteristiche e le proprietà di una pagina di Toolbook

Cliccate ora nella casella **Name** e quindi scrivete il nome che volete dare alla pagina. Continuando con il nostro esempio, si potrebbe scrivere "inizio", perché questa è proprio la pagina iniziale del nostro ipertesto. Ma avremmo potuto chiamarla in qualunque altro modo. Confermate successivamente le modifiche apportate alle proprietà della pagina cliccando sul pulsante **OK**. La finestra di dialogo si chiuderà e si tornerà alla pagina bianca, che ora, però, ha anche un nome: "inizio". un nome che manterrà finchè non decideremo di cambiarlo.

Inserire un'immagine sulla pagina...

Quali altri oggetti possiamo adesso "mettere" su questa pagina ? Sempre sulla scia del nostro esempio, diciamo che vorremmo applicare sul foglio, che è ancora bianco, almeno l'autoritratto di Leonardo (o qualche altra sua opera emblematica) e un riquadro che contenga il testo di una breve biografia sul genio di Vinci.

Cominciamo a vedere come si possono applicare le immagini sulla pagina. Le immagini possono essere tratte da foto o illustrazioni di libri, giornali, ma devono essere inserite nel computer tramite un particolare dispositivo, detto *scanner*, una macchina che con un comportamento analogo a quella di una fotocopiatrice riproduce qualunque immagine *digitalizzando* il corrispondente analogico, ovvero cartaceo. Le immagine digitalizzate attraverso lo *scanner* verranno archiviate nel computer (più propriamente, nel disco rigido del computer) e potranno essere ritrovate e riutilizzate dai programmi - come Toolbook - attraverso il nome che è stato dato loro al momento del "salvataggio", al momento

cioè dell'archiviazione su disco. Gli autori di ipertesti, generalmente, oltre al computer con le caratteristiche minime sopra esposte, dispongono anche di uno *scanner* per poter immagazzinare e digitalizzare le foto e le immagini da utilizzare negli ipertesti. Se non avete uno *scanner* (quelli di buona qualità sono ancora piuttosto costosi), potrete sempre rivolgervi a uno dei tanti *service* esterni, ormai diffusi in tutta Italia perfino nei paesi con "meno di 15.000 abitanti". Il *service* penserà a "scannerizzare" le immagini (ma in italiano si dovrebbe dire "scandire", per analogia con il processo, che si chiama "scansione") e ve le restituirà su dischetti che potrete copiare successivamente sul disco rigido del vostro computer. Attenzione, però ! Non tutti gli scanner funzionano nello stesso modo e non tutti i *formati* grafici sono uguali, nel senso che il processo di digitalizzazione e di salvataggio delle immagini su *files* può avvenire secondo diverse modalità (numero di colori, compressione dei dati ecc.). Toolbook, così come altri programmi, è in grado di "leggere" immagini in diversi formati grafici, ma non in tutti. Bisognerà dunque aver cura di trattare e far trattare le immagini secondo modalità che Toolbook è in grado di riconoscere. A questi problemi sarà dedicato, più avanti, un apposito approfondimento [**A29-A30**].

Andiamo avanti. Supponiamo che voi disponiate già nel vostro computer delle stesse immagini che noi utilizzeremo nell'esempio, o perché le avete "scannerizzate" direttamente o perchè vi siete rivolti ad un *service*. Supponiamo anche che il formato grafico di quelle immagini sia compatibile con quelli che Toolbook può "leggere". Si tratta ora di inserire sulla pagina bianca l'immagine che ci interessa. Per inserire un'immagine sulla pagina scegliete dal menu **File** la voce **Import Graphic**, e quindi andate a cercare l'immagine - a titolo di esempio diciamo che era stata precedentemente archiviata con il nome "leonardo.bmp" (dove *leonardo* è il nome che abbiamo dato al file e *.bmp* l'estensione che indica il formato grafico utilizzato) - che rappresenta, appunto, l'autoritratto di Leonardo con la barba lunga. Andarla a cercare significa frugare all'interno del disco rigido del computer (o all'interno di un dischetto), fino a che non si entrerà nella *directory* su cui l'immagine è stata archiviata (nella finestra di dialogo **Import Graphic** un apposito spazio consente questa operazione). Selezionate, dall'elenco, il nome dell'immagine che volete "importare" e confermate la scelta cliccando sul pulsante **OK**. La finestra di dialogo si chiuderà, e sulla pagina bianca verrà applicata, con il primo vertice in alto a sinistra, la nostra immagine. Cliccate ora su di essa, e, tenendo premuto il pulsante del mouse, trascinatela e spostatela a vostro piacimento fino a che non sarà sistemata nella posizione più opportuna (ad esempio a sinistra, in modo da lasciare dello spazio, sulla destra, per le note biografiche sull'autore). A questo punto la pagina non è più vuota: il primo *oggetto* è stato inserito.

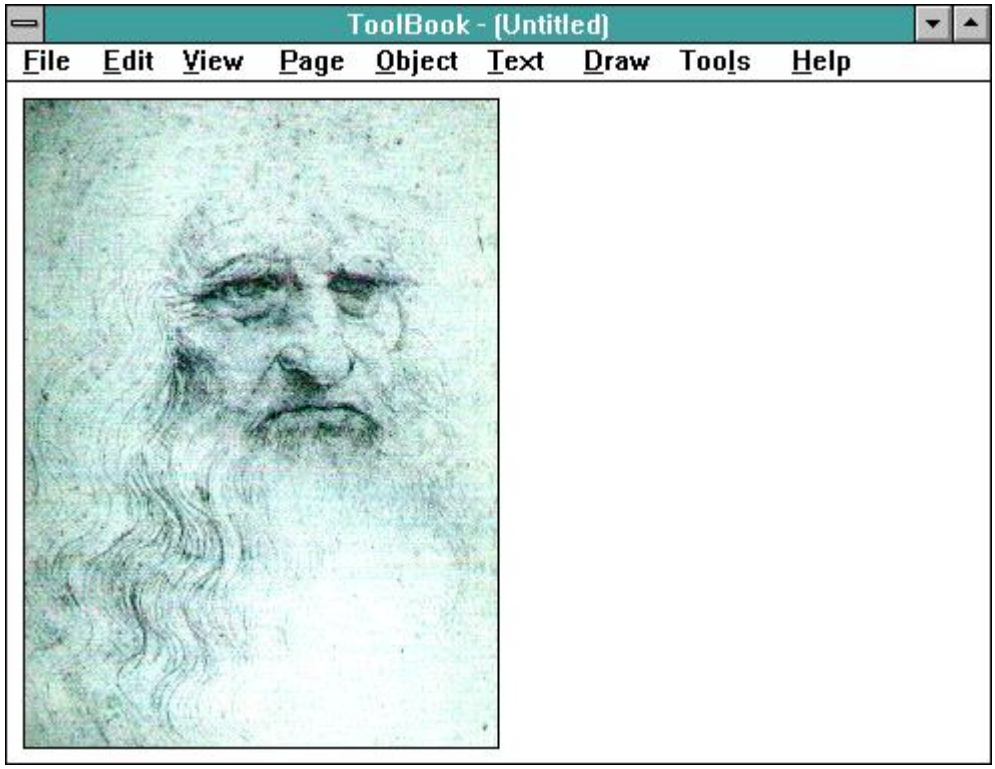

Figura 6: un'immagine è stata importata sulla pagina

In realtà, è il secondo, perchè la pagina stessa, come abbiamo visto, è un oggetto. Anche all'immagine possiamo dare un nome, seguendo la stessa procedura indicata per le pagine: cliccando sulla voce **Object** della barra di menu di Toolbook e successivamente sulla voce **Paint Object properties**.

Creare dei campi, scrivere dei testi...

Sulla pagina abbiamo inserito un'immagine. Ora è la volta del testo. Il testo, in Toolbook - ormai dovreste cominciare a capire il meccanismo - deve essere inserito in un altro *oggetto*. Così come nell'ipertesto cartaceo abbiamo ritagliato dei rettangolini sui quali abbiamo scritto dei testi (non abbiamo mai scritto direttamente sul foglio), allo stesso modo in Toolbook esistono dei "contenitori" di testi di forma rettangolare che noi possiamo creare e inserire sulla pagina. Questi "contenitori" vengono chiamati *campi* (**field**) prendendo a prestito un parola del gergo informatico legata soprattutto al mondo dei *database* (archivi elettronici). In realtà, un campo, in Toolbook, è più simile al foglio di una videoscrittura di quanto non lo sia ai campi dei *database*. In pratica il *campo* è il tipo di oggetto che dobbiamo creare per poter inserire dei testi sulla pagina. Sulla pagina, in quanto *oggetto* dotato di sue proprie caratteristiche, non si può infatti scrivere. Si deve invece dapprima creare un *campo* (cioé un contenitore), quindi inserisce il testo nel campo che è stato creato, scrivendolo direttamente tramite la tastiera del computer.

Per creare un campo bisogna usare un'apposito *strumento*. Gli strumenti di lavoro, in Toolbook, sono rappresentati da un box che "galleggia" sulla pagina e di cui si è già parlato in precedenza. Il box si

presenta come una finestra, sulla cui barra colorata appare o una freccia, che corrisponde al mouse, o il simbolo di un oggetto. Ogni strumento per creare oggetti è rappresentato, nella finestra, da una icona che in qualche modo cerca di evocarne il significato e le caratteristiche. Quello che a noi interessa in questo momento è lo strumento per costruire un campo di testo. Clicchiamo dunque sull'icona corrispondente.

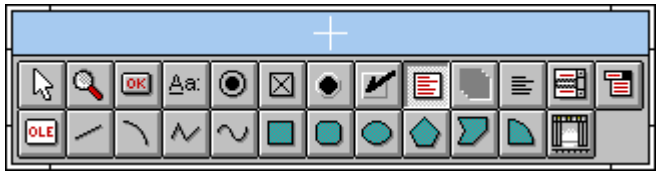

Figura 7: il box degli strumenti con in evidenza l'icona per creare i campi di testo

Notiamo adesso che la forma del cursore del mouse è cambiata: non è più la consueta freccia, ma una crocetta, simile a quella di un reticolo collimatore. Ciò indica che possiamo creare il campo. Spostiamoci sulla pagina e clicchiamo su una zona bianca, o dove vogliamo che il *campo* che stiamo per disegnare abbia il suo vertice superiore-sinistro. Tenendo premuto il pulsante del mouse scendiamo ora sempre più in basso a destra, fino a che non avremo delineato la dimensione del *campo* (vi sembrerà proprio di disegnare un rettangolo !). Non preoccupatevi, al momento, se non riuscite a dare al *campo* le giuste dimensioni o la giusta posizione. Il *campo* è un *oggetto*, e come tutti gli oggetti di Toolbook può essere modificato, spostato e ridimensionato in un momento successivo. Dopo aver rilasciato il pulsante del mouse, e quindi dopo aver disegnato il *campo*, è possibile infatti andare a cliccare nuovamente sul box degli strumenti in corrispondenza dell'icona che rappresenta la freccia del cursore del mouse. A quel punto sarà possibile cliccare su di esso, selezionarlo ed intervenire per spostarlo o per effettuare eventuali modifiche sulle sue proprietà e caratteristiche.

Se dunque, ora, volete riposizionare meglio il *campo* o modificarne le dimensioni, cliccate sul *campo* in modo da selezionarlo. Verranno visualizzati attorno al *campo* dei piccoli quadrati, cliccando sui quali (e tenendo premuto il pulsante del mouse) potrete allargare, rimpicciolire e in generale ridimensionare il rispettivo lato del rettangolo. Cliccando invece in una zona interna al *campo*, sempre tenendo premuto il mouse, potrete spostare l'oggetto in un'altra posizione. Basta un po' di esercizio per imparare a modificare rapidamente la forma, la dimensione e la posizione di un *campo*. Le stesse procedure, del resto, valgono per qualunque altro *oggetto* Toolbook, le cui caratteristiche esteriori possono essere modificate nello stesso modo, fatta eccezione per le immagini, che essendo "importate" e non "disegnate" direttamente con gli strumenti che il programma mette a disposizione, possono essere spostate ma non ridimensionate (salvo alcuni casi particolari). Per ulteriori informazioni sul modo di selezionare e modificare gli oggetti e le loro proprietà rimandiamo alle schede di approfondimento [**A11-A22**].

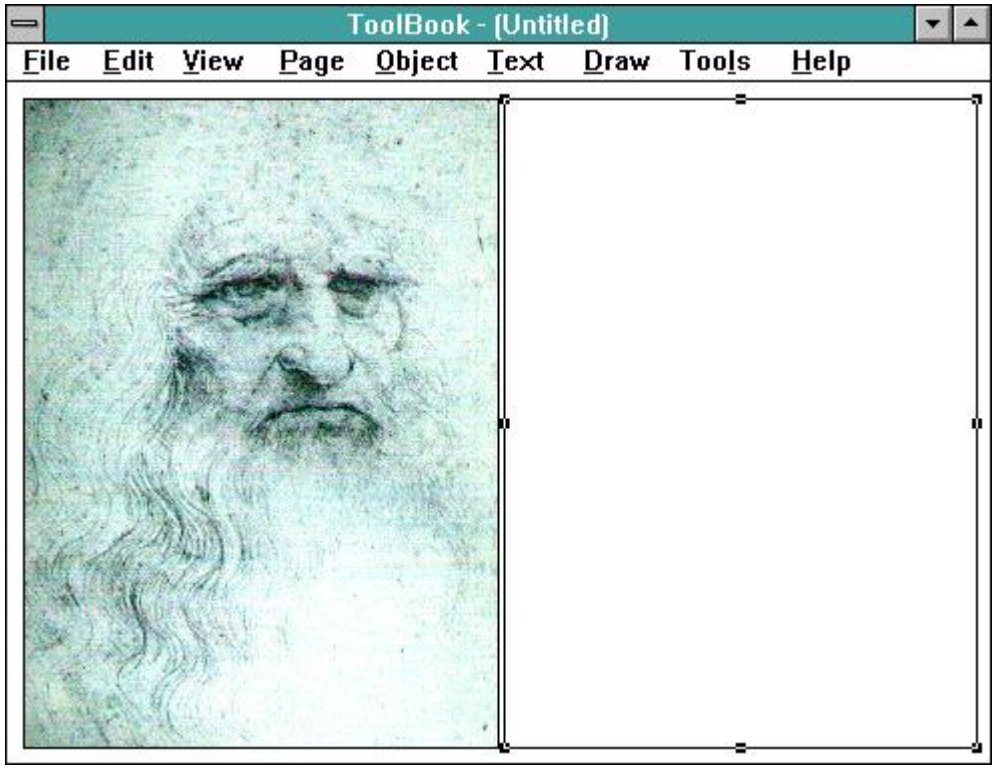

Figura 8: sulla pagina è stato creato un campo di testo che può essere ridimensionato cliccando e trascinando il mouse sui quadratini che lo circondano quando risulta selezionato

Per il momento torniamo al nostro *campo*. Ora, ammettiamo che siate soddisfatti della sua dimensione e della sue posizione, e che vogliate scriverci un testo. Per inserire un testo in un campo è sufficiente fare un *doppio clic* veloce all'interno di esso, fino a che il cursore non prenderà la forma di una barretta verticale, la stessa che di solito indica il documento o il foglio attivo in un qualsiasi programma di videoscrittura. A quel punto potrete scrivere direttamente usando la tastiera. Non preoccupatevi, per adesso, della grandezza dei caratteri, dell'impaginazione, dei colori, e nemmeno della lunghezza del testo rispetto alla grandezza del campo (fatto salvo il principio che in un libro elettronico i messaggi scritti dovrebbero essere brevi e chiari !): Toolbook tratterà e impaginerà il testo che state scrivendo automaticamente, e vi permetterà, in seguito, di modificare anche caratteristiche come il tipo di carattere, la forma esteriore del *campo*, l'impaginazione ecc. Si tratta infatti, in tutti questi casi, di agire sulle *proprietà* dell' *oggetto* "campo", cosa che si può fare facilmente selezionando l'oggetto stesso e richiamando dal menu **Object** la voce corrispondente, **Field properties**. E'questo, peraltro, uno dei principi generali del funzionamento di Toolbook: tutto ciò che Toolbook gestisce e tutto ciò che si può inserire nel "libro" elettronico *è* un *oggetto*, e come tale, possiede degli attributi e delle caratteristiche (non necessariamente le stesse) che possono essere modificate in qualunque momento. Si vedano, in proposito, le schede di approfondimento **[A23-A28].** Il campo di testo, naturalmente, è un *oggetto* più complesso di altri: si potrà quindi intervenire sulla sua dimensione, sulla sua forma esteriore (selezionando ciò che ci interessa all'interno della finestra di dialogo richiamata dal menu **Field properties**), ma anche sulle caratteristiche del testo che vi abbiamo scritto, che a sua volta potrà essere "formattato", ovvero cambiato in dimensione, stile, tipo

di carattere, ecc. sia relativamente all'intero contenuto che per una precisa porzione di esso. Queste funzioni possono essere richiamate dal menu **Text,** alle voci **Character** (corpo, tipo e stile) e **Paragraph** (impaginazione e formattazione vera e propria), e verranno spiegate meglio in seguito **[A26-A27].**

Il primo nodo, gli altri nodi...

Alla fine delle operazioni appena descritte la pagina dovrebbe presentarsi più o meno così.

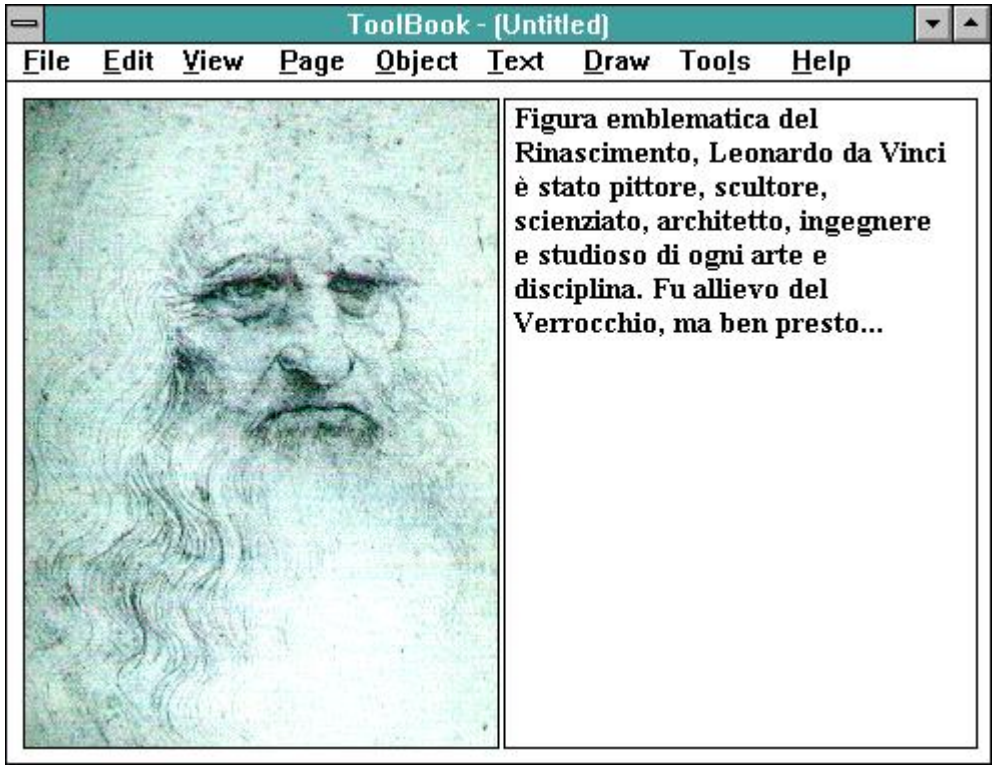

Figura 9: nel campo di testo che è stato creato sulla pagina sono state inserite delle informazioni scritte

Bene! Abbiamo dunque inserito i contenuti relativi a ciò che avevamo idividuato come il primo nodo dell'ipertesto. Ora non ci resta che aggiungere le altre pagine (gli altri nodi) e creare i *link*: avremo costruito un piccolo ipertesto.

Ci serve un'altra pagina. Per creare una nuova pagina bisogna selezionare la voce **New Page** dal menu **Object** (si veda, in proposito, anche quanto riportato nelle schede di approfondimento [**A6**]). Apparirà una nuova pagina bianca: siamo adesso sulla seconda pagina del nostro ipertesto (e sulla barra di *status* sarà indicato "Page 2" e "2 of 2"), una pagina ancora vuota. Su questa pagina ripeteremo esattamente le stesse operazioni effettuate sulla prima pagina. Le daremo un nome: "Leonardo Pittore"; vi importeremo un'immagine (ad esempio la famosa "Gioconda", precedentemente acquisita con uno *scanner* e inserita con **Import Graphic...**); quindi disegneremo un *campo* su cui scriveremo un testo che riguarda la figura di Leonardo pittore. Ora anche la seconda pagina è pronta. Passiamo

alla terza. Sempre scegliendo **New Page** dal menu **Object** aggiungeremo una nuova pagina a quello che ormai sta prendendo la forma di un libro, anche se elettronico. Anche su questa nuova pagina ripeteremo ciò che abbiamo fatto per le pagine precedenti. Avrà anch'essa un nome ("Leonardo Scienziato"), conterrà un'immagine rappresentativa dell'argomento e un campo con un testo che parla degli studi scientifici intrapresi da Leonardo.

Colorare le pagine e gli oggetti...

Il nostro ipertesto sta dunque prendendo forma. Tuttavia, non siamo ancora del tutto soddisfatti. Vorremmo infatti poter migliorare l'aspetto grafico delle pagine (non ci piace, ad esempio, la loro uniforme colorazione bianca, su cui i testi, anch'essi a fondo bianco, non risaltano come dovrebbero). Toolbook, in tal senso, ci mette a disposizione una serie di strumenti specifici, che ci consentono di dare un tocco di colore, anche piuttosto sofisticato, al nostro ipertesto. Prima di tutto si possono "colorare" le pagine. In realtà, ciò a cui possiamo attribuire un colore non sono le pagine in quanto tali, ma il loro "sfondo" comune. Lo "sfondo" comune a una serie di pagine viene detto **Background**: il **Background** è un *oggetto* come gli altri (vedremo meglio, in seguito, alcune delle sue particolari proprietà). Una delle sue caratteristiche principali è costituita dal fatto che esso è, per così dire, "condiviso" da tutte le pagine, si intende da tutte quelle collegate a quello sfondo, poichè in Toolbook, così come è possibile creare nuove pagine, è possibile anche creare "nuovi sfondi". Il colore è una delle proprietà del **Background**. Per modificarlo è sufficiente selezionare la voce **Background Properties** dal menu **Object** e cliccare, dopo che sarà apparsa la solita finestra di dialogo, sul pulsante **Color**. Apparirà una *palette*, letteralmente una "tavolozza", sulla quale sarà possibile scegliere il colore che più vi piace. Le pagine collegate a quel **Background** assumeranno automaticamente la colorazione attribuita al **Background** stesso. Analogamente, la stessa palette dei colori potrà essere richiamata in qualunque momento facendo clic sul pulsante che mostra la relativa icona sul box delle funzioni (quello con la barra colorata vuota) o selezionando la voce di menu **View**... **Palette**... **Color**. Quando la "tavolozza" è sulla scrivania (**fig. 10**), essa potrà essere utilizzata per cambiare il colore di un qualsiasi *oggetto*, a patto che esso sia selezionato (ricordate come si fa ?) e non sia un'immagine importata. Potremo, ad esempio, selezionare un campo e cambiarne il colore del fondo (bastano un clic sull'icona che mostra un secchiello e successivamente uno sul colore prescelto) e quello del testo (un clic sull'icona con il pennello e uno sul colore prescelto). Le combinazioni possibili ? Dipendono soltanto dalla vostra fantasia e dal vostro senso grafico.

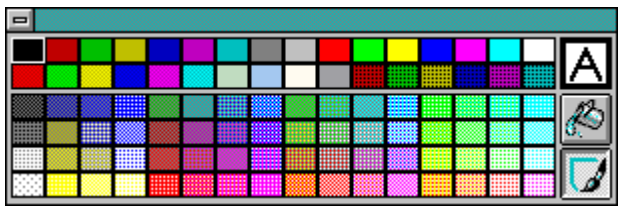

Figura 10: la palette dei colori

Sfogliare le pagine...

Il nostro ipertesto è ora costituito da tre pagine, che per il momento, tra loro, sono in rapporto di stretta sequenza, come quelle di un normale libro: pagina 1, pagina 2, pagina 3 **[A7-A9].** Queste pagine contengono immagini e campi di testo. Non è ancora un vero e proprio ipertesto, tuttavia comincia ad assumerne l'aspetto. In Toolbook, infatti, le pagine possono essere sfogliate come quelle

di un libro, in normale sequenza precedente-successiva, ma è possibile, senza bisogno di interventi di programmazione da parte dell'autore dell'ipertesto, anche andare direttamente alla prima o all'ultima delle pagine create, tornare all'ultima pagina sfogliata e richiamare la sequenza *non lineare* dei salti che abbiamo effettuato tra le pagine. Questi semplici "strumenti di navigazione" sono costituiti dalle rispettive voci del menu **Page**:

Next (va alla pagina successiva)
Previous (va alla pagina precedente)
First (va alla prima pagina del libro)
Last (va all'ultima pagina del libro)
Back (torna all'ultima pagina sfogliata)
History (mostra la sequenza con cui il lettore ha sfogliato le pagine)

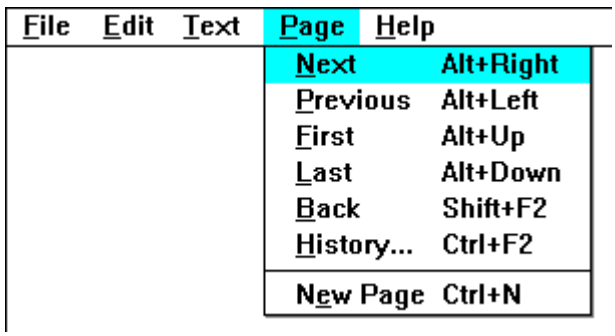

Figura 11: il menu "Page" e le corrispondenti voci per la "navigazione" nel libro

Il nostro libro elettronico può quindi essere sfogliato e consultato in modo *non lineare*, ovvero *ipertestuale*, selezionando le voci di menu qui ricordate (o usando le corrispondenti combinazioni di tasti, indicate accanto alle rispettive voci). Ma non basta: non sono questi gli unici strumenti di navigazione che Toolbook ci mette a disposizione. Se nelle pagine ci sono dei testi, ad esempio, con Toolbook è possibile cercare in modo automatico una particolare parola o frase contenuta in un qualsiasi campo in modo da visualizzare e consultare solo le pagine che contengono quella parola o quella frase, attivando così un nuovo strumento di navigazione, con caratteristiche spiccatamente ipertestuali. Per attivare questa funzione è sufficiente selezionare la voce **Find** del menu **Edit** e scrivere la parola e il testo che vogliamo cercare sui campi delle varie pagine [**A28**]. Possiamo riuscire, così, sfruttando queste particolari funzioni, proprie del programma, a "navigare" con una certa libertà attraverso le pagine. Si potrebbe perfino affermare che, a livello minimo, con Toolbook si può costruire un ipertesto effettuando le semplici operazioni che abbiamo già descritto (creare pagine, importare immagini, inserire campi di testo) e utilizzando, per la navigazione, gli strumenti già a nostra disposizione. Il nostro sarebbe un ipertesto molto elementare, ma vale la pena sottolineare questa possibilità, per capire che con questi strumenti di lavoro costruire libri elettronici può essere talmente semplice che anche il totale neofita, anche chi non ha mai messo un dito sulla tastiera di un computer e non ha mai toccato un *mouse*, potrebbe riuscirci, e in pochissimo tempo.

Salvare e riaprire un libro...

L'ipertesto che stiamo costruendo, naturalmente, alla fine sarà ben più complesso e articolato di quanto non lo sia adesso. Tuttavia decidiamo, per il momento, che, anche così com'è, può essere

sufficiente ai nostri scopi e sufficientemente definito per le possibilità e le esigenze di un principiante. Si tratta ora di "salvarlo", perchè possa essere riutilizzato, "navigato" e modificato in un secondo momento, e anche per evitare, nella malaugurata ipotesi di un improvviso *black-out*, di perdere tutto ciò che è stato fatto finora. "Salvare" il vostro ipertesto significa registrarlo nella memoria permanente del vostro disco rigido. Scegliete la voce **Save** del menu **File**. Quindi date un nome al vostro ipertesto, che non superi gli 8 caratteri e non contenga spazi, punti o caratteri strani come l'asterisco o il punto interrogativo. Un nome adatto potrebbe essere proprio "Leonardo".

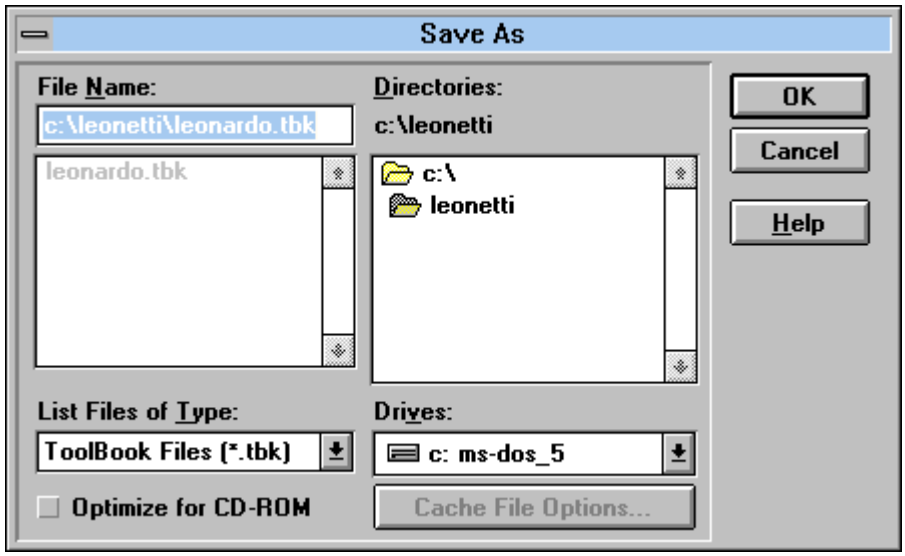

Figura 12: la finestra di dialogo per "salvare" il libro

Fatto questo, dovrete posizionarvi sulla "cartella" del disco rigido (directory) dove volete che il vostro lavoro venga memorizzato. Per semplicità confermiamo la stessa directory in cui si trova Toolbook. Quindi, premete "invio" per confermare il salvataggio.

Quello che avete appena salvato è un *file* che conterrà tutto ciò che avevate inserito nel vostro ipertesto. Per Toolbook, però, ciò che avete appena salvato è prima di tutto un **Book**, un "libro": il **Book** è un *oggetto*, come la pagina, le immagini e i campi di testo. E' il primo degli oggetti, quello che contiene tutti gli altri. Attenzione ! Non è necessariamente *un* ipertesto. Il **Book** è un singolo *file* costruito usando Toolbook, e può corrispondere anche al *vostro* ipertesto, ma non obbligatoriamente: la vostra applicazione, infatti, potrebbe anche essere composta di più libri, di più **Book**. Il Book è tuttavia l'unità elementare che con Toolbook può essere salvata, richiamata e modificata.

Quando sarà il momento di riaprire il vostro ipertesto, per usarlo o per modificarlo, dopo aver lanciato Toolbook, dal menu **File** dovrete scegliere la voce **Open** e quindi scrivere il nome del **Book** da aprire ("Leonardo", nel nostro caso), ovvero cliccare sul nome corrispondente nell'elenco che apparirà nella finestra di dialogo.

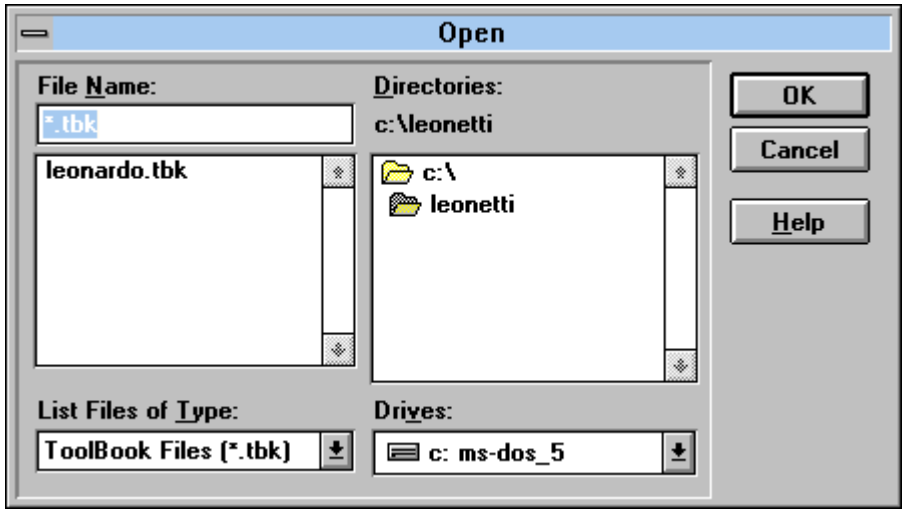

Figura 13: la finestra di dialogo per "aprire" il libro

Ma di tutto questo si parlerà più estesamente negli approfondimenti [**A2-A4**]. Ora proviamo piuttosto a prendere confidenza con un vero e proprio *linguaggio di programmazione*, che finora non abbiamo neppure sfiorato. Riapriamo il nostro **Book**, dunque, e andiamo avanti.

Creare un link: i pulsanti...

Abbiamo già visto come sia possibile "sfogliare" le pagine del **Book** utilizzando determinate voci di menu. In realtà, queste modalità di navigazione non sono molto intuitive, nè ortodosse: perchè il nostro sia un *vero* ipertesto dobbiamo inserire dei *link* che ci consentano di consultare le pagine secondo una sequenza più libera e nello stesso tempo più organica rispetto ai contenuti. In un ipertesto, infatti, la sequenza fisica delle pagine non sempre corrisponde (anzi quasi mai) alla sequenza dettata dallo schema ipertestuale realizzato con i *link*, che è una sequenza esclusivamente logica. Ricordate i fili di spago che legavano tra loro i fogli impilati ? Permettevano di saltare direttamente da pagina 1 a pagina 3, senza passare per la 2. Si può fare la stessa cosa anche attivando una funzione **Find**, come abbiamo visto, ma l'eventuale salto non sequenziale tra le pagine sarebbe casuale, dipenderebbe dalla presenza o meno in un campo di testo di una determinata parola. Il salto, invece, dovrebbe essere possibile sulla base di precise associazioni - *link* - tra i contenuti dei vari nodi. Nel nostro caso, partendo dalla prima pagina, dovrebbe essere possibile approfondire, ovvero richiamare, sia gli aspetti legati al tema "Leonardo Pittore" che quelli legati al tema "Leonardo Scienziato", sulla base della libera scelta del lettore, una libertà che un buon ipertesto dovrebbe tra l'altro saper assecondare.

Immaginiamo dunque che le nostre tre pagine siano, schematicamente, così collegate.

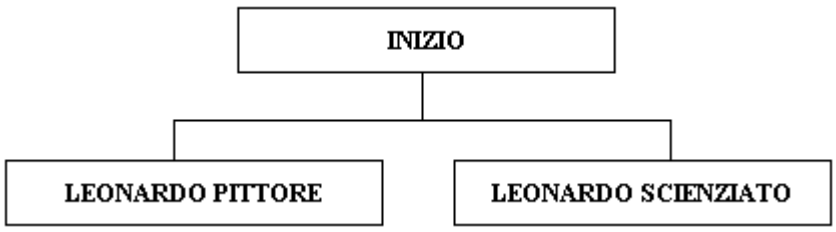

Figura 14: la "mappa" dei collegamenti tra le pagine dell'ipertesto

Vediamo ora come è possibile, con Toolbook, creare questi collegamenti. Innanzitutto dobbiamo ritornare alla prima pagina. Poichè abbiamo appena riaperto il **Book** non è necessario "navigare" nell'ipertesto per farlo: in Toolbook, infatti, il numero corrispondente alla posizione della pagina può essere cambiato in qualsiasi momento (basta scrivere il numero corrispondente alla posizione che vogliamo far assumere alla pagina nell'apposito spazio **Number** della finestra di riepilogo delle proprietà della pagina [**A7**]), ma un **Book**, quando viene aperto, mostra sempre per prima la pagina che il programma riconosce come la prima, ovvero quella che in quel momento porta il numero 1. Poichè non abbiamo cambiato l'ordine "fisico" delle pagine, avremo ora sullo schermo la prima pagina del nostro ipertesto, quella con l'autoritratto di Leonardo. In che modo si possono creare i legami che permetteranno di saltare direttamente da questa pagina alla seconda o alla terza ?

L'azione di saltare direttamente ad una certa pagina deve essere decisa dall'utente, il quale agisce, generalmente, scegliendo di cliccare su quelle aree di ciascuna videata che rispondono agli eventi che l'utente stesso provoca (il "clic" del mouse, ad esempio). L'autore dell'ipertesto, in tal senso, dovrebbe da un lato offrire all'utente una gamma di possibilità di interazione reale, dall'altro rendere in qualche modo evidenti le possibilità che all'utente vengono offerte. Abbiamo già detto che le aree dello schermo che reagiscono ad un evento-utente vengono dette *hot-spot* (zone calde). Le zone calde si possono distinguere, grossolonamente, in *pulsanti* ed *hot-word*. Nel senso che pulsanti e hot-word rispondono in modo efficace all'esigenza di rendere manifesta all'utente una possibilità di interazione. Toolbook, tuttavia, va ben oltre questa elementare classificazione degli *hot-spot*. Per Toolbook, infatti, ogni *oggetto* inserito nell'ipertesto può essere "caldo", può cioé prevedere un'azione in risposta ad un evento generato dall'utente. Lo stesso "book", le pagine, le immagini e i campi, gli oggetti che abbiamo già imparato a definire, possono essere quindi sensibili al *click* del mouse (o ad altro) ed istruiti a compiere determinate azioni in risposta all'evento, azioni come il saltare da una pagina ad un'altra. Questo è uno degli assiomi del programma: in Toolbook tutto ciò che possiamo creare e manipolare è un *oggetto*, e tutto ciò che è un *oggetto* può essere *programmato*, ovvero istruito a reagire in un determinato modo ad un determinato evento.

Cominciamo, dunque, a *programmare*. A definire, cioé, il comportamento di un *oggetto* da *noi stessi* creato sulla base di criteri da *noi stessi* definiti. Lavoreremo, a titolo di esempio, su un *oggetto* che, per antonomasia, è un *hot-spot*: il *pulsante*. Questo nuovo tipo di oggetto può essere creato così come avevamo fatto per i campi. Si clicca sull'icona corrispondente del box degli strumenti.

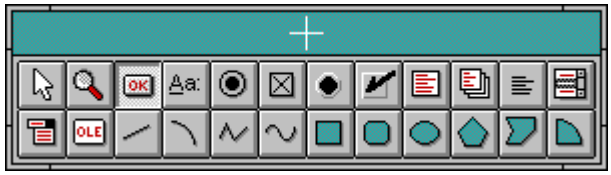

Figura 15: il box degli strumenti con, in evidenza, l'icona per creare i pulsanti

Quindi, cliccando sulla pagina e trascinando il mouse, si definiscono la dimensione e la posizione del pulsante. Il pulsante avrà una forma rettangolare, sarà di colore grigio e di aspetto leggeremente rilevato, proprio come i pulsanti di un telecomando o di un registratore. Avrà anche un' *etichetta* piuttosto generica: **Button**. Come al solito, potrete selezionarlo e spostarlo in qualunque momento cliccando sull'icona dello strumento "freccia" sull'apposito box; allo stesso modo, potrete cliccare sui quadratini posti lungo il suo perimetro, una volta selezionato l'oggetto, e trascinarli per ridimensionarne i lati. Niente di nuovo, vero ? Avevamo già imparato a fare le stesse cose con i campi. Ciò significa, lo abbiamo già detto ma è bene ripeterlo, che in Toolbook campi, pulsanti e quant'altro ancora sono e restano *oggetti* dei quali è possibile modificare sia le proprietà generali (la posizione, la dimensione, il nome) sia quelle particolari, specifiche per ciascun tipo di oggetto. Tra le proprietà particolari del *campo* rientra il *testo* che esso può contenere ? Bene, tra quelle del *pulsante* merita una certa attenzione l' *etichetta*. Per modificare le proprietà di un pulsante è sufficiente selezionare la voce **Button Properties** dal menu **Object** , come al solito. **(cfr.fig. 4)**. Nella finestra di dialogo che apparirà noterete però, a differenza di quanto accade nel riepilogo delle proprietà del campo, che le caselle su cui si potrà scrivere sono due: **Name** e **Caption**. Attenzione ! Nella casella **Name** si potrà inserire il *nome* che intendiamo dare al pulsante, quello stesso genere di nome che avevamo dato agli altri oggetti, un nome logico, che ci permetta di ritrovare l'oggetto, se mai dovessimo perderlo (e può succedere), o di indirizzare un evento verso di lui. Per cambiare l'etichetta del pulsante, sostituendo il generico "Button" con qualcosa di più esplicativo, si deve invece agire sulla casella **Caption**. La casella **Caption** consente infatti di modificare quella particolare proprietà dell'oggetto che è l' "etichetta", e tutto ciò che scriverete su quella casella verrà riportato sul pulsante come *etichetta*. Scriviamo dunque "Leonardo pittore", cliccando successivamente sul pulsante **Ok** per chiudere la finestra di dialogo delle proprietà. L'aspetto del pulsante, ora, è cambiato. Volendo, avremmo potuto anche scegliere di attribuire all'oggetto forme diverse, o di colorarlo. Ma non è il caso di ripetere ancora una volta come si fa ciò che ormai dovreste già essere in grado di fare da soli.

Autore, lettore...

Il fatto di aver cambiato l'etichetta del pulsante non è certo sufficiente a far sì che il pulsante, se cliccato, "salti" direttamente alla pagina contenente le notizie sulla figura di Leonardo pittore. Per il momento, infatti, abbiamo definito soltanto una delle proprietà "esteriori" dell'oggetto (l'etichetta), e non il comportamento che esso dovrà assumere in risposta ad un eventuale *clic*. Il pulsante, sotto certi aspetti, reagirà comunque all'evento, ma non produrrà gli effetti desiderati. Vogliamo provare ? Per "provare" il pulsante bisogna cambiare ambiente. Ora, da "autori", dobbiamo diventare semplici "lettori". E' il "lettore", infatti, che consulta l'ipertesto cliccando sulle aree calde. Voi autori, finchè rimarrete al livello in cui vi trovate adesso, potete cliccare quanto volete sul pulsante, ma non otterrete altro effetto che quello di selezionarlo e non potrete verificare le sue eventuali reazioni al *clic*. Si introduce qui un concetto fondamentale di Toolbook: Toolbook lavora su due *livelli* diversi, due *modalità*, che vengono dette proprio "autore" e "lettore". A livello "autore" si possono creare oggetti, modificarne le proprietà, programmarli. A livello "lettore", invece, gli oggetti non potranno essere nè creati nè modificati, ma potranno "agire". Per cambiare modalità è sufficiente scegliere dal menu **Edit**

la voce **Reader** o, più rapidamente, premere il tasto <**F3**>. Eccoci trasformati in "lettori". Noterete subito che dallo schermo sono scomparsi i box e la barra dello *status* e che anche la barra del menu si è notevolmente ridotta (e certe voci sono cambiate). La pagina è finalmente come volevamo che fosse. Libera, colorata, intera. Noterete anche che, per quanti sforzi possiate fare, non riuscirete a selezionare gli oggetti. Provate, a livello "lettore", a fare un *clic* sul pulsante che avete appena creato. Noterete che il pulsante simula tridimensionalmente l'effetto caratteristico di un vero e proprio *pulsante-che-viene-premuto*. Reagisce, cioè, all'evento che abbiamo provocato. Ciò nonostante, non succede assolutamente nulla. Rimaniamo sempre sulla pagina iniziale, mentre ci aspetteremmo che, una volta premuto, il pulsante richiamasse la pagina che avevamo chiamato "Leonardo pittore". Dopo tutto, era stato creato per questo scopo e perchè lo scopo fosse intuitivo sulla sua *etichetta* avevamo scritto "Leonardo pittore", un'indicazione *logica*. Come ottenere, dunque, questo comportamento ? Associando al pulsante un *programma*. Ritorniamo in modalità "autore" premendo nuovamente il tasto <**F3**> o scegliendo la voce **Author** sempre da menu **Edit**, e vediamo come.

Definire il comportamento di un oggetto: lo script...

Per Toolbook gli oggetti che costituiscono un ipertesto (pagine, campi, pulsanti, ecc.) sono come gli "attori" di un film. Certo, si tratta di un film anomalo: un film in cui spetta a voi scrivere il soggetto e la sceneggiatura, decidere i costumi, le scene e gli ambienti del teatro di posa, ma in cui il vero regista (colui che dovrà dire: "ciak, si gira") sarà l'utente. Solo l'utente potrà decidere quando un "attore" presente sulla scena dovrà cominciare a recitare la sua parte. Sarà lui, quindi, a definire la trama (che altro è un *percorso ipertestuale* se non una trama indefinita ?) "interpretando" il film sulla base della sua sensibilità e dei suoi propri interessi. Tuttavia, il vostro ruolo rimarrà di fondamentale importanza. Voi, gli "autori", dovrete infatti assumervi l'onere di istruire gli "attori", dare loro il copione da recitare, stabilire che cosa dovranno o non dovranno fare al momento del ciak, o, meglio, del *click*. In Toolbook, definire il comportamento di un oggetto significa definirne lo *script*, che in inglese significa, non a caso, "copione", proprio quello che seguono gli attori quando devono recitare una parte. Lo *script* altro non è che una delle tante *proprietà* dell'oggetto-attore.

Per poter definire uno *script* si dovrà quindi prima di tutto richiamare la finestra di dialogo delle *proprietà* dell'oggetto. Selezionate il pulsante (ormai lo sapete fare...), quindi, dal menu **Object**, selezionate la voce **Button Properties** (anche questo, ormai, dovreste saper fare...). Nella finestra notate, sulla destra, un pulsante con l'etichetta **Script**: è proprio quello che bisogna premere per definire la "parte" del pulsante, cioé il comportamento che dovrà tenere in risposta agli eventi provocati dall'utente o, più in generale, in risposta a qualunque tipo di evento. Cliccate dunque su **Script** e vedrete apparire un foglio bianco con una barra di menu e una serie di icone: su questo foglio proveremo a scrivere il "copione" del pulsante "Leonardo Pittore". In pratica, sulla base di quanto scriveremo su questo foglio, impareremo a *programmare* il pulsante.

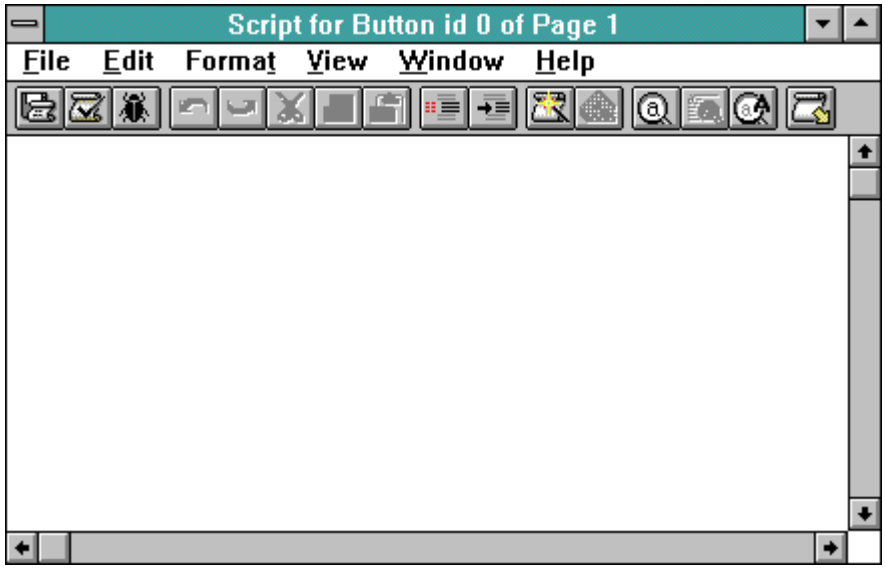

Figura 16: come si presenta la finestra dell'editor degli script degli oggetti

Quando si scrive un copione per un attore, bisogna fare in modo che l'attore capisca ciò che gli si vuole dire. Si dovrà quindi usare un linguaggio comune, comprensibile sia a noi che all'attore. Se il nostro attore fosse giapponese e la sua parte fosse scritta in pugliese, ben difficilmente egli potrebbe recitarla. Bisogna parlare la stessa lingua, dunque. Ma poichè recitare una parte implica qualcosa di più della semplice "lettura", bisogna anche adottare un medesimo "stile" di comunicazione, uno stile che ci permetta di comunicare all'attore quale atteggiamento tenere mentre recita una battuta. Forse qualcuno di voi ha avuto modo di vedere com'è fatta una sceneggiatura: ci sono parti scritte in corsivo o tra parentesi, distinte da altre parti in grassetto, o allineate in un certo modo. Tutto ciò serve a indicare all'attore nono solo ciò che dovrà dire, ma anche quando e come dovrà farlo.
....
(entra il ladro dalla finestra)
Lui: *(concitato)* Chi è? Cosa vuole?
(il ladro si gira di scatto e porge la mano)
Lui: *(più rilassato)* Molto lieto. Piacere di conoscerla.
....
Se date questo pezzo di copione ad un attore, questi inizierà la sua parte nel momento in cui si verifica l'evento indicato (*il ladro entra della finestra*) e quindi reciterà la battuta assumendo l'espressione indicata tra parentesi (*in modo concitato*). Ora, anche per definire il comportamento di un pulsante bisogna tener conto di queste elementari esigenze comunicative: bisogna cioè dire al pulsante *quando* e *cosa* esso dovrà fare, ma bisogna dirlo in un linguaggio che lui possa comprendere. Il linguaggio usato in Toolbook per definire il comportamento degli oggetti, ovvero per programmarli e istruirli, si chiama **Openscript**: tecnicamente, esso può essere definito un "linguaggio di programmazione event-driven" (ovvero, guidato dagli eventi). Proviamo dunque a vedere il "copione" del nostro pulsante, in che modo, insomma, possiamo dirgli che, quando verrà premuto, dovrà saltare alla pagina chiamata "Leonardo pittore":

```
to handle ButtonClick
        go to page "Leonardo pittore"
end ButtonClick
```

Questo sequenza di istruzioni è un piccolo **Script**, scritto seguendo la *sintassi* dell'unico linguaggio che il pulsante è in grado di comprendere, **Openscript**. Assomiglia un pò all'inglese corrente, vero ? Che cosa vi aspettavate da un linguaggio della "quarta generazione" ? Codici complicati e formule astruse ? Quelle sono cose vecchie ! Oggi, con questi strumenti, programmare non è difficile, e il significato di molte istruzioni, a patto di avere una minima confidenza con la lingua inglese, si può anche intuire abbastanza facilmente. Si potrebbe anzi affermare che usare Openscript per *programmare* le reazioni degli *oggetti* è *quasi* come parlare loro, ovviamente in quella che è ormai considerata la lingua della comunicazione universale.

Vediamo, in dettaglio, come è fatto questo *script*. La prima riga dice al pulsante quando deve attivarsi, ovvero, in conseguenza di quale evento egli deve cominciare a recitare la sua parte (*to handle* ....): il "ciak si gira" sarà dato nel momento in cui l'utente farà *clic* sul pulsante. Il *clic* del mouse genera infatti un evento che Toolbook identifica come *ButtonClick* e che viene subito notificato, inviato al bersaglio dell'evento (ad un **target**, in questo caso il pulsante). Il pulsante controlla se nel suo *script* è stata prevista una reazione a quell'evento (cerca cioé una riga di istruzioni che inizia con *to handle buttonClick*): in caso di esito positivo esegue tutte le azioni indicate di seguito nello *script* fino alla fine, ovvero fino a che non incontra la parola *end*. Nel nostro caso l'unica azione che il pulsante deve eseguire è quella corrispondente al salto di pagina programmato, che è stata definita semplicemente con *go to* ("vai a") *page* ("alla pagina") "Leonardo pittore" (il *nome* della pagina). Il gioco è fatto. Il pulsante, ora, ha un suo "copione", è stato istruito, programmato. Salvate lo *script* scegliendo **Save and exit** dal menu **File** della finestra dello *script* stesso (oppure cliccando sulla prima icona della barra della finestra, quella con un dischetto e un foglio di carta): lo *script* verrà registrato all'interno del pulsante e vi ritroverete sulla finestra di dialogo delle proprietà dell'oggetto. Fate *clic* su **Ok** per confermare tutto ciò che avete fatto e vi ritroverete sulla prima pagina [sul modo di richiamare e modificare uno script si veda anche **A22**]. Lo *script*, ora, fa parte delle *proprietà* del pulsante, e come tale, *agirà* (perchè questo è il suo scopo), così come potrà essere riaperto o modificato.

A questo punto possiamo provare il funzionamento del pulsante. Andate a livello "lettore" (premendo il tasto <**F3**>) e cliccate sul pulsante: vi ritroverete esattamente alla pagina che avevate chiamato "Leonardo Pittore". Non solo avete creato un *link*, ma avete anche creato un meccanismo per percorrerlo. E' un piccolo passo per un autore di ipertesti, ma un grande passo in avanti per tutti coloro che finora sono stati soltanto dei "lettori". Coraggio: le cose, naturalmente, non finiscono qui.

Tornare indietro...

Il *link* che abbiamo realizzato ci porta dalla pagina iniziale a quella chiamata "Leonardo Pittore". E per tornare indietro ? La funzione "tornare indietro" fa parte degli strumenti di navigazione essenziali in un ipertesto. Insieme alle funzioni "uscire" (smettere di lavorare con l'ipertesto [**B9**]), "tornare alla pagina iniziale", "richiamare una mappa dell'ipertesto" (con la quale l'utente possa orientarsi) fa parte di un *set* che dovrebbe essere sempre disponibile, su tutte le pagine, e che potremmo far corrispondere ad altrettanti pulsanti.

Riflettiamo meglio su che cosa significa "tornare indietro". Spesso il concetto viene confuso, per assonanza, con "andare alla pagina precedente". In realtà si tratta di due opzioni diverse. Supponiamo

di leggere un libro. Ad un certo punto della lettura, magari a pagina 3, troviamo una nota del tipo "cfr. pp. 25-27" che ci invita a sospendere la lettura di questa pagina per "saltare" direttamente alla pagina 25 dello stesso volume. Facciamo il salto. Ora siamo a pagina 25, dove troviamo un altro invito ad un salto: "crf. pag. 34". Andiamo così a pagina 34 e leggiamo quello che c'è scritto. A questo punto vorremmo poter "tornare indietro", ma questo significa tornare alla pagina da cui eravamo partiti, ovvero a pagina 25, e poi, da pagina 25, tornare ancora indietro fino a pagina 3, quella da cui eravamo partiti. La "pagina precedente" alla 34, invece, è la 33, che non ci interessa affatto e che non abbiamo ancora letto. Per "pagina precedente", dunque, si intende la pagina "fisicamente" precedente a quella in esame, mentre "tornare indietro" si riferisce alla pagina *precedente* a quella in esame relativamente allo sviluppo del *percorso ipertestuale* da me seguito in quel momento. Toolbook, da buon strumento per la costruzione di ipertesti, mantiene una memoria dei salti ipertestuali effettuati con comandi del tipo *go to page "...."*, per cui, se vogliamo percorrere a ritroso il cammino effettuato è sufficiente chiedere a Toolbook di "tornare indietro", senza specificare a quale pagina, perché lui lo sa già, un pò come accade nella storia Pollicino, che ad ogni passo lascia una traccia per ritrovare la strada. Abbiamo già accennato, in precedenza, alla presenza, tra i menu, di una voce che permette di richiamare la traccia dell'intero percorso ipertestuale, così come esso è stato memorizzato da Toolbook (la funzione **History**). Ora, comunque, si tratta di vedere in che modo il "passo indietro" può essere programmato come specifico comportamento di un pulsante.

Il pulsante per "tornare indietro", si è detto, dovrebbe essere presente in ogni pagina. Finché le pagine sono due o tre, come nel nostro esempio, la cosa, in sè, non costituisce un grande problema. In fondo, si tratta di costruire 3 pulsanti ! Ma supponiamo di aver arricchito il nostro ipertesto con altre 100, 200 pagine. Del resto, sviluppando un argomento come Leonardo da Vinci, è facile lasciarsi prendere la mano... Che cosa dovremmo fare ? Dovremmo forse creare su ogni pagina un pulsante, per di più lo *stesso* pulsante ?! Non è necessario, perchè Toolbook ci viene incontro. Vi ricordate come si fa a cambiare il colore di una pagina ? Si richiamano le *proprietà* di un oggetto *particolare*, che si chiama **Background**, "sfondo". Si è accennato al fatto che lo "sfondo" è condiviso da più pagine. Bene, aggiungiamo ora che lo "sfondo" permette di definire alcuni contenuti *comuni* a tutte le pagine. In pratica, se vogliamo che un determinato *oggetto* (un pulsante, ma anche un campo) sia visibile su ogni pagina del nostro ipertesto, dobbiamo creare quell'oggetto *sullo* sfondo. E' un po' come se avessimo a disposizione un substrato rigido e opaco su cui tutte le pagine si sovrappongono come se fossero dei lucidi. Questo è lo "sfondo". Scegliete dunque la voce **Background** dal menu **View** o, più rapidamente, premete il tasto <**F4**>. Noterete che, di colpo, il foglio, la videata, torna ad essere bianca, vuota. Non preoccupatevi ! Non avete perduto nulla di ciò che avete già costruito. State semplicemente lavorando su un "piano" diverso: osservate la barra dello *status*: nell'angolo in basso a destra è scritto **Background**. State lavorando sullo "sfondo" della pagina, anzi, sullo "sfondo" comune a *tutte* le pagine che fino a questo momento avete aggiunto all'ipertesto. Tutti gli *oggetti* che deciderete di costruire su questo "piano" speciale (sarebbe improprio definirlo un "foglio", anche se ne ha tutto l'aspetto) saranno visibile su ogni pagina che condivide quello stesso sfondo, a meno che, naturalmente, sulla pagina non ci siano oggetti che "coprono" fisicamente quelli posizionati sul **Background**. Creiamo ora, sul **Background**, il pulsante per "tornare indietro". Sappiamo già come fare. Posizionamolo in basso a destra, perchè non finisca "sotto" le immagini, i campi e i pulsanti delle pagine, e scriviamo, sulla sua *etichetta*: "Indietro". Lo script di questo pulsante sarà:

```
to handle buttonClick
        send Back
end buttonClick
```

Dove *send Back* è l'istruzione che dice a Toolbook di "richiamare l'ultima pagina sfogliata nel

percorso ipertestuale". Toolbook ripristinerà la pagina *precedente* a quella attuale, ma non quella "fisicamente" precedente, bensì quella che nel cammino *percorso* è stata toccata immediatamente prima di quella corrente.

Premete nuovamente il tasto <**F4**> o scegliete la voce **Foreground** (letteralmente "primo piano" dal menu **View**. Ora siete tornati a lavorare sulla pagina e noterete che il pulsante "Indietro" si è aggiunto agli oggetti già presenti. Quel pulsante, però, a differenza degli altri, sarà visibile su *ogni* pagina, perché si trova sullo "sfondo" comune ad esse. Provate la funzione "tornare indietro". Passate in modalità "lettore" (<**F3**>) e quindi cliccate sul pulsante "indietro". Tornerete alla pagina iniziale, ovvero a quella immediatamente precedente nel *percorso*, anche se in questo caso - in fondo si tratta di un esempio molto semplice e articolato su tre sole pagine - essa è anche la prima del libro e quella *fisicamente* precedente. Provate comunque a cliccare ancora sul pulsante "Leonardo Pittore" e quindi a tornare ancora indietro. Il *link* tra le due pagine vi apparirà per quello che è: completo.

Copiare, incollare...

Ora che abbiamo *linkato* la pagina iniziale alla pagine "Leonardo Pittore", non ci resta che attivare gli stessi tipi di legami con la pagina "Leonardo Scienziato". Sulla pagina iniziale dovremo quindi costruire un un altro pulsante, del tutto analogo a quello la cui etichetta è "Leonardo Pittore": cambieremo solo l'etichetta e, nello *script*, il *nome* della pagina a cui il pulsante dovrà "saltare". Ora che avete imparato a costruire e programmare il primo pulsante, costruire e programmare gli altri vi sembrerà noioso. Ma come, penserete, devo davvero ripetere tutte quelle operazioni ogni volta che devo attivare un *link* ? In realtà, in Toolbook possiamo facilmente sfruttare il lavoro già fatto.

Passate nuovamente in modalità "autore" (<**F3**>), tornate alla pagina iniziale e selezionate il pulsante "Leonardo Pittore". Dal menu **Edit** scegliete ora la voce **Copy** (copia). Che cosa è successo ? E' successo che il pulsante, con tutte le sue *proprietà*, è stato copiato sfruttando la cosidetta *clipboard*, uno strumento a cui tutti i programmi Windows possono accedere (lo avrete forse usato per copiare una parte di un testo usando una videoscrittura). Dopo aver copiato il pulsante, sempre dal menu **Edit**, scegliete la voce **Paste** (incolla) [**A17**]. Non noterete nulla di particolare, ma in realtà avete "duplicato" il pulsante "Leonardo Pittore". Il pulsante copiato, per il momento, si trova nella stessa posizione dell' "originale". Prendetelo e spostatelo, trascinandolo. Ora richiamate le **Button Properties** dal menu **Object** e cambiate l'etichetta del pulsante copiato in "Leonardo Scienziato": Poi cliccate su **Script** e notate che la sequenza di istruzioni è stata mantenuta. In pratica, il copione, il *programma*, è già pronto: dovete solo cambiare il nome della pagina a cui si vuole che acceda l'istruzione *go to page* in "Leonardo Scienziato". Salvate e confermate il tutto con **Ok**. Si può fare tutto anche molto più velocemente. Sul box delle funzioni, ad esempio, c'è un pulsantino con un'icona costituita da una piccola pila di schedine bianche (è il quarto a partire da sinistra): copia e incolla direttamente, ovvero *duplica* l'oggetto selezionato. Anche lo *script* può essere modificato direttamente, senza richiamare la finestra di dialogo delle proprietà dell'oggetto [**A22**].

Per provare il nuovo pulsante, tornate ancora in modalità "lettore". Cliccando su di esso si presenterà la pagina che era stata chiamata "Leonardo Scienziato". Su quella pagina ci sarà già il pulsante "Indietro", perché esso si trova sul **Background**. Ora, le possibilità di navigazione all'interno del nostro ipertesto cominciano ad essere molteplici. L'utente ha già a sua disposizione una serie di scelte, una certa libertà.

Per completare rapidamente lo schema di quell'ipertesto cartaceo che inizialmente avevamo preso a modello potremmo ora limitarci a inserire nelle pagine "Leonardo Pittore" e "Leonardo Scienziato"

altri pulsanti, che richiamino reciprocamente le rispettive pagine: dalla pagina "Leonardo Pittore" si dovrà poter andare alla pagina "Leonardo Scienziato", e viceversa. Sapete come fare. Consideratelo un esercizio. E considerate pure quello che avete appena realizzato il vostro primo ipertesto.

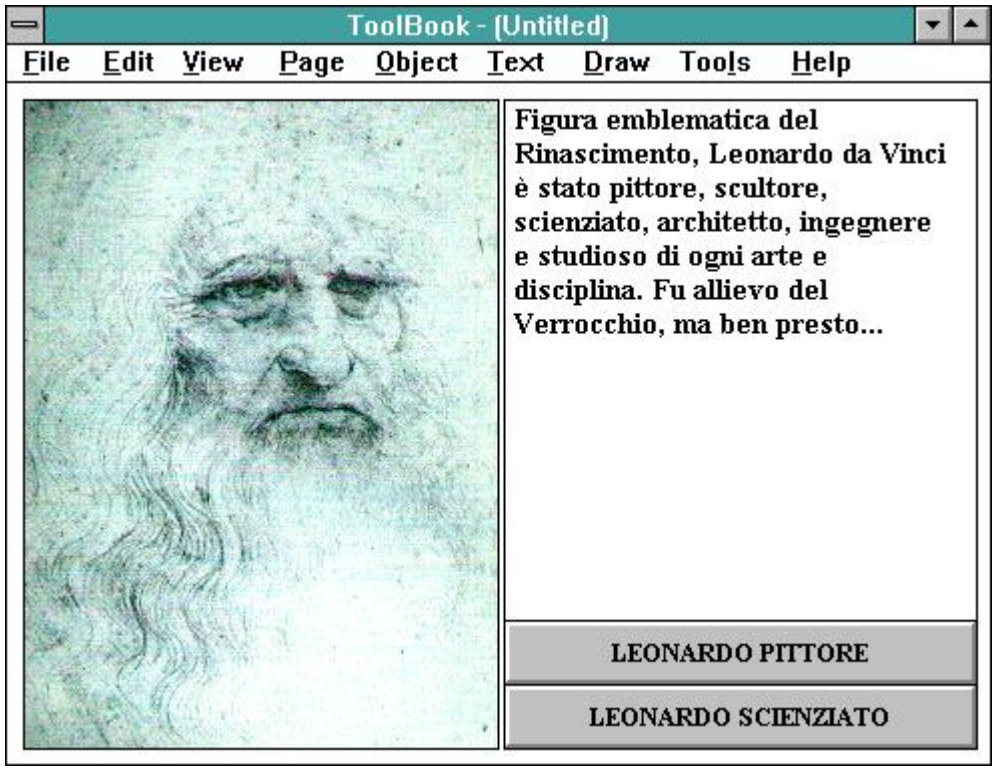

Figura 17: esempio di pagina con pulsanti di link ad altre pagine

I "SEGRETI" DEL LINGUAGGIO

Parole calde...

Naturalmente, potremmo andare avanti e complicare l'ipertesto aggiungendo pagine e costruendo altri pulsanti per effettuare i *link* tra di esse. Potremmo anche, ed è quello che faremo, costruire dei *link* basati su altri oggetti, programmare altri eventi, rendere insomma l'ipertesto più completo, più dinamico, più interessante.

Creare e programmare un pulsante non è l'unico modo di creare e percorrere i *link* di un ipertesto. Un *link*, ad esempio, può anche essere identificato e attivato attraverso delle parole calde, delle *hotword*. Toolbook consente di definire e programmare delle *hotword*: sono anch'esse *oggetti*, che tuttavia possono essere creati solo se si è già creato un campo e si è inserito, in esso, un testo. Torniamo alla pagina iniziale del nostro ipertesto. Nel campo su cui avevamo scritto la biografia di Leonardo, ad un certo punto, si parla della sua importanza come pittore. Selezionate ora la parola "pittore". Per selezionare una parola basta fare *clic* immediatamante prima della "p" e quindi, tenendo premuto il pulsante del mouse, trascinare il mouse verso destra fino a che l'intera parola non risulterà evidenziata. Basta avere un minimo di esperienza di videoscrittura per riuscire a compiere questa operazione con una certa disinvoltura. Inoltre, se il testo che vogliamo rendere "caldo" è costituito da una sola parola, essa potrà essere selezionata anche cliccando velocemente due volte al suo interno. A questo punto, nel menu **Text** selezionate la voce **Create Hotword**. Il testo selezionato è diventato una parola calda, ovvero un *oggetto* diverso dal campo di testo, un oggetto che, in quanto tale, possiede caratteristiche sue proprie (tra le altre cose, può avere un *colore* diverso dal resto del testo del campo) e può essere istruito per reagire ad un evento, proprio come il pulsante e tutti gli altri oggetti. Quando il testo della parola calda è evidenziato, nel menu **Object** scegliete la voce **Hotword Properties** e definite pure lo *script* della parola calda, che potrebbe essere identico a quello del pulsante "Leonardo Pittore", visto che il tipo di link che vogliamo attivare è lo stesso. Fatelo, e provate l'effetto. Attenzione, però ! La parola calda è un oggetto anomalo: perchè il suo *script* possa essere eseguito si devono verificare alcune condizioni. In particolare, si dovrà modificare una delle *proprietà* del campo che la contiene. Il campo, infatti, salvo indicazioni contrarie da parte nostra, è in genere "aperto". Provate a costruire un campo e a passare in modalità "lettore": cliccando sul campo, il cursore diventerà una barra e noterete che, anche a livello "lettore", vi sarà data la possibilità di scrivere su di esso. Questo significa che un evento come il *Button Click*, anche se opportunamente definito e programmato, non avrà esito, nè, di conseguenza, avrà esito un *clic* sulle eventuali parole calde. Se vogliamo rendere attivo lo *script* di un campo, e conseguentemente delle sue eventuali parole calde, quindi, dobbiamo "bloccare" il contenuto del campo. Selezionate il campo e richiamate quindi le **Field Properties** dal menu **Object**. Tra le voci selezionabili, cliccate su **Activated (typing disabled)**. Confermate, e a quel punto il campo sarà "bloccato", non consentirà cioè al lettore di modificarne il testo. Consentirà invece l'esecuzione degli *script* associati, ad esempio, alle parole calde. Provate ora, in modalità lettore, il funzionamento della *hotword* che avevate costruito e attivato. Noterete che essa si comporta esattamente come il pulsante e che, tra le altre cose, il cursore del mouse assume una caratteristica forma, una "manina", non appena tocca la zona dello schermo corrispondente alla parola calda. Questo comportamento viene attribuito automaticamente da Toolbook alle parole calde, a meno che non gli si dica, espressamente, di fare il contrario. Tuttavia, esso può essere attribuito a qualunque *oggetto*, per rendere evidente la sua eventuale natura di *hot-spot*. Vedremo meglio, più avanti, in che modo.

Nascondere e mostrare gli oggetti...

Proviamo, adesso, a rendere più dinamico e più attraente il nostro ipertesto. Sulla pagina dedicata a "Leonardo Pittore", dove abbiamo già l'immagine della *Gioconda* e un testo, vorremmo, ad esempio, che fosse possibile richiamare una scheda sul celebre dipinto. Ci piacerebbe, tuttavia, che questo non comportasse necessariamente un salto di pagina, poichè la scheda non rappresenta un nodo, ma un approfondimento del nodo su cui stiamo lavorando. Vorremmo, in sostanza, che la scheda sulla *Gioconda* fosse "nascosta" sulla stessa pagina e visibile solo quando l'utente deciderà di richiamarla. Proviamo dunque a creare un nuovo *campo*, su cui scriveremo la scheda sulla *Gioconda*. Definiamone il nome, chiamiamolo "Notizie". Posizioniamolo ora in modo tale che risulti parzialmente sovrapposto all'immagine del dipinto.

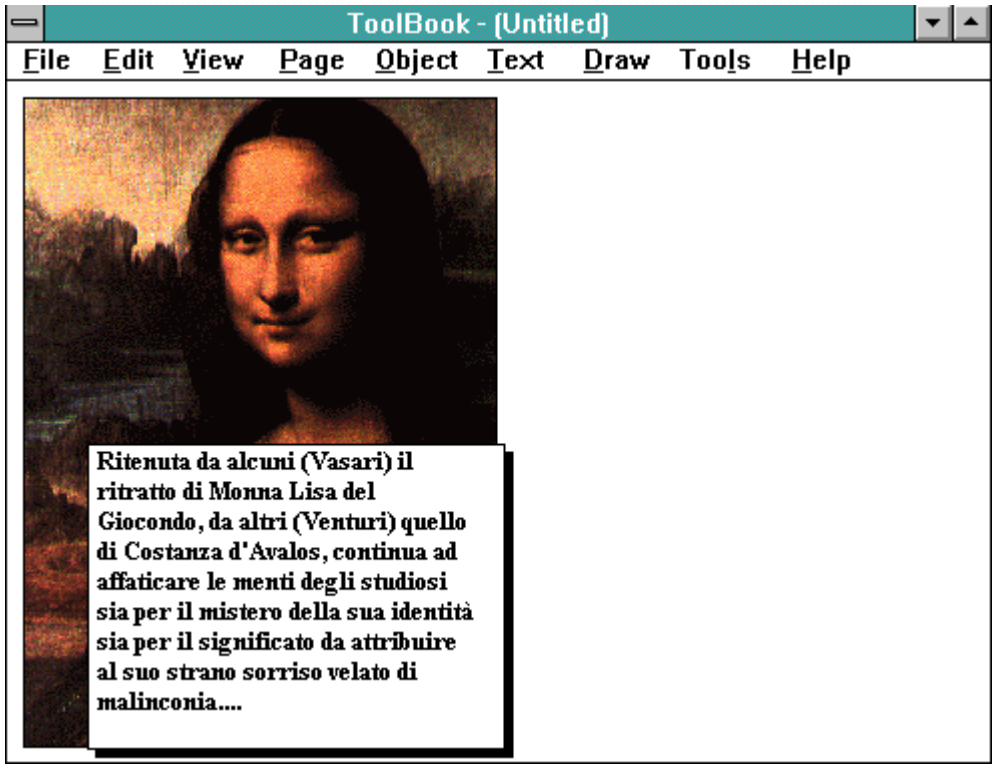

Figura 18: un campo di testo è stato creato e posizionato sopra l'immagine. Ora potrà essere mostrato o nascosto attraverso il clic del mouse...

"Blocchiamo" il campo, nel modo che sappiamo, in modo che il testo non sia modificabile dal lettore. Questo campo, in realtà, non dovrà essere visibile sulla pagina, perchè altrimenti coprirebbe l'immagine, che a noi interessa invece mantenere in evidenza. Il campo dovrà rimanere nascosto finché l'utente non deciderà di richiamarlo per avere maggiori notizie sulla *Gioconda*. Per "nascondere" il campo, per il momento, possiamo richiamare un semplice strumento che serve a inviare istruzioni a Toolbook, la finestra di comando, o **commandWindow**. Per richiamarla basta selezionare la voce **Command** dal menu **View** [**A1**]. Apparirà una piccola finestra di tipo Windows, che potrete spostare a piacimento sullo schermo e ingrandire o rimpicciolire a seconda delle necessità.

Selezionate il campo che volete "nascondere" e, sulla finestra di comando, scrivete **hide**. Premete il tasto < **Invio** > (per qualcuno sarà < **Enter** >) e vedrete "sparire" l'oggetto selezionato.

Il campo, ora, non è più visibile, ovvero, c'è, ma non si vede. Vogliamo che a mostrarlo sia il lettore, quando farà *clic* sull'immagine della *Gioconda*. Selezioniamo perciò l'immagine: è anch'essa un *oggetto*, per cui tra le sue proprietà avrà anche lo *script*, che potrà essere richiamato e modificato attraverso l'apposito pulsante dalla relativa finestra di dialogo. Definiamone dunque il comportamento:

```
to handle buttonClick
        show field "Notizie" of this page
end
```

L'istruzione *show* serve per rendere visibili (mostrare) oggetti che, quando si verifica l'evento, visibili non sono. All'istruzione *show* segue, come sempre in Toolbook, un riferimento preciso all' *oggetto* su cui si vuole che il comando abbia effetto: in questo caso si tratta di un "campo" (*field*) che si chiama "Notizie" e si trova su questa pagina (*of this page*). Quest'ultima precisazione non è strettamente necessaria, nel senso che Toolbook, in mancanza di indicazioni precise sull' *indirizzo* dell' *oggetto*, andrà sempre e comunque a cercarlo prima di tutto sulla pagina. Tuttavia, il campo poteva trovarsi sullo "sfondo". In quel caso, una specificazione del tipo *of this background* sarebbe stata obbligatoria.

Proviamo ora questo *script*. Passate in modalità "lettore" e cliccate sulla *Gioconda*, che ora è un *hot-spot* e reagisce al *clic* eseguendo lo *script*: verrà mostrato, cioè reso "visibile", il campo "notizie", che contene la scheda che avevamo scritto.

Ora dobbiamo far sì che l'utente possa nuovamente nascondere quel campo, perché magari ha già letto ciò che gli interessa e vuole rivedere con calma l'immagine del dipinto. Tra le tante tecniche utilizzabili per ottenere questo risultato, una delle più semplici e intuitive (per il "lettore") consiste nel fare in modo che cliccando sul campo, in risposta al *clic* esso stesso si nasconda. Ormai abbiamo capito perfettamente il meccanismo: dobbiamo associare uno *script* all' *oggetto* campo. Ritorniamo dunque in modalità "autore", selezioniamo il campo e modifichiamone lo *script* in questo modo:

```
to handle ButtonClick
        hide self
end
```

Il comando **hide** (che abbiamo già usato nella finestra di comando) è il reciproco del comando **show**: nasconde l' *oggetto* al quale si fa riferimento. Avremmo potuto scrivere *hide field "notizie" of this page* (funzionerebbe perfettamente), ma poichè l'oggetto su cui si vuole agire è lo stesso su cui è stato definito l'evento che attiva le istruzioni, è sufficiente usare la parola *self*. La parola **self**, infatti, per Toolbook indica l'oggetto che è stato "bersagliato" dall'evento (il *ButtonClick*), ovvero il **target** dell'evento: in questo caso il campo non deve far altro che nascondere sé stesso, e può quindi essere identificato e riconosciuto con il termine **self**, ad ulteriore riprova dell'elasticità e della somiglianza di *Openscript* con la lingua inglese.

Come sempre, proviamo il funzionamento dello *script*. Passate in modalità "lettore" e cliccate sul campo. Esso scompare. Per farlo riapparire, a questo punto, sarà sufficiente cliccare sulla *Gioconda*, e quindi cliccare ancora sul campo per nasconderlo. Abbiamo costruito un meccanismo piuttosto efficiente ! Abbiamo usato un comando *show* per mostrare l'oggetto e un comando *hide* per

nasconderlo. Su questa base potremmo divertirci a costruire infinite situazioni, rendendo così più dinamico e accattivante il nostro ipertesto.

Raggruppare gli oggetti...

Vediamo ora di prendere confidenza con un altro *oggetto* particolare, che potrebbe rivelarsi molto utile: il gruppo, in Toolbook **Group**. E' uno dei tanti oggetti (sfondi, pagine, campi, pulsanti) che Toolbook permette di creare. Al contrario degli altri, però, il gruppo si può creare soltanto partendo da *oggetti* già esistenti, "raggruppati", appunto, in un unico *oggetto*.

Proviamo ad esemplificare questo nuovo concetto intervenendo ancora sul nostro ipertesto. Andiamo alla pagina "Leonardo Scienziato". Supponiamo di voler consentire all'utente di poter approfondire alcuni argomenti legati alla figura di Leonardo da Vinci in quanto scienziato. Sulla falsariga di quanto abbiamo fatto finora, creeremo due pulsanti: uno con l'etichetta "Le macchine da guerra" e l'altro con l'etichetta "Gli studi sul volo". Su ciascuno dei pulsanti applicheremo uno *script* con un'istruzione del tipo **go to page...** (se vogliamo che l'utente richiami delle pagine che tratteranno di quegli argomenti) o del tipo **show field...** (se vogliamo che l'utente richiami delle piccole schede di approfondimento nascoste nel contesto della pagina).

Ora, selezionate entrambi i pulsanti: dovete cliccare su uno di essi, e quindi, tenendo premuto il tasto "shift" (quello che di solito si usa per scrivere le lettere maiuscole), sull'altro. I quadratini che indicano che l'oggetto è selezionato sono ora evidenti su entrambi gli oggetti [**A19**]. Dal menu **Object** scegliete quindi la voce **Group**. Avete così "raggruppato" i due pulsanti in un unico oggetto, detto appunto **Group**. Selezionando questo nuovo oggetto, notate che sul menu **Object** compare anche la voce **Group Properties**, scegliendo la quale richiamerete la consueta finestra di dialogo, su cui potrete, ad esempio, scrivere il nome che volete dare al nuovo oggetto (chiamiamolo "approfondimenti"). Noterete ora che, selezionando il gruppo e trascinandolo, tutti gli *oggetti* di cui esso è composto (nel caso specifico i due pulsanti) si spostano di conseguenza. Raggruppare gli oggetti, quindi, può essere utile anche a fini pratici, per rendere più veloci le operazioni di impaginazione, ad esempio. Noterete, tuttavia, che benchè "raggruppati", i due pulsanti non perdono le loro distinte identità, nel senso che continuano a conservare *tutte* le loro *proprietà* (non solo l' *etichetta*, ma anche lo *script*) [**A19**].

Vogliamo ora che questi pulsanti, riuniti in un gruppo, siano visibili solo se l'utente intende approfondire l'argomento "Leonardo Scienziato". Rendiamo dunque invisibile il gruppo (esattamente come avevamo fatto per il campo). Quindi, creiamo un altro pulsante con l'etichetta "Approfondimenti". Il pulsante potrebbe contenere questo *script*:

```
to handle buttonClick
        set the visible of group "Approfondimenti" of this page \
        to not the visible of group "Approfondimenti" of this page
end buttonClick
```

Avremmo potuto ricorrere a dei comandi del tipo *show* e *hide*, come in precedenza (salvo il fatto che, non conoscendo ancora alcuni elementi di *Openscript*, avremmo avuto bisogno di due pulsanti, "mostra approfondimenti" e "chiudi approfondimenti"). In realtà, i comandi *show* e *hide*, agiscono esplicitamente su una delle *proprietà* degli oggetti, **Visible**. Come tutte le *proprietà* degli oggetti, essa può essere modificata attraverso uno *script*, usando il comando **Set**. "Mostrare" un oggetto, in sostanza, significa, di fatto, attribuire alla sua proprietà **Visible** un particolare valore. Poichè la

proprietà in questione presuppone soltanto due alternative (l'oggetto o è "visibile" o non lo è), il valore di **Visible** sarà di tipo *logico*, ossia "vero" quando l'oggetto è visibile (in Openscript: **true**), "falso" quando l'oggetto è nascosto (in Openscript: **false**). Quindi, per "mostrare" un oggetto posso sì dire a Toolbook *show...*, seguito dal riferimento all'oggetto, ma posso anche dirgli di "settare" (*set*) la proprietà che ne definisce la visibilità (*visible of...*) a "vero" (*to true*). Nel caso appena riportato, poi, il "settaggio" viene in qualche modo reso automatico, poichè il riferimento *visible....to not visible* attribuirà alla proprietà il valore "vero" o "falso" in relazione al valore precedentemente espresso. In sostanza, sposterà il "settaggio" su "vero" quando esso su "falso" e viceversa. Un piccolo trucco (ma a pensarci bene, in inglese il concetto potrebbe essere espresso proprio con le stesse parole), uno *script* che consente di mostrare e nascondere alternativamente il gruppo "approfondimenti" cliccando sullo stesso pulsante.

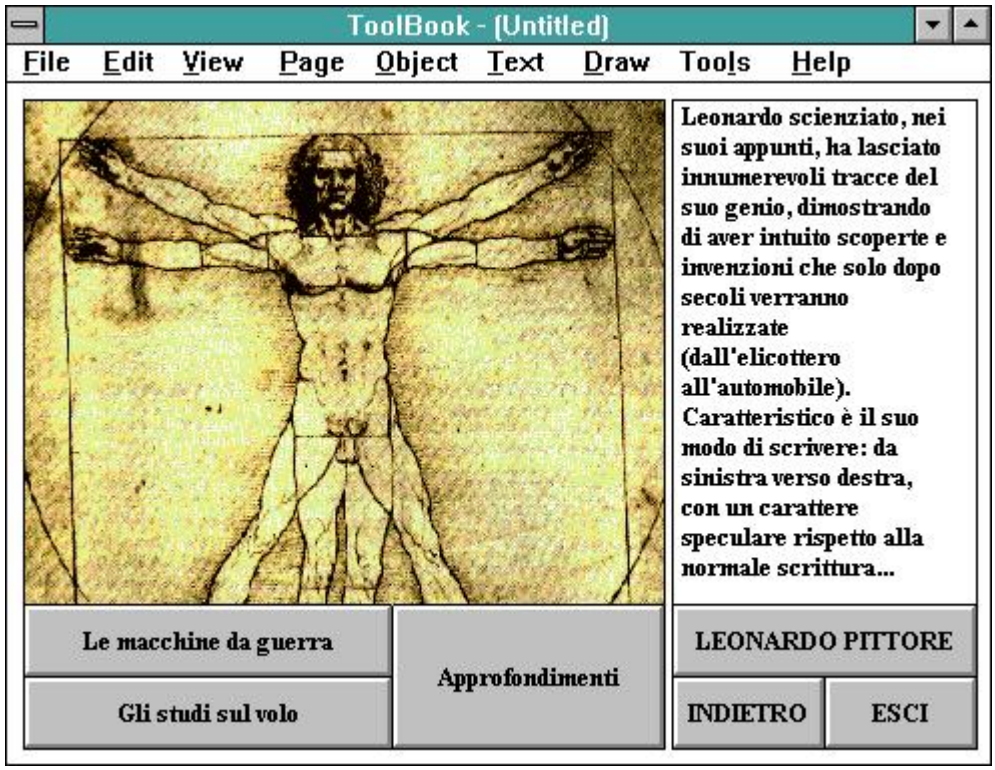

Figura 19: esempio di pagina su cui è stato inserito un pulsante che richiama un "gruppo" costituito da altri pulsanti

Gestire gli eventi, cliccare, sfiorare...

Abbiamo dunque costruito degli *oggetti*, e li abbiamo programmati perchè reagissero ad un determinato *evento*. L'evento prescelto è stato il *clic* del mouse, il **buttonClick**, nella lingua di Toolbook. A questo punto è lecito domandarsi se si possano programmare le reazioni degli oggetti ad altri eventi, e a quali. In realtà, in Toolbook, gli eventi a cui si possono associare delle istruzioni sono molteplici. Vedremo più avanti di puntualizzare meglio questo argomento. Per adesso, limitiamoci a ricordare che gli eventi più facilmente riconoscibili sono quelli che l'utente determina usando il *mouse*. Il **buttonClick**, ad esempio, se ci pensate attentamente, è la somma di due situazioni: quando

facciamo *clic* con il *mouse*, infatti, il pulsante, in un primo momento, viene abbassato, e in un secondo momento torna nella posizione originaria. Toolbook è in grado di riconoscere questi eventi: quando l'utente preme il pulsante del mouse, capisce che è avvenuto un **buttonDown**; quando il pulsante viene rilasciato riconosce un **buttonUp**. Il **buttonClick** che abbiamo finora usato non è che l'insieme delle parti di questa sequenza.

Tutto ciò che può essere riconosciuto da Toolbook come *evento* può essere *gestito* dallo *script* di un oggetto: il senso della terminolgia *to handle...* che, come abbiamo visto, è una costante di tutti gli *script* realizzati finora, è proprio questo. L' *handler* è una modalità di gestione dell'evento che ad esso è associato, e, in quanto tale, può riferirsi a molteplici eventi. Attenzione ! Non si deve confondere l' *handler* con lo *script*. Nello script di un oggetto possono essere indicati più "gestori" di eventi, possono cioè convivere più cicli di istruzioni gestite da un *to handle...*, a patto che ciascun ciclo si chiuda con un *end* e che l'evento associato al gestore non sia mai lo stesso, perchè in tal caso si aprirebbe un conflitto logico che Toolbook non sarebbe in grado di risolvere. Più eventi, quindi, possono essere programmati in uno *script*, ma non più volte lo stesso evento: se si vuole che la risposta ad un evento sia una complessa serie di operazioni, esse dovranno essere dettagliate e messe in sequenza all'interno di un unico gestore *to handle..... end*, oppure associate ad eventi diversi, ciascuno dei quali gestito da un proprio *handler*. Torniamo al nostro ipertesto. Sulla base di questa logica, ad esempio, l'immagine della *Gioconda*, invece dello *script* che le avevamo attribuito (*to handle buttonClick... end*), potrebbe contenere questo script:

```
to handle buttonDown
        show field "notizie" of this page
end buttonDown

to handle buttonUp
        hide field "notizie" of this page
end buttonUp
```

In questo script vengono gestite le risposte a due eventi diversi, il *buttonDown* (il pulsante del mouse che si abbassa) e il *buttonUp* (il pulsante del mouse che viene rilasciato). Provate l'effetto: *finchè* il lettore tiene premuto il pulsante del mouse (**buttonDown**) sull'immagine della *Gioconda*, il campo "notizie", quello con la scheda, verrà mostrato, non appena il lettore lo rilascerà (**buttonUp**) esso verrà immediatamente nascosto.

Analogamente, possono essere gestiti e programmati in Toolbook altri eventi generati dal *mouse*: quando il cursore "sfiora" l'area occupata da un oggetto, anche se l'utente non clicca, si avrà comunque un **mouseEnter**. Quando l'utente riporta il cursore fuori dalla stessa area si avrà invece un **mouseLeave**. Anche a questi eventi si possono associare delle reazioni. Provate, ad esempio, a intervenire ancora sullo *script* dell'immagine della Gioconda, sostituendolo con il seguente:

```
to handle mouseEnter
        show field "notizie" of this page
end mouseEnter

to handle mouseLeave
        hide field "notizie" of this page
end mouseLeave
```

Otterrete sempre lo stesso effetto (il campo viene mostrato - il campo viene nascosto), ma avrete gestito gli eventi in modo tale che le notizie sul dipinto appaiano all'utente anche senza che, da parte sua, sia necessario intervenire con un *clic* (sarà sufficiente che "tocchi" l'immagine con il cursore).

Cambiare la forma del cursore...

Definire il comportamento di Toolbook di fronte ad eventi di questo genere può tuttavia comportare dei rischi: rischi "tecnici" (non avete notato, ad esempio, che anche lo spostamento del cursore sull'oggetto che è stato mostrato implica l'esecuzione del *mouseLeave* ?), e rischi di natura comunicativa, poichè a volte, mostrando e nascondendo oggetti sulla base del semplice "sfioramento", si può "disorientare" l'utente. In realtà, gli eventi mouseEnter e mouseLeave, e la tecnica della programmazione di una reazione allo "sfioramento" che essi implicano, sono molto importanti nell'economia generale di un ipertesto. Questi eventi, infatti, vengono comunemente utilizzati non tanto per rivelare oggetti (e meno che mai per "saltare" da una pagina all'altra !), quanto per far capire all'utente che sullo schermo ci sono degli *hot-spot* attivi. In che modo ? Semplicemente facendo sì che lo "sfioramento" dell'area in questione comporti la modifica della forma del cursore, così come avviene quando si "toccano" le parole calde. Come è possibile cambiare la forma del cursore ? Non è difficile: la forma è una delle *proprietà* del cursore, e come tale può essere cambiata e "settata" anche attraverso uno *script*. Il cursore, tuttavia, non è un *oggetto* Toolbook, ma una caratteristica dell'intero *sistema*. La proprietà che dovremo modificare, quindi, sarà la proprietà *di sistema* che identifica e determina la forma del cursore. Quelli che seguono sono due *handlers* che gestiscono gli eventi legati allo "sfioramento" e reagiscono cambiando la forma del cursore. Possiamo metterli (o aggiungerli) ad uno qualsiasi degli *script* che abbiamo finora definito.

```
to handle mouseEnter
        set sysCursor to 44
end mouseEnter

to handle mouseLeave
        set sysCursor to 1
end mouseLeave
```

Il significato di questi due *handlers* è facilmente comprensibile: quando si "sfiora" un oggetto (*mouseEnter*) al cursore viene data (*set*) la forma (*sysCursor*, perchè è una proprietà di sistema) che Toolbook identifica attraverso il numero 44, la classica "manina". Il valore del "settaggio", in questo caso, non è "logico", come avevamo visto in precedenza a proposito della proprietà *visible*, ma è un numero, perchè il cursore può assumere molte forme, e Toolbook è in grado di identificarle solo così. Quando il cursore esce dall'area dell'oggetto (*mouseLeave*) al cursore viene restituita la forma originaria, identificata dal numero 1. Divertitevi pure a verificare a quali forme corrispondono tutti gli altri numeri compresi tra 1 e 44.

Pulsanti trasparenti...

Pensate adesso a quante possibilità si aprono, nella messa a punto definitiva del vostro ipertesto, ora che vi sono chiare (speriamo !) le tecniche di gestione degli eventi e quello dello "sfioramento". Pensate, ad esempio, a come potreste intervenire sull'immagine della *Gioconda*: potreste, ad esempio, ritagliare nell'immagine stessa delle aree calde, che l'utente potrebbe "scoprire" sfiorando l'immagine stessa e osservando i cambiamenti della forma del cursore; e potreste fare in modo che ad un *clic* su di ciascuna di esse venga mostrata un'altra immagine, magari l'ingrandimento di un particolare. Se avete

seguito attentamente il filo che lega gli esempi che abbiamo illustrato, dovreste essere già in grado di complicare in tal senso l'ipertesto, senza particolari difficoltà. Avrete tuttavia bisogno di qualche chiarimento a proposito dei *pulsanti trasparenti*. Già, perchè per trasformare in *hot-spot* delle aree che si sovrappongono all'immagine senza nasconderla (o che, in generale, si sovrappongono ad un *oggetto* senza nasconderlo, a meno che non siano parole calde in un campo) dovrete ricorrere a dei pulsanti, o a degli oggetti, *trasparenti*. Fino a questo momento avete imparato a "nascondere" gli oggetti, non a renderli *trasparenti*. E tra le due cose c'è una bella differenza. Un oggetto "nascosto", infatti, non è attivo, non può cioè ricevere dei messaggi e reagire a degli eventi. Un oggetto "trasparente", invece, è del tutto *attivo* e si comporta come uno qualsiasi dei vostri pulsanti. Rendere trasparente un pulsante (così come qualsiasi altro oggetto) non è difficile. Che cos'è, infatti, la trasparenza, se non una *proprietà* ? Prendete un pulsante, dunque, e sulla finestra di dialogo delle sue proprietà cliccate sulla voce **transparent** per selezionarla (non avrete fatto altro che "settare" la *proprietà* **transparent** al valore *true*). Noterete che il pulsante diventerà trasparente, ma non scomparirà del tutto (in effetti, non verrà "nascosto"). Rimarranno visibili il contorno, l'etichetta, e, in parte, con effetti a volte graziosi, il colore di fondo. Per rendere un oggetto completamente trasparente bisogna intervenire ancora sulle sue *proprietà*: il pulsante, ad esempio, non dovrà avere nessun bordo (sulla casella corrispondente alla sua proprietà **borderStyle** bisogna selezionare la voce **none**), non dovrà avere etichetta e dovrà essere di colore bianco su bianco. Solo a quel punto sarà del tutto *trasparente* al lettore. Tuttavia rimarrà attivo (oltre che "fisicamente" selezionabile): potrà essere ridimensionato e spostato, in modo da sormontare solo il volto della Gioconda, ad esempio. Provate ad associare a quel pulsante uno script in cui vengono definite le reazioni allo "sfioramento" (appare la "manina" - o una lente d'ingrandimento) e quella al *clic* (viene mostrato un particolare del celebre sorriso di Monna Lisa): il lettore avrà l'illusione di sfiorare e il volto della Gioconda e di interagire con il suo sorriso. In realtà sfiorerà e interagirà soltanto con quel pulsante *trasparente*.

Costruire un archivio...

Il nostro ipertesto ha preso ormai definitivamente forma, e comincia ad essere divertente e interessante. Tuttavia, gli manca ancora qualcosa. Questo "qualcosa" potrebbe consistere in un piccolo "archivio", magari facilmente aggiornabile, fatto di piccole schede su cui sono riportate, ad esempio, brevi notizie su alcuni personaggi vissuti al tempo di Leonardo, gli stessi che magari, nei testi inseriti nelle pagine dell'ipertesto, sono stati più volte citati. Per costruire e usare un archivio siffatto, Toolbook ci mette a disposizione due *oggetti* molto interessanti, di cui vedremo adesso di scoprire le caratteristiche: questi due oggetti si chiamano **recordField** e **viewer**. Non sono necessariamente associati, ma parlarne insieme può rendere più agevole la loro comprensione. Cominciamo dai **recordField**, i "campi di record". Si tratta di campi di testo speciali, che possono essere creati e definiti soltanto sullo "sfondo" delle pagine. Ciò significa che un **recordField** è visibile su ogni pagina che condivide quello stesso "sfondo". Tuttavia, noterete ben presto che, se provate a scrivere su uno di questi campi mentre siete ancora posizionati sul **Background**, non ci riuscirete, mentre, tornando alla pagina, il campo si comporterà come un qualsiasi campo di testo *aperto*. Questa è la particolarità del **recordField**: è un oggetto esclusivo dello "sfondo" che, relativamente ad una delle sue proprietà, il *testo*, può assumere un valore diverso su ogni pagina. E' in sostanza l'oggetto ideale per costruire uno schedario, poichè è simile, in tutto e per tutto, ai campi dei *database*. Tra l'altro, i testi inseriti nei campi di record sono gli unici, in Toolbook, che possono essere stampati *come testi* (scegliendo la voce **Print Report** dal menu **File**), mentre i normali campi di testo possono essere stampati solo in modalità grafica insieme al resto della schermata (scegliendo la voce **Print Pages** dal menu **File**).

Come possiamo, dunque, costruire il nostro archivio, e come possiamo richiamarlo dalle pagine del

nostro ipertesto (sullo sfondo delle quali, a pensarci bene, non possiamo nemmeno inserire dei campi di record, perchè esse sono già quasi completamente "piene") ? Ci può aiutare, in tal senso, il secondo oggetto di cui parleremo, il **viewer**. Il **viewer** è, a tutti gli effetti, una *finestra* Windows, proprio una di quelle finestre che si vedono sullo schermo quando si accende il computer. Più specificamente, il **viewer** è una finestra Windows che "contiene" una pagina di Toolbook, con relativo sfondo. Anche la finestra su cui avete lavorato finora è un **viewer**, all'interno del quale Toolbook vi ha mostrato, di volta in volta, una pagina. Ora, se si possono creare dei viewers, e se un viewer può contenere una pagina, non dovrebbe essere difficile trovare una soluzione al nostro attuale problema.

Cominciamo, quindi, a costruire il nostro "archivio": poichè non possiamo creare dei **recordField** sullo sfondo delle pagine che abbiamo finora creato, e poichè non ci servirebbe a nulla nemmeno creare una nuova pagina (che avrebbe lo stesso "sfondo" delle altre), dovremo creare un *nuovo* sfondo. Dal menu **Object** è sufficiente scegliere la voce **New Background**. Un nuovo sfondo verrà creato e, automaticamente, ci apparirà una nuova pagina, vuota, senza la quale lo sfondo non potrebbe esistere. Costruiremo il nostro archivio su questo nuovo sfondo, a cui, volendo, possiamo dare un nome (chiamiamolo "archivio", per comodità), un colore e una dimensione diversa da quella dello sfondo precedente, che verrà applicata a tutte le pagine che da questo momento in poi creeremo. Per fare tutto questo basta richiamare la finestra relativa alle proprietà del nuovo sfondo (come sappiamo). Ora dobbiamo posizionarci sul **Background** e creare un **recordField**, come un qualsiasi altro oggetto. Dovremo poi tornare sulla pagina e cominciare a inserire sul **recordField** il testo che vogliamo. Abbiamo già un archivio, composto di una sola scheda. Non ci credete ? Create una nuova pagina (come sapete fare) e osservate che cosa succede: nella nuova pagina il **recordField** appare vuoto, è pronto, cioè, per l'inserimento di un altro testo. Aggiungete pure quante "schede" volete al vostro archivio: per farlo sarà sufficiente creare nuove pagine sullo sfondo che abbiamo chiamato "archivio".

Il problema, ora, consiste nel richiamare queste "schede", ovvero le pagine dello sfondo "archivio" su cui abbiamo scritto dei testi utilizzando un **recordField**, *dalle* pagine del nostro ipertesto. Ci serve un **viewer**, evidentemente, diverso dalla finestra principale, perchè ciò che vogliamo è che le pagine dello sfondo "archivio" si *sovrappongano* a quelle che avevamo chiamato "Leonardo pittore", "Leonardo scienziato" ecc. Il **viewer** si costruisce cliccando su un pulsante del box degli strumenti che mostra l'icona di una finestra.

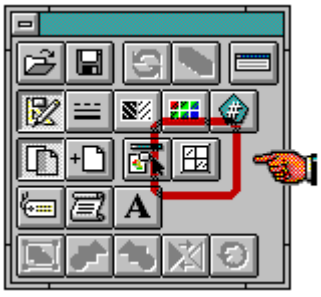

Figura 20: il box dei menu con, in evidenza, l'icona attraverso la quale si possono costruire o modificare i "viewers"

Apparirà un box di dialogo sul quale dovremo premere il pulsante **New**. Confermiamo le opzioni successive, fino a che non ci troveremo davanti la finestra **Viewer properties**.

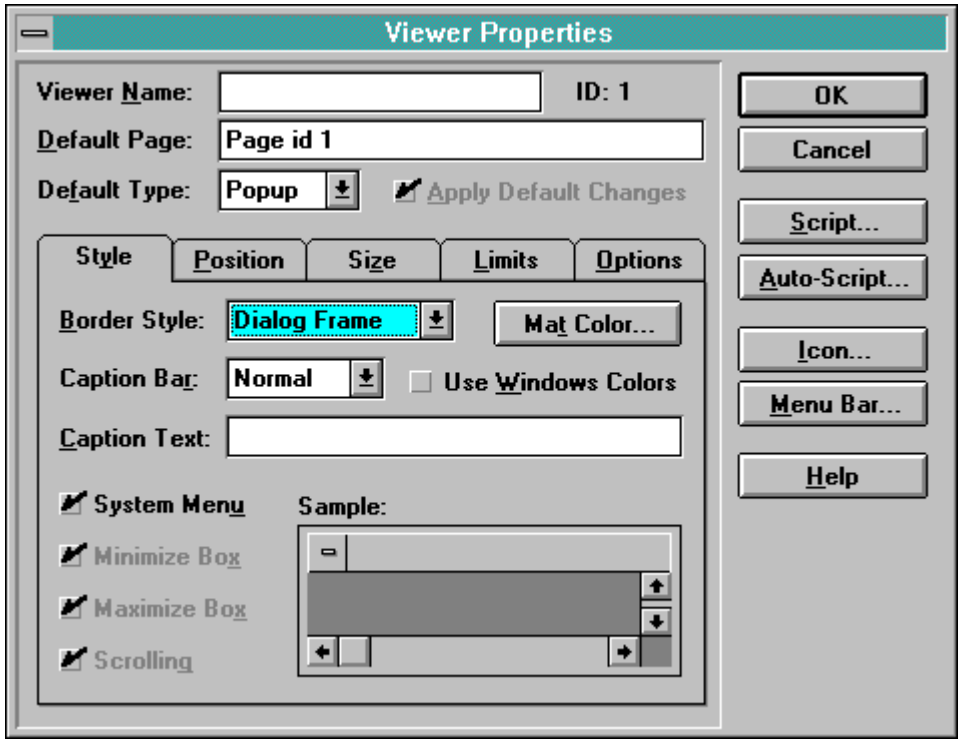

Figura 21: la finestra di dialogo che consente di definire le caratteristiche, le proprietà e il comportamento di un "viewer"

Essendo un *oggetto*, il viewer può essere definito e modificato a piacimento. Chiamiamolo "scheda", ad esempio, diamogli una forma, una dimensione, una posizione (**Center**, ad esempio, se vogliamo che venga visualizzato sempre al centro dello schermo) e, tra le opzioni, selezioniamo **Close on button Click**, perchè vogliamo che questa particolare finestra si chiuda, una volta richiamata, ad un semplice *clic* del mouse. Confermiamo tutto, finchè il **viewer** non apparirà sullo schermo, sotto forma di una finestra sovrapposta alla pagina su cui stavamo lavorando. Una finestra che, per il momento, mostrerà soltanto una pagina vuota. Per chiuderla, sarà sufficiente passare in modo "lettore" e fare un *clic*, poichè avevamo selezionato l'opzione *Close on button Click*. Ulteriori informazioni sull'oggetto **viewer** verranno date negli approfondimenti [**A31-A32**]

Ora abbiamo una serie di pagine con dei testi (le pagine che abbiamo aggiunto allo sfondo "archivio") e una finestra (il **viewer** "scheda") che, come tale, potrebbe contenerle e visualizzarle in qualunque momento. Si tratta soltanto di capire come questo sia possibile. Andiamo, a titolo di esempio, alla pagina "Leonardo pittore", e supponiamo che nel testo che avevamo inserito in quella pagina ci sia un riferimento al maestro di Leonardo, Verrocchio. Senza perdere di vista il contesto (la pagina "Leonardo pittore") vogliamo far sì che l'utente, cliccando su quella parola, richiami una "scheda" su Verrocchio, una di quelle "schede" che abbiamo inserito nel **recordField** di una pagina dello sfondo "archivio", una pagina che, magari, avremo provveduto a chiamare proprio "Verrocchio". Trasformiamo la parola "Verrocchio" in **hotword**. Richiamiamo lo script della parola calda e modifichiamolo nel modo che segue:

```
to handle buttonClick
        open viewer "scheda"
        currentPage of viewer "scheda" = page "Verrocchio" of background "archivio"
        show viewer "scheda"
end buttonClick
```

Che cosa succede quando l'utente clicca sulla parola ? Toolbook "apre", cioè rende disponibile, una finestra, quella che abbiamo chiamato "scheda" (*open viewer "scheda"*). Subito dopo, "mette" nella finestra una pagina (*currentPage of viewer*), in particolare la pagina dello sfondo "archivio" che abbiamo chiamato "Verrocchio". Infine, mostra (*show*) il viewer, e quindi la pagina ad esso associata. Provate l'effetto passando a livello "lettore".

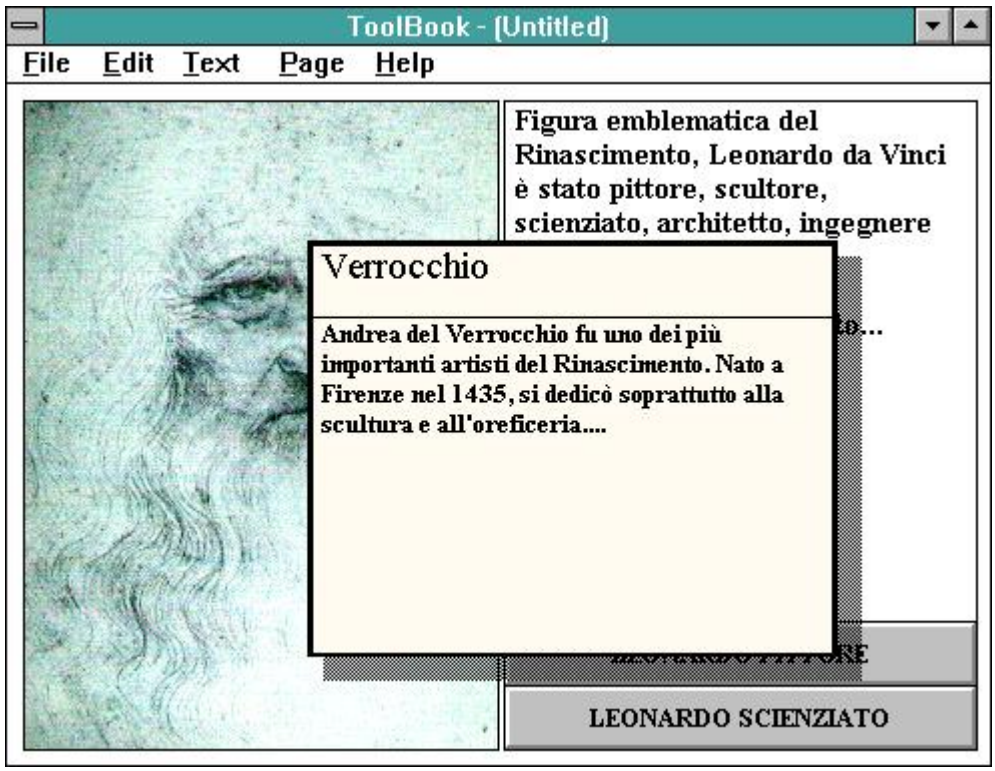

Figura 22: una piccola pagina, contenente una scheda di approfondimento, è stata richiamata e sovrapposta alla pagina principale utilizzando un "viewer"

Certo, avremmo potuto ottenere il medesimo risultato creando un normale campo di testo nascosto sulla pagina e mostrandolo con un comando *show*... Ma se il nostro ipertesto avesse decine e decine di pagine, con relativi testi, e se certe parole calde si ripetessero spesso, quanto tempo perderemmo solo a copiare e incollare gli script e i campi a cui essi fanno riferimento ? Pensate inoltre alla facilità con cui, in questo modo, sarà possibile arricchire e modificare il nostro archivio.

Una semplice variabile...

A poco a poco il nostro ipertesto ha assunto un aspetto più articolato. Ora, vorremmo tuttavia entrare

maggiormente nel merito dei contenuti: i testi dei campi che abbiamo inserito nelle pagine cominciano ad essere costellati di parole calde, ognuna delle quali mostra una scheda del nostro archivio o un piccolo campo di testo con approfondimenti e osservazioni varie. Cominciamo così a costruire, nel testo della pagina "Leonardo pittore", una *hotword* con la parola "tempera", e la programmiamo in modo tale che quando l'utente *cliccherà* su di essa apparirà un campo di testo che spiega che cos'è la tecnica della pittura a tempera. Ormai sappiamo come fare: si apre lo script dell' *oggetto*, si definisce l' *evento* (un *buttonClick*), si dice a Toolbook di mostrare (*show*) un oggetto (un *field*, in questo caso) che si chiama, magari, proprio "tempera". Facciamo poi la stessa cosa con altre parole calde, ma ripetendo le operazioni descritte finora, ci accorgiamo che, benchè siano relativamente semplici da compiere, esse comportano comunque, man mano che aumenta la quantità degli *oggetti* su cui vogliamo programmare degli eventi, un notevole dispendio di tempo. Si richiede inoltre una particolare attenzione da parte del *programmatore* (ormai lo siete quasi !), in particolar modo per quel che riguarda la corretta identificazione degli oggetti a cui le istruzioni degli *script* si riferiscono. A questo punto, qualcuno si domanderà se questo è davvero l'unico modo di procedere. Possibile che non si possa andare avanti più velocemente ? Vedremo ora di indicare una strada per rendere più veloce e più semplice la messa a punto definitiva dell'ipertesto. Nell'esempio appena indicato lo *script* della parola calda potrebbe essere questo:

```
to handle buttonClick
        show field "tempera"
end buttonClick
```

Provate a sostituirlo con il seguente:

```
to handle buttonClick
        get text of target
        show field it
end buttonClick
```

Come può essere interpretato questo nuovo *script* ? Abbiamo già avuto modo di usare il comando *set*, e abbiamo capito che esso serve, ad esempio, a modificare una proprietà, a cambiare un valore, insomma, a "scrivere" qualcosa, direttamente in un oggetto o nella memoria del computer. Nell'esempio appena proposto si usa invece il comando **get**, che può essere considerato il reciproco di *set*. Così come *set* serve a "scrivere", si può dire che **get** serve a "prendere", o meglio, a "leggere". Leggere che cosa ? Ad esempio il valore di una *proprietà*, o qualunque altra caratteristica di un oggetto che possa essere "letta", ad esempio un testo, nella fattispecie il testo dell'oggetto "bersaglio", del **target**, che nel nostro caso è la parola calda, poichè lo *script* è stato inserito in una *hotword*. Toolbook è quindi in grado di "leggere" delle cose, siano esse un testo o un valore logico o numerico. Ma che cosa succede dopo che, attraverso un comando di tipo **get**, si è detto a Toolbook di "leggere" qualcosa ? Succede che Toolbook "memorizza" ciò che ha letto, in attesa di nuove istruzioni. E poichè ciò che ha letto potrebbe essere importante, perchè non vada perduto lo deposita in una *variabile*. E' assolutamente necessario affrontare, anche a livello di sguardo generale, il problema delle variabili, perchè, prima o poi, chiunque si trovi a lavorare in Toolbook dovrà fare i conti con esse. Il problema verrà successivamente approfondito [**B1-B2**]. Per il momento ci limiteremo a parlare di una particolare variabile, caratteristica di Toolbook, che il programma usa, appunto, per "memorizzare" ciò che di volta in volta gli abbiamo detto di "leggere". Questa variabile è talmente generica e semplice che in *Openscript* si chiama **it**, letteralmente "esso". Non è difficile capire che cos'è **it**: **it** è una specie di barattolo vuoto, che Toolbook riempie, di volta in volta e quando è necessario, con ciò che *deve* ricordare, ad esempio con tutto ciò che consegue all'esecuzione di un

comando di tipo **get**. Fuori di metafora, possiamo dire che **it** è una *variabile pura*: un contenitore che, provvisoriamente, conserva dei dati che potranno così essere riutilizzati. Per tornare all'esempio illustrato, quando diciamo a Toolbook di "leggere" (*get*) il testo dell' *oggetto* su cui l'utente ha cliccato (*text of target*), Toolbook provvede, automaticamente, a "conservare" il testo che ha appena letto in quel suo particolare e caratteristico contenitore a contenuto *variabile* che chiama **it**. Da quel momento in poi, potremo "utilizzare" facilmente quel valore, nel nostro caso un testo, come termine di paragone per eventuali verifiche, ad esempio, o, più semplicemente - è il caso dell'esempio proposto - per dire a Toolbook di mostrare (*show*) un campo di testo (*field*) il cui *nome* è uguale al *testo* della parola calda, ovvero uguale al testo che il programma ha appena "letto" e "memorizzato" in un contenitore e che quindi è sufficiente chiamare **it**. Il vantaggio che può derivare dall'uso di una *variabile* è evidente: per mostrare o nascondere *oggetti*, ad esempio, non sarà più necessario riaprire lo *script* delle parole calde (o dei pulsanti) attraverso i quali le operazioni *show* e *hide* vengono attivate e modificare il riferimento al *nome* dell' *oggetto* da mostrare o nascondere. Sarà sufficiente copiare su ciascuna parola calda lo *script* così com'è (ad esempio, uno *script* come quello sopra indicato), e aver cura di dare agli oggetti che si vogliono "mostrare" un *nome* uguale al *testo* della parola calda, al testo, cioè, che viene utilizzato come elemento *variabile*. Si guadagna certamente in velocità e in efficienza. Attenzione, però ! Non dimenticate mai che **it** serve a memorizzare dei dati solo *provvisoriamente*. Ogni volta che un nuovo comando comporterà l'uso di **it** da parte di Toolbook, l'eventuale precedente contenuto della variabile verrà inesorabilmente distrutto.

Dialogare con il lettore...

Una volta assimilati questi concetti, non dovrebbe essere difficile imparare ad estendere l'uso della *variabile* ad una gamma piuttosto ampia di situazioni: provate, a titolo di esercizio, a "rivedere" alla luce di queste ulteriori informazioni lo *script* che prima avevamo associato ad una parola calda per richiamare un *viewer* con una pagina del nostro archivio. Pensate, inoltre, a certe possibilità che si aprono, a come può aumentare, ad esempio, la soglia del dialogo che gli "autori" possono instaurare con i "lettori" che stanno consultando l'ipertesto. Supponiamo, ad esempio, di voler rendere maggiormente interattivo il rapporto tra l'utente e l'immagine della *Gioconda*. Costruiamo un pulsante, che metteremo sotto o accanto al dipinto. L' *etichetta* potrebbe essere "Particolari del dipinto". Porgrammiamo il pulsante con uno *script* come quello che segue:

```
to handle buttonClick
        request "Quale particolare del dipinto vuoi osservare ?" \
        with "Il sorriso" or "Le mani" or "Il paesaggio"
        show paintObject it
end buttonClick
```

Request è un comando che consente di far apparire sullo schermo un messaggio. Il messaggio sarà quello che noi stessi definiremo, scrivendolo, subito dopo l'istruzione *request*, tra virgolette. Esso comparirà in una finestra con caratteristiche date dal sistema Windows, a fondo grigio e con una barra colorata, e, di norma, sarà accompagnato da un pulsantino con la scritta "Ok", premendo il quale la finestra si chiude. In realtà i pulsanti che accompagnano il messaggio possono essere più di uno (ma non più di tre), e possono essere definiti e pilotati attraverso lo *script*. In questo caso abbiamo definito noi stessi la presenza di tre pulsanti (*with... or... or...*) e ne abbiamo determinato le *etichette* ("Il sorriso" ecc.). Ora, ogni volta che l'utente *clicca* su uno di questi pulsanti (se non lo fa, peraltro, la finestra con il messaggio non si chiude), Toolbook "memorizza" nella variabile *it* l' *etichetta* del pulsante. Possiamo quindi facilmente usare il valore della *variabile* come riferimento ad un *oggetto*. Nell'esempio, diciamo a Toolbook, dopo che l'utente avrà premuto uno dei tre pulsanti, di mostrare

(*show*) un'immagine (*paintObject*) che si chiama *esattamente* come l'etichetta del pulsante premuto, e che naturalmente, avremo avuto cura di importare nella pagina, di posizionare e di nascondere. Provate. L'utente avrà la sensazione di "dialogare" con il computer. E voi potrete, sviluppando ulteriormente questo spunto, dialogare con l'utente.

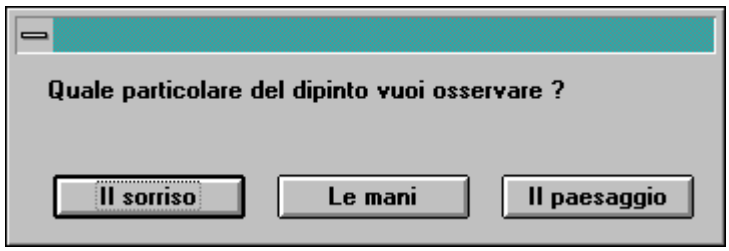

Figura 23: come si presenta una finestra di "request"

Infine, per completare il quadro delle più semplici applicazioni derivate dall'uso della *variabile*, proponiamo uno script nuovamente finalizzato ad instaurare una forma di rapporto interattivo tra "autore" e "lettore". Lo scopo dello *script* è far sì che su tutte le pagine dell'ipertesto appaia il nome di chi, in quel momento, è il "lettore". Allo scopo, avremo opportunamente creato e posizionato, sullo sfondo, un campo di testo (*field*) chiamato "nome":

```
to handle enterBook
        ask "Qual'è il tuo nome ?"
        set text of field "nome" of this background to it
end buttonClick
```

EnterBook ? Che cosa significa ? Nulla di particolare: anche il "libro" è un *oggetto*, no ? E se è un *oggetto*, allora possono essere programmati degli eventi che lo riguardano. Richiamate la finestra di dialogo delle proprietà del libro (**Book Properties**, dal menu **Object**), e vedrete che anche il libro può avere uno *script*. Aprite quello script e inserite in esso l' *handler* indicato. *EnterBook* significa semplicemente che le istruzioni che abbiamo appena definito verranno eseguite *quando* il libro verrà aperto, prima ancora, cioè, che il "lettore" faccia un qualsiasi *clic* con il mouse. Apparirà invece una finestra, attivata dal comando **Ask**, nella quale verrà riportato il messaggio "Qual'è il tuo nome ?", che voi stessi avete definito attraverso lo *script*. La finestra richiamata dal comando Ask avrà poi uno spazio bianco su cui l'utente potrà scrivere e due pulsanti automaticamente previsti dal *sistema*, uno con la scritta "annulla" (chiude la finestra), uno con la scritta "Ok".

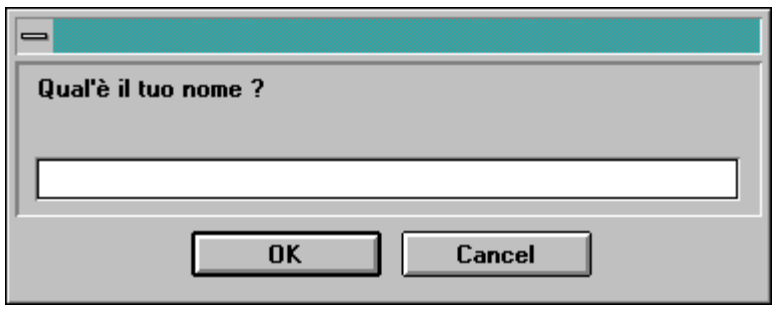

Figura 24: come si presenta una finestra di tipo "ask"

Il pulsante "Ok" chiude la finestra e "legge" ciò che l'utente ha eventualmente scritto nell'apposito spazio a sua disposizione. Legge e "memorizza". Dove ? Nella solita *variabile* **it**, naturalmente ! Una volta chiusa la finestra, quindi, dentro il "barattolo" rappresentato da **it** ci sarà quello che l'utente ha scritto, il suo nome. Il successivo comando provvederà a "scrivere" (*set text*) su un campo (*field*) che avremo creato allo scopo sullo sfondo delle pagine (*of this background*, perchè sia visibile in tutte) il nome dell'utente, ovvero il contenuto di *it* in quel momento. L'utente vedrà così il suo nome su tutte le pagine, e avrà la sensazione di un approccio più amichevole e meno "freddo" con l'ipertesto.

Strutture di controllo...

L'uso avanzato della *variabile* richiede, necessariamente, una certa pratica di "programmazione logica". Abbiamo visto che essa non è necessaria per costruire un ipertesto, ma solo per perfezionarlo e per renderlo più completo. Se siete arrivati fin qui, tuttavia, vuol dire che ormai avete preso confidenza con Toolbook, e non vi spaventa più l'idea di "sporcarvi le mani" con un linguaggio informatico. Comincerete dunque a ipotizzare molte "complicazioni" nel vostro ipertesto: ad esempio, vi verrà da pensare che, poichè è possibile leggere, scrivere e usare un dato *variabile*, allora dovrebbe essere possibile far sì che le reazioni di un *oggetto* ad un *evento* dipendano dalle variazioni di quel certo dato. Naturalmente, in Toolbook tutto questo è possibile (*OpenScript*, del resto, è a tutti gli effetti un completo linguaggio di programmazione, e anche piuttosto potente): è possibile attivando le cosidette **strutture di controllo**.

Le **strutture di controllo** sono delle istruzioni *Openscript* che consentono di modificare una sequenza di azioni in relazione a determinate condizioni o di eseguire più volte determinate operazioni, magari in dipendenza di particolari condizioni che possono essere a loro volta controllate. E' un concetto strettamente legato alla "programmazione logica", che ritroviamo peraltro in tutti i linguaggi informatici. Non è questa la sede per sviluppare il tema. Questa parte è solo uno sguardo generale alle potenzialità di Toolbook nella costruzione di ipertesti. Vogliamo però mettervi questa pulce nell'orecchio, suggerendovi di approfondire la questione se volete attribuire ai vostri pulsanti o alle vostre *hotword* un comportamento più "intelligente" di quanto non lo sia l'esecuzione di una semplice sequenza di azioni. Alle strutture di controllo dedicheremo un'apposita sezione di approfondimenti [**B3-B8**].

La gerarchia degli oggetti...

Ormai abbiamo dato uno sguardo a quasi tutti gli oggetti di Toolbook. Viene ora da chiedersi in che rapporto essi siano tra loro, se tra essi non esista ad esempio una precisa relazione gerarchica, dalla quale dipende il modo in cui un *evento* (un *ButtonClick*, ad esempio) viene notificato all'oggetto. Che cosa succede, ad esempio, se l'oggetto su cui l'utente effettua un *buttonClick* non prevede un *handler* per quel tipo di *evento* ? A prima vista (provate !), sembra che non succeda nulla. Ma siamo proprio sicuri ?

In effetti un preciso rapporto gerarchico lega tra loro gli *oggetti* che compongono un *book*, e da esso dipende quella che possiamo definire "modalità di propagazione dei messaggi". Che ci sia una gerarchia degli *oggetti*, è del resto abbastanza facilmente intuibile. Perchè possa esistere un *background*, ad esempio, ci *deve* essere un *book*. Perchè possa esistere una pagina ci *deve* essere un *background*. Perchè si possa costruire un pulsante, ci *deve* essere una pagina (o un *background*). Perchè si possa costruire una *hotword*, ci *deve* essere un campo di testo. Il libro, quindi, è l'oggetto gerarchicamente più alto, seguito dal *background*, dalla pagina e via via dagli altri oggetti.

Se questo è vero, allora in che modo si propagano i messaggi (gli *eventi*) ? In prima linea troveremo gli oggetti costruiti sulla pagina (che del resto coprono anche "fisicamente" sia la pagina, che gli oggetti posti sullo sfondo): campi, pulsanti, immagini, grafici, sono, lo sappiamo, oggetti sui quali l'utente può "puntare" direttamente, lanciando il relativo l'evento (ad esempio un *ButtonClick*). Tuttavia, se il *target* non prevede un gestore (*handler*) per l' *evento* generato, se cioè non esiste un *to handle buttonClick* nello *script* dell'oggetto il messaggio inviato attraverso il *clic* del mouse non si ferma. Esso viene infatti inviato sempre più in alto nella gerarchia che lega tra loro tutti gli oggetti, secondo una precisa sequenza che dipende proprio dal rapporto gerarchico: prima di tutto il *buttonClick* verrà inviato all'eventuale *gruppo* di cui l'oggetto "cliccato" potrebbe far parte. Se neanche nel gruppo trova un *to handle buttonClick* il viaggio continuerà verso l'alto. L' *evento* verrà prima notificato alla pagina. Poi al *background*. Poi al *book*. Se nemmeno il book prevede il relativo *handler*, il messaggio continuerà a viaggiare fino ad arrivare al cosidetto *System Book*, un "libro" speciale che un programmatore può costruire e definire a supporto della sua applicazione. Infine, se proprio non trova un *to handle buttonClick* il messaggio si perde senza produrre alcun risultato. Tutto questo, naturalmente, avviene alla velocità della luce ! Nella **figura 25** viene mostrato uno schema sintetico di questa gerarchia e illustrato il modo in cui un *evento* "viaggia" attraverso di essa.

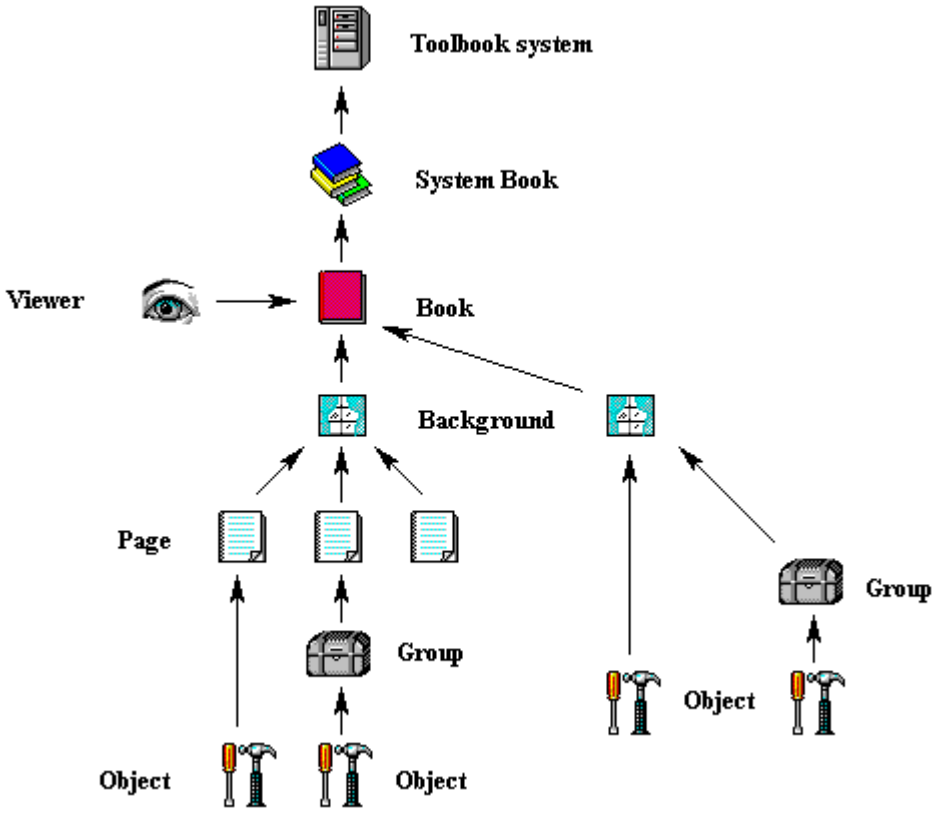

Figura 25: la struttura gerarchica delle realzioni tra gli oggetti di Toolbook: i messaggi seguono il percorso indicato dalle frecce fino a raggiungere il sistema

Messaggi di sistema, messaggi utente...

Ora, poichè il gestore (*handler*) di un *evento* viene "cercato" da Toolbook sull'intera struttura gerarchica di un'applicazione, ciò significa che esso non deve necessariamente essere associato al singolo *oggetto* su cui vogliamo che l' *evento* produca un effetto. Potremmo ad esempio portarlo ad un livello in cui possa essere opportunamente "intercettato". Con notevoli vantaggi. Ad esempio, inseriamo le istruzioni che seguono nello *script* di una pagina, la pagina che avevamo chiamato "Leonardo pittore":

```
to handle buttonDown
        hide field "notizie" of this page
end buttonDown
```

Sono le stesse istruzioni che in precedenza avevamo associato al campo "notizie". Solo che adesso, associate allo *script* della pagina, produrranno un effetto molto interessante: sarà infatti sufficiente *abbassare* il pulsante del mouse (*buttonDown*) in *qualsiasi* punto della pagina per ottenere che il campo "notizie", se in quel momento è visibile, venga nascosto. Attenzione ! In qualsiasi punto della pagina, *a meno che*, nel punto in cui il mouse viene cliccato non ci sia un *oggetto* che, nel suo *script*, prevede una serie di istruzioni ad un *evento* del tipo *to handle buttonDown* !

La struttura gerarchica di Toolbook può essere agevolmente e proficuamente "sfruttata" in sede di programmazione. Anzi, possiamo affermare che uno dei "segreti" della programmazione in Toolbook consiste proprio nella capacità di dosare attentamente la collocazione dei gestori degli eventi nella gerarchia. Non si deve mai dimenticare che, in quest'ottica, più *in alto* viene definito il gestore dell'evento, maggiore sarà il numero degli oggetti che potranno condividere la serie di istruzioni ad esso collegate. Più in basso il gestore dell'evento verrà definito, più limitato e "indirizzato" sarà l'effetto prodotto dall'attivazione di quell'evento. Ovviamente, non è particolarmente conveniente "spostare" su un piano gerarchicamente più alto la gestione di quegli eventi che l'utente genera più spesso, a meno che, come nel caso portato ad esempio, questo non comporti molti vantaggi e pochi rischi. Può però essere molto conveniente poter disporre, su un piano gerarchicamente elevato, di quegli *handlers* a cui abbiamo associato serie di istruzioni particolarmente complesse e che magari devono essere eseguite in relazione a molti oggetti. Vediamo come.

Finora abbiamo fatto riferimento, negli esempi allegati, ad *eventi* gestiti da Toolbook perchè legati a precise caratteristiche del *sistema*, i movimenti del mouse, il *clic*, l'apertura di un "libro" (*enterBook*). Questo tipo di *eventi* fa parte dei cosidetti "messaggi di sistema". Sono "messaggi di sistema" tutti gli eventi legati all'uso del mouse (*buttonClick*, *buttonDown*, *buttonUp*, *mouseEnter*, *mouseLeave*, *buttonStillDown* ecc.), tutti quelli legati alla gestione degli oggetti da parte di Toolbook (*enterBook*, *enterPage*, *enter**Field***, *enterBackground*, *enterButton*, *enterRecord**Field***, *enterSystem*, *leaveBook*, *leavePage*, *leave**Field*** ecc.), gli eventi legati all'uso della tastiera (*keyDown*, *keyChar*, *keyUp* ecc.) e perfino l' *assenza* di altri eventi (*idle*). Ognuno di questi eventi può prevedere un *handler* a cui è associata l'esecuzione di una serie di specifiche azioni.

Rientrano inoltre tra i messaggi di sistema tutte le voci del menu di Toolbook (*Copy*, *Paste*, *NewPage*, *Next*, *Previous* ecc.), nel senso che selezionare una di quelle voci significa, di fatto, attivare un handler ad esse corrispondente: selezionare la voce *Copy*, ad esempio, equivale a lanciare un *to handle Copy*, la sequenza di istruzioni collegata al quale è predefinita a livello di sistema.

La gamma degli eventi è dunque piuttosto ampia. Tuttavia, Toolbook offre all'utente stesso la possibilità di definire propri eventi, ovvero di impostare un numero *illimitato* di eventi da gestire

attraverso un eventuale *handler*. In pratica, noi possiamo tranquillamente associare una serie di istruzioni ad un gestore del tipo **to handle Saluta**, dove l' *evento* **Saluta** è stato determinato nel momento stesso in cui abbiamo scritto l' *handler*. Ovviamente, **Saluta** non è un "messaggio di sistema", non è previsto da Toolbook e non corrisponde a nessuna delle azioni che l'utente può compiere, nè usando il mouse, nè usando la tastiera. Perchè quell' *evento* possa verificarsi, quindi, esso dovrà essere "chiamato", comunque, attraverso un "messaggio di sistema", come un *clic* del mouse. Per "chiamarlo" Toolbook ci mette a disposizione il comando **send**, che significa "invia" (è sottinteso "un messaggio a..."), attraverso il quale, peraltro, possono essere "chiamati" anche i "messaggi di sistema". Vediamo un esempio:

```
to handle buttonClick
        send Saluta
end buttonClick
```

Ammettiamo che questo sia lo *script* di un pulsante. Che cosa succede quando si preme il pulsante ? Semplicemente, viene generato un "messaggio di sistema" (*ButtonClick*), che invia (*send*) una "chiamata" ad un evento che è stato chiamato **Saluta**. Un evento che viene detto "messaggio utente", poichè non è il sistema a definirlo. Ora, ricordate quanto abbiamo detto a proposito della gerarchia ? Bene. Dovete sapere che la "chiamata" all' *evento* **Saluta** procederà proprio lungo la catena gerarchica precedentemente descritta. Toolbook cercherà, cioè, un gestore dell' *evento* sul pulsante, poi sulla pagina, sul *background* e così via, fino a che non incontrerà un *oggetto* il cui script contenga un *to handle Saluta...* A quel punto le istruzioni collegate a quell'handler verranno eseguite. Il "messaggio utente", quindi, dovrà essere collocato su un *oggetto gerarchicamente superiore* a quello dal quale parte il "messaggio di sistema" che lo richiama. Dovrà inoltre essere "costruito" e definito. Seguendo l'esempio, potremmo aver inserito l' *handler* relativo al "messaggiio utente" nello *script* della pagina su cui si trova il pulsante:

```
to handle Saluta
        request "Ciao, come stai ?"
end Saluta
```

In risposta all' *evento* **Saluta** (richiamato attraverso il *buttonClick* generato sul pulsante della pagina) verrà mostrato nell'apposita finestra il saluto "Ciao, come stai ?". Certo, l'esempio portato non permette di capire i vantaggi che possono derivare dall'adozione di questi "meccanismi" di programmazione. Essi consistono prevalentemente in un risparmio di memoria e di tempo: se le istruzioni associate all' *evento* **Saluta** fossero non una semplice riga ma un complesso *programma*, infatti, e se quello stesso *programma* dovesse essere lanciato da molti pulsanti, risulterebbe particolarmente conveniente spostare l' *handler* sullo *script* della pagina o su quello del *book*. Sarebbe sufficiente un solo comando per richiamarlo, ed eventuali modifiche (normale amministrazione, quando un *programma* diventa complesso) non dovrebbero essere apportate su tutti i relativi pulsanti, ma soltanto sull' *handler* che avevamo opportunamente collocato più in alto. Meditate, gente, meditate !

Estensioni multimediali...

E la multimedialità ? Di cui tanto si parla, spesso senza nemmeno sapere di che cosa si tratta esattamente ? Il nostro ipertesto (ma ormai si può considerare vostro...) utilizza sia testi che immagini statiche. Per farlo diventare un *Ipermedia*, basterebbe estendere la gamma delle sue modalità di comunicazione delle informazioni all'audio e alle immagini in movimento. Si può fare, in Toolbook

3.0 , ma non è particolarmente semplice. Se quindi si pensa che sia necessario comunicare alcune informazioni anche attraverso l'audio o le immagini in movimento, sarà bene imparare ad utilizzare **Asymetrix Multimedia Toolbook 3.0**, che è una versione del programma specificamente orientata alla gestione della multimedialità [**C11-C18**]. Il linguaggio *OpenScript* che questa versione supporta è lo sempre lo stesso e tutto quello di cui abbiamo parlato finora poteva tranquillamente essere realizzato, nello stesso modo, con la versione **Multimedia**. Cambia soltanto il modo in cui vengono gestite le cosidette *estensioni* multimediali. La versione **Multimedia** del programma mette infatti a disposizione degli utilizzatori e dei programmatori ulteriori oggetti ed istruzioni in *Openscript* per la definizione ed il controllo di precise funzionalità multimediali, come l'esecuzione di un brano musicale digitalizzato (in formato **WAV**), l'esecuzione di un brano direttamente da un **CD** musicale, l'esecuzione di un brano in formato **MIDI** (uno standard molto noto ai musicisti soprattutto di tastiere e sintetizzatori elettronici), la visualizzazione di un filmato preventivamente digitalizzato (in formato **AVI**, **MPEG**, **Quicktime**, ecc.) o l'attivazione di una animazione realizzata interamente con il computer, ecc [**C11-C18**]. Insomma, tutto ciò che può rendere ancora più accattivante e coinvolgente l'utilizzo di un ipertesto.

In realtà, finchè si tratta di emettere dei semplici suoni registrati (in formato **WAV**), della durata massima di 10 secondi circa, Toolbook 3.0 è ancora sufficiente. Provate infatti a inserire questo *script* in un pulsante:

```
to handle buttonClick
        get playSound("c:\windows\tada.wav")
end buttonClick
```

Lo *script* lancia ed esegue un file audio ("tada.wav", che in genere si trova nella directory di windows, un piccolo suono caratteristico che evoca un'entrata in scena, o qualcosa del genere), utilizzando (ovvero richiamando attraverso un comando *get*) una *funzione* specifica, **playSound("..")**, che è disponibile anche nella versione non multimediale del programma. [**C1**]. Sempre nella versione non multimediale di Toolbook, è inoltre possibile inserire estensioni multimediali ricorrendo a specifiche "librerie" di funzioni, alle quali è dedicata un'apposita sezione di approfondimento [**C2-C10**].

La funzione **playSound("..")** è sufficiente per eseguire un brano sonoro delle dimensioni massime di 100 Kb (circa 10 secondi). Per brani più lunghi, o per filmati e animazioni complesse, **Multimedia Toolbook 3.0** diventa indispensabile. Nella versione multimediale del programma è infatti disponibile un oggetto nuovo: lo **stage**. E' una sorta di palcoscenico che è possibile "montare" sulla pagina e su cui possono essere eseguiti, ad esempio, dei video. Così come i campi servono per contenere dei testi, lo **stage** serve a contenere dei filmati. Nel box dei strumenti della versione **Multimedia** potete facilmente individuare il pulsante che consente di creare uno **stage.**

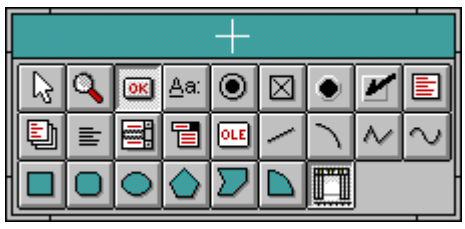

Figura 26: il box degli strumenti con, in evidenza, l'icona per creare uno "stage"

Lo **stage** si costruisce allo stesso modo dei campi e dei pulsanti e anche per esso si possono definire delle proprietà, come la posizione e la dimensione. Richiamando la finestra **Stage Properties**, inoltre, sarà possibile assegnare un nome all' *oggetto*, ad esempio "schermo".

A questo punto, per poter utilizzare un filmato all'interno dello **stage** è necessario importarlo come **clip**, in modo che Multimedia Toolbook possa riconoscerlo come sua *risorsa*. In pratica, tutte le fonti multimediali (suoni, filmati, animazioni, ecc.), prima di poter essere utilizzate, devono essere *definite* come **clip** del nostro *book*. Nel menu **Object** selezionate la voce **Clips**.

Figura 27: la finestra di dialogo per creare e definire un clip

Cliccate poi sul pulsante **New** e quindi selezionate il **file** corrispondente al filmato che volete importare (ad esempio, un filmato che si chiama "samples.avi", uno di quelli che di solito viene dato in omaggio quando si acquistano una *scheda* video e un programma *Video for Windows*). Al clip appena importato assegnate ora un nome, in questo caso "esempio". Ora è un *oggetto* come gli altri, e in quanto tale può essere facilmente gestito. Sulla pagina dove avete costruito lo **stage**, non rimane che creare un pulsante con uno *script* come quello che segue:

```
to handle buttonClick
        mmPlay clip "esempio" in stage "schermo"
end buttonClick
```

Il significato dello *script* ci sembra tanto chiaro da non aver bisogno di ulteriori spiegazioni. Per ulteriori informazioni sulla gestione degli oggeti multimediali rimandiamo comunque ad un'apposita sezione degli approfondimenti [**C11-C18**].

APPROFONDIMENTI

Una domanda che l'utilizzatore di Toolbook si pone relativamente spesso consiste nel cercare di capire se esista o meno una soluzione "ottimale" per la risoluzione di determinati problemi che possono presentarsi durante l'assemblaggio di un ipermedia. In realtà *OpenScript* è un linguaggio piuttosto "aperto", e si può affermare che non esiste mai una sola soluzione ai problemi dati. Ci sono comandi che attivano determinate operazioni, funzioni che calcolano dei valori, variabili che possono essere utilizzate in un certo modo. Ma le strade per ottenere lo stesso risultato (un particolare "effetto" che vorremmo inserire in un ipertesto, ad esempio) possono essere molteplici. In questa sezione del volume cercheremo di descrivere in modo più approfondito ulteriori funzioni e comandi, di chiarire meglio concetti legati all'uso degli strumenti di Toolbook e di affrontare una serie di questioni relative alla programmazione degli oggetti e alla gestione degli eventi. Gli approfondimenti verranno supportati da un certo numero di *script* esemplificativi, che ognuno potrà provare a realizzare. Useremo, ovviamente, un linguaggio più tecnico di quello che abbiamo usato finora, ben sapendo che a questo livello ci rivolgiamo ormai, se non a programmatori con una certa esperienza, quanto meno a utilizzatori smaliziati. Nell'insieme, vorremmo offrirvi, più che un completo manuale di Toolbook, delle idee che potrebbero rivelarsi utili durante la realizzazione pratica dei vostri progetti ipertestuali e multimediali. Ricordiamo che l' *help* di Toolbook spiega, anche se in lingua inglese, il funzionamento ed il significato di tutti i comandi *OpenScript*.Se state lavorando ad uno *script* premete il tasto **F1** per richiamare l' *help*, o, meglio ancora, selezionate un particolare comando che avete scritto: premendo **F1** Toolbook richiamerà direttamente la relativa scheda.

APPROFONDIMENTI DI I LIVELLO

Libri, Pagine, Oggetti, Testi

La finestra *command*

A1. Per lanciare direttamente i comandi OpenScript

I comandi *Openscript* possono essere eseguiti sia creando un pulsante che li contenga (e associando l'esecuzione ad un *evento*) che attraverso una particolare finestra, detta *command Window*. All' editor di script di ogni oggetto si accede tramite il pulsante *script* del dialog box delle proprietà dello stesso. Alla finestra *command* si accede con:

View --> Command (Shift+F3)

(dal menu "**View**" scegliere la voce "**Command**" o, più rapidamente, premere il tasto <**Shift**> e il tasto <**F3**>)

Tutti i comandi *OpenScript* possono essere scritti su questa finestra ed eseguiti direttamente premendo il tasto <**Invio**>.

Il book

A2. Come creare un nuovo libro

Per creare un nuovo libro scegliere Nuovo (**New**) dal menu **File**. Se è già aperto un libro Toolbook lo chiude e ne apre uno nuovo con una prima pagina vuota. La prima cosa da fare è dare un nome al nuovo libro scegliendo Salva col nome ... (**Save As** ...) dal menu **File** (dare un nome DOS valido).

A3. Come salvare i cambiamenti di un libro

Se state lavorando su un libro e avete apportato delle modifiche sarà bene eseguire regolarmente il salvataggio perché i cambiamenti non vadano perduti. Per farlo scegliete il comando **Save** del menu **File**, se volete salvare il libro con il suo stesso nome. Scegliete invece il comando **Save As** ... del menu **File** se volete salvare il libro con un nome diverso (un nome DOS valido: non più di otto caratteri e senza caratteri particolari). Questa seconda opzione è conveniente, perchè salvando il libro con un nome diverso esso verrà "compattato" in modo da occupare minor spazio su disco.

A4. Come chiudere un libro

Per chiudere un libro scegliere il comando Esci (**Exit**) del menu **File**. Vi sarà chiesto di salvare o non salvare le modifiche: cliccate su NO se non volete sovrascrivere il file originario con le modifiche fatte, scegliete YES in caso contrario. Si può settare la proprietà **saveOnClose** in modo da controllare il modo con cui si esce dal book:

saveOnClose of this book = Yes
(salva automaticamente prima di uscire)

saveOnClose of this book = No
(non salva quando si esce)

saveOnClose of this book = ask
(salva a seconda della risposta data)

saveOnClose of this book = system
(è usata la proprietà di sistema sysChangesDB)

sysChangesDB = true
(prima di uscire mostra un dialog-box chiedendo se si vuol salvare o no)

sysChangesDB = false
(prima di uscire non mostra un dialog-box chiedendo se si vuol salvare o no)

Pagine e cambi di pagina

A5. Per stabilire la dimensione (Page size) delle pagine del book e del background

Le pagine di un book possono assumere la dimensione massima di 196 pollici quadrati (1263,8 centimetri quadrati) ovvero un quadrato di 14 per 14 pollici di lato (circa 35,6 cm per 35,6 cm)
In assenza di precise indicazioni (cioé per default) Toolbook stabilisce la pagina di 6 pollici per 4 pollici (15,2 cm per 10,2 cm).

Una pagina è costituita dal **foreground** (primo piano) e dal **background** (sfondo) che hanno sempre la stessa dimensione. Tuttavia un libro può avere più e diversi *background*. La dimensione di un *background* (**size**) può cambiare, e di conseguenza cambieranno le dimensioni delle rispettive pagine. La dimensione (**size**) può quindi essere stabilita per ogni *background* per cui il libro apparirà con pagine di dimensione diversa ogni volta che cambia il *background*. La dimensione di default delle pagine del *book* si definisce con:

Object --> Book proprierties (Shift+F8) --> Page size

Per definire invece la dimensione delle pagine di un certo background:

Object --> Background proprierties (Ctrl+F7) --> Page size

Attenzione ! Più piccole sono le pagine e maggiore è la velocità con cui si può accedere ad esse.

Per effettuare operazioni sulle dimensioni delle pagine in *Openscript*:

size of this book = 10000,15000
(definisce la dimensione della pagine del book a 10000 punti in larghezza e 15000 in altezza)

size of this background = 10000,15000
(definisce la dimensione della pagine del background a 10000 punti in larghezza e 15000 in altezza).

L'unità di misura che Toolbook usa per calcolare le dimensioni delle pagine e degli oggetti in generale si chiama *PageUnits*. 1 centimetro corrisponde a circa 568 *PageUnits*, 1 pollice a 1440.

Per sapere qual'è la dimensione di una pagina in *pageUnits* in modo da coprire uno schermo VGA da 640x480 pixels:

get clientToPageUnits("640,480")

oppure:

request clientToPageUnits("640,480")

Per settare la dimensione della pagina per valori definiti in pixels 640x480 VGA:

size of this book = clientToPageUnits("640,480")
-- setta la dimensione della pagina alla dimensione del VGA standard 640x480

Per conoscere la dimensione delle pagine di un libro:

get size of this book
request it
-- ottiene la dimensione delle pagine del book e pone il risultato nella varibile it, quindi mostra il risultato. Queste due linee, per poter essere eseguite, devono essere inserite nello script di un pulsante o nella finestra *command*, ma in quest'ultimo caso devono essere separate dal punto e virgola, poichè nella finestra *command* non si può "inviare" a capo.

Per conoscere la dimensione delle pagine di uno sfondo:

request size of this background
-- mostra direttamente la dimensione delle pagine del background.

Per portare una pagina alla dimensione massima per essa stabilita usare:

Page --> Size to page (F11)

In *OpenScript*:

send sizeToPage

A6. Per aggiungere o copiare una nuova pagina

Facendo clic su **New Page** o **Ctrl+N** (New Page) del menu **Object**: una nuova pagina sarà aggiunta immediatamente dopo la pagina attiva e diviene la pagina attiva; essa avrà lo stesso Sfondo (**background**) della pagina precedente. Una pagina può anche essere copiata in una diversa posizione del libro o in un altro libro. Per fare questo selezionare la pagina da copiare facendo Clic sullo **status box** (è la barra grigia in basso) in corrispondenza dell'icona pagina (sulla destra), poi scegliere il menu Modifica (**Edit**) e di esso la voce Copia (**Copy**) o premere **Ctrl**+**Ins** : la pagina sarà così copiata negli appunti (*clipboard*). Per incollarla in altro punto del libro (od altro libro) posizionarsi sulla pagina immediatamente precedente alla posizione in cui vogliamo inserire la pagina, scegliere dal menu Modifica (**Edit**) la voce Incolla (**Paste**) o premere **Shift**+**Ins**: la pagina desiderata sarà richiamata dagli appunti, incollata dopo la pagina attiva e mostrata.

In *OpenScript*:

send newpage
-- crea una nuova pagina.

send newbackground
-- crea un nuovo sfondo.

import book "c:\jacopo\prova.tbk"
-- copia nel book corrente tutte le pagine del book indicato e relativi sfondi.

import pages 2 to 4 of book "c:\f\titolo.tbk"
-- copia nel book corrente solo le pagine da 2 a 4 del book indicato e relativi sfondi.

import pages 2 to 4 of book "c:\f\titolo.tbk" without background
-- copia nel book corrente solo le pagine da 2 a 4 del book indicato, escludendo però i relativi sfondi.

A7. **Per dare un nuovo numero d'ordine ad una pagina**

Per cambiare il numero d'ordine di una pagina posizionarsi sulla pagina a cui si vuol cambiare il numero, aprire il dialog-box delle proprietà della pagina e inserire nell'apposito spazio un nuovo numero d'ordine. La pagina verrà rinumerata e tutte le altre verranno "scalate" di conseguenza.

In *OpenScript*:

set pageNumber of this page to 3
-- cambia il numero d'ordine della pagina attuale portandolo a 3.

set pageNumber of page 2 to 3
-- cambia il numero d'ordine della pagina 2, che diventa pagina 3.

A8. Per selezionare e cancellare una pagina

Una pagina per poter essere cancellata deve prima essere selezionata. Si può fare in diversi modi:

- facendo click sull'icona-pagina dello status-box

(premere **F12** per far comparire lo status box se in quel momento non risulta visibile)
- scegliendo **select Page** dal menu **Edit**
- premendo **Shift**+**F12**.

Quando la pagina è selezionata e l'icona-pagina appare evidenziata in neretto, per cancellarla scegliere l'opzione Taglia (**Cut**) del menu Modifica (**Edit**): la pagina sarà cancellata e posta negli appunti di Windows, da dove sarà poi possibile incollarla in un altro punto del libro. Se invece si usa il comando Cancella (**Clear**) del menu Modifica (**Edit**) (o si preme il tasto **Canc** o **Canc**+**Shift**) la pagina sarà cancellata senza essere copiata negli appunti.

In *OpenScript*:

```
select this page
send clear
-- seleziona la pagina attuale e quindi la cancella.

select page 3
send clear
-- seleziona la pagina 3 e quindi la cancella.
```

A9. Per "navigare" da una pagina all'altra

I comandi *OpenScript* di navigazione tra le pagine sono:

```
go to next page          -- va alla pagina successiva
go to previous page      -- va alla pagina precedente
go to first page         -- va direttamente alla prima pagina del book
go to last page          -- va direttamente all'ultima pagina del book
send back                -- ripropone l'ultima pagina vista lungo il percorso ipertestuale
go to page "mia"   -- va direttamente alla pagina chiamata "mia"
go to page id 3          -- va direttamente alla pagina con ID 3
```

Il passaggio da una pagina all'altra può essere accompagnato da un "effetto speciale" usando invece che *go* i seguenti comandi:

```
fxDissolve fast to next page        -- va alla pagina successiva con effetto "dissolvenza"
fxZoom fast to next page    -- va alla pagina successiva con effetto "zoomata"
fxWipe fast to next page            -- va alla pagina successiva con effetto "tendina"
```

Invece che **fast** si possono usare anche **slow** o **normal**, in modo da indicare anche con quale velocità deve essere effettuato il cambio.

L'effetto speciale può essere applicato anche senza effettuare nessuno spostamento, rimanendo cioé sulla pagina attiva ma attribuire un colore finale all'effetto. Ad esempio:

```
fxDissolve fast to yellow
-- effettua una dissolvenza fino al colore "giallo".
```

Naturalmente, anzichè *yellow* si potrebbe scrivere **white** o **black**.

E' possibile inoltre usare un comando che regola particolari effetti di "transizione". Il comando è **transition**, e permette, associato a vari parametri, numerosi altri effetti speciali, qui esemplificati:

```
transition "split out horizontal fast" to previous page   -- effetto "sipario che si apre"
transition "split in horizontal fast" to previous page    -- effetto "sipario che si chiude"
transition "blinds  slow" to previous page                -- effetto "tenda alla veneziana"
transition "dissolve  fast" to next  page                       -- effetto "mosaico"
transition "drip  fast" to previous page                        -- effetto "goccia"
transition "fade  fast" to previous page                        -- effetto "dissolvenza incrociata"
transition "iris  fast" to previous page                        -- effetto "otturatore"
transition "push left fast" to previous page              -- effetto "spinta verso sinistra"
transition "push right fast" to previous page             -- effetto "spinta verso destra"
transition "push top fast" to previous page               -- effetto "spinta verso l'alto"
```

transition "push bottom fast" to previous page -- effetto "spinta verso il basso"
transition "rain" to previous page -- effetto "pioggia"
transition "puzzle" to previous page -- effetto "puzzle"
transition "slide in left fast" to previous page -- effetto "diapositiva"
transition "slide in right fast" to previous page -- effetto "diapositiva"
transition "slide out top fast" to previous page -- effetto "diapositiva"
transition "slide out bottom fast" to previous page -- effetto "diapositiva"
transition "spiral out speed 3" to previous page -- effetto "spirale"
transition "spiral in fast" to previous page -- effetto "spirale"
transition "tear fast" to previous page -- effetto "pennellata"
transition "turnPage left fast" to previous page -- effetto "sfoglia pagina verso sinistra"
transition "turnPage right fast" to previous page -- effetto "sfoglia pagina verso destra"
transition "wipe left fast" to previous page -- effetto "tendina"
transition "zoom in left fast" to previous page -- effetto "zoom"
transition "zoom out lowerleft fast" to previous page -- effetto "zoom"

L'effetto di transition può essere usato per cancellare la pagina con un colore:

transition "puzzle" to red -- effetto "puzzle" a fondo rosso

A10. Per "bloccare" l'accesso a una pagina

Se si desidera impedire all'utente di accedere ad una pagina, ad esempio usando i menu, settare la proprietà **skipNavigation** tramite il box di dialogo delle proprietà della pagina.

In *OpenScript*:

set skipNavigation of page id 23 to true

In questo modo la pagina ID 23 non sarà mai visitabile dall'utente, rimanendo comunque accessibile all'autore.

A proposito di *oggetti*

A11. Identificazione degli oggetti

Quando un oggetto viene creato ad esso è assegnato un esclusivo numero di riconoscimento chiamato ID. Questo numero serve a Toolbook per identificare gli oggetti. Toolbook non assegna mai a due oggetti dello stesso tipo lo stesso numero ID. Per "identificare" un oggetto (far riferimento ad esso in uno *script*) è possibile utilizzare questo numero.

Tuttavia, poichè sarebbe difficile ricordarsi il numero attribuito da Toolbook ad ogni oggetto, è sempre opportuno dare un nome all'oggetto, un nome che possa essere usato per identificarlo in uno *script*. Si può dare un nome ad un oggetto riempendo l'apposito spazio nel box degli attributi dello stesso. Il nome non può essere più lungo di 32 caratteri e non si deve dare uno stesso nome ad oggetti identici sulla stessa pagina (in generale, è bene non usare lo stesso nome per più oggetti). Il nome dell'oggetto, in quanto proprietà dello stesso, può essere anche definito in *Openscript*:

set name of field id 5 to "note"
-- il campo a cui è stato assegnato il numero ID 5 viene chiamato "note".

A12. Spostare un oggetto da uno strato (layer) all'altro

Ogni oggetto occupa "fisicamente" uno strato (**layer**). Creando un nuovo oggetto, questi verrà posto automaticamente su uno strato più superficiale rispetto a tutti gli altri oggetti già presenti sulla pagina o sullo sfondo. Lo strato a cui appartiene un oggetto può essere modificato semplicemente scrivendo un altro numero nello spazio relativo del box degli attributi di un oggetto. Quando si fa questo Toolbook sposta automaticamente gli altri oggetti su altri strati, poichè non possono esserci due oggetti sullo stesso strato. Un oggetto posto sullo strato numero 1 è quello più "lontano". Il numero di tutti gli strati occupati è indicato nel box degli attributi della pagina o dello sfondo. Si può spostare un oggetto da uno strato all'altro anche usando i comandi del menu **Draw** : Avvicina

(**Bring Closer**) porta l'oggetto sullo strato immediatamente più vicino al primo piano, Allontana (**Send Farther**) porta l'oggetto sullo strato immediatamente più lontano verso lo sfondo, Porta sopra (**Bring To Front**) porta l'oggetto sull'ultimo strato, il più lontano dallo sfondo, Porta sotto (**Send To Back**) porta l'oggetto sullo strato più lontano, il più vicino allo Sfondo.

In *Openscript*:

set layer of button "av" to 3
-- porta il pulsante "av" sul terzo strato a partire dal fondo

A13. Come creare un nuovo oggetto

Per creare un oggetto si deve essere in modo Autore (**Author**). Si avrà a disposizione un box di strumenti (che normalmente si trova sulla sinistra in alto della pagina, ma che può anche essere spostato, cliccando sulla barra scura di esso e trascinando), con cui si possono creare o modificare gli oggetti. La parte superiore di tale box indica sempre il tipo di oggetto selezionato. Per creare un nuovo oggetto si deve scegliere dal box degli strumenti: fare click sul pulsante con l'icona corrispondente all'oggetto (pulsante, campo, od altro ...) che si vuole creare. Dopo questo clic nell'indicatore di selezione appare una croce, l'icona dell'oggetto nel box diviene evidenziata e il cursore assume la forma di una croce: siamo pronti per creare l'oggetto. Posizionare il cursore (la croce) sul punto dello schermo dove si vuole che inizi l'oggetto. Fare click e con il pulsante del mouse premuto spostare il cursore (la croce) fino al punto dello schermo dove si vuole che termini il secondo angolo dell'oggetto, lasciare il pulsante quando l'oggetto avrà raggiunto la dimensione desiderata.

In *OpenScript*:

draw button from 100,250 to 350, 550
-- disegna direttamente un pulsante partendo dalle coordinate 100, 250 (espresse in *PageUnits*) e "trascinandolo" fino alle coordinate 350, 550.

A14. Come selezionare un oggetto

Per poter modificare un oggetto bisogna prima selezionarlo. Per selezionare un oggetto deve essere attivata la *freccia* (strumento per selezionare) nel box degli strumenti. Fare quindi click sulla freccia, che sarà evidenziata ed apparirà anche nell'Indicatore di Selezione. Il cursore assumerà la forma di una freccia. A questo punto per selezionare un oggetto è sufficiente fare click su di esso. Per sapere se un oggetto è selezionato controllare se ai suoi angoli appaiono dei piccoli rettangoli e se l'icona corrispondente al tipo di oggetto appare nell'Indicatore di Selezione. Per annullare la selezione è sufficiente fare click in qualunque altro punto dello schermo fuori dell'oggetto selezionato.
Gli oggetti possono essere selezionati uno alla volta, ma se si tiene premuto il tasto *Shift* si possono selezionare più oggetti consecutivamente ed insieme, cliccando ogni volta su quelli che vogliamo selezionare. Per annullare la selezione di un solo oggetto è sufficiente fare di nuovo click su di esso tenendo Shift premuto. Per selezionare più oggetti contemporaneamente) selezionare la freccia dal box degli strumenti, fare click in un punto dello schermo che al di fuori di tutti gli oggetti da selezionare e, tenendo premuto il pulsante, trascinare il mouse; si formerà un rettangolo tratteggiato, che può essere ingrandito per effetto del trascinamento; quando il rettangolo tratteggiato ingloba tutti gli oggetti che si vogliono selezionare rilasciare il pulsante del mouse. Tutti gli oggetti contenuti nel rettangolo tratteggiato appariranno come selezionati.

Per selezionare tutti gli oggetti presenti su una pagina premere **Shift**+**F9** o scegliere Seleziona tutto (**Select All**) dal menu Modifica (**Edit**). Non è possibile selezionare contemporaneamente oggetti che si trovano sul Primo piano e sullo Sfondo: o si selezionano gli oggetti del Primo piano o gli oggetti dello Sfondo.

In *OpenScript*:

send selectAll
-- seleziona tutti gli oggetti della pagina.

select button "prova"
-- seleziona un pulsante chiamato "prova" che si trova sulla pagina corrente.

selection = button "prova"
-- seleziona un pulsante chiamato "prova" che si trova sulla pagina corrente.

selection = button "prova", field "campo"
-- seleziona contemporaneamente il pulsante di nome "prova" e il campo di nome "campo".

selection = null
-- de-seleziona tutti gli oggetti selezionati in quel momento.

unselect selection
-- de-seleziona tutti gli oggetti selezionati in quel momento.

A15. Come spostare un oggetto sullo schermo

La posizione di un oggetto sullo schermo può essere cambiata selezionando l'oggetto, facendo click su di esso e, tenendo premuto il pulsante sinistro del mouse, trascinando l'oggetto fino al punto desiderato: quando il pulsante del mouse viene rilasciato l'oggetto assume sullo schermo la nuova posizione e rimane selezionato.

In *OpenScript*:

move button "mio" by -1440,1440
-- sposta il pulsante "mio" di 1440 punti a sinistra e 1440 punti in basso rispetto alla posizione attuale.

move button "mio" to 2000,3000
-- il pulsante "mio" viene spostato esattamente sulla posizione corrispondente alle coordinate 2000,3000.

Per "trascinare" un oggetto a livello "lettore", inserire nel suo *script* questo *handler*:

to handle buttonStillDown
set my position to sysMouseposition
end
-- finchè il mouse viene tenuto premuto l'oggetto viene spostato alle coordinate corrispondenti alla posizione del cursore.

A16. Come modificare la dimensione di un oggetto

Per modificare la dimensione di un oggetto (lo spazio che occupa sullo schermo) bisogna prima selezionare l'oggetto, fare click su uno dei quadratini che lo contornano e, tenendo premuto il pulsante sinistro del mouse, trascinare. L'oggetto cambia di dimensione secondo l'angolo che si è scelto e la direzione in cui si trascina il mouse.

In *OpenScript*:

set bounds of field "prova" to 450,500, 1350,2000
-- i vertici estremi dell'oggetto (*bounds*) vengono modificati: il vertice posto in alto a sinistra viene portato alle coordinate 450,500, quello in basso a destra alle coordinate 1350,2000. L'oggetto assume la nuova dimensione.

A17. Come copiare gli oggetti

Dei nuovi oggetti possono essere creati come copia di oggetti già esistenti: i nuovi oggetti copiati conservano tutte le proprietà e lo *script* degli oggetti originali.

Per copiare un oggetto: selezionare l'oggetto; scegliere Copia (**Copy**) dal menu Modifica (**Edit**) o premere **Ctrl**+**Ins**. Una copia dell'oggetto è posta negli appunti (*clipboard*). Posizionarsi sulla pagina su cui si vuol ricopiare l'oggetto (o sullo sfondo): scegliere Incolla (**Paste**) dal menu Modifica (**Edit**) o premere **Shift**+**Ins**. L'oggetto sarà richiamato dagli appunti e posto sullo stesso punto dello schermo che occupava prima di essere copiato. Spostarlo quindi nel punto voluto.

In *OpenScript*:

set focus to null -- elimina eventuali conflitti con oggetti selezionati
select button "avanti" of this page -- seleziona l'oggetto
send copy -- copia l'oggetto nella clipboard
send paste -- "incolla" l'oggetto copiato sulla pagina

Per "duplicare" direttamente un oggetto selezionarlo e scegliere Duplica (**Duplicate**) dal menu Modifica (**Edit**) o premere **Ctrl**+**D**. Nel box delle funzioni c'è anche un pulsante che effettua la setssa operazione.

A18. Come cancellare gli oggetti

Per cancellare un oggetto (o più oggetti insieme): selezionare l'oggetto (o gli oggetti), premere **Canc** o scegliere Cancella (**Delete**) dal Menu Modifica (**Edit**). Per annullare l'operazione immediatamente dopo la cancellazione (e non più tardi) premere **Alt**+**Backspace** o scegliere Annulla (**Undo**) dal menu Modifica (**Edit**).

In *OpenScript*:

```
set focus to null                          -- elimina eventuali conflitti con oggetti selezionati
select button "avanti" of this page  -- seleziona l'oggetto
send clear                                 -- cancella l'oggetto
```

oppure:

```
send cut                                   -- cancella l'oggetto dopo averlo copiato nella clipboard
```

Per cancellare tutti gli oggetti del background costruire un pulsante con questo *script*:

```
to handle buttonClick
  send background                  -- si sposta sul background
  select all                       -- seleziona tutti gli oggetti dello sfondo...
  send clear                       -- ... e li cancella
  send foreground            -- torna al livello della pagina
end
```

A19. Come raggruppare gli oggetti in un unico altro oggetto

Più oggetti possono essere raggruppati in un unico altro oggetto. Questo nuovo oggetto creato può essere trattato come un qualsiasi altro oggetto. Un oggetto che deriva dal raggruppamento di altri oggetti è un nuovo tipo di oggetto di nome Gruppo (**Group**). Caratteristica essenziale di un Gruppo è che può essere munito di un suo *script* conservando tuttavia, per ogni oggetto che lo compone il relativo *script* di cui esso è eventualmente dotato. Lo *script* del Gruppo può essere attivato in modo separato dallo script di ogni oggetto, nel senso che si può attivare lo *script* del Gruppo così come si possono attivare gli *script* di ogni singolo oggetto.

Per creare un Gruppo: selezionare tutti gli oggetti da raggruppare; scegliere Raggruppa (**Group**) dal menu Oggetto (**Object**) oppure premere **Ctrl**+**G**. Per separare gli oggetti di un gruppo selezionare il gruppo e poi scegliere Separa (**Ungroup**) dal menu **Object** o premere **CTRL+G**. Separando gli oggetti di un gruppo l'eventuale *script* del gruppo viene perduto.

In *OpenScript*:

```
to handle buttonClick
  select button "avanti", field "note"   -- seleziona il pulsante "avanti" e il campo "note"
  send group                                 -- i due oggetti vengono raggruppati
  set name of selection to "gruppomio"       -- il gruppo appena creato viene chiamato "gruppomio"
end
```

Ed ecco come il gruppo può essere separato, sempre attraverso uno *script* da associare ad un pulsante:

```
 to handle buttonClick
  select group "gruppomio"
  send ungroup
 end
```

Per selezionare un singolo oggetto che appartiene ad un gruppo fare doppio clic (in modalità *autore*) su di esso. Per aggiungere un oggetto ad un gruppo copiare nella *clipboard* l'oggetto da aggiungere (**Copy**), fare doppio clic su un oggetto del gruppo e poi scegliere **Paste** (o premere **Shift+Ins**). Per cancellare un oggetto da un gruppo fare doppio clic sull' oggetto desiderato e poi scegliere **Clear** (o premere **Canc**).

A20. Come richiamare le Proprietà o Attributi (Properties) di un oggetto

In Toolbook ogni oggetto ha delle proprietà o attributi (**properties**) che gli derivano dall'appartenenza ad una categoria di oggetti specifica. Il riepilogo di queste proprietà può essere richiamato tenendo premuto il tasto **Shift** mentre si fa **doppio clic** sull'oggetto stesso oppure scegliendo il comando ... Attributi... (**Properties**) dal menu Oggetto (**Object**): si aprirà un box di dialogo relativo alle proprietà dell'oggetto (diverso, a seconda del tipo di oggetto), che può essere usato per "settare" o modificare le proprietà stesse.

A21. Come modificare le Proprietà o Attributi (Properties) di un oggetto

Una volta mostrato il box degli attributi di un oggetto è sufficiente posizionarsi nello spazio relativo del box per poterli modificare, con **OK** si confermano le modifiche con Annulla (**Cancel**) si esce senza effettuare modifiche.

L'intero riepilogo delle proprietà di un oggetto può essere richiamato facendo clic sull'oggetto con il pulsante **destro** del mouse. Il riepilogo sarà in questo caso in forma di menu a tendina. Il menu di riepilogo conterrà inoltre una pulsantiera attraverso la quale si potrà accedere velocemente al normale box di dialogo delle proprietà dell'oggetto, all'editor dello script o (penultimo pulsante, con un'etichetta sovrapposta ad una finestra come icona), ad una tabella che mostra il riepilogo complessivo di tutte le proprietà dell'oggetto, con relativi valori. Su questa tabella l'utente può modificare tutte le proprietà e, con relativa facilità, definire e "costruire" ulteriori proprietà dell'oggetto, dette **User Properties**, a cui dare un valore.

A22. Come modificare gli script di un oggetto senza aprire l'editor di script

Lo *script* di un oggetto può essere modificato o letto tramite l'editor di script, a cui si accede per mezzo del menu Oggetto... Attributi oggetto (**Object** ...**Properties**). Può però essere "aperto" anche facendo **doppio clic** sull'oggetto stesso tenendo premuto il tasto **Ctrl**. Un altro modo per *editare* lo *script* di un oggetto è quello di usare la finestra dei comandi (**commandWindow**). Per mostrare la finestra dei comandi premere **Shift+F3**. In questo caso, per aprire uno script scrivere i comandi relativi dentro tale finestra e premere <**invio>**. Ad esempio, se si vuol vedere lo script della pagina attiva scrivere:

edit script of this page

Analogamente:

edit script of this book
edit script of selection
edit script of button "Mostra" of this page
select button "Mostra" of this page; edit script of selection

Si può anche richiamare lo script di un book non aperto indicando correttamente il nome del file:

edit script of book "c:\toolbook\animate.tbk"

Si può infine aprire uno script e/o modificarlo direttamente da un altro *script*, ad esempio dallo script di un pulsante:

```
to handle buttonClick
        set script of this page to null
end
-- apre e cancella il contenuto dello script della pagina corrente.
```

L'apertura e la modifica degli *script* degli oggetti non è però ammessa quando si è a livello "lettore", nè quando l'applicazione viene lanciata attraverso il modulo **Runtime** di Toolbook (una serie di file che rendono possibile l'esecuzione dell'applicazione anche su computer dove il programma non è installato).

Lavorare con i testi

A23. Per selezionare e modificare il testo

Un campo (**Field**) è un oggetto rettangolare che può contenere del testo. Se il campo non è "protetto" anche il "lettore" può inserire del testo, selezionarne una parte, cancellarlo, ecc. I campi che appaiono identici su ogni pagina (e si trovano sul background) si chiamano Campi di record (**Record Fields**). In modalità **Reader** (Lettore) il testo può essere selezionato solo in campi non-protetti.

Per selezionare del testo di un campo o di un campo di record: selezionare il campo facendo click sopra di esso. Se si è in modo *autore* fare doppioclic su di esso per entrare in modo *modifica testo*. Una barra verticale apparirà in coincidenza del punto del campo dove si trova il cursore Fare click e trascinare (o premere *Shift* e poi i tasti freccia) per evidenziare il testo desiderato (che sarà selezionato). Un' altra tecnica di selezione consiste nel fare click all'inizio del testo da selezionare, premere poi il tasto *Shift*, posizionarsi con il mouse alla fine del testo da selezionare e fare di nuovo clic; oppure fare clic all'inizio del testo e tenendo premuto il tasto *Shift* premere ripetutamente i tasti freccia.

Per selezionare tutto il testo premere **SHIFT+F9**.

Una volta che il testo è stato selezionato (evidenziato) esso può essere cancellato o copiato: per cancellarlo premere **Canc** o **Shift+Canc**, oppure scegliere il comando Cancella (**Clear**) o Taglia (**Cut**) del menu Modifica (**Edit**) (con **Cut** si cancella e si pone negli appunti il testo selezionato, con **Clear** si cancella semplicemente). Per copiarlo premere **Shift+Canc** o scegliere Copia (**Copy**) dal menu Modifica (**Edit**).

A24. Per spostare il cursore o cancellare un carattere in un campo

Premendo il tasto **Canc** si cancella il carattere che si trova nella posizione immediatamente a destra della barra cursore: premendo il tasto **Backspace** invece si cancella il carattere che si trova nella posizione immediatamente a sinistra della barra cursore. Quando si scrive un testo Toolbook va automaticamente a capo a fine riga. Una nuova riga può essere creata premendo **Shift+Invio** ed un nuovo paragrafo premendo **Invio**.

Il cursore può essere spostato con il mouse semplicemente facendo clic nella nuova posizione. Il cursore può anche essere spostato con la tastiera:

- freccia in alto: porta alla linea sopra
- freccia in basso: porta alla linea sotto
- tasto **Home**: porta all'inizio della linea;
- tasto **End**: porta alla fine della linea;
- tasto **PgUp**: sposta di una pagina in alto;
- tasto **PgDn**: sposta di una pagina in basso;
- **Ctrl+Home**: porta all'inizio del campo;
- **Ctrl+End**: porta alla fine del campo.

Quando il cursore è nel campo (si sta modificando il testo) si può passare da un campo ad un altro, oltre che con il clic del mouse sopra il nuovo campo, anche premendo i tasti **TAB** o **TAB+SHIFT**.

Se un testo è stato cancellato per errore si può recuperare immediatamente (se non sono stati eseguite altre operazioni dopo la cancellatura) scegliendo Annulla (**Undo**) dal menu Modifica (**Edit**): si può anche premere contemporaneamente **Alt+Backspac**e.

A25. Come copiare un testo

Per copiare un testo bisogna prima selezionarlo e poi procedere nel modo seguente: scegliere Copia (**Copy**) dal menu Modifica (**Edit**) o premere su **Ctrl+Ins** in modo da copiare il testo selezionato negli appunti (*clipboard*) di Windows; posizionare il cursore nel punto desiderato, sullo stesso campo o su un altro campo, e scegliere Incolla (**Paste**) dal menu Modifica (**Edit**) o premere contemporaneamente **Shift+Ins**. Se non è selezionato nessun testo il comando **Copia** è non-attivo, così come è inattivo il comando **Incolla** se gli appunti sono vuoti.

A26. Scelta del Tipo, dello Stile e della Dimensione dei caratteri di un testo

Formattare un testo significa, tra le altre cose, scegliere il tipo di carattere (o **font**) con cui il testo deve apparire, nonchè la dimensione del carattere (Tipo, Stile e Dimensione). Toolbook differenzia fra formattare il Carattere (**Character**) e formattare il Paragrafo (**Paragraph**). Quando la formattazione è fatta sul Paragrafo i valori assunti valgono per tutto il testo del campo. Quando invece è scelta la formattazione del Testo la formattazione è valida solo per il testo selezionato o quello che sarà scritto dopo le nuove scelte (e non per il testo già scritto) immediatamente dopo il punto in cui si trova il cursore al momento delle nuove scelte. I cambiamenti sono sempre

validi per il testo selezionato o, se non è selezionato nessun testo, per i caratteri che saranno scritti.

Per formattare o modificare il carattere del testo selezionare il testo desiderato o posizionarsi con il cursore sul punto da dove si vuol iniziare a modificare il Tipo, lo Stile e la Dimensione dei caratteri.
Scegliere poi le caratteristiche volute dal box di dialogo che è mostrato premendo **F6** o selezionando la voce Carattere (**Character**) del menu Testo (**Text**). Toolbook usa i **fonts** tipici di Windows: *System*,*Terminal*, *Helvetica*, *Courier*. Una volta scelti il Tipo, lo Stile e la Dimensione del carattere è possibile controllare l'effetto della modifica facendo click nel box di dialogo su Applica (**Apply**): le scelte possono essere poi confermate con **OK** o annullate con Annulla (**Cancel**). Anche dopo aver premuto su **OK** è sempre possibile annullare la modifica scegliendo Annulla (**Undo**) dal menu Modifica (**Edit**).

Per scegliere direttamente uno stile di scrittura è possibile usare anche delle combinazioni di tasti, che dipendono però dalla versione del programma (italiana o inglese). Nella versione 3.0 in inglese:

- **Ctrl+B** sceglie il Grassetto (**Bold**)
- **Ctrl+I** sceglie il Corsivo (**Italic**)
- **Ctrl+U** sceglie il Sottolineato (**Underline**)
- **Ctrl+K** sceglie il Barrato (**Strikeout**)

In *OpenScript*:

```
set fontFace of field "A" to Helv      -- cambia in "helvetica"il testo del campo "A"
set fontSize of field "A" to 24        -- cambia il corpo del testo del campo "A"
```

A27. Come formattare un campo di testo

Formattare un campo di testo significa stabilire alcune caratteristiche con cui esso deve apparire come l' *allineamento* (Sinistro, Destro, Centrato, Giustificato o a sinistra e destra), l' *interlinea* (valore dello spazio fra le linee), i *rientri* dal bordo del campo (valore della distanza del testo dal bordo del campo della prima linea di un paragrafo, e di tutto il testo da sinistra e da destra) ed il tipo e valore della *tabulazione*. Se tali valori non vengono cambiati sono assunti quelli di *default* (quelli assunti come standard alla partenza del sistema Toolbook). In modo Lettore (**Reader**) è possibile formattare il testo solo nei campi non-protetti.

Per formattare il testo di un campo premere **F7** o scegliere Paragrafo (**Paragraph**) dal menu Testo (**Text**). Viene mostrato il box di scelta: scegliere quindi l'Allineamento (**Alignement**), l'Interlinea (**Spacing**), ed il Rientro (**Indents**). Anche in questo caso possono essere usati **OK**, **Apply** o **Cancel**.
L'Allineamento può essere a Sinistra (**Left**), a Destra (**Right**), Centrato (**Center**) o Giustificato (**Justify**). L'interlinea può essere Singola, 1/2 o Doppia. Il Rientro corrisponde allo spazio con cui la prima linea del paragrafo rientra rispetto al bordo destro ed allo spazio che vi è in tutto il paragrafo rispetto al bordo sinistro e destro del campo.

In *OpenScript*:

```
set textAlignment of field "A" to justify      -- "giustifica" l'allineamento del testo del campo "A"
```

A28. Come cercare e/o cercare e cambiare un testo in una pagina o in tutto il libro

Può essere necessario cercare una parola, una frase o una qualsiasi sequenza di caratteri nei campi di una pagina o in quelli di tutto il libro. Per fare questo premere **F5** o scegliere Cerca ... (**Find** ...) dal menu Modifica (**Edit**). Viene mostrato un box di dialogo nel quale oltre ad inserire il testo da cercare si può decidere anche di ordinare a Toolbook di cambiarlo con un altro testo. Si deve indicare poi se la ricerca deve avvenire solo sulla pagina attiva o su tutto il libro, se devono essere considerate o meno le maiuscole, se devono essere considerate solo le parole intere (**Match Whole Word only**) e se devono essere esclusi i campi dello sfondo. Si può anche stabilire che la ricerca avvenga solo sui campi di record. Premere quindi su Trova (**Find**) per cominciare la ricerca e su Cambia (**Change**) o Annulla (**Cancel**) quando la parola è stata trovata. Toolbook comincia la ricerca dalla pagina attiva e continua (se è stata ordinata la ricerca su tutto il libro) dalla pagina seguente fino alla fine del libro: domanda quindi se deve ricominciare la ricerca dalla prima pagina o si deve fermare. Nel caso in cui si ordini di ricominciare Toolbook ripete la ricerca dalla prima pagina fino alla pagina attiva. Una ricerca può esser interrotta premendo sul tasto **Esc**.

Le immagini

A29. Formati grafici

Toolbook 3.0 può trattare immagini esterne, ovvero richiamare e importare attraverso il comando **Import Graphic** (inviato da menu o da *script*) in molti formati grafici, tra i quali:

*.*bmp*: formato *bitmapped* standard (usato, ad esempio, dal programma Paintbrush)
*.*dib*: altro formato *bitmapped*
*.*pcx*: formato vettoriale (usato da Paintbrush)
*.*wmf*: formato vettoriale
*.*drw*: formato vettoriale (usato, ad esempio, dal programma Microsoft Draw)
*.*dxf*: formato vettoriale (usato nei programmi di CAD)
*.*gif*: formato vettoriale compresso (usato spesso nell'acquisizione di immagini statiche da video)
*.*tif*: formato vettoriale compresso (usato dal software di molti *scanner*)
*.*eps*: formato vettoriale
*.*cdr*: formato vettoriale (usato dal programma Corel Draw)

Quando si importano in Toolbook immagini in formato *bitmapped* Toolbook crea un oggetto **PaintObject**, quando si importano immagini in formato vettoriale Toolbook crea un oggetto **Picture**. Gli oggetti di tipo **Picture** possono essere ridimensionati una volta importati in Toolbook, al contrario degli oggetti di tipo **PaintObject**. Tuttavia, i **Picture**, dopo essere stati importati, possono essere convertiti in **PaintObject**, scegliendo l'apposita opzione nella finestra di dialogo delle loro proprietà. Gli oggetti di tipo **PaintObject**, anche se comportano interventi di modifica esterni a Toolbook (ovvero non possono essere ridimensionati una volta importati) sono tuttavia consigliabili, perchè vengono visualizzati più rapidamente e "affaticano" di meno la memoria del computer.

Attenzione ! Relativamente all'importazione di immagini nei formati *.bmp*, *.dib* e . *wmf* Toolbook utilizza filtri di lettura e di conversione che presuppone già presenti nel sistema. Bisogna essere sicuri della presenza di questi filtri, altrimenti l'importazione di quei formati grafici non potrà essere effettuata. Per quel che riguarda gli altri formati Toolbook dispone invece di suoi proprio filtri, che vengono installati insieme al programma. In questo caso potrebbero verificarsi dei problemi nella lettura di immagini compressetrattate - ad esempio quelle in formato *.tif* o *.gif* - poichè il sistema di compressione usato in sede di trattamento e salvataggio delle immagini potrebbe non essere standard, o diverso da quello che Toolbook è in grado di leggere e convertire.

A30. Il problema della palette e del trattamento delle immagini

Ogni immagine *bitmapped* contiene una "tavola" di informazioni sui colori usati nell'immagine stessa. La tavola dei colori di un'immagine viene chiamata **palette**. Quando un'immagine viene salvata in formato 8 bit, ovvero 256 colori (il minimo, se si vogliono inserire immagini di qualità nell'applicazione), a seconda del programma con cui viene acquisita o elaborata, essa può essere "convertita" sulla base di una palette *standard*, o di sistema, o di una palette "ottimizzata" e specifica, nella quale verranno calcolati e definiti soltanto i colori che il programma di trattamento grafico riconosce come i più adatti per una corretta visualizzazione dell'immagine, indipendentemente dalle esigenze del sistema. Nel primo caso, tutte le immagini avranno quindi la medesima *palette*, pur risultando, ciascuna, di qualità inferiore, nel secondo ogni immagine avrà invece una "sua" *palette* e la qualità di ciascuna risulterà superiore.

Se si utilizzano immagini a 256 colori che non usano la *palette* standard di Windows, quando si tenta di mostrarle contemporaneamente sulla stessa pagina Toolbook, o nel passaggio tra due pagine, si possono verificare dei conflitti fra le *palette* "ottimizzate" delle immagini: in particolare, accadrà che la *palette* dell'ultima immagine mostrata venga assegnata anche alle altre, con spiacevoli effetti di "snaturamento" o di *flash* nei cambi di pagina. Nella sostanza, la *palette* dell'ultima immagine mostrata o richiamata viene assunta come *palette-video*.

Per ovviare a tali inconvenienti, il sistema più semplice consiste nell'assegnare la stessa *palette* a tutte le immagini che devono essere contemporaneamente mostrate sullo schermo. E' sufficiente, in tal senso, scegliere una conversione di tipo *standard* o di sistema in sede di trattamento delle immagini. Ciò, tuttavia, comporta una perdita di qualità delle singole immagini, che non sempre può essere accettata (ad esempio, in applicazioni a soggetto storico-artistico). Se è dunque necessario che durante la conversione a 256 colori l'immagine venga "ottimizzata", si dovrà intervenire sull'applicazione stessa per evitare i problemi legati al "salto" di *palette* tra un'immagine e un'altra. In alcuni casi, quando il problema si presenta soltanto nei cambi di pagina, può essere sufficiente inserire tra una pagina e l'altra un "effetto" di passaggio ad un colore standard, ad esempio il nero. Ciò può essere ottenuto attraverso l'aggiunta di un semplice comando:

```
to handle buttonClick
    fxDissolve fast to black
    fxDissolve to page 5
end
```

Nei casi, invece, in cui più immagini con *palette* diverse vengano ad essere visualizzate contemporaneamente sulla stessa pagina o sullo stesso sfondo, non resta altro da fare che applicare a *tutte* la stessa *palette*. In tal senso, se l'uso di una *palette standard* comportasse un'eccessiva perdita di qualità delle singole immagini, va detto che Toolbook offre la possibilità di applicare alle immagini anche una eventuale *palette* appositamente costruita e calcolata, migliore di quella di sistema. Richiamando la finestra di dialogo delle proprietà del libro, infatti (**Book properties**), si può assegnare all'applicazione (pulsante **Color palette**) una qualsiasi *palette*, che verrà utilizzata come *risorsa* e automaticamente assegnata a tutte le immagini comprese nell'applicazione stessa, evitando i problemi di cui sopra. Si potrebbe quindi assegnare al libro una *palette* studiata in modo tale da adattarsi nel migliore dei modi alle immagini in esso contenute, "costruendola" o direttamente, attraverso un editor fornito insieme a Toolbook, o, meglio ancora, ricorrendo a delle *utilities* in grado di calcolare e definire la *palette* media ottimale applicabile ad un certo numero di immagini, in genere comprese nei migliori pacchetti di *software* per *computer graphic*.

Naturalmente, se il sistema è in grado di gestire immagini con un numero di colori superiore a 256, ovvero se le caratteristiche della scheda video sono tali da consentire la visualizzazione contemporanea sullo schermo di 64000 o 16 milioni di colori (24 bit), i problemi sopra indicati non si verificheranno o risulteranno impercettibili, e si potrà fare tranquillamente uso di immagini con *palettes* ottimizzate.

Si faccia attenzione, infine, al valore della proprietà **solidColorsEnabled** del **Book**. Quando il valore della proprietà risulta *true* (o la casella corrispondente nelle **Book properties** selezionata), Toolbook tratta la *palette* di sistema riservando 32 colori su 256 come *standard* e 64 come "colori solidi" assegnabili agli oggetti. Quando il valore è *false*, 20 colori su 256 vengono riservati come *standard* di sistema Windows. Nel primo due casi, potrebbero quindi verificarsi problemi di incompatibilità di *palette* anche tra oggetti Toolbook "colorati" e immagini trattate con *palette standard*.

Il viewer

A31. Che cosa sono i "viewers"

I "viewers" sono finestre che possono essere utilizzate per mostrare delle pagine di un libro Toolbook soprapponendole ad altre pagine. In pratica, attraverso un *viewer* una pagina può essere utilizzata come risorsa e richiamata e mostrata insieme a tutti gli oggetti che contiene. Questa possibilità è caratteristica della versione 3.0 di Toolbook, e non era prevista nella precedente versione.

La finestra principale all'interno della quale vengono visualizzate le pagine di un'applicazione Toolbook è anch'essa un *viewer*. Ad essa ci si può riferire con il termine **Main window**, attraverso il quale Toolbook la identifica per *default*. Alla **Main window** si può comunque dare un "nome" per riferirsi ad essa come ad un qualsiasi altro *viewer*.

I viewers possono essere usati per mostrare:

- box di dialogo
- *palettes*
- box di strumenti (*toolbars*)
- il *logo* o il riepilogo dei *credits* dell'applicazione

- finestre che mostrano altre pagine dell'applicazione o pagine di altre applicazioni

Per creare un *viewer* si clicca sull'apposita icona del box dei menu. Viene mostrata una finestra di dialogo in cui è riportato l'elenco dei viewers disponibili nell'applicazione corrente. In questo elenco è sempre riportata almeno la **Main window**.

Per creare un nuovo viewer bisogna premere sul pulsante **New**. Nella successiva finestra di dialogo si possono selezionare alcune opzioni utili: scegliere la voce **custom** se si desidera definire e personalizzare il viewer che verrà creato passo dopo passo, la voce **template** per lasciare a Toolbook il compito di attribuire al *viewer* determinate caratteristiche predefinite, a seconda dell'uso che si vorrà fare della finestra (selezionabile attraverso l'apposito menu a tendina). Si può scegliere o meno di creare un nuovo sfondo e una nuova pagina insieme al *viewer*: per *default*, Toolbook creare sempre uno sfondo e una pagina per ogni nuovo *viewer*, ma questo non è obbligatorio.

Premendo **Ok** sulla finestra di dialogo **New viewer**, viene mostrato il riepilogo delle *proprietà* del *viewer*, attraverso il quale si possono definire tutte le caratteristiche della finestra. Attenzione ! Questo stesso box di dialogo può essere utilizzato anche per modificare le caratteristiche della **Main window**, a patto che la voce corrispondente sia stata precedentemente selezionata sulla finestra **Viewers** per richiamarne le **properties** attraverso l'apposito pulsante. Quando si crea un *viewer* si può assegnare ad esso una pagina di *default*, che verrà richiamata automaticamente (salvo diversa indidcazione) quando il *viewer* verrà aperto e mostrato. L'indicazione della pagina di default non è obbligatoria. E' invece buona norma dare al *viewer* un nome, come si fa con tutti gli altri oggetti.

A32. Caratteristiche e proprietà dei "viewers"

Un viewer può essere **child** o **popup**. Se è *child*, esso potrà essere trascinato e spostato sullo schermo solo all'interno della finestra "madre", ovvero della **Main window** della stessa applicazione (o del *viewer* da cui viene richiamato). Se è *popup*, esso potrà essere trascinato e spostato anche al di fuori della finestra principale e si comporterà come una qualsiasi finestra Windows.

Rispetto alle caratteristiche e alle proprietà dei "viewers" ci si può comportare secondo lo schema che segue.

Style: è la sezione della finestra **Viewer properties** che serve a definire l'aspetto esterno e il comportamento generale del *viewer*. Relativamente all'aspetto, si possono selezionare le seguenti opzioni:

None: nessun bordo
Thin Frame: il bordo è costituito da una linea sottile
Thick Frame: l'aspetto è quello della normale finestra Windows
Dialog Frame: il bordo è in rilievo e l'aspetto è quello delle finestre di dialogo in Windows
Shadowed: il bordo è costituito da una linea sottile e la finestra proietta un'ombra

La barra della finestra, o **Caption Bar**, può essere definita a parte (**None**, **Normal**, **Thin**). A seconda dell'aspetto scelto, si può decidere anche di assegnare o meno al *viewer* gli elementi caratteristici delle finestre Windows (barra di scorrimento, casella di chiusura, caselle per ingrandire la finestra e per ridurla a icona). Si può assegnare al *viewer*, attraverso l'apposito pulsante, un'icona che verrà utilizzata quando esso verrà "ridotto" utilizzando l'eventuale, apposito pulsantino in alto a destra.

Position: è la sezione della finestra **Viewer properties** che serve a definire la posizione del *viewer* al momento di essere mostrato. Se si vuole che il viewer venga posizionato esattamente al centro dello schermo, indipendentemente dalla risoluzione di quest'ultimo, scegliere **Center**. Se si sceglie l'opzione **Custom** la posizione dovrà essere espressa in coordinate a seconda dell'unità di misura prescelta. Attenzione ! In quest'ultimo caso, poichè alla posizione viene attribuito un valore *assoluto* e non relativo, eventuali cambiamenti nella risoluzione dello schermo (ad esempio da 640 x 480 pixels a 800 x 600) comporteranno uno spostamento del *viewer*.

Size: è la sezione della finestra **Viewer properties** che serve a definire la grandezza di default del *viewer*. In realtà, scegliendo l'opzione **Auto... Size to Page**, la dimensione del *viewer* verrà automaticamente adattata alla dimensione della pagina ad esso assegnata quando verrà mostrato.

Limits: è la sezione della finestra **Viewer properties** che serve a definire i limiti massimo e minimo della dimensione del *viewer*. Le opzioni migliori sono quelle proposte per *default*.

Options: è la sezione della finestra **Viewer properties** che serve a definire determinati comportamenti del *viewer*, nonchè a decidere la presenza o meno di determinate caratteristiche (come la barra di *status*). Tra le opzioni a cui

bisogna fare particolare attenzione si segnalano almeno:

Close On ButtonClick: quando è selezionata l'opzione, il *viewer* si chiude automaticamente al primo clic del mouse. Quanto non è selezionata il *viewer* deve essere chiuso attraverso un'istruzione *OpenScript* del tipo: *close viewer* < *nome* >.

Always on Top: quando è selezionata l'opzione, il *viewer* rimane sempre sullo strato più "alto" dello schermo, anche in presenza di "viewers" richiamati successivamente (a meno che anch'essi non abbiano la stessa proprietà selezionata).

Center Client: quando è selezionata l'opzione, la pagina mostrata nel *viewer*, ovvero l'area riservata all'applicazione, viene centrata automaticamente nello schermo anche se la finestra viene ingrandita o ridimensionata secondo modalità Windows (pulsante "maximize" o trascinamento dei bordi).

Un *viewer* può essere mostrato in modalità **modal** oppure **nonmodal**. In modalità *modal* il *viewer* rimane attivo fino a quando non è chiuso o nascosto (con comandi del tipo *hide...* o *close...*): l'utente non può così interagire con il resto dell'applicazione mentre il *viewer* viene mostrato. In modalità *nonmodal* la possibilità di interagire con il resto dell'applicazione viene invece mantenuta. Le *palettes* sono un esempio tipico di *viewer nonmodal*, perché si può interagire con il resto dell'applicazione quando esse sono in mostra. La modalità di visualizzazione del *viewer* può essere definita nello *script* che lo richiama:

```
to handle buttonClick
        show viewer "prova" as modal
end
```

APPROFONDIMENTI DI II LIVELLO

Strutture di controllo e variabili

Le variabili

B1. Le variabili

Spesso può essere necessario conservare alcuni dati in delle caselle della memoria. Toolbook, allo scopo, ci mette a disposizione la variabile di sistema **it**. Tuttavia, essa potrebbe non essere sufficiente per i dati che vogliamo memorizzare. Dovremmo quindi riservarci altre caselle, e dare loro un nome per poi poterci riferire ad esse nei nostri *script*. Queste caselle di memoria vengono dette *variabili*. Esse avranno dunque un nome (che daremo noi, e sarà fisso) e potranno contenere un valore (che potrà *variare*, e che potrà essere "scritto" con un comando ti tipo **set** o "letto" con un comando di tipo **get**).

Toolbook è molto flessibile nella definizione di variabili. Se abbiamo bisogno di una casella di memoria per contenere un valore, basta introdurne il nome nello script e la variabile è già definita. Alla fine dello script (in corrispondenza del comando "end") tali variabili vengono però distrutte e le relative caselle di memoria svuotate. Queste variabili vengono dette "variabili locali", **local** in *OpenScript*. Se si vuole evitare che i dati memorizzati vengano eliminati al termine dello script bisogna invece esplicitamente definire delle variabili"di sistema", in *OpenScript* **system**. Queste ultime conserveranno il loro contenuto fino a quando non si chiuderà il book o fino a quando non verrà inviato un comando del tipo **restore system**.

Per dichiarare una variabile locale:

local A

Per dichiarare una variabile di sistema:

system A

In entrambi i casi la riga di istruzioni crea e rende subito disponibile il "contenitore" rappresentato dalla variabile. Attenzione ! In *OpenScript* non è possibile dichiarare due volte la stessa variabile all'interno di uno stesso *script*.

Ecco un esempio di utilizzo della variabile locale sullo script di un pulsante:

```
to handle buttonClick
        local A,B,C
        ask "dammi un numero..."
        A = it
        ask "dammene un altro..."
        B = it
        C = A + B
        request "la somma dei due numeri è " && C
end
```

L'istruzione **ask** chiede all'utente di inserire un valore. Quando l'utente preme il pulsante **Ok** per confermare la sua immissione il valore viene memorizzato nella variabile **A**. Sarebbe stato sufficiente, in questo particolare caso e trattandosi di variabili "locali", istruzione *A = it*, senza la precedente dichiarazione *local A*. Successivamente, l'intero ciclo si ripete e il nuovo valore inserito dall'utente nella nuova finestra **ask** viene memorizzato nella variabile **B**. A questo punto nella variabile **C** facciamo confluire il risultato della somma dei contenuti delle variabili A e B. Il valore della variabile **C**, infine, viene concatenato (**&&**) con il messaggio dell'istruzione **request** per comunicare all'utente il risultato dell'operazione. Al termine dello script tutte e tre le variabili (A, B e C) vengono distrutte e svuotate.

Se avessimo voluto conservarle fino alla chiusura del book avremmo dovuto scrivere:

```
to handle buttonClick
        system A, B, C
        ....
```

end

Il comando **system** definisce le variabili e fa sì che queste rimangano conservate e disponibili da tutti gli script di tutti gli oggetti del book fino alla sua chiusura. Il valore di A, B e C, in questo caso, avrebbe potuto essere richiamato anche dallo script di un altro pulsante, a patto di richiamare nuovamente i nomi delle variabili in questione con una riga **system A, B, C**.

B2. Applicazioni delle variabili: pulsanti "intelligenti"

Proviamo a definire qualche piccolo "trucco" utile per consentire che un pulsante, una volta cliccato, porti, ad esempio, ad una pagina che si chiama esattamente come la sua *etichetta* o come il suo *nome*. Il trucco consiste nel "simulare" la presenza di una variabile, usando il nome o l'etichetta dell'oggetto come se fossero elementi di una funzione:

```
to handle buttonClick
    go to page (name of self)
end buttonUp
```
-- il pulsante va in questo caso ad una pagina che si chiama come il suo *nome*.

```
to handle buttonClick
   go to page (caption of self)
end buttonUp
```
-- il pulsante va in questo caso ad una pagina che si chiama come la sua *etichetta* (**caption**).

Per ottenere lo stesso risultato si può naturalmente far riferimento al contenuto di una variabile locale:

```
to handle buttonClick
   local Destinazione
   set Destinazione to name of target
   go to page Destinazione
end buttonUp
```
-- va ad una pagina che si chiama come il suo *nome*, dopo che questo è stato memorizzato nella variabile "destinazione".

Le strutture di controllo

B3. Le strutture di controllo

Le strutture di controllo sono dei cicli di comandi che, all'interno di uno script, permettono di attivare delle istruzioni in relazione al verificarsi di determinate condizioni o di ripeterle per un certo numero di volte. Nel primo caso si avrà una struttura di controllo *condizionale*. Nel secondo una struttura di controllo *iterativa*. Nelle strutture di controllo è quasi assolutamente necessario far riferimento a delle variabili e al loro contenuto. Essendo dei cicli, le strutture di controllo devono essere "chiuse" prima che si chiuda l' *handler* in cui esse sono inserite. Lo script, cioè, avrà due istruzioni **end**, una in corrispondenza della chiusura del ciclo aperto dalla struttura di controllo, una in corrispondenza della fine dell' *handler*.

B4. La struttura di controllo "If"

Spesso si pone il problema di dover eseguire una particolare azione o sequenza di azioni in alternativa ad un'altra, in conseguenza di una certa condizione. Ad esempio, supponiamo che quando si entra in una pagina si voglia chiedere all'utente il suo sesso ed in conseguenza della risposta dare un messaggio diverso, se l'utente è maschio inviare un messaggio "Ciao bello!", se invece è femmina un messaggio "Ciao bella!". In questo caso è necessario aprire una struttura di controllo detta **if**.

in *OpenScript*:

```
to handle enterPage
        request "Qual è il tuo sesso?" with "Maschio" or "Femmina"
        if it is "Maschio" then
```

```
                request "Ciao bello !"
        else
                request "Ciao bella !"
        end if
end enterPage
```

Questo script deve appartenere alla pagina (viene infatti attivato in corrispondenza dell'evento *enterPage*). La prima istruzione chiede (*request*) all'utente il sesso suggerendo due sole possibili risposte ("maschio" o "femmina"). Quando Toolbook esegue questa istruzione mostra all'utente una finestra con il testo della domanda e due pulsanti etichettati con le rispettive possibili risposte. L'utente deve cliccare su uno di questi pulsanti e l'etichetta del pulsante cliccato viene conservata nella particolare "variabile di sistema" identificata con il nome di **it**. L'istruzione successiva (**if**) apre un ciclo di controllo sul contenuto della variabile **it**. *Se* questa contiene la parola "Maschio" *allora* (**then**, ma non è obbligatorio specificarlo) viene eseguita l'istruzione successiva (inviare il messaggio "Ciao, bello !"), *altrimenti* (**else**) viene eseguita l'istruzione alternativa. Il comando **end if** è essenziale e serve a indicare la fine del ciclo di controllo iniziato con il comando **if**.
Una istruzione **if** deve avere sempre un'istruzione corrispondente al *then* ma potrebbe anche non prevedere nessuna alternativa: potrebbe quindi anche non esserci il comando **else**. Ad esempio:

```
to handle enterPage
        request "Qual è il tuo sesso?" with "Maschio" or "Femmina"
        if it = "Femmina" then
                request "Ciao bella"
        end if
end enterPage
```

In questo caso il messaggio "Ciao bella" verrà inviato solo per le femminucce, mentre ai maschietti non verrà inviato alcun messaggio.

B5. La struttura di controllo "Conditions"

Con un ciclo **if** è possibile scegliere una delle due possibili strade in corrispondenza dell'esito di una condizione: la condizione su cui viene effettuato il controllo, cioè, può essere soltanto *vera*, al che si procede secondo la strada indicata da **then**, o falsa, nel qual caso si procede sulla strada de **else**. Può accadere però che sia necessario scegliere non tra due sole possibili alternative, ma tra tre o più sequenze di azioni scelte sempre sulla base dell'esito di particolari "condizioni". Per questo scopo, in *OpenScript* è prevista una struttura di controllo detta **conditions**. Illustriamo il concetto con un esempio di *script*:

```
to handle buttonClick
        ask "Scrivi un nome..."
        conditions
                when it = "Pietro"
                        request "Pietro è il nome del fondatore della Sacra Romana Chiesa"
                when it = "Francesco"
                        request "Francesco è il nome del Santo Patrono d'Italia"
                when it = "Giovanni"
                        request "Giovanni era il nome dell'aposto prediletto da Gesù"
                when it = "Nicola"
                        request "Nicola è il nome del macellaio giù all'angolo"
                else
                        request it && " chi é costui...?!"
        end conditions
end buttonClick
```

Con il comando **ask** si chiede all'utente di inserire un nome (e non di scegliere tra due opzioni). Il nome scritto dall'utente viene conservato nella variabile **it**. A questo punto vogliamo che lo *script* reagisca controllando se il nome inserito dall'utente è quello di un personaggio "famoso". Con l'istruzione **conditions** si iniziail controllo. Quando si verifica una delle condizioni definite in sede di *script*, ciascuna preceduta da **when**, vengono eseguite le istruzioni ad essa collegate e quindi si salta direttamente alla istruzione **end conditions**, obbligatoria anche in questo caso. Se non viene verificata nessuna delle condizioni previste si eseguono le istruzioni associate all'istruzione **else** (che non è obbligatoria: se omessa, lo *script* termina direttamente con **end conditions**). Non ci sono limiti al numero di alternative che è possibile verificare e controllare in un ciclo di tipo **conditions**.

B6. La struttura di controllo iterativa "while...end while"

Le strutture di controllo appena descritte ci permettono di differenziare il percorso di istruzioni da seguire all'interno di uno *script*, ma ancora non ci consentono di *ripetere* una o più istruzioni fino a quando non si verificano determinate condizioni. Supponiamo ad esempio di voler proteggere l'accesso ad una pagina chiedendo all'utente una parola d'ordine (*password*) e di voler continuare a chiederla, all'infinito se necessario, fino a quando l'utente non si decide a dare quella giusta. In questo caso abbiamo l'esigenza di ripetere un'azione (la richiesta della parola d'ordine) per un numero indefinito di volte (dipende da quando l'utente scriverà quella giusta). Per questo genere di esigenze è opportuno usare la struttura **while...end while**. Ecco uno *script* che esemplifica la soluzione del problema:

```
to handle enterPage
        ask "Parola d'ordine!"
        while it < > "apriti Sesamo"
                ask "Sbagliato! riscrivi la parola d'ordine!"
        end while
        request "Giusto, puoi passare."
end
```

Lo *script* va naturalmente associato alla pagina da proteggere. Non appena si entra nella pagina il comando **ask** chiede all'utente di scrivere la parola d'ordine. Viene quindi attivato il "ciclo": si controlla se **it** (ovvero il contenuto della variabile su cui viene memorizzato il testo inserito dall'utente nella finestra **ask**) è diverso (< >) dalla parola giusta (nel nostro caso è "apriti Sesamo"). Se ciò che ha scritto l'utente è diverso si deve chiedere ancora la parola (un nuovo comando **ask**). Che cosa succede ? Una volta giunto all' **end while** Toolbook torna al **while** e verifica se la condizione indicata è *falsa*: solo in questo caso continua con l'istruzione successiva all' **end while**, *altrimenti* continua a ripetere tutto ciò che c'è tra il **while** e l' **end while**. In pratica, con un ciclo di tipo **while**, si possono ripetere dei comandi finchè la condizione iniziale data (specificata all'inizio del ciclo, dopo il comando **while**) si mantiene *vera*.

B7. La struttura di controllo iterativa "do...until"

Questa struttura è analoga alla precedente ma presenta una sottile, seppure importante differenza. Nell'esempio precedente l'utente poteva già al primo **ask** inserire la parola d'ordine giusta, per cui l'azione "riscrivi la parola d'ordine", poteva anche non essere mai eseguita, saltando direttamente all' **end while**. La struttura **while...end while** si usa quindi quando la sequenza di istruzioni da ripetere potrebbe anche non essere mai eseguita. Se invece vogliamo che una sequenza di istruzioni venga eseguito almeno una volta si userà questa seconda struttura iterativa. Vediamo, in questo caso, come cambia lo *script*:

```
to handle enterPage
        do
                ask "Parola d'ordine!"
        until it =  "apriti Sesamo"
        request "Giusto, puoi passare."
end
```

In pratica, in questo caso, per uscire dal ciclo la condizione deve essere *vera* e non *falsa*, come nel caso precedente. Tra tutte le strutture di controllo **do** è l'unica che non necessita di un'istruzione di chiusura end, poichè il ciclo che essa apre si chiude automaticamente quando si verifica la condizione specificata nell'istruzione **until**.

B8. La struttura di controllo iterativa "step... end step"

Quest'ultima struttura di controllo può invece essere usata quando è già noto in anticipo quante volte deve ripetersi una determinata sequenza di azioni. Se per esempio vogliamo ripetere una certa azione esattamente per 10 volte realizzeremo uno *script* di questo tipo:

```
to handle buttonClick
        step i from 1 to 10
                request i
        end step
end
```

Viene usata una variabile locale (**i**) per contare le ripetizioni, i *passi* (**step**) che vanno da 1 a 10; ad ogni passo, in questo caso, viene eseguita l'istruzione **request** che mostra all'utente il valore progressivamente raggiunto dalla variabile **i**. Noterete che questo valore viene automaticamente incrementato di 1 ad ogni *passo* fino a giungere a 10 e quindi terminare l'esecuzione. Il ciclo può essere particolarmente utile per realizzare semplici ma efficaci animazioni, ad esempio per far muovere in modo regolare un oggetto sullo schermo:

```
to handle buttonClick
        step i from 1 to 50
                move target by 100, 0
        end step
end
```

L'oggetto su cui viene posto questo script, al clic del mouse, si sposta per 50 volte di 100 punti sull'asse orizzontale delle sue coordinate (x) e di 0 su quello verticale (y). Il risultato sarà un movimento fluido e regolare sul piano orizzontale dello schermo, da sinistra verso destra.

B9. Il pulsante per uscire

E' uno di quei pulsanti che dovrebbero essere presenti su ogni pagina (va quindi messo sullo sfondo). In pratica, il lettore, premendo su questo pulsante, chiude il libro uscendo da Toolbook. E' necessario però attivare un piccolo ciclo di controllo, per evitare che l'utente esca dal libro pur non desiderandolo, magari perchè a premuto inavvertitamente il pulsante.

in *OpenScript*:

```
to handle buttonClick
        request "Confermi la chiusura del libro?" with "Si" or "No"
        if it is "Si" then
                send exit
        end if
end
```

L'istruzione vera e propria per uscire è **send exit**. L'esecuzione di questa istruzione è però condizionata dall'esito della risposta data dall'utente alla richiesta di conferma.

APPROFONDIMENTI DI III LIVELLO

Estensioni Multimediali

La multimedialità con Toolbook 3.0

C1. La funzione playsound( ) in Toolbook 3.0

In Toolbook 3.0, versione non multimediale, il comando **playSound**() gestisce il suono in modo predefinito (formato .wav). Questo comando funziona al meglio con files di lunghezza inferiore a 100K ed accede automaticamente alla funzione standard MMSYSTEM.DLL di Windows 3.1. La funzione può essere attivata con uno *script* come quello che segue:

```
to handle buttonClick
  get playSound("c:\windows\chord.wav")      -- riproduce il suono specificato
end
```

Per interrompere l'esecuzione del file audio:

```
to handle buttonClick
  get playSound(null)                        -- ferma il suono
end
```

Se alla fine della linea di comando si aggiunge il parametro *true* la riproduzione audio non avviene in background ed il controllo torna al sistema solo alla fine della riproduzione. Per default tale parametro è settato come false.

In *OpenScript*:

```
to handle buttonClick
  get playSound("c:\windows\chord.wav", true)      --riproduce il suono ma non in background
end

to handle buttonClick
  get playSound("c:\windows\chord.wav", false)     --riproduce il suono  in background
end
```

Il comando **playSound( )** è disponibile anche in Multimedia Toolbook v.3.0, e può essere usato comunemente per riprodurre suoni di tipo *waveaudio* (.WAV)

C2. Per controllare le periferiche multimediali usando MMSYSTEM.DLL

Se non disponete di Multimedia Toolbook v.3.0 e volete controllare e usare le periferiche multimediali potete provare nel modo che indichiamo di seguito. Le periferiche multimediali possono essere controllate da Toolbook facendo diretto riferimento alle librerie di funzioni **DLL** proprie di Windows 3.1. In particolare, all'interno di un *handler*, si devono definire delle istruzioni per collegarsi (*link*) alla libreria di funzioni **mmsystem.dll**. Il *link* va effettuato *immediatamente* prima di usare la periferica multimediale e chiuso *immediatamente* dopo. In alternativa, può essere realizzato in un *enterBook* (o *enterSystem*) e chiuso in *leaveBook*.

C3. Linkare e dichiarare le funzioni DLL

Se nel book si vogliono usare le dynamic link librairies (DLL) di Windows il loro uso deve essere preliminarmente dichiarato. Perché un book possa usare una DLL si deve dichiarare in esso quali delle funzioni della DLL saranno usate; dopo tale dichiarazione le funzioni potranno essere usate come qualsiasi altra funzione Toolbook. Attenzione ! Dichiarare funzioni DLL occupa memoria, per cui è bene DICHIARARE SOLO LE FUNZIONI CHE SARANNO EFFETTIVAMENTE USATE e CHIUDERLE QUANDO NON SONO PIU' NECESSARIE

Il *link* si realizza con un ciclo di istruzioni **linkDLL**: le funzioni richiamate rimangono attive fino al comando opposto (**unlinkDLL)** oppure fino al comando **restore system** (che annulla tutti i *link* alle DLL). Sarebbe buona regola porre nell'handler di chiusura del book le istruzioni:

```
to handle leaveBook
 restore system
end leaveBook
```

Il *link* può essere stabilito in qualunque punto di uno script: bisogna comunque fare attenzione, perché il processo di **linkDLL** richiede del tempo e può rallentare l'esecuzione di uno *script*. Buona regola è stabilire il **linkDLL** all'inizio di una applicazione (in **enterBook**) e togliere il *link* (**unlinkDLL**) alla fine dell'applicazione (**leaveBook**).

In *OpenScript*:

```
to handle enterBook
 linkDLL "mmsystem.dll"
    INT mciExecute(STRING)
 end linkDLL
 forward
end enterBook

to handle leaveBook
 unlinkDLL "mmsystem.dll"
end leaveBook
```

E' sottinteso che **mmsytem.dll**, o si trova nella directory di lavoro o è accessibile tramite una *pathname* definita nella configurazione della macchina (come normalmente accade).

C4. Controllo degli errori nell'uso delle periferiche

Per controllare ed evitare eventuali errori si può procedere nel modo che segue:

```
to handle enterBook
   set sysSuspend to true
   clear sysError
     linkDLL "mmsystem.dll"                          -- attiva il link
       INT suona = mciExecute(STRING)          -- usa un alias (suona) per evitare conflitti d'uso
     end linkDLL
       if sysError < > null
          request sysError & CRLF \
          & "USO della periferica non possibile"          -- indica se ci sono problemi con la periferica
       end if
    set sysSuspend to false
  forward
end enterBook

to handle leaveBook
 unlinkDLL "mmsystem.dll"                        -- disattiva il link
 restore system
 forward
end leaveBook
```

E' chiaro, comunque, che le periferiche multimediali devono esistere "fisicamente" nel sistema: il loro uso deve essere dichiarato in *win.ini* e in *system.ini* e pilotato tramite appositi *drivers* (files) che si devono trovare nella directory *system* di Windows 3.1 con i relativi files (*sbpsnd.drv*, *indeov.drv*, ecc...). **Ripetiamo:** perché le periferiche multimediali siano correttamente disponibili all'uso devono essere correttamente installate in windows 3.1

C5. Riproduzione di un suono usando le DLL

Dopo aver attivato un **linkDLL** come il precedente, per riprodurre, ad esempio, un suono, si può procedere inserendo in un pulsante uno *script* come quello che segue:

```
to handle buttonClick
          get suona("sound c:\windows\chord.wav")
end
```

-- Apre e riproduce un file audio: è meglio che il files .wav non superi la dimensione di 100 KB.

Per riprodurre un cd audio:

```
to handle buttonClick
	linkDLL "mmsystem.DLL"
		INT mciExecute(STRING)
	end linkDLL
	get mciExecute("open cdAudio")
	get mciExecute("play cdAudio from 1:00 to 1:37:24")
	request “Fai click per stop”
	get mciExecute("stop cdAudio")
	get mciExecute("close cdAudio")
	unlinkDLL "mmsystem.DLL"
end buttonClick
```

Per riprodurre un cd audio usando un *alias*:

```
to handle buttonClick
	linkDLL "mmsystem.DLL"
		INT mciExecute(STRING)
	end linkDLL
	get mciExecute("open cdAudio alias prova")
	get mciExecute("play prova")
	request “Fai click per stop”
	get mciExecute("stop prova")
	get mciExecute("close prova")
	unlinkDLL "mmsystem.DLL"
end buttonClick
```

Oppure:

```
to handle buttonClick
	linkDLL "mmsystem.DLL"
		INT mciExecute(STRING)
	end linkDLL
	get mciExecute("open prova.wav type waveAudio alias prova1")
	get mciExecute("play prova1")
	request "Fai click per pausa"
	get mciExecute("pause prova1")
	request "Fai click per riprendere"
	get mciExecute("resume prova1")
	request "Fai click per stop"
	get mciExecute("stop prova1")
	get mciExecute("close prova1 ")
	unlinkDLL "mmsystem.DLL"
end buttonClick
```

Chiudere sempre la periferica, attivando un pulsante con uno *script* come quelli che seguono:

```
to handle buttonClick
	get mciExecute("close all")
	unlinkDLL "mmsystem.DLL"
end buttonClick

to handle buttonClick
	get mciExecute("close prova")
	unlinkDLL "mmsystem.DLL"
end buttonClick

to handle buttonClick
	get mciExecute("close waveAudio")
```

```
        unlinkDLL "mmsystem.DLL"
end buttonClick

to handle buttonClick
        get mciExecute("close cdAudio")
        unlinkDLL "mmsystem.DLL"
end buttonClick
```

C6. Per riprodurre un video usando le DLL

Per riprodurre in una applicazione Toolbook una sequenza video catturata in formato Video for Windows (AVI), usando le DLL, si deve:

- disporre di Windows 3.1
- aver correttamente installato in questo ambiente i files *runtime* (i *drivers*) di *Video for Windows*
- aver scritto un *handler* del tipo:

```
to handle buttonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
         get mciExecute("open c:\avi\prova.avi type aviVideo alias prova2")
                 -- open apre il files
                 -- wait riproduce il filmato senza tornare
                 -- type aviVideo e' il tipo di periferica indicato in [mci] di win.ini
                -- se non e' indicato il tipo e' accettato il valore di default
                -- al posto di type si puo' mettere il carattere !
                -- prova2 e' l'alias
        get mciExecute("play prova2 wait")
                -- riproduce il files il cui alias e' prova2
        get mciExecute("close prova2")
                -- chiude la periferica il cui alias e' prova2
        unlinkDLL "mmsystem.DLL"
end buttonClick
```

Resta sottinteso che tutte le periferiche devono risultare correttamente installate.

C7. Per registare un suono

La registrazione di un suono o di una sequenza audio può essere attivata da un'applicazione Toolbook anche utilizzando le DLL. Lo "start" della registrazione può essere affidato ad un pulsante con uno script come quello che segue:

```
to handle buttonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute ("open new type waveaudio  alias registra","")     -- apre
        get mciExecute("record registra")                                 -- registra
        request "Fai clic quando la registrazione è terminata .."
        get mciExecute("save registra prova.wav")
                -- salva  la registrazione in un files di nome prova.wav
        get mciExecute("stop registra")
        get mciExecute("close registra")
        unlinkDLL "mmsystem.DLL"
end buttonClick
```

C8. Esempi di funzioni multimediali realizzate usando le DLL: file audio

Per aprire, riprodurre e chiudere un audio:

```
to handle buttonClick
```

```
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute("open prova.wav type waveaudio alias prova2")
        get mciExecute("play prova2 wait")
        get mciExecute("close prova2")
        unlinkDLL "mmsystem.DLL"
end buttonClick
```

Per aprire un audio:

```
to handle buttonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute("open prova.wav type waveaudio alias prova2")
end buttonClick
```

Per riprodurre un audio:

```
to handle buttonClick
        get mciExecute("play prova2")
end buttonClick
```

Per riprodurre un audio da ... a ...:

```
to handle buttonClick
        get mciExecute("play prova2 from 656 to 1500")
end buttonClick
```

Per fare una pausa nell'esecuzione di un audio:

```
to handle ButtonClick
        get mciExecute("pause prova2")
end ButtonClick
```

Per fermare un audio:

```
to handle ButtonClick
        get mciExecute("stop prova2")
end ButtonClick
```

Per chiudere la periferica audio:

```
to handle ButtonClick
        get mciExecute("close prova2")
        unlinkDLL "mmsystem.DLL"
end ButtonClick
```

C9. Esempi di funzioni multimediali realizzate usando le DLL: CD audio

Per aprire la periferica CD audio ed iniziare dall'inizio la riproduzione:

```
to handle ButtonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute("close all ")
        get mciExecute("open cdAudio")
        get mciExecute("play cdAudio")
end ButtonClick
```

Per fare una pausa durante la riproduzione:

```
to handle ButtonClick
        get mciExecute("pause cdAudio")
end ButtonClick
```

Per riprendere l'esecuzione e l'ascolto dopo la pausa:

```
to handle ButtonClick
        get mciExecute("play cdAudio")
end ButtonClick
```

Per fermare l'esecuzione (corrisponde alla pausa):

```
to handle ButtonClick
        get mciExecute("stop cdAudio")
end ButtonClick
```

Per posizionarsi alla fine del CD audio:

```
to handle ButtonClick
        get mciExecute("seek cdAudio to end")
end ButtonClick
```

Per posizionarsi all'inizio del CD audio:

```
to handle ButtonClick
        get mciExecute("seek cdAudio to start")
end ButtonClick
```

Per posizionarsi su di una traccia specifica del CD audio:

```
to handle ButtonClick
        get mciExecute("seek cdAudio to  08:05:45")
end ButtonClick
```

Per chiudere la periferica:

```
to handle ButtonClick
        get mciExecute("close all")
        unlinkDLL "mmsystem.DLL"
end ButtonClick
```

C10. Esempi di funzioni multimediali realizzate usando le DLL: video

Per riprodurre e chiudere un video:

```
to handle ButtonClick
        linkDLL "mmsystem.DLL"
        INT mciExecute(STRING)
        end linkDLL
        get mciExecute("open c:\avi\prova.avi type aviVideo alias prova2")
        get mciExecute("play prova2 wait")
        get mciExecute("close prova2")
        unlinkDLL "mmsystem.DLL"
end ButtonClick
```

Per aprire un video:

```
to handle ButtonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
```

```
        end linkDLL
        get mciExecute("open c:\avi\prova.avi type aviVideo alias prova2")
end ButtonClick
```

Per aprire un video in modo *popup*:

```
to handle ButtonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute("open c:\avi\prova.avi type aviVideo alias prova2 style popup")
end ButtonClick
```

Per aprire un video in modo *overlapped*:

```
to handle ButtonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute("open c:\avi\prova.avi type aviVideo alias prova2 style overlapped")
end ButtonClick
```

Per aprire un video in modo *child*:

```
to handle ButtonClick
        linkDLL "mmsystem.DLL"
                INT mciExecute(STRING)
        end linkDLL
        get mciExecute("open c:\avi\prova.avi type aviVideo alias prova2  style child parent"  \
        &&  sysWindowHandle)
end ButtonClick
```

Per riprodurre un video:

```
to handle ButtonClick
        get mciExecute("play prova2")
end ButtonClick
```

Per fare una pausa in un video:

```
to handle ButtonClick
        get mciExecute("pause prova2")
end ButtonClick
```

Per fermare un video:

```
to handle ButtonClick
        get mciExecute("stop prova2")
end ButtonClick
```

Per riprodurre una precisa serie di frames di un video:

```
to handle ButtonClick
        get mciExecute("play prova2 from 6 to 15")
end ButtonClick
```

Per riprodurre un video al contrario:

```
to handle ButtonClick
        get mciExecute("play prova2 reverse")
end ButtonClick
```

Per riprodurre un frame alla volta di un video:

```
to handle buttonDown
        get mciExecute("step prova2 by 1")
end
```

Per riprodurre una frame alla volta di un video all'indietro:

```
to handle buttonDown
        get mciExecute("step prova2 by 1 reverse")
end
```

Per chiudere un video:

```
to handle ButtonClick
        get mciExecute("close prova2")
end ButtonClick
```

La multimedialità con Multimedia Toolbook 3.0

C11. Requisitit di sistema per poter usare Multimedia Toolbook v. 3.0

La configurazione minima del sistema richiesto per poter lavorare con Multimedia Toolbook v.3.0 è la seguente:

- Windows 3.1 o superiore
- 80486 (raccomandato 80486 DX2 66 MHz o superiore)
- HD capiente con 8 --> 24 MB di spazio libero
- 6 MB di RAM (8MB raccomandati, meglio se 16 MB)
- scheda grafica SVGA
- mouse
- scheda audio MCI compatibile 8-->16 bit, sintetizzatore e riproduzione MIDI
- altoparlanti e microfono
- lettore CD-ROM MCI2 compatibile (almeno a doppia velocità => 300 KB/secs < 300 milliseconds, photoCD compatibile, CD-AUDIO)

C12. Particolarità di Multimedia Toolbook v.3.0

Multimedia Toolbook v.3.0 prevede, rispetto a Toolbook 3.0, due oggetti in più, in particolare due *gestori* multimediali: i **clips** e gli **stages**. Ai **clips** si accede cliccando sull'apposita icona della barra dei menu oppure con:

Object --> clips

Agli **stages** si accede facendo clic sull'apposito pulsante del box degli strumenti: essi possono essere costruiti come un qualsiasi altro oggetto.

Nel linguaggio *OpenScript* supportato da Multimedia Toolbook v.3.0 sono disponibili alcuni comandi specificamente dedicati alla gestione dei due oggetti:

mmPlay -- riproduce un clip. Se non è specificato uno *stage* il clip è riprodotto in una finestra *popup*

```
mmPlay clip "film"
mmPlay clip "film"  in stage "scena"
mmPlay clip "film"  from 1000 to 3000
mmPlay clip "film"  in stage "scena" wait
mmPlay clip "film"  in stage "scena" from 23 to 56
mmPlay clip "film"  in stage "scena" hold
        -- il parametro hold pone in pausa il clip alla fine della registrazione
mmPlay clip "film"  in stage "scena" release
       -- il parametro release pone in stop il clip alla fine della registrazione
```

mmOpen -- carica un *clip* in memoria

mmOpen clip "film"

mmClose -- scarica un *clip* dalla memoria

mmClose clip "film"

mmStop -- ferma e chiude un *clip*

mmStop clip "film"

mmPause -- fa fare una pausa al *clip*

mmPause clip "film"

mmCue -- porta al punto di partenza il *clip* (*frame* 0)

mmOpen clip "film" wait
mmCue clip "film"

mmSeek -- porta ad un punto specificato del *clip*

mmTimeFormat of clip "film" = "frames" -- setta il conteggio in *frames*
mmSeek clip "film" to 12 -- posiziona il clip sul *frame* 12 contando dalla prima
mmSeek clip "film" to 12 from end -- posiziona il clip sul *frame* 12 contando a partire dall'ultimo

mmStep -- incrementa il *clip* di un valore specificato

mmTimeFormat of clip "film" = "frames" -- setta il conteggio in *frames*
mmStep clip "film" by "1" -- avanza di 1 *frame*
mmStep clip "film" back by "1" -- torna indietro di 1 *frame*

mmRewind -- ferma un *clip* e torna alla posizione di partenza

mmRewind clip "film"
mmRewind waveAudio
mmRewind digitalVideo
mmRewind animation
mmRewind all

mmShow -- mostra un *clip*
mmHide -- nasconde *clip*
mmStatus -- controlla lo *status* di un *clip*
mmIsOpen -- indica se un *clip* è aperto
mmTimeFormat -- setta il modo di controllare il *clip* (milliseconds, *frames* ecc..)

C13. Inserimento degli oggetti multimediali in Multimedia Toolbook v.3.0

Per inserire in Toolbook oggetti multimediali, come video o animazioni, bisogna definire un **clip** ed uno **stage** in cui il video o l'animazione possano essere riprodotti. Un **clip** è un oggetto di Toolbook, e come tutti gli oggetti possiede *proprietà*. In generale si deve assegnare un *nome* (unico) ad un **clip** per potersi riferire ad esso in modo inequivocabile (altrimenti si deve usare il suo *uniquename*). Per i video, le animazioni, l'audio si può anche indicare, tra le proprietà dell'oggetto, il punto di inizio (*start point*) e di fine (*end point*) della riproduzione (**timingTab** nel dialog box del **clip**). La procedura per l'inserimento degli oggetti multimediali è la seguente:

- si crea un **clip**
- si crea uno **stage**
- si prepara uno script (in *openScript*) che controlla il **clip** nello **stage**

Supponiamo di voler riprodurre un file digitale "prova.avi" (un video) che si trova nella directory "c:\media" del disco rigido. Si può procedere nel modo che segue:

1 Si crea il **clip**:

- si fa clic sul pulsante dei **clip** della barra dei menu oppure si seleziona **Object --> clips**
- sul dialog box che segue si sceglie **New**
- sul successivo dialog-box si sceglie il *file* da riprodurre cioè "c:\media\prova.avi"
- si conferma la scelta (**Ok**)
- si dà un nome al **clip**, ad esempio "provaclip"
- si fa clic su **timingTab** e si scelgono i *frames* da riprodurre
- si clicca su **>** per "provare" il **clip**
- si conferma (**OK**) ed il **clip** è creato

2 Si crea lo **stage** in cui il **clip** sarà riprodotto:

- si fa clic sul pulsante apposito del box degli strumenti
- si disegna lo **stage** sullo schermo (sulla pagina)
- si attiva il dialog-box delle *proprietà* dello **stage** e se ne definiscono le proprietà e il nome (ad esempio "stageDiProva")
- si fa clic su **Ok** e lo **stage** è creato

3 Si predispone lo *script* che controlla il **clip** nello **stage**. Ad esempio, si crea un pulsante e vi si pone lo *script*:

```
to handle buttonClick
  mmOpen clip "provaclip"                              -- apre il clip
  mmPlay clip "provaclip" in stage "stageDiProva" wait -- riproduce il clip fino alla fine
                                                       -- senza tornare all'applicazione
                                                       -- non in background (wait)
  mmClose "provaclip"                                  -- chiude il clip
end buttonClick
```

Più in generale, la procedura per l'inserimento e l'uso degli oggetti multimediali in Multimedia Toolbook v.3.0 può essere così schematizzata:

1 creare una *directory* dove porre gli oggetti multimediali da usare come clips (*media path*)
2 usare l'editor di **clip** per creare i clips che verrano usati nell'applicazione
3 per clips tipo video ed animazioni creare uno **stage** appropriato
4 scrivere gli *script* che attivano i clips

Bisogna sempre ricordarsi che possono essere usate solo le periferiche che sono state correttamente installate sul sistema.

Non sempre è necessario usare uno **stage**. Ad esempio, proviamo a creare un **clip** audio (*.wav*) usando il file "c:\windows\tada.wav", in modo che tutte le volte che si entra in una pagina si abbia un segnale sonoro. Seguiamo la consueta procedura:

- fare clic sul pulsante crea-**clip** della barra dei menu;
- sul dialog-box che segue scegliere **New** e poi **Sound** (Files);
- sul successivo dialog-box scegliere "c:\windows\tada.wav" e confermare;
- in **Name** dare un nome al **clip**, ad esempio "avviso"
- confermare con **Ok** in modo da creare il **clip**
- associare il seguente *script* allo *sfondo*:

```
to handle enterPage
  mmPlay clip "avviso"
end
```

Quando si distribuisce una applicazione si devono inserire in essa *tutti* gli oggetti multimediali che l'applicazione userà (quelli a cui i **clips** si riferiscono). Questi oggetti sono "esterni", e non vengono incorporati nell'applicazione, per cui o sono copiati nella directory in cui si trova il book che li usa e dove vengono cercati per *default*, oppure

sono copiati in apposite *directory* opportunatamente indicate come "media path" nell'applicazione stessa. Quando si inizia la costruzione di una applicazione è buona regola porre tutti gli oggetti multimediali in *directory* che saranno le stesse dell'applicazione finale. Per esempio se si lavora in una directory di nome "c:\firenze" e gli oggetti multimediali sono posti in "c:\avi", "c:\fli", "c:\wav" la *media path* definita nell'applicazione dovrà contenere le indicazioni "c:\avi, c:\fli, c:\wav", in modo che tutti i clip possano essere automaticamente ricercati in queste directory (qualcosa di simile alla definizione della *path* nei files di configurazione del DOS). Creare una *media search path* significa indicare alla applicazione dove cercare (oltre alla directory attiva) gli oggetti multimediali.

In *OpenScript*:

push "c:\avi, c:\fli, c:\wav" onto HDMediaPath

C14. Per riprodurre un oggetto multimediale

Quando si riproduce un oggetto multimediale è bene osservare alcune regole generali:

- caricare sempre in memoria il **clip** con un comando **mmOpen** (ma non è obbligatorio)
- riprodurre il **clip** con un comando **mmPlay**
- chiudere il **clip** con un comando **mmClose** (ma non obbligatorio)
- aver cura di consentire all'utente di interrompere l'esecuzione del **clip**

Ad esempio, sullo script di una pagina si potrebbero inserire questi riferimenti:

```
to handle enterpage
 mmOpen clip "film"
 forward
end
 -- carica in memoria il clip quando si entra nella pagina

to handle leavepage
 mmClose clip "film"
 forward
end
-- scarica dalla memoria il clip quando si esce dalla pagina
```

Su un pulsante posto nella pagina, invece:

```
to handle buttonClick
 mmPlay clip "film"
end
 -- riproduce il clip
```

Su un altro pulsante posto nella stessa pagina, ancora:

```
to handle buttonClick
 mmPause clip "film"
end
 -- fa fare una pausa al clip
```

Su un terzo pulsante posto nella stessa pagina, infine:

```
to handle buttonClick
 mmStop clip "film"
end
 -- ferma il clip
```

C15. Considerazioni generali sugli oggetti multimediali di Toolbook

Gli oggetti multimediali che è possibile gestire in Multimedia Toolbook v.3.0 sono:

-- audio (files)
-- CD audio

-- video digitale (files)
-- animazioni
-- video da videodisco
-- video da videoregistratore

Tutti questi oggetti multimediali, per essere riprodotti, hanno in generale bisogno di supporti *hardware* e *software* (*driver*) indipendenti da Toolbook. E' perciò importante che ciascun oggetto multimediale sia correttamente installato e che il sistema possieda i dispositivi hardware e software capaci di guidarlo. Ad esempio, i files audio necessitano di una scheda audio (*hardware*) e di un *driver* (*software*) correttamente installati sul sistema (computer + Windows 3.1.)

C16. Gli oggetti di tipo "audio"

L'audio riproducibile da Toolbook può essere in formato:

-- waveaudio files (*.wav)
-- midi files (*.mid)
-- CD audio

La registrazione in diretta (conversione analogico - digitale tramite microfono o *line in*) può essere effettuata normalmente in formato waveaudio e registrata in files *.wav*. Se si dispone di un dispositivo adeguato si può registrare in formato *midi* (files *.mid*). In generale vale il seguente criterio:

-- per la voce e per i suoni semplici usare files *.wav*
-- per la musica usare files *.mid* o CD - AUDIO

Per usare formato *midi* si deve fare particolarmente attenzione allo standard, nel senso che non è sufficiente avere a disposizione un buona stazione di produzione, ma è necessario produrre l'oggetto in "modo standard" e disporre di una scheda di riproduzione con un sintetizzatore (*sequencer*) al livello della qualità con cui il file è stato prodotto. Per esempio le comuni schede *Sound Blaster* riescono a riprodurre *midi* standard, ma non con una qualità elevata (spesso non accettabile, dipende anche dal tipo di scheda). I files *midi* occupano molto meno spazio in *bytes*.
L'audio registrato e/o riprodotto dal computer è in formato digitale (numerico): il ruolo della scheda audio è quello di convertire l'audio da analogico a digitale (registrazione) e/o da digitale ad analogico (riproduzione).

La qualità dell'audio digitale è data dal tipo di campionamento (8 o 16 bit), dalla frequenza di campionamento (11025, 22050, 44100 Khz) e dal modo (mono, stereo) con cui è stato registrato.
La massima qualità è data dal CD-AUDIO che è registrato a 16 bit, 44100 Khz in modo stereo, tuttavia non è possibile registrare un CD - AUDIO (se non disponendo dell'attrezzatura adeguata, il CD-AUDIO è a sola lettura) ed è necessario disporre in lettura di un lettore di CD-ROM (audio compatibile). Maggiore è la qualità con cui si registra e maggiore è lo spazio occupato dal file audio. Ad esempio, per 1 minuto (60 secondi) di registrazione audio in formato *.wav* si può tener conto della seguente tabella:

Tipo di registrazione		spazio occupato	qualità
8 bits	11025 kHz	660 KB	bassa
8 bits	22050 Khz	1323 KB	buona
8 bits	44100 Khz	2640 KB	più che buona
16 bits	11025 kHz	1323 KB	accettabile
16 bits	22050 Khz	2640 KB	più che buona
16 bits	44100 Khz	4892 KB	alta qualità

C17. Gli oggetti di tipo "video"

Gli oggetti video che è possibile gestire in Toolbook possono essere:

-- files video digitali in formato MCI compatibile
    tipo *.avi (Video for Windows)
    tipo *.mov (Apple Quick Time for Windows)
    tipo *.mpg (MPEG)
-- videodisco
-- videoregistratore

Per essere "catturati" e registrati i files video digitali necessitano di un apposito dispositivo *hardware-software* (schede cattura video tipo *Video Blaster*, ad esempio), ma non necessitano di un *hardware* particolare per la loro riproduzione. Per riprodurre un video digitale è sufficiente aver installato sul sistema i *drivers* appropriati al video da riprodurre. Il video digitale può essere registrato su hard disk o su CD-ROM. I files video digitali occupano generalmente molto spazio in relazione alla qualità (numero dei colori e *frames* al secondo - da 15 a 30) e alla dimensione della finestra video rispetto allo schermo. La qualità di riproduzione di un file video digitale dipende molto dal sistema sul quale esso è riprodotto (CPU, velocità HD e/o CD-ROM, velocità scheda video ecc..). Va considerato che 1 minuto di video digitale può occupare anche 20 MB, per cui, per distribuire una applicazione contenente video si deve pensare, come supporto, al CD-ROM.

Il video digitale contenuto in videodischi può essere riprodotto ad alta qualità in modo *overlay* (in sovrapposizione) o su monitor separato, a pieno schermo (*full video*) ed a pieno regime (*full motion*). Esso necessita tuttavia, in fase di riproduzione, oltre che del videodisco (che deve essere distribuito con l'applicazione) anche di un apposito *hardware* (lettore di videodisco - normalmente collegato alla porta seriale - e scheda *overlay*). Lo stesso discorso vale per la riproduzione video tramite videoregistratore, sul quale il video è in forma analogica, registrato su cassetta. Anche questo dispositivo che può essere controllato da Toolbook, ma ha il difetto di una estrema lentezza nella ricerca (che è sequenziale): Inoltre deve essere compatibile con il computer e si deve disporre di una scheda typo *overlay*.

C18. Gli oggetti di tipo "animazione"

Si parla di animazioni quando invece di avere a che fare con filmati veri e propri, si riproducono una serie di disegni in rapida successione. La natura dell'animazione fa sì che essa sia riproducibile sullo schermo su finestre di dimensioni maggiori rispetto al video. Le animazioni, inoltre, occupano meno spazio in *bytes*. In generale, è preferibile l'animazione al video, anche perché essa richiede, in fase di riproduzione, un sistema meno potente. Tuttavia, la costruzione di un'animazione può essere più costosa della registrazione di un video, dato che ha bisogno dell'intervento di grafici esperti (capaci di *fare* grafica con il computer).

Per riprodurre animazioni non è richiesto *hardware* aggiuntivo, ma i *drivers* dello specifico tipo di animazione devono essere correttamente installati sul sistema.

I tipi di animazione supportati da multimedia toolbook sono:

*.fli *.flc (Autodesk Animator e 3D Studio)
*.mmm (Macromind Director)

PICCOLO DIZIONARIO DI OPENSCRIPT

GLI OGGETTI

AngledLine

E' una linea spezzata con vertici ad angolo. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Arc

E' un arco di cerchio. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Background

Il background o sfondo è uno degli oggetti fondamentali dell'applicazione Toolbook. Tutti gli oggetti posizionati sul background verranno visualizzati su tutte le pagine ad esso riferite. Per cambiare gli attributi e le caratteristiche di un background scegliere la voce background properties dal menu **object** a livello autore. Ogni libro scritto in Toolbook contiene almeno un background, o sfondo. Per aggiungere un nuovo background ad un libro selezionare la voce new background dal menu page a livello autore.

Book

Il book è il file applicativo generato grazie a Toolbook. Un libro, che contiene un certo numero di sfondi o backgrounds e un certo numero di pagine ad essi collegate. Per cambiare gli attributi e le caratteristiche di un libro scegliere la voce book properties dal menu **object** a livello autore. Il libro è l'oggetto Toolbook che contiene tutti gli altri oggetti. Ogni volta che Toolbook viene lanciato genera un libro senza titolo, che potrà essere salvato e riaperto successivamente.

Button

E' l'oggetto a cui generalmente vengono associati eventi e comandi. Per cambiare gli attributi e le caratteristiche di un pulsante selezionarlo e scegliere la voce **object** properties dal menu **object** a livello autore. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Clip

E' uno degli oggetti multimediali disponibili in Multimedia Toolbook 3.0. I clips si creano con un pulsante posto sulla barra dei menu. Di fatto, sono oggetti che servono a Toolbook a identificare, e quindi a rendere eseguibile, il file multimediale a cui necessariamente devono essere associati.

Combobox

E' un oggetto particolare, che consiste in un box al quale è associato un menu a tendina. L'utente può definire il testo che dovrà comparire sul box e il testo delle varie righe del menu a tendina. La reazione del combobox ad un'azione dell'utente è del tutto simile a quella di un campo di testo del tipo "casella di riepilogo". Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Curve

E' una linea ad andamento curvo. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Ellipse

E' un oggetto grafico circolare o ellittico. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Field

Ogni testo, in Toolbook, deve essere scritto su un campo (**field**) per poter essere visualizzato. Per scrivere nel campo basta cliccare due volte il mouse su di esso. Per cambiare gli attributi e le caratteristiche di un campo di testo selezionarlo e scegliere la voce **object** properties dal menu **object** a livello autore. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Group

Il gruppo è un oggetto composto di più oggetti, ognuno dei quali conserva le proprie caratteristiche e proprietà. Raggruppare gli oggetti facilita in particolare le operazioni di copia e di impaginazione.
Selezionare più oggetti trascinando il mouse su tutta l'area da essi occupata. Scegliere la voce **group** dal menu **object** a livello autore.

Hotword

La parola chiave è un oggetto caratteristico di tutti i linguaggi ipertestuali. Funziona come un pulsante di selezione a cui può essere associato un evento e può essere collegata con specifiche pagine del libro. Selezionare una parola o un gruppo di parole sul testo di un campo. Scegliere la voce **Create Hotword** dal menu **Text** a livello autore.

IrregularPolygon

E' un oggetto grafico poligona con lati irregolari. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse. Ogni volta che il mouse viene rilasciato la linea del perimetro riparte nella direzione desiderata. Per interrompere cliccare due volte il mouse.

Line

E' una linea retta. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E' uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Page

La pagina è uno degli elementi fondamentali del libro scritto in Toolbook. Su di essa si possono posizionare vari oggetti. Per cambiare gli attributi e le caratteristiche di una pagina scegliere page properties dal menu **object** a livello autore. Un libro Toolbook è composto almeno di una pagina. Per aggiungere una pagina ad uno sfondo scegliere **new Page** dal menu **Page**

PaintObject

Le immagini sono parte integrante della struttura multimediale del libro scritto in Toolbook.
Importare l'oggetto grafico scegliendo la voce **import graphic** dal menu **file**. Vengono riconosciute come oggetti *PaintObject* le immagini salvate in formato bitmapped (.bmp, .pcx ecc.)

Picture

Le immagini sono parte integrante della struttura multimediale del libro scritto in Toolbook. Importare l'oggetto grafico scegliendo **import graphic** dal menu **file**. Vengono riconosciute come oggetti *Picture* le immagini salvate in formato vettoriale (.wmf, .eps ecc.), oltre che le immagini incollate in Toolbook attraverso la *clipboard*.

Pie

E' un settore di cerchio o di ellisse. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Polygon

E' un poligono regolare, a tre o più lati (massimo 50). Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse. Per definire il numero dei lati di un Polygon prima di disegnarlo, richiamare la *palette* dei poligoni dall'apposita voce di menu.

RecordField

Il campo di record è simile ai campi di database. Il testo in esso contenuto è diverso in ogni pagina dello sfondo su cui è stato generato e può essere stampato o utilizzato per ricerche. Per cambiare gli attributi e le caratteristiche di un campo di record selezionarlo e scegliere la voce **object properties** dal menu **object** a livello autore. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse. Il campo di record può essere generato sultanto sullo sfondo del libro.

Rectangle

E' un poligono regolare a 4 lati. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse. Può essere molto utile anche come "pulsante" nascosto.

RoundedRectangle

E' un poligono regolare a 4 lati con angoli smussati. Può essere utile per disegnare oggetti complessi o per costruire e visualizzare grafici. E'uno strumento indispensabile nell'impaginazione e per la corretta impostazione grafica del libro. Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Stage

E' uno degli oggetti di Multimedia Toolbook 3.0. Definisce un'area dello schermo all'interno della quale può essere eseguito un **clip** (ad esempio un video). Selezionare il simbolo corrispondente sulla Toolbar e procedere trascinando il mouse.

Viewer

E' una finestra Windows all'interno della quale può essere visualizzata una pagina di un libro Toolbook. Anche la finestra principale di Toolbook (*MainWindow*) è un **viewer**. Per costruire un viewer premere sul pulsante con un'icona a forma di finestra nel box dei menu e scegliere **New**. Verrà mostrata una finestra di dialogo sulla quale potranno essere definite tutte le caratteristiche del **viewer**. Rispetto alla *Main Window* (e più in generale rispetto alla finestra da cui viene richiamato), un viewer può essere *child*, ovvero "interno", o *popup*: in quest'ultimo caso esso può essere trascinato anche al di fuori dello spazio occupato dalla finestra "madre".

GLI HANDLERS

NotifyAfter

Gestisce un messaggio che viene inviato ad un oggetto *dopo* che si è verificato un determinato evento. L'handler agisce in conseguenza dell'evento indicato, generalmente un evento del tipo *enter* o *leave*. Non si può usare questo handler sullo script di pagine, sfondi o libri.

```
NotifyAfter < evento >
 < comando >
end
```

NotifyBefore

Gestisce un messaggio che viene inviato ad un oggetto *prima* che si verifichi un determinato evento. L'handler agisce anticipando l'evento indicato, generalmente un evento del tipo *enter* o *leave*. Non si può usare questo handler sullo script di pagine, sfondi o libri.

NotifyBefore < evento >
 < comando >
end

To get

Definisce una procedura speciale per valutare il risultato di una funzione o per ottenere la definizione di una proprietà. E' obbligatorio indicare, dopo gli eventuali comandi, il valore di return della procedura, ovvero ciò che l' handler, quando viene attivato, deve calcolare o valutare.
Un handler **to get** può essere usato da qualunque altro handler richiamandolo con il suo nome.

to get < nome > < parametri >
 < comando >
 return < valore >
end

To handle

To handle è il gestore dei messaggi. Ogni serie di comandi deve essere inserita all'interno di un *handler*, la cui struttura è compresa tra l'istruzione *to handl*e, seguita dal nome del gestore del messaggio che si vuole associare ad un evento, e l'istruzione *end*.

to handle < gestore del messaggio >
 < comando >
end

To set

Definisce i parametri per il settaggio delle proprietà e degli attributi definiti dall'utente. Il parametro e il valore specificati verranno usati dagli altri handlers come variabili, e i relativi comandi verranno eseguiti quando un comando **set**, in un qualsiasi *handler*, definirà il settaggio della proprietà *nome* sulla base del parametro e del valore specificati in to set.

to set < nome > < parametri > to < valore >
  < comandi >
end

I MESSAGGI

Author

Porta Toolbook a livello autore. Corrisponde alla voce relativa nel menu **Edit** e può essere attivato con il tasto **F3**. Non si può inviare questo messaggio ad un libro aperto in versione *Runtime*.

to handle ButtonClick
 send author
end

Back

Torna all'ultima pagina sfogliata in ordine cronologico. Corrisponde alla voce ultima nel menu
pagina.

to handle ButtonClick
 send back
end

BringCloser

Avvicina l'oggetto selezionato di uno strato, ovvero aumenta di 1 unità il valore del suo **layer**.

```
to handle ButtonClick
 select rectangle  “test”
 send BringCloser
end
```

BringToFront

Porta l'oggetto selezionato o che riceve il messaggio sullo strato più vicino della pagina o dello sfondo. Corrisponde alla voce **BringToFront** nel menu **object**.

```
to handle ButtonClick
 select rectangle  “test”
 send bringtofront
end
```

ButtonClick

Attiva il messaggio associato quando il tasto sinistro del mouse è stato premuto e successivamente rilasciato.

```
to handle ButtonClick
 request  “Messaggio ricevuto”
end
```

ButtonDoubleClick

Attiva il comando associato quando il tasto sinistro del mouse viene premuto rapidamente due volte.

```
to handle ButtonDoubleClick
 request  “Messaggio ricevuto”
end
```

ButtonDown

Attiva il comando associato quando il tasto sinistro del mouse viene premuto.

```
to handle ButtonDown
 request  “Messaggio ricevuto !”
end
```

ButtonStillDown

Attiva il comando associato quando il tasto sinistro del mouse viene tenuto premuto per almeno 2 secondi.

```
to handle ButtonStillDown
 request  “Messaggio ricevuto ! “
end
```

ButtonUp

Attiva il comando associato quando il tasto sinistro del mouse viene rilasciato.

```
to handle ButtonUp
 request  “Messaggio ricevuto !”
end
```

Clear

Cancella l'oggetto selezionato. Corrisponde alla voce **Cancel** nel menu **Edit**.

```
to handle ButtonClick
 set focus to null
 select rectangle “test”
 send clear
end
```

Copy

Copia l'oggetto selezionato nella *clipboard* di Windows. Corrisponde alla voce **Copy** nel menu **Edit** .

```
to handle ButtonClick
 set focus to null
 select rectangle “test”
 send copy
 request “Il rettangolo è stato copiato nella clipboard”
end
```

Cut

Cancella l'oggetto selezionato dopo averlo copiato nella clipboard di Windows. Corrisponde alla voce **Cut** nel menu **Edit**.

```
to handle ButtonClick
 set focus to null
 select rectangle “test”
 send cut
end
```

DrawDirect

Rende l'oggetto selezionato visibile sullo strato superficiale disegnandolo direttamente sullo schermo. Corrisponde alla voce **DrawDirect** nel menu **Draw**.

```
to handle ButtonClick
 set focus to null
 select rectangle  “test “
 send drawdirect
end
```

EnterBackground

Attiva il comando associato al momento di entrare nello sfondo che incorpora l'istruzione.

```
to handle EnterBackground
< istruzioni o comandi >
end
```

EnterBook

Attiva i comandi associati al momento dell'apertura del libro. Eventuali parametri di apertura del libro (centratura, presenza delle barre dei menu ecc.) definiti in sede di programmazione devono essere inseriti nello script del libro all'interno dell' *handler* **EnterBook**.

```
 to handle EnterBook
 < istruzioni o comandi >
end
```

EnterButton

Attiva il comando associato nel momento in cui il *focus* passa all'oggetto che incorpora l'istruzione.

```
to handle enterButton
```

```
 request  “messaggio ricevuto !”
end
```
enterField

Attiva il comando associato nel momento in cui il *focus* passa all'oggetto che incorpora l'istruzione.

```
to handle Enter<b>Field</b>
 request   “messaggio ricevuto !”
end
```

enterPage

Attiva il comando associato al momento di entrare nella pagina che incorpora l'istruzione.

```
to handle enterPage
 request  "messaggio ricevuto !"
end
```

enterRecordField

Attiva il comando associato nel momento in cui il *focus* passa all'oggetto che incorpora l'istruzione.

```
to handle enterRecord<b>Field</b>
 request  “messaggio ricevuto !”
end
```

enterSystem

Attiva il comando associato *prima* che venga aperto il libro, al momento dell'avvio del sistema Toolbook gerarchicamente superiore a tutti gli altri oggetti.

```
to handle enterSystem
 < istruzioni o comandi >
end
```

Exit

Esce dal libro e dal programma chiedendo, a meno che non siano state date indicazioni contrarie, di salvare o meno le ultime modifiche effettuate. Corrisponde alla voce **Exit** nel menu **File**.

```
to handle ButtonClick
 send exit
end
```

FlipHorizontal

Rovescia l'oggetto selezionato in senso orizzontale. Corrisponde alla voce **FlipHorizontal** del menu **Draw**. Non agisce sull'orientamento dei testi.

```
to handle ButtonClick
 select group “test”
 send flipHorizontal
end
```

FlipVertical

Rovescia l'oggetto selezionato in senso verticale. Corrisponde alla voce **FlipVertical** nel menu **Draw**. Non agisce sull'orientamento dei testi.

```
to handle ButtonClick
 select group “test “
 send flipVertical
```

```
end
```

History

Mostra l'elenco delle pagine sfogliate in ordine cronologico. Corrisponde alla voce **History** nel menu **Page**.

```
to handle ButtonClick
 send history
end
```

Idle

Agisce in mancanza di altri eventi. Qualunque altro evento interrompe immediatamente l'esecuzione di un **idle**. Perchè l'handler corrispondente sia sempre attivo deve essere collocato su un piano gerarchico piuttosto alto, almeno al livello della pagina. I comandi associati ad un **idle** vengono ripetuti reiteratamente a meno che non intervengano altri eventi.

```
To handle idle
 < istruzioni o comandi >
end
```

Import

Apre una finestra di dialogo dalla quale è possibile selezionare un testo da importare nel libro. Corrisponde alla voce **Import** nel menu **File**. Il testo deve essere in formato ASCII. Il termine import può essere utilizzato come comando, associandolo al path e al nome specifico del testo da importare.

```
to handle ButtonClick
 send import
end
```

ImportGraphic

Apre una finestra di dialogo dalla quale è possibile selezionare un'immagine da importare nel libro. Corrisponde alla voce **ImportGraphic** nel menu **File**. L'immagine può essere sia bitmapped che vettoriale. Il termine **importGraphic** può essere utilizzato come comando, associandolo al *path* e al nome specifico dell'immagine da importare.

```
to handle ButtonClick
 send importGraphic
end
```

KeyChar

Attiva uno o più comandi associandoli ad un tasto. Il tasto deve essere specificato sulla base della tabella ANSI.

```
to handle keyChar key
 if key = 27
  request  “Messaggio ricevuto !”
 end
end
```

KeyDown

Attiva uno o più comandi associandoli ad un tasto. Il tasto deve essere specificato sulla base del nome risultante dalla comparazione con il numero corrispondente nella tabella ANSI.

```
to handle keyDown key
 if key = KeySubtract
   request  “Messaggio ricevuto !”
 end if
end
```

KeyUp

Attiva uno o più comandi associandoli ad un tasto. Il tasto deve essere specificato sulla base del nome risultante dalla comparazione con il numero corrispondente nella tabella ANSI.

```
to handle keyUp key
 if key = KeyAdd
  request  “Messaggio ricevuto !”
 end if
end
```

leaveBackground

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

leaveBook

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

leaveButton

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

leaveField

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

leavePage

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

leaveRecordField

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

leaveSystem

Si veda, in proposito, quanto riportato a proposito del reciproco messaggio di tipo *enter*...

Tutti i comandi di tipo *leave*... attivano il comando associato prima di lasciare l’oggetto che incorpora l'istruzione nel suo script.

```
to handle leaveBackground
 request   “Messaggio ricevuto !”
end
```

mouseEnter

Attiva il comando associato quando il cursore entra nel perimetro dell'oggetto che incorpora lo *script*.

```
to handle MouseEnter
 request   “Messaggio ricevuto !”
end
```

mouseLeave

Attiva il comando associato quando il cursore esce dal perimetro dell'oggetto che incorpora lo *script*.

```
to handle MouseLeave
 request  “Messaggio ricevuto !”
end
```

NewPage

Aggiunge una nuova pagina allo sfondo corrente. Corrisponde alla voce **NewPage** nel menu **Object**.

```
to handle ButtonClick
 send NewPage
end
```

Next

Va alla pagina successiva. Corrisponde alla voce **Next** nel menu **Page**.

```
to handle ButtonClick
 send next
end
```

Open

Apre un nuovo libro dopo aver chiuso quello aperto. Corrisponde alla voce **Open** nel menu **File**.

```
to handle ButtonClick
 send open
end
```

Paste

Incolla sulla pagina o sullo sfondo l'oggetto o gli oggetti che erano stati precedentemente copiati nella clipboard di Windows. Corrisponde alla voce **Paste** nel menu **Edit**.

```
to handle ButtonClick
 send paste
end
```

PrintPages

Apre la finestra di dialogo per l'avvio della stampa del libro in formato grafico. Corrisponde alla voce **PrintPages** nel menu **File**.

```
to handle ButtonClick
 send printPages
end
```

PrintReport

Apre la finestra di dialogo per l'avvio della stampa dei testi contenuti nei campi di record del libro. Corrisponde alla voce **PrintReport** nel menu **File**.

```
to handle ButtonClick
 send printReport
end
```

Reader

Passa al livello lettore di Toolbook, permettendo così di rendere attivi gli handlers e gli script. Corrisponde alla voce **Reader** nel menu **Edit** .

```
to handle ButtonClick
 send reader
end
```

RightButtonDoubleClick

Attiva il comando associato quando il tasto destro del mouse viene premuto rapidamente due volte.

```
to handle RightButtonDoubleClick
 request  “Messaggio ricevuto !”
end
```

RightButtonDown

Attiva il comando associato quando il tasto destro del mouse viene premuto.

```
to handle RightButtonDown
 request   “Messaggio ricevuto !”
end
```

RightButtonUp

Attiva il comando associato quando il tasto destro del mouse viene rilasciato.

```
to handle RightButtonUp
 request   “Messaggio ricevuto !”
end
```

RotateLeft

Ruota l'oggetto selezionato a sinistra per 90 gradi. Corrisponde alla voce **RotateLeft** nel menu **Draw**.

```
to handle ButtonClick
 set focus to null
 select group “test”
 send rotateLeft
end
```

RotateRight

Ruota l'oggetto selezionato a destra per 90 gradi. Corrisponde alla voce **RotateRight** nel menu **Draw**.

```
to handle ButtonClick
 set focus to null
 select group  “test”
 send rotateRight
end
```

Run

Apre un altro libro Toolbook senza chiudere quello già aperto. Corrisponde alla voce **Run** nel menu **File**.

```
to handle ButtonClick
 send run
end
```

SendFarther

Allontana l'oggetto selezionato di uno strato, ovvero diminuisce di 1 unità il valore del suo **layer**.

```
to handle ButtonClick
 select rectangle  “test”
 send SendFarther
end
```

SendToBack

Porta l'oggetto selezionato sullo strato più basso della pagina o dello sfondo (**layer** 1). Corrisponde alla voce **SendToBack** nel menu **Object**.

```
to handle ButtonClick
 select rectangle “test”
 send sendToBack
end
```

SizeToPage

Centra la finestra che racchiude il libro Toolbook aperto sullo schermo, ovvero porta la dimensione del **viewer** che racchiude la pagina alle dimensioni della pagina stessa. Corrisponde alla voce **SizeToPage** nel menu **Page** e al tasto **F11**. La centratura della pagina sullo schermo e la sua corrispondenza con il **viewer** possono comunque essere definite dal dialgo box delle proprietà del **book**.

```
to handle ButtonClick
 send sizetopage
end
```

Transparent

Attiva la proprietà di trasparenza dell'oggetto selezionato. Corrisponde alla voce **Transparent** nel menu **Draw**.

```
to handle ButtonClick
 set focus to null
 select field  “test”
 send transparent
end
```

I COMANDI

Activate menuItem

Rende attiva ed utilizzabile una voce di menu che è stata disattivata con il comando *Deactivate menuItem*.

```
handle EnterBook
 activate menuItem  “Antipasto” at  reader
end
```

Add menu

Aggiunge un menu a quelli predefiniti. E' possibile specificare la posizione del menu nella barra della finestra. Se essa non viene specificata il menu aggiunto verrà automaticamente posizionato all'ultimo posto. E'obbligatorio specificare il livello al quale si vuole aggiungere il menu.

```
to handle EnterBook
 add menu “Piatti del giorno” position 5 at reader
end
```

Add menuItem

Aggiunge una voce di menu ad un menu esistente o definito dall'utente. E' possibile specificare la posizione del menu nella tendina. Se essa non viene specificata il menu aggiunto verrà automaticamente posizionato all'ultimo posto. E' obbligatorio specificare il livello al quale si vuole aggiungere la voce di menu.

```
to handle EnterBook
 add menuItem  “Antipasto” at menu “Piatti del giorno” at reader
end
```

Ask

Mostra una finestra di sistema (predefinita, con caratteristiche Windows) sulla quale è possibile inserire dei dati. La finestra è formata da un testo definito nello *script*, dall'area di *input* e dai pulsanti OK e Annulla. Il dato inserito verrà automaticamente acquisito nella variabile *it*.

```
to handle ButtonClick
 ask “Scrivi quello che vuoi, purchè siano parole sensate”
 set the text of field “test” to it
end
```

Beep

Emette un beep per una durata specificata. La durata è il numero dei beep intesi come singole unità di emissione del suono da parte del computer.

```
to handle ButtonClick
 beep 100
end
```

Attenzione ! Se si vuole associare ad un evento un suono particolare (ad esempio, uno dei suoni disponibili in Windows) non si deve ricorrere a questo comando, ma ad una funzione di esecuzione di un file audio come **playsound( )**: se il computer su cui viene utilizzata l'applicazione è configurato in modo da associare un file audio ad un evento critico (questa possibilità fa parte delle opzioni previste dal "pannello di controllo" di Windows), il beep di Toolbook verrà associato a quello stesso evento, per cui il file audio verrà eseguito tante volte quanti sono i beep richiesti nello script. In questo caso è bene dare al beep il valore 1.

Break

Interrompe l'esecuzione di uno script al momento specificato o quando si verifica una determinata condizione.

```
to handle ButtonClick
 set the text of field “a” to OK
 set the text of field “b” to OK
 break
 set the text of field “c” to OK
end
```

Continue

Ripete un comando precedentemente specificato, e in particolare i comandi delle strutture di controllo **do / until**, **while** e **step**. Non può essere usato se non associato ad un altro comando.

```
to handle ButtonClick
  do
    beep 20
    continue do
  until it = 100
end
```

Deactivate menuItem

Rende inattiva e non utilizzabile una voce di menu che è stata attivata con il comando Activate menuItem. La voce di menu disattivata apparirà più chiara rispetto a quelle attive e non produrrà effetti se selezionata.

```
to handle EnterBook
 deactivate menuItem  “Antipasti” at reader
end
```

Decrement

Sottrae al contenuto di una variabile un valore specificato nello script. Il valore deve essere numerico, così come il contenuto della variabile sul quale agisce il decremento.

```
to handle ButtonClick
 get (the text of field “n”)
 decrement it by 5
 set the text of field “n” to it
end
```

Draw

Disegna un oggetto specificato nelle dimensioni definite dall'istruzione. Le dimensioni sono quelle comprese tra le coordinate x,y del vertice in alto a sinistra e quelle del vertice in basso a destra. Le caratteristiche dell'oggetto disegnato con il comando draw sono quelle previste per default dal sistema Toolbook. L'oggetto appena disegnato viene automaticamente selezionato.

```
to handle ButtonClick
 draw a button from 5376,3920 to 8008,4816
end
```

Extend Select

Estende la selezione di un oggetto ad un'altro o a più oggetti correttamente specificati. Il comando può essere usato solo in relazione al comando select. Non si possono selezionare pagine, sfondi o campi di record.

```
to handle ButtonClick
 select rectangle “a”
 extend select rectangle “b”
 extend select rectangle “c”
 set fillColor of selection to yellow
end
```

Flip

Sfoglia un numero specificato di pagine del libro a partire da quella corrente. Nello sfogliare le pagine con il comando flip non vengono prodotti effetti di tendina, dissolvenza o zoom.

```
 to handle ButtonClick
  flip 3
end

to handle ButtonClick
  flip all
end
```

Forward

Fa sì che Toolbook, dopo aver eseguito le istruzioni associate all'evento programmato, continui la ricerca di un eventuale gestore dell'evento lungo la gerarchia degli oggetti.

```
to handle buttonClick
 go to next page
 forward
end
```

FxDissolve

Va ad una pagina specifica aprendola con un effetto di dissolvenza incrociata. La pagina può essere specificata in base al suo numero o al suo nome o essere la successiva (next) o la precedente (previous) nel libro. Il comando **FxDissolve** accetta parametri di velocità (fast, medium o slow) non obbligatori. Gli effetti speciali nel cambio di

pagina possono essere impostati con il comando transition.

```
to handle ButtonClick
 fxDissolve slow to next page
end
```

FxWipe

Va ad una pagina specifica aprendola con un effetto a tendina verso la direzione specificata (right, left, top, bottom). La pagina può essere specificata in base al suo numero o al suo nome o essere la successiva (next) o la precedente (previous) nel libro. Il comando **FxWipe** accetta parametri di velocità (fast, medium o slow) non obbligatori. Gli effetti speciali nel cambio di pagina possono essere impostati con il comando transition.

```
 to handle ButtonClick
  fxWipe top slow to next page
end
```

FxZoom

Va ad una pagina specifica aprendola con un effetto di zoom a partire dal centro. La pagina può essere specificata in base al suo numero o al suo nome o essere la successiva (next) o la precedente (previous) nel libro. Il comando **FxZoom** accetta parametri di velocità (fast, medium o slow) non obbligatori. Gli effetti speciali nel cambio di pagina possono essere impostati con il comando transition.

```
to handle ButtonClick
 fxZoom fast to previous page
end
```

Get

Valuta una qualsiasi espressione, un oggetto, un testo o il contenuto di una variabile e inserisce il valore risultante nella variabile it. E' uno dei comandi fondamentali per la programmazione in Toolbook.

```
to handle ButtonClick
 get the text of recordField "esempio di sintassi" of this page
 set the text of field “test” to it
end
```

Go

Va ad una pagina specifica direttamente e senza effetti di tendina o di dissolvenza. La pagina può essere specificata in base al suo numero o al suo nome. In quest'ultimo caso il nome deve essere virgolettato come una stringa definita. Con il comando go la pagina desiderata viene visualizzata sullo schermo ridisegnando tutti gli oggetti che contiene.

```
to handle ButtonClick
 go to page 96
end
```

Hide

Nasconde un oggetto. Il comando **hide** equivale all'istruzione *set visible of* < *oggetto* > *to false*. Un oggetto nascosto non può essere selezionato e non può ricevere messaggi dal mouse per l'esecuzione del proprio script.

```
to handle ButtonClick
 hide group “test”
end
```

Increment

Somma al contenuto di una variabile un valore specificato nello script. Il valore deve essere numerico, così come il contenuto della variabile sul quale agisce l'icremento.

```
to handle ButtonClick
 get (the text of field “n”)
 increment it by 5
 set the text of field “n” to it
end
```

Move

Muove un oggetto selezionato o specificato verso una destinazione precisa sullo schermo. La destinazione deve essere specificata in punti secondo le coordinate x,y desiderate. Lo spostamento porterà l'angolo alto a sinistra dell'oggetto sulla posizione specificata. E' il sistema più semplice per costruire una animazione.

```
to handle ButtonClick
 select group astronave
 move the selection to 7616, 2968
 move the selection to 7616, 2408
 move the selection to 7560, 1624
end ButtonClick
```

Pause

Effettua una pausa nell'esecuzione dello script per la durata specificata. La durata deve essere definita con un numero seguito dalla precisazione < seconds >. Se la precisazione viene omessa la pausa sarà del numero equivalente in ticks, ovvero in centesimi di secondo. Il comando pause accetta un valore massimo di 231 e non accetta valori negativi.

```
to handle ButtonClick
 show field “pausa”
 pause 5 seconds
 hide field “pausa”
end
```

Pop

Cattura un elemento, un numero o una stringa di una lista o di un contenitore e lo inserisce nella variabile **it**, a meno che non venga specificata una destinazione diversa aggiungendo al comando l' istruzione: *into* < *variabile di destinazione* >

```
to handle ButtonClick
 pop item 3 of the text of field “elenco”
end
```

Push

Aggiunge una voce, un numero o un qualsiasi valore (anche una proprietà di sistema) ad una lista o ad un contenitore. Il posizionamento è automatico (il primo posto nella lista), ma può essere definito nello script dopo il comando *onto* (es. *onto item 3*).

```
 to handle ButtonClick
 push "Eccetera" onto the text of field “elenco”
end
```

Put

Inserisce un numero o una stringa specificata o il contenuto di una variabile in una variabile, nel testo di un campo o in un contenitore (come il *commandWindow*). In un testo si può inserire una stringa prima o dopo (before, after) qualsiasi posizione specificata. Per inserire un valore in una variabile usare il comando *put* < *valore* > *into* <

variabile >

```
to handle ButtonClick
 put pausa after item 5 of the text of field “test”
end
```

Remove Menu

Elimina un menu da quelli predefiniti. E' obbligatorio specificare il livello dal quale si vuole rimuovere il menu.

```
to handle EnterBook
 remove Menu “Aiuto” at reader
end
```

Remove MenuItem

Elimina una voce di menu ad un menu esistente o definito dall'utente. E' obbligatorio specificare il livello dal quale si vuole eliminare la voce di menu.

```
to handle EnterBook
 remove menuItem “Esercitazione” at reader
end
```

Request

Attiva una finestra di request (predefinita, con caratteristiche Windows) nella quale può essere visualizzato un messaggio. Il comando accetta fino a tre pulsanti, definiti nell'handler, ognuno dei quali può essere associato ad una serie di comandi. Se i comandi non vengono specificati, premendoli si chiuderà la finestra. Se i pulsanti non vengono definiti, automaticamente, verrà proposto un pulsante OK.

```
to handle ButtonClick
 request  “Ho eseguito fedelmente il comando” \
 with “Non era necessario” or  “Va bene così” or “Non so”
end
```

Restore system

Ripristina il sistema Toolbook azzerando tutte le variabili e annullando tutte le opzioni definite dall'utente. In genere, si usa prima di chiudere un'applicazione per evitare che alcuni parametri, proprietà e modalità di esecuzione vengano associati ad altre sessioni di Toolbook.

```
to handle LeaveBook
 restore system
end
```

Return

E' il comando che in un handler to get definisce quale funzione deve essere attivata e che permette di ottenerne il risultato quando, in un qualsiasi *handler*, essa viene richiamata.

```
to get Lettere LettA, LettB
 return LettA & LettB
end
```

Search

Cerca una stringa di testo all'interno di un percorso specificato. La ricerca avviene su tutte le pagine, su tutti i testi compresi nei campi e nei campi di record, ma si possono aggiungere dei parametri (es. page o excluding background) per limitarne il raggio. Se la stringa richiesta esiste essa viene evidenziata e la ricerca si ferma.

```
to handle ButtonClick
 search page for “esiste”
```

```
end
```

Seed

Definisce il punto di partenza effettivo di una funzione numerica random. Il numero-radice può essere direttamente specificato (tra 0 e 32767) o può fare riferimento ad una proprietà di sistema, come l'orologio interno del computer, per aumentare il raggio del fattore di casualità, dopo aver definito il *formato* della proprietà specifica.

```
to handle ButtonClick
 set sysTimeFormat to seconds
 seed sysTime mod 14267
 get random(20000)
 set the text of field “n” to it
end
```

Select

Seleziona un determinato oggetto. Se di esso non viene specificata la collocazione il comando cercherà di selezionarlo sulla pagina corrente, e invierà un messaggio di errore in caso di tentativo fallito. Se l'oggetto si trova sullo sfondo occorre aggiungere il parametro < *oggetto* > *of this background*. Sfondi, pagine e campi di record non sono selezionabili attraverso questo comando.

```
to handle ButtonClick
 select rectangle “test”
 request “Ho selezionato il rettangolo. Ora diventerà giallo”
 set fillColor of selection to yellow
end
```

Select all

Seleziona tutti gli oggetti presenti all'interno dell'area specificata sulla pagina corrente. Se nell'area specificata non ci sono oggetti, Toolbook invierà un messaggio di errore. L'area definita è sempre quella compresa tra le coordinate x,y del punto in alto a sinistra e del punto in basso a destra. Sfondi, pagine e campi di record non sono selezionabili.

```
to handle ButtonClick
 select all from 4215, 2970 to 7815, 5700
 request “Ho selezionato tutti gli oggetti compresi nel mio spazio”
 set fillColor of selection to yellow
end
```

Send

Rende eseguibile un handler non legato ad un evento automatico o invia un messaggio/evento (es. un ButtonClick) ad un oggetto. Tutte le voci di menu possono essere attivate da uno script con il comando send. Un handler definito dall'utente (esclusi quelli di tipo to get e to set) verrà a sua volta attivato ed eseguito con lo stesso comando.

```
to handle ButtonClick
 send < messaggio >
end
```

Set

Cambia il valore di un testo, di un contenitore, di una variabile o di una proprietà. E'uno dei comandi fondamentali di Toolbook, poichè consente di modificare tutte le caratteristiche degli oggetti presenti in un libro.

```
to handle ButtonClick
 set the text of field “test” to the text of recordField  “esempio di sintassi” of this page
 set fillColor of field “test” to red
end
```

Show

Rende visibile un oggetto nascosto. Il comando **show** equivale all'istruzione *set visible of* < *oggetto* > *to true*. Ricordiamo che un oggetto non visibile non può essere selezionato e non può ricevere messaggi dal mouse per l'esecuzione del proprio *script*.

```
to handle ButtonClick
 show group “test”
end
```

Sort

Ordina le pagine specificate sulla base del parametro assegnato. Il parametro deve essere il contenuto di un campo di record. L'ordinamento può essere sia crescente che decrescente, sia numerico che alfabetico.

```
to handle ButtonClick
 sort pages 1 to 100 by ascending text text of recordField “comando espressione o messaggio”
end
```

Transition

Regola gli effetti di transizione in un cambio di pagina o gli effetti di transizione da una pagina ad un colore. Contrariamente a quanto accade negli altri comandi, l'effetto di transizione, compresi i suoi parametri, variabili a seconda del tipo di effetto scelto (velocità, direzione ecc.), viene definito e impostato subito dopo il comando come se fosse una stringa di testo.

```
to handle buttonClick
 transition "puzzle fast" to page 3
end
```

LE ESPRESSIONI

Costante

Le espressioni costanti sono elementi specifici del linguaggio cui vengono attribuite determinate proprietà e che non è possibile sostituire con un'espressione numerica o con una stringa. I nomi dei colori principali, ad esempio, sono delle costanti, e possono essere specificati soltanto indicando correttamente il termine speciale loro corrispondente.

```
to handle < messaggio >
 set < attributo > to black
end
```

Logica

L'espressione logica è la risposta alla richiesta di definizione di un parametro che ne esclude un altro. Le espressioni logiche sono quindi:
< *true* > quando si vuole che l'attributo impostato sia attivato.
< *false* > quando si vuole che l'attributo impostato non sia attivato.
Le espressioni logiche sono richieste nei settaggi delle proprietà di sistema.

```
to handle < messaggio >
 set < attributo > to true
end
```

Numero

L'espressione numerica è la risposta ad una richiesta di definizione di un parametro che richiede di essere impostato sulla base di un numero. Viene usata nei settaggi di molte proprietà e può essere costituita da un numero intero o decimale, positivo o negativo. Nei calcoli si usano sempre espressioni numeriche.

```
to handle < messaggio >
 set < attributo > to 5
end
```

Stringa

La stringa è la risposta ad una richiesta di definizione di un parametro che richiede di essere impostato sulla base di un testo definito. Tutti i testi sono composti di stringhe e il risultato della variabile it, a meno che non venga diversamente specificato, viene di fatto utilizzato come una stringa. La stringa deve essere definita tra due virgolette.

```
to handle < messaggio >
 set < attributo > to "prova"
end
```

LE VARIABILI

It

It è una variabile locale predefinita e prevista dal sistema Toolbook. Ogni volta che viene attivato un comando che prevede l'uso di una variabile (ad esempio get o push, request o ask) il risultato dell'esecuzione viene immagazzinato da Toolbook in questa variabile, il cui contenuto, in modo simile a ciò che avviene per la clipboard, è sempre aggiornato all'ultima operazione effettuata.
Il contenuto di **it** può essere analizzato, copiato, usato per confronti condizionali e manipolato con quasi tutti i comandi di Toolbook.

Local

Local è il comando che serve a definire una variabile locale. La caratteristica della variabile locale è l'uso limitato alle funzioni previste dallo script in cui essa viene definita. Tuttavia, a differenza di it, essa conserva il valore acquisito anche in presenza di altre operazioni sulle variabili nello stesso script. Ogni volta che Toolbook esegue un comando local azzera il contenuto della variabile relativa (o lo setta sulla base delle istruzioni ricevute). Per utilizzare il contenuto della variabile locale si fa riferimento al nome con il quale è stata definita.

```
to handle < messaggio >
 local < nome >
 set < nome > to < espressione >
end
```

System

System è il comando che serve a definire una variabile di sistema. La caratteristica della variabile di sistema è la capacità di immagazzinare e conservare dati dal momento della sua prima dichiarazione e definizione fino a che il libro non viene chiuso o non viene attivato un messaggio restore system. Il contenuto della variabile non sarà quindi continuamente aggiornato come negli altri casi, ma conserverà la memoria di tutte le operazioni effettuate, a meno che non intervenga ad azzerarlo un'istruzione affidata al comando **set**.

```
to handle < messaggio >
 system < nome >
 set < nome > to < espressione >
end
```

GLI ATTRIBUTI O PROPRIETA'

Activated

Definisce se un campo di testo o un campo di record possano essere modificati dall'utente o risultino "bloccati",

ovvero possano consentire l'attivazione di uno *script*. Il parametro relativo deve essere di tipo logico. La stessa proprietà può essere definita attraverso il box delle proprietà dell'oggetto selezionando o de-selezionando la voce *Activated(typing disabled)*.

```
to handle buttonClick
 set activated of field "A" to true
end
```

Bounds

Definisce la dimensione e la posizione di un oggetto sulla base delle coordinate x,y dei suoi vertici in alto a sinistra e in basso a destra. I bounds di un oggetto possono essere settati su quelli di un altro oggetto (es. set the bounds of < oggetto > to the bounds of < oggetto >). Anche un singolo elemento delle 4 coordinate specificate può essere settato (item < numero > of bounds of < oggetto >). Le coordinate delle dimensioni estreme di un **viewer** sono espresse in *pixel* e non in *PageUnits*.

```
to handle ButtonClick
 set bounds of < oggetto > to < parametro >
end
```

Caption

Definisce l'etichetta di un pulsante. Come valore di default, Toolbook assegna al pulsante l'etichetta pulsante o button ogni volta che esso viene disegnato. L'etichetta può essere omessa settando il parametro a < null >.

```
to handle ButtonClick
 set the caption of button “test” to  < nome >
end
```

CaptionPosition

Definisce la positione dell'etichetta di un pulsante rispetto agli oggetti grafici eventualmente inseriti in esso. Il parametro deve essere una costante (*left*, *top*, *auto* ecc.).

```
to handle ButtonClick
 set captionPosition of button “test” to  < parametro >
end
```

Char

Definisce il singolo carattere alfanumerico di un testo escludendo dal calcolo spazi e parti della punteggiatura. Il singolo carattere, definito dal numero corrispondente alla sua posizione, può essere letto o settato.

```
to handle ButtonClick
 ask  “ Scrivi un numero tra 1 e 100”
 get char it of the text of field  “test”
 request  “La lettera corrispondente è”   &&   it   &&   “.”
end
```

CurrentPage

Definisce la pagina che deve essere visualizzata in un **viewer** quando esso verrà aperto e mostrato. Il parametro è il nome completo della pagina, accompagnato dall'eventuale "indirizzo", nel caso che la pagina da richiamare si trovi su un altro *book*.

```
to handle buttonClick
 currentPage of viewer "prova" = page 4 of book "Firenze.tbk"
end
```

FillColor

Definisce il colore di fondo di un oggetto o di uno sfondo (non si può settare questa proprietà sulla pagina). Settare questa proprietà equivale a cambiare il colore dell'oggetto sulla palette a livello autore. Il parametro è una costante (es. red, yellow, cyan) per i colori di base o tre numeri percentuali separati da virgole (es. 10, 35, 85).

```
to handle ButtonClick
 set FillColor of < oggetto > to < parametro >
end
```

Focus

Definisce l'oggetto che sta ricevendo il messaggio in un determinato momento. Il focus è anche la proprietà che consente ai campi di testo e di record di ricevere un input e ai pulsanti di apparire con il caratteristico alone attorno all'etichetta. Prima di inviare il messaggio Paste è bene premettere l'istruzione < set focus to null >.

```
to handle ButtonClick
 set focus to < parametro >
 set focus of < oggetto > to < parametro >
end
```

FontFace

Definisce il tipo di carattere usato in un campo di testo, un campo di record o un pulsante. Settare questa proprietà equivale a cambiare il carattere del testo attivando l'apposita finestra a livello autore. Il parametro è una costante, ovvero il nome del carattere desiderato o < system > per il valore di default. Il carattere prescelto deve essere disponibile in Windows.

```
to handle ButtonClick
 set FontFace of the text of field "test" to < parametro >
end
```

FontSize

Definisce la grandezza (il corpo) del carattere di un campo di testo, un campo di record o un pulsante. Settare questa proprietà equivale a cambiare la grandezza del carattere attivando l'apposita finestra a livello autore. Il parametro è un valore numerico espresso in punti e compreso tra 1 e 72.

```
to handle ButtonClick
 set FontSize of the text of field "test" to < parametro >
end
```

FontStyle

Definisce lo stile del carattere usato in un campo di testo, un campo di record o un pulsante. Settare questa proprietà equivale a cambiare lo stile del carattere del testo attivando l'apposita finestra a
livello autore. Il parametro è una costante, ovvero il nome dello stile desiderato o < null > per il valore di default. Lo stile prescelto deve essere disponibile in Windows.

```
to handle ButtonClick
 set FontStyle of the text of field “test” to < parametro >
end
```

IdNumber

Definisce il numero di riconoscimento assegnato automaticamente da Toolbook a qualsiasi oggetto quando l'oggetto viene generato. Ogni oggetto mantiene inalterato il proprio idNumber, che in mancanza di una specificazione diversa costituisce il suo solo elemento di riconoscimento ai fini dell'invio di un messaggio o dell'effetto di un comando.

```
to handle ButtonClick
 get idNumber of self
```

```
 request “Il mio numero di id è”   && it &&   “.”
end
```

Invert

Definisce l'effetto positivo-negativo di una *hotword* di fronte ad un evento. Il parametro deve essere un valore logico. Il valore *true* implica che il testo della parola calda appaia in negativo rispetto al testo del campo.

```
to handle buttonClick
 set my invert to true
 show field "A"
 set my invert to false
end
```

IsOpen

Definisce se un **viewer** è correntemente "aperto" o no. Il parametro deve essere un valore logico.

```
to handle buttonClick
 if IsOpen of viewer "prova" is true
  close viewer "prova"
 else
 end
end
```

Item

Definisce una stringa di testo alfanumerica compresa tra due virgole in un elenco. Attenzione ! Il riferimento ad una parola del testo non è l'attributo item ma l'attributo word. Un item può essere letto o settato definendone il numero di posizione nell'elenco.

```
to handle ButtonClick
 ask    “Scrivi un numero tra 1 e 20”
 get item it of the text of field   “test”
 request  “La stringa corrispondente è”   && it &&  “.”
end
```

Layer

Definisce lo strato sul quale è posizionato un oggetto sulla pagina o sullo sfondo. Settare il layer di un oggetto equivale a scegliere voci di menu come Bring to Front, Bring Closer, Send Farther, Send Back a livello autore. Il settaggio del layer deve corrispondere ad un numero intero positivo.

```
to handle ButtonClick
 set layer of < oggetto > to < parametro >
end
```

LineStyle

Definisce lo spessore di una linea o del bordo di un oggetto grafico. Settare questa proprietà equivale ad agire sull'oggetto con la palette delle linee a livello autore. I parametri sono numeri interi compresi tra 0 e 8.

```
to handle ButtonClick
 set LineStyle of < oggetto > to < parametro >
end
```

Name

Definisce il nome proprio di un qualsiasi oggetto come elemento di riconoscimento primario insieme al numero id. Solo un oggetto che ha un nome definito può essere richiamato in uno script senza far riferimento al suo numero **id**.

```
to handle ButtonClick
 ask  “Dammi un nome”
 set the name of self to it
 get my name
 request   “Il mio nome è”  && it &&   “.”
end
```

ObjectFromPoint

Definisce il tipo di oggetto che si trova in corrispondenza della posizione specificata o della posizione del mouse e associa il nome della sua tipologia (es. button o rectangle) ad it. La proprietà è utile per limitare il numero degli script in un libro o per consentire ad oggetti diversi di compiere azioni diverse facendo riferimento al medesimo script.

```
to handle ButtonClick
 get ObjectFromPoint (sysMousePosition)
  if object of target is rectangle
   request  “Ciao. Sono io l'oggetto che cercavi !”
 end if
end
```

PageNumber

Individua il numero di una pagina, ovvero la sua posizione "fisica" all'interno del libro. La proprietà può essere "settata" attraverso un valore numerico.

```
To handle buttonClick
 get pageNumber of this page
 request it
end
```

Parent

Individua l'oggetto che nella gerarchia di Toolbook è immediatamente superiore ("genitore") all'oggetto su cui la proprietà viene richiamata.

```
to handle buttonClick
 hide parent
end
```

Pattern

Definisce l'aspetto del fondo di un oggetto o di uno sfondo. Settare questa proprietà equivale ad agire con l'apposita palette sull'oggetto selezionato a livello autore. I parametri sono un numero intero compreso tra 1 e 128.

```
to handle ButtonClick
 set pattern of < oggetto > to < parametro >
end
```

Position

Definisce la posizione di un oggetto in termini di coordinate x,y del suo vertice superiore sinistro.
Non si può leggere o settare la posizione di una pagina o di uno sfondo. 1 pollice equivale a circa 1440 punti, 1 centimetro a 568. Le coordinate della posizione di un **viewer** sono espresse in *pixel* e non in *PageUnits*.

```
to handle ButtonClick
 move rectangle  “test1”  to the position of rectangle   “test2”
end
```

SaveOnClose

Definisce il comportamento di Toolbook, al momento della chiusura di un libro, rispetto all'eventuale salvataggio

delle modifiche apportate al libro stesso. Oltre ai valori logici *true* e *false*, il parametro associabile a questa proprietà possono essere le costanti *system* (comportamento secondo il default) o *ask* (richiesta di conferma del salvataggio delle modifiche).

```
to handle LeaveBook
 set SaveOnClose of this book to false
end
```

Script

Definisce tutte le istruzioni e i comandi associati ad un qualsiasi oggetto e gestiti da un handler. Lo script può essere letto e settato dall'esterno come qualsiasi altro attributo. Tuttavia, non si può agire su uno script quando l'applicazione viene eseguita in Runtime.

```
to handle ButtonClick
 set the script of < oggetto > to < script completo >
end
```

SelectedText

Definisce il testo selezionato in un campo o in un campo di record. Il comando **get** seguito da **SelectedText** associa ad **it** il testo della linea selezionata su un campo del tipo "casella di selezione".

```
to handle ButtonClick
 get selectedText
 request  “Il testo selezionato è”  && it &&  “.”
end
```

SelectedTextLines

Definisce la linea di testo selezionata in un campo a caselle di selezione singole o multiple. Il comando **get** seguito da **SelectedTextLines** associa ad **it** il numero della linea selezionata e non il testo in essa contenuto.

```
to handle ButtonClick
 get selectedTextLines
 request  “La linea selezionata è”  && it &&  “.”
end
```

Selection

Definisce l'oggetto o gli oggetti selezionati con il comando select. Questa proprietà non può essere settata. La selezione è valida finchè non ne viene operata un'altra o non si verifica un cambio di target e di focus. Non possono essere selezionati lo sfondo, il libro o la parola chiave.

```
to handle ButtonClick
 select self
 hide selection
end
```

Self

Definisce l'oggetto il cui script è in corso di esecuzione. Questa proprietà non può essere settata.

```
to handle ButtonClick
 hide self
 pause 50
 show self
end
```

Spacing

Definisce la grandezza dell'interlinea nel testo di un campo o di un campo di record. Settare questa proprietà

equivale a scegliere l'interlinea dall'apposita finestra a livello autore. I parametri sono 1, 1.5 e 2.

```
to handle ButtonClick
 set Spacing of the text of field   “test”   to < parametro >
end
```

StrokeColor

Definisce il colore del bordo di un oggetto o di uno sfondo e quello di un testo o di un'etichetta (non si può settare questa proprietà sulla pagina). Settare questa proprietà equivale a cambiare il bordo dell'oggetto usando la palette a livello autore. Il parametro è una costante (es. red, yellow, cyan) per i colori di base.

```
to handle ButtonClick
 set strokeColor of < oggetto > to < parametro >
end
```

SysChangesDB

Definisce se debba essere o non essere visualizzata la finestra di controllo che al momento di chiudere un libro avverte di salvare eventuali modifiche. Se la proprietà è settata < false > la finestra non viene visualizzata e il comando exit agirà senza salvare eventuali modifiche apportate al libro. Per visualizzare la finestra il settaggio è < true >.

```
to handle ButtonClick
 set sysChangesDB to < parametro >
end
```

SysCursor

Definisce la forma e l'aspetto del puntatore mouse, in generale o durante l'esecuzione di un handler o di una parte di esso. Il cursore può assumere un numero definito di aspetti compreso tra 1 e 44.
Il valore normale (aspetto a freccia) è 1 o 2 ma può essere chiamato anche < default >.

```
to handle ButtonClick
 set sysCursor to < parametro >
end
```

SysHistory

Definisce il modo in cui viene visualizzata la finestra prevista dalla voce di menu History e sulla quale sono memorizzate le pagine sfogliate durante l'istanza di apertura del libro. I parametri possono essere < name > o < default >.

```
to handle ButtonClick
 set sysHistory to < parametro >
end
```

SysLockScreen

Definisce la possibilità di evitare che lo schermo venga ridisegnato continuamente durante l'esecuzione di uno script. Se i cambiamenti dello schermo non vengono visualizzati (settaggio < true >) l'intero processo avviene a livello di memoria random, guadagnando in velocità. Il settaggio < false > mostra tutti i cambiamenti intervenuti nell'esecuzione.

```
to handle ButtonClick
 set sysLockScreen to < parametro >
end
```

SysMousePosition

Definisce la posizione del puntatore mouse sullo schermo in termini di coordinate x,y.
Un oggetto può essere spostato facilmente sulla posizione del mouse associando il comando ad un evento quale il

ButtonClick. La posizione del cursore può essere settata. 1 pollice equivale a circa 1440 punti. 1 centimetro a circa 568.

```
to handle ButtonClick
 move button test to sysMousePosition
end
```

SysSuspend

Definisce la presenza o meno della finestra che visualizza il messaggio di errore durante l'interruzione dell'esecuzione di uno script. Se si vuole che la finestra venga visualizzata la proprietà deve essere settata < true >, altrimenti dovrà essere settata < false >.

```
to handle ButtonClick
 set sysSuspend to < parametro >
end
```

SysTime

Definisce la lettura del clock interno del computer dopo che è stata formattata settando la proprietà SysTimeFormat. Può essere utile nelle routines random o per collegare dei comandi all'orologio. Questa proprietà non può essere settata.

```
to handle ButtonClick
 get sysTime
 request   “Sono le ore”   && it &&    “.”
end
```

SysTimeFormat

Definisce il modo in cui dovrà essere formattato il tempo collegato al clock interno del computer prima di essere letto attraverso la proprietà SysTime. Formattare la visualizzazione del tempo è utile nelle routines random o nelle operazioni che richiedono un collegamento con il clock.

```
to handle ButtonClick
 set sysTimeFormat to < parametro >
end
```

Target

Definisce l'obbiettivo diretto del messaggio che viene eseguito. Non è possibile settare questa proprietà. Il comando get target è usato in uno script complesso per rendere più veloce e immediato l'indirizzamento dei messaggi contenuti nello script.

```
to handle ButtonClick
 get target
 request   “Il bersaglio è” && it &&    “.”
end
```

Text

Definisce il contenuto complessivo di un campo di testo (anche se è settato sulla proprietà caselle di selezione) o di un campo di record. La proprietà Text può essere settata con qualsiasi valore alfanumerico.

```
to handle ButtonClick
 ask    “Scrivi quello che vuoi”
 set the text of field   “test”  to it
end
```

TextAlignment

Definisce le modalità di allineamento del testo in un campo o in campo di record. Settare questa proprietà equivale

a scegliere la formattazione del testo da menu a livello autore. I parametri sono delle costanti come left o right.

```
to handle ButtonClick
 set textAlignment of the text of field   “test “  to < parametro >
end
```

TextLine

Definisce la stringa compresa tra due invii a capo (tasto enter o costante CRLF) durante l'input di un testo. Come tutte le parti del testo può essere letta o settata da programma. Se il campo di testo è di una sola linea o con a capo automatico l'unica TexLine disponibile sarà la numero 1. Il comando get TextLine inserisce nella variabile it il numero (e non il testo) della linea.

```
to handle ButtonClick
 ask   “Scrivi una frase”
 set textLine 2 of the text of field   “test”  to it
end
```

unLinkDLL

Rimuove l'associazione tra l'applicazione e una determinata libreria di funzioni (*DLL*) precedentemente aperta e "linkata" attraverso una struttura di controllo **linkDLL**.

```
to handle buttonClick
 unLinkDLL "mmsystem.dll"
end
```

Vertices

Definisce le coordinate di tutti i vertici di un oggetto attraverso una lista di numeri che, a due a due, individuano i singoli vertici partendo da quello in alto a sinistra. Le coordinate dei vertici di un **viewer** sono espresse in *pixel* e non in *PageUnits*.

```
to handle buttonClick
 get vertices of angledLine "A"
 request it
end
```

Word

Definisce una stringa di testo alfanumerica separata da uno spazio. Attenzione ! Il riferimento ad un termine di un elenco non è l'attributo *word* ma l'attributo *item*. Una parola può essere letta o settata facendo riferimento al numero corrispondente alla sua posizione all'interno di un testo.

```
to handle ButtonClick
 ask   “Scrivi un numero tra 1 e 40”
 get word it of the text of field   “test”
 request   “La parola corrispondente è”    && it &&  “ .”
end
```

LE STRUTTURE di CONTROLLO

Conditions ... when ... else

Stabilisce le condizioni in base alle quali un comando viene eseguito. Il numero delle condizioni when è teoricamente illimitato. L'istruzione else esegue i comandi associati se nessuna delle condizioni predefinite si è verificata. La struttura di questo ciclo condizionale deve sempre essere chiusa da un *end*.

```
to handle ButtonClick
 conditions
    when my name is "rosso"
      request "Sono il pulsante rosso"
    when my name is "verde"
      request "Sono il pulsante verde"
    else
     request  "Non sono nè rosso nè verde"
```

```
  end  Conditions
end ButtonClick
```

Do ... until

Ripete il comando specificato dopo la prima istruzione (**do**) fino a che (**until**) non si verifica una determinata condizione. Il comando continue consente a do di agire fino alla condizione data.

```
to handle ButtonClick
 do
  flip 1
  get the name of this page
 continue do
 until it = "Arrivo"
end ButtonClick
```

If ... else

Verifica la condizione in base alla quale un comando viene eseguito. Il verificarsi di una condizione esclude l'altra. L'istruzione else esegue i comandi associati se la condizione predefinita non si è verificata. La struttura di questo ciclo condizionale deve sempre essere chiusa da un end.

```
to handle ButtonClick
 ask "Scrivi una parola magica"
 if it is "Apriti sesamo"
  request "Era la parola giusta"
 else
  request "Non riuscirai ad entrare nella caverna"
 end if
end ButtonClick
```

In

Consente di agire all'interno di un **viewer** anche se l'invio del comando parte da un altro **viewer**. Il ciclo di istruzioni collegato alla struttura **in** può quindi consentire di attivare effetti o di effettuare controlli indipendentemente dalla pagina attiva. La struttura deve essere chiusa con un *end in*.

```
to handle buttonClick
 in viewer "scheda"
 transition "puzzle" to page 5 of book "schede.tbk"
 end in
end
```

LinkDLL

Permette di utilizzare le funzioni predefinite nelle cosidette librerie disponibili sia in Toolbook che in Windows. Qualsiasi comando che faccia riferimento alla funzione di una libreria deve essere preceduto gerarchicamente da un handler di collegamento tra il libro e la libreria di funzioni che si vuole utilizzare.

```
to handle EnterBook
 linkDLL < nome >
 < parametri >
 end linkDll
end EnterBook
```

Start spooler

Attiva lo spooler della stampante e lo chiude dopo aver eseguito tutte le operazioni di stampa. La struttura di controllo start / end spooler è necessaria perchè Toolbook usa il Print Manager di Windows per inviare i dati alla stampante, cedendogli parte delle sue funzioni in completo automatismo.

```
to handle ButtonClick
```

```
 start spooler
 < istruzioni e parametri >
 end spooler
end ButtonClick
```

Step

Ripete esattamente uno o una serie di comandi fino al limite definito, ma a differenza del ciclo while non verifica la presenza di condizioni date. Il ciclo step è molto utile per operazioni quali la messa a punto di un contatore e nelle animazioni. Un ciclo step non si può interrompere su base condizionale.

```
to handle ButtonClick
 step i from 1 to 100
  get i
  set the text of field test to it
 continue step
 end step
end ButtonClick
```

While

Ripete o esegue uno o più comandi finchè una determinata condizione predefinita si mantiene inalterata. A differenza del ciclo step il ciclo while si interrompe in modo condizionale. E' molto utile nelle animazioni e in generale in tutti gli script che cercano di effettuare operazioni di controllo su pagine, testi o valori.

```
to handle ButtonClick
select ellipse test
put 7616 into it
 while item 1 of position of selection <= it
   move selection by 50, 48
    continue while
 end while
end ButtonClick
```

LE FUNZIONI

&

Somma e associa il contenuto di due stringe senza separarle con uno spazio.

```
to handle ButtonClick
 get  “Buon”  &  “giorno”
 request   “Il risultato dell'operazione è:”   && it &&  “.”
end
```

&&

Somma e associa il contenuto di due stringhe separandole con uno spazio.

```
to handle ButtonClick
 get  “Buon”  &&  “giorno”
 request   “Il risultato dell'operazione è:”   && it &&  “.”
end
```

*****

Moltiplica due espressioni numeriche.

```
to handle ButtonClick
 get   456 * 654
```

```
  request   “Il risultato dell'operazione è:”   && it &&   “.”
end
```

+

Somma due espressioni numeriche.

```
to handle ButtonClick
 get 4563 + 1654
request   “Il risultato dell'operazione è:”   && it &&   “.”
end
```

-

Sottrae due espressioni numeriche.to handle ButtonClick

```
 get 3456 - 654
request   “Il risultato dell'operazione è:”   && it &&   “.”
end
```

/

Divide due espressioni numeriche.

```
to handle ButtonClick
 get 45643 / 654
request   “Il risultato dell'operazione è:”   && it &&   “.”
end
```

<

Valuta se l'espressione numerica successiva è minore della precedente.

```
to handle ButtonClick
 get random(10)
 if it < 5
  request   “Il numero è:”  && it &&  “.”
 end
end
```

< =

Valuta se l'espressione numerica successiva è minore o uguale alla precedente.

```
to handle ButtonClick
 get random(10)
 if it <= 5
    request   “Il numero è:”  && it &&  “.”
 end
end
```

< >

Valuta se l'espressione numerica successiva è diversa dalla precedente o se il contenuto di una stringa è diverso dal contenuto di un'altra stringa.

```
to handle ButtonClick
 get random(10)
 if it < > 5
    request   “Il numero è:”  && it &&  “.”
 end
end
```

=

Valuta se l'espressione numerica successiva è uguale alla precedente.

```
to handle ButtonClick
 get random(10)
 if it = 5
     request   “Il numero è:”  && it &&  “.”
 end
end
```

>

Valuta se l'espressione numerica successiva è maggiore della precedente.

```
to handle ButtonClick
 get random(10)
 if it > 5
    request   “Il numero è:”  && it &&  “.”
 end
end
```

> =

Valuta se l'espressione numerica successiva è maggiore o uguale alla precedente.

```
to handle ButtonClick
 get random(10)
 if it >= 5
   request   “Il numero è:”  && it &&  “.”
 end
end
```

Average

Calcola la media statistica esatta tra i numeri di una lista.

```
to handle ButtonClick
 get average(10,24,34,12,45,65)
  request   “ Il risultato dell'operazione è: “  && it &&   “.”
end
```

CharCount

Calcola il numero dei caratteri di cui è composto un testo escludendo gli spazi.

```
to handle ButtonClick
 get CharCount(the text of field “test”)
  request   “Nel testo ci sono”   && it && “lettere.”
end
```

cos

Calcola il coseno dell'angolo specificato nella funzione.

```
to handle ButtonClick
 get cos(75)
  request  “ Il risultato dell'operazione è:”  && it &&  “.”
end
```

Exp

Calcola il risultato della costante e (2.7182818) elevato alla potenza specificata nella funzione.

```
to handle ButtonClick
 get exp(10)
  request “Il risultato dell'operazione è:”  && it && “.”
end
```

ItemCount

Calcola il numero di elementi in una lista.

```
to handle ButtonClick
 get itemCount(the text of field test)
  request   “Nel testo ci sono”  && it && “elementi.”
end
```

Log

Calcola il logaritmo del numero specificato nell'espressione e nella base specificata.

```
to handle ButtonClick
 get log(10,3)
  request   “Il risultato dell'operazione è:”  && it && “.”
end
```

Max

Individua il numero massimo presente in una lista.

```
to handle ButtonClick
 get max(10,45,12,234,15,26,345,112,87)
  request  “Il numero più alto è:”  && it && “.”
end
```

Min

Individua il numero minimo presente in una lista.

```
to handle ButtonClick
 get min(10,45,12,234,15,26,345,112,87)
  request  “ Il numero più basso è: “ && it &&  “.”
end
```

PageCount

Calcola il numero delle pagine di cui è composto il libro.

```
to handle ButtonClick
  get PageCount of this book
  request  “Le pagine di questo libro sono”  && it && “.”
end
```

PlaySound

Richiama ed esegue un file audio di tipo .wav. Il nome e l'eventuale "indirizzo" del file costituiscono il parametro della funzione. Il file richiamato non dovrebbe essere più di 100 Kb.

```
to handle buttonClick
 get PlaySound("c:\windows\chord.wav")
end
```

sin

Calcola il seno dell'angolo specificato nella funzione.

```
to handle ButtonClick
 get sin(75)
  request  “Il risultato dell'operazione è:”  && it && “.”
end
```

tan

Calcola la tangente dell'angolo specificato nella funzione.to handle ButtonClick

```
 get tan(75)
  request  “Il risultato dell'operazione è:”  && it && “.”
end
```

TextFromPoint

Definisce il rapporto tra una data coordinata (ad esempio la posizione del cursore) e una precisa posizione in un campo di testo: il valore che la funzione è in grado di individuare consiste in due numeri, il primo dei quali esprime la linea di testo corrispondente alla coordinata data, il secondo il numero del carattere corrispondente alla stessa coordinata data su quella linea.

```
to handle buttonClick
 get TextFromPoint(sysMousePosition)
 request it
end
```

WordCount

Calcola il numero di parole presenti in un testo.

```
to handle ButtonClick
 get wordCount(the text of field  “test”)
  request   “Nel testo ci sono”  && it && “parole.”
end
```

GLOSSARIO

Allineamento
Uno degli attributi di un testo. L'allineamento può essere a sinistra, centrato, a destra e giustificato.
Questi indicano rispettivamente un testo con le righe spostate a sinistra, centrate rispetto al foglio, a destra o distribuite in modo da occupare omogeneamente i margini del foglio.

Animazione
Sequenza di immagini create direttamente al computer con specifici programmi (come Macromind Director) in grado di essere presentate all'interno di Toolbook dando la sensazione di un "cartone animato".

ASCII
Codice standard con il quale vengono trasformati simboli del nostro alfabeto in sequenze di cifre binarie (bit) per la loro rappresentazione all'interno del computer.

AVI
Uno degli standard usati da Windows per archiviare sequenze video.

Bit
Contrazione di "binary digit", cifra binaria. Il codice binario usa due soli simboli (0 e 1). Ciascun simbolo è detto appunto bit. In un computer le informazioni sono rappresentate come sequenze di bit.

Book
Un ipertesto creato con Toolbook. Esso ha di solito l'estensione TBK. Es.: Leonardo.tbk

Byte
Sequenza di 8 bit. Nel codice ASCII un carattere del nostro alfabeto viene rappresentato da un byte.

Campionamento (frequenza)
Si riferisce alla fase di acquisizione di un suono attraverso una scheda audio. Più è alta questa frequenza, maggiore sarà la qualità del suono registrato (ma maggiore sarà anche lo spazio occupato dal suono in memoria). Per i CD musicali tale frequenza è di solito di 44.1 Khz.

Campo
E' l'oggetto di Toolbook deputato a contenere testi. Tutti i testi presenti in un book sono contenuti in rispettivi campi.

Cartella
In Windows la cartella è un sinonimo di Directory.

CD audio
Compact Disc musicale, che di solito si ascolta con i normali lettori di Compact disc, non necessariamente collegati/collegabili ad un computer.

CD-ROM
Compact Disc - Read Only Memory. Compact Disc leggibile sono usando un lettore di CD collegato ad un computer, contenente dati (immagini, suoni) e informazioni consultabili attraverso programmi che in genere risiedono nello stesso CD. I CD-ROM per la loro grande capacità di immagazzinamento costituiscono il supporto ideale per la distribuzione di programmi e iperstesti multimediali, che non potrebbero trovare posto in un semplice dischetto.

Cliccare
Azione legata all'uso del mouse. Si "clicca" premendo il pulsante del mouse (in genere quello sinistro, per i mouse con più pulsanti) quando il cursore sullo schermo (di solito a forma di freccia) punta un particolare oggetto o disegno.

Copia-Incolla
Una delle funzioni standard dei sistemi multi-finestra. L'utente seleziona un oggetto (ovvero evidenzia del testo), lo copia e quindi può incollarlo (duplicarlo) in un'altra finestra o in un'altra parte della stessa finestra.

Digitalizzazione
Indica in generale la fase di trasformazione di una informazione (testi, immagini, suoni, sequenze video) in forma numerica trattabile dal computer. Con lo "scanner" (cfr) ad esempio, l'immagine di una foto viene trasformata in numeri, rappresentanti gli attributi di colore di ciascun punto dell'immagine. Questa versione numerica viene utilizzata dal computer per poter riprodurre l'immagine sullo schermo o sulla stampante.

Dimensione
Uno degli attributi di un testo. E' possibile aumentare o diminuire la grandezza di un carattere agendo sulla sua dimensione.

Directory
Come in un tradizionale archivio cartaceo è possibile raggruppare documenti in fascicoli o cartelle (es. bollette enel, sip, ecc.) allo stesso modo documenti e programmi possono essere archiviati su disco in "directory", in cartelle con le quali possiamo organizzare meglio i contenuti.

Dischetto
E' il supporto magnetico più comune per l'archiviazione di programmi e dati. Il formato standard è ormai il 3.5" (tre pollici e mezzo) con una capacità che va da un minimo di 720 KB (DS DD) ad un massimo di 1.44MB (DS HD).

Disco fisso
E' il disco contenuto all'interno del computer, di solito di grande capacità, etichettato da MS-DOS come C: Il sistema operativo è generalmente presente sul disco fisso insieme con i programmi e i documenti che si usano più spesso. I dischetti (cfr) per la loro bassa capacità vengono usati solo per contenere copie di riserva dei programmi e dei dati (distribuiti su più dischetti) mentre in genere si lavora direttamente dal disco fisso che è anche più veloce di un dischetto nel recuperare e registrare i dati.

Disco rigido
vedi Disco fisso

Doppio-clic
Cliccare (cfr) due volte consecutivamente su un oggetto. Questa azione comporta generalmente l'apertura dell'oggetto. Se è una cartella questa viene aperta mostrando il suo contenuto; se è un programma questi viene lanciato (eseguito).

Drive
E' il dispositivo contenuto nel computer nel quale inseriamo il dischetto. Il drive contiene parti meccaniche ed elettroniche per la lettura e scrittura dei dati dal dischetto.

Driver
Particolari programmi che gestiscono direttamente dei dispositivi hardware aggiunti al computer (ad es. scheda audio, scheda video, ecc.). Il sistema operativo (DOS Windows) piuttosto che indirizzarsi direttamente a tali dispositivi si riferisce al suo driver il quale provvede ad eseguire le operazioni richieste. In questo modo si rende il sistema operativo indipendente dall'hardware.

File
Sia i programmi che i dati (es. documenti, lettere, ecc.) sono registrati sul disco in forma di file (archivio). Ciascun file ha un nome (per MS-DOS massimo di 8 caratteri) ed una estensione (max. 3 caratteri) che di solito viene usata per indicare il tipo di file. Esempi di nomi di file: autoexec.bat, win.exe, lettera.doc

Font
Tipo di carattere. Esistono ormai migliaia di tipi di caratteri (times, helvetica, algerian, ecc.) con i quali è possibile scrivere un testo. La scelta di una font è spesso dettata da regole di strategie di comunicazione che conoscono molto bene i grafici pubblicitari. La font è uno degli attributi di formattazione di un testo (oltre a stile, dimensione, allineamento, interlinea)

Hardware
Indica tutti i componenti meccanici ed elettronici di un computer: il monitor, la tastiera, il mouse, ecc. costituiscono l'hardware.

Hard disk
vedi disco fisso

Hotspot
Zona di una videata sensibile al clic.

Hotword
E' un hotspot costituito da una parola presente sulla videata.

Ipermedia
Ipertesto multimediale. I nodi dell'ipertesto possono contenere immagini, suoni, animazioni, sequenze video oltre ai testi.

Ipertesto
Testo non lineare. E' costituito da una rete di documenti (nodi) in relazione tra loro tramite un insieme di legami (link).

Kbyte
Kilo Byte. Indica 1024 byte.

Link (legame)
In un ipertesto, il link collega un nodo ad uno o più nodi dell' ipertesto. La percorrenza del link è resa possibile generalmente usando hotspot.

Macintosh
Personal Computer della casa Apple, che per prima ha introdotto l'uso del mouse e l'adozione di un sistema operativo ad interfaccia grafica. Gran parte del software oggi disponibile per Windows, proviene dall'ambiente Macintosh.

Mbyte
Mega Byte. Indica 1.048.576 byte

Messaggi
In Toolbook gli eventi provocati dal mouse o di altro tipo vengono comunicati tramite rispettivi messaggi che possono essere intercettati e trattati.

Midi
Standard di comunicazione di informazioni musicali.

Mouse
Dispositivo per il puntamento e l'attivazione di oggetti presenti sullo schermo. Indispensabile per usare un sistema ad interfacciamento grafico.

MPEG
Formato di archiviazione di sequenze video che risolve in modo ottimale il problema della grande occupazione di spazio che queste sequenze richiedono.

MS-DOS
Microsoft Disc Operating System. E' il sistema operativo più diffuso per i PC IBM compatibili. Windows, fino alla versione 3.11, richiede la presenza di MS-DOS per funzionare. Il nuovo sistema operativo Windows 95 può invece farne a meno.

Multimedia
Integrazione di più codici comunicativi (suoni, immagini, animazioni, ecc.) fruibili in unico ambiente in modo interattivo.

Navigazione
Esplorazione dei contenuti di un ipertesto.

Nodo
Componente di un ipertesto, tipicamente una videata.

Oggetto
In Toolbook indica in generale un componente dell'ipertesto.Book, sfondo, pagina, pulsante, campo, ecc. sono esempi di oggetti con i quali si può comporre un ipertesto.

PC IBM compatibile
Personal Computer conforme allo standard imposto da IBM, generalmente consente l'utilizzo del sistema operativo MS-DOS e di conseguenza di tutti i programmi scritti per quel sistema operativo.

Pulsante
E' un tipo di hotspot. E' utilizzato per rappresentare esplicitamente un link di un ipertesto. L'utente clicca sul pulsante e il link viene attivato richiamando il nodo ad esso collegato.

Quicktime
Standard di archiviazione di sequenze video usato su computer Macintosh ma anche disponibile in Windows.

RAM
Random Access Memory. Indica la memoria interna del computer, quella dove vengono caricati programmi e dati pronti per essere eseguiti. Da non confondere con la memoria del disco fisso o dei dischetti. Per poter essere eseguito un programma deve essere trasferito dal disco alla memoria RAM. E' proprio quello che fa il sistema operativo quando gli si da il comando di lanciare un programma. Più e' capiente la memoria RAM più siamo capaci di far girare programmi complessi, specialmente quelli che fanno un pesante utilizzo di grafica.

Scanner
Dispositivo per la digitalizzazione (cfr.) di immagini residenti su supporti cartacei (foto, pagine di un libro, ecc.). Per poter trasferire tali immagini e archiviarle nel computer come file (cfr.) e' necessario "scannerizzarle". Questa fase assomiglia molto ad una fotocopia, solo che il risultato non e' una copia su carta ma una immagine digitale dell'originale.

Scansione
Detta anche "scannerizzazione". Digitalizzazione (cfr.) effettuata tramite lo Scanner (cfr.)

Scheda audio
Componente aggiuntivo dell'hardware di un PC che consente di collegare al computer sorgenti sonore (impianti stero, walkman, microfono, ecc.) che possono essere digitalizzate e salvate in file sonori (vedi waveaudio file). Consente inoltre di utilizzare lettori di CD. L'uscita audio del computer e' inoltre collegabile grazie alla scheda audio a della casse o ad un amplificatore.

Scheda video
Componente aggiuntivo dell'hardware di un PC che consente di collegare al computer sorgenti video (videocamera, videoregistratore, ecc.) che possono essere digitalizzate e salvate in file video (vedi AVI, MPEG, QuickTime). Tali schede consentono generalmente di catturare una sequenza video, ma non il viceversa (per riversare ad esempio su videocassetta le immagini del monitor), per la cui operazione è richiesta un altro tipo specifico dispositivo (detto "encoder").

Sfioramento
Passare con il cursore del mouse su un oggetto senza cliccare.

Sistema operativo
Componente software di un PC essenziale per il suo fuzionamento. Il sistema operativo più diffuso e' MS-DOS. Senza questo il computer non puo' assolutamente funzionare. E' grazie al sistema operativo che e' possibilie eseguire programmi, archiviare dati, organizzare i file, ecc. Windows e' impropriamente definito sistema operativo ad interfaccia grafica. In realta' e' un programma che gira grazie ad MS-DOS e che fornisce gli stessi servizi di un sistema operativo ma in modo piu' efficace per l'utente. Il sistema operativo del Macintosh e' invece un vero sistema operativo ad interfaccia grafica "nativa", senza richiedere cioe' livelli piu' bassi di software per funzionare. Altri sistemi ooperativi di questo tipo sono OS/2 Warp della IBM Windows '95 della Microsoft, che elimina l'ormai superato MS-DOS per PC.

Software
Questa parola sta ad indicare i programmi che il computer esegue per servire le nostre esigenze. Esistono programmi di base (tipicamente quelli che compongono il sistema operativo) e programmi applicativi (quali Word, Excel, Wordstar, ecc.). Il computer si comporta seguendo pedissequamente le istruzioni contenute nel programma che gli si e' detto di eseguire.

Stile
Uno degli attributi assegnabili ad un testo. Lo stile puo' essere ad esempio grassetto, sottolineato, corsivo, ecc.

SVGA
Scheda grafica (cfr.) in grado di visualizzare più colori e con risoluzione superiore rispetto alla normale VGA (cfr.)

VGA
Scheda grafica (cfr.) standard che visualizza 16 colori con una risoluzione di 640x480 pixel.

Video digitale
Sequenze video rappresentate in forma numerica (digitalizzate) conservate in un file, tipicamente contenuto in un CD o anche su un disco rigido se sufficientemente capiente.

Videodisco
Apparecchio per la lettura di dischi ottici (analoghi ai CD) ma con un diametro di 30 cm (come i vecchi LP), su cui possono essere registrate immagini in formato digitale ad alta qualità.

Waveaudio files
File di suoni digitalizzati tramite la scheda audio (cfr.). Tipicamente questi file hanno l'estensione .WAV

Windows
Applicazione di MS-DOS che fornisce gli stessi servizi di un sistema operativo ma offrendo una interfaccia grafica analoga a quella di un Macintosh.

BIBLIOGRAFIA ESSENZIALE

Sugli ipertesti in generale

D.CORCIONE, G. DI TONTO, *Dal testo all'ipertesto. Teoria, utilizzo e aree applicative*, Milano, Jackson, 1989.
A.CALVANI, *Dal libro stampato al libro multimediale*, Firenze, La Nuova Italia, 1990.
F.ANTINUCCI, *Sulla natura dell'ipertesto*, in "Golem", 3, 5, 1991, pp. 21-23.
F.SCAVETTA, *Le metamorfosi della scrittura. Dal testo all'ipertesto*, Firenze, La Nuova Italia, 1992.
J.BOLTER, *Lo spazio dello scrivere*, Milano, Vita e Pensiero, 1993.
G.LANDOW, *Ipertesto, il futuro della scrittura*, Bologna, Baskerville, 1993.
A.CALVANI, *Iperscuola. Tecnologia e futuro dell'educazione*, Padova, Muzzio, 1994.
D.CESARENI, *Gli ipertesti, cosa sono, a cosa servono*, Roma, Garamond, 1995.
B.M.VARISCO, *Alle radici dell'ipertestualità*, in *Costruire / decostruire significati. Micromondi, ipertesti e orizzonti formativi*, a cura di A.Calvani, Padova, CLEUP, 1995.

Sul problema dell'interfaccia e della multimedialità

T.NELSON, *Hypertext & Hypermedia*, Boston-London, Academic Press, 1990.
M.GIUSTINIANI, R.BONAZZI, *Comunicazione e multimedialità. Guida teorico-pratica alla realizzazione di sistemi multimediali efficaci*, Milano, Angeli, 1992.
R.MARAGLIANO, *Manuale di didattica multimediale*, Bari, Laterza, 1994.
M.CAMMARATA, *Appunti per un corso di comunicazione multimediale*, in "Microcomputer", 1994-1995.

Su Toolbook

L.TWAY, *Multimedialità. Come costruire un'applicazione*, Milano, Tecniche Nuove, 1993.
Uno dei pochissimi testi in italiano che illustra le caratteristiche e le funzionalità di Toolbook, prendendo in esame la versione 1.5.

INDICE

Pagine e cambi di pagina

Per stabilire la dimensione (page Size) delle pagine
Per aggiungere o copiare una nuova pagina
Per dare un nuovo numero d'ordine ad una pagina
Per "navigare" da una pagina all'altra
Per "bloccare" l'accesso ad una pagina

A proposito di oggetti

Identificazione degli oggetti
Come creare un nuovo oggetto
Come selezionare un oggetto
Come spostare un oggetto sullo schermo
Come modificare la dimensione di un oggetto
Come copiare gli oggetti
Come cancellare gli oggetti
Come raggruppare gli oggetti in un unico altro oggetto
Come richiamare le proprietà o attributi (properties) di un oggetto
Come modificare le proprietà o attributi (properties) di un oggetto
Come modificare gli script di un oggetto senza usare l'editor di script

Lavorare con i testi

Per selezionare e modificare il testo
Per spostare il cursore o cancellare un carattere in un campo
Come copiare un testo
Scelta del tipo, dello stile e della dimensione dei caratteri di un testo
Come formattare un campo di testo
Come cercare e/o cercare e cambiare un testo

Le immagini

Formati grafici
Il problema della palette e del trattamento delle immagini

Il viewer

Che cosa sono i "viewers"
Caratteristiche e proprietà dei "viewers"

Approfondimenti di II livello: strutture di controllo e variabili

Le variabili

Le variabili
Applicazioni delle variabili: pulsanti "intelligenti"

Le strutture di controllo

Le strutture di controllo
La struttura di controllo "if"
La struttura di controllo "Conditions"
La struttura di controllo iterativa "While...end while"
La struttura di controllo iterativa "do...until"
La struttura di controllo iterativa "step...end step"
Il pulsante per uscire

Approfondimenti di III livello: estensioni multimediali

La multimedialità con Toolbook 3.0

La funzione playsound( ) in Toolbook 3.0
Per controllare le periferiche multimediali usando MMSYSTEM.DLL
Linkare e dichiarare le funzioni DLL
Controllo degli errori nell'uso delle periferiche

Elenco delle illustrazioni: