di Raffaello De Masi

Object Logo versione 2.0

Ricordo che, quando cominciai ad interessarmi di microinformatica abbandonando i monumentali mainframe universitari i tempi erano ben diversi da quelli in cui ci si ritrova adesso; tempi oscuri e pionieristici in cui per forza di cose ognuno doveva arrangiarsi a procurarsi e prepararsi da sé le piccole cose di cui aveva bisogno.

Parte fondamentale della macchina era il linguaggio con cui era fornito, tipicamente il Basic, da cui la macchina stessa era inscindibile (tant'è che sistema operativo ed idioma erano sovente integrati e presenti su ROM). L'utente doveva essere un buon conoscitore della sua macchina e, come i gentleman driver di una volta agli inizi del novecento, capace di mettere le mani con competenza sulla sua macchina, alla bisogna.

Perciò conosceva i suoi tool (leggi linguaggi) a perfezione, e per macchine diffuse come il PET, l'Apple II Europlus, il TRS 80 (lo ricordate?), i linguaggi abbondavano; successivamente la cosa si estese anche alle scatoline più piccole, come Commodore 64, Sinclair vari, e così via, tanto che il linguaggio, vuoi per effettiva necessità, vuoi per una stupida moda dei tempi, in cui chi conosceva a malapena un po' di Basic amava definirsi «programmatore» (beatus monoculus in terra caecorum) divenne il pacchetto più diffuso e acquistato (o pirateggiato). Per il mio Spectrum, a quanto mi ricordo, possedevo una decina di linguaggi diversi tra cui addirittura tre Forth (allora mi interessavo di controllo di strumentazione per la SNIA, utilizzando un PDP/11, e mi venne la curiosità di vedere quanta differenza (non molta per la verità) esisteva tra le grosse e le piccole implementazioni) e quattro o cinque compilatori Pascal, oltre a cose strane, come un BPCL, un miniAda e così via. Il boom della moda del programmatore fu raggiunta, credo, tra gli anni '84-'85 ma la sempre maggiore diffusione sul mercato di professionismo efficiente nel campo delle applicazioni stand-alone fece finalmente giustizia di tanto ciarpame (ricordate alcune riviste dell'epoca piene zeppe di listati inviati dai lettori?) e la buona messe dei programmatori domenicali si ritirò in buon ordine, con i suoi listatoni in Basic e Pascal, facendo largo a software di qualità, soprattutto polivalente ed efficiente.

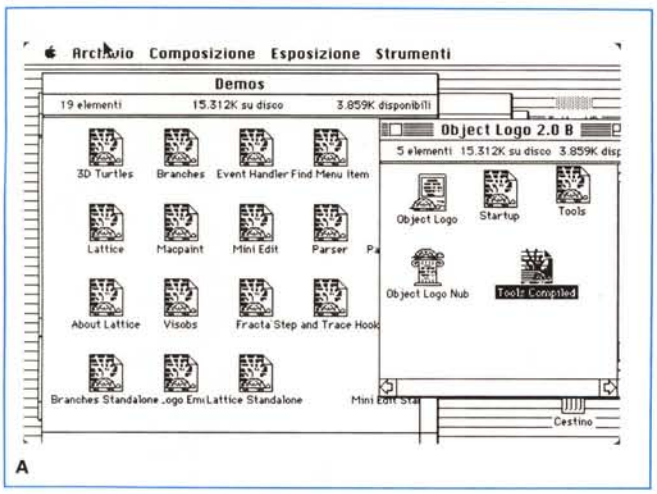
Ovviamente questo segnò il rapido declino della richiesta, da parte dell'utenza, dei linguaggi; per la spietata legge della domanda e dell'offerta la sparizione di questo genere di utenti impoverì l'offerta dei linguaggi sul mercato; case sorte come produttori di idiomi più o meno specializzati (in ogni caso si trattava quasi sempre di produttori dedicati alle scatolette di cui sopra) finirono miseramente nell'oblio; il linguaggio di programmazione non fu più il «business» dell'anno e poté essere supportato solo da case con solide strutture, capaci di affrontare senza problemi un investimento a lungo termine come quello di un idioma informatico. Ecco quindi restare padroni del mercato nomi come Borland, Microsoft, TLM, Zedcor, Atzec, Consulair, e pochi altri, capaci di produrre, aggiornare e supportare linguaggi a lungo termine, di eseguire periodici remake ed aggiornamenti e di curare il cliente nella maniera più adatta ed efficace.

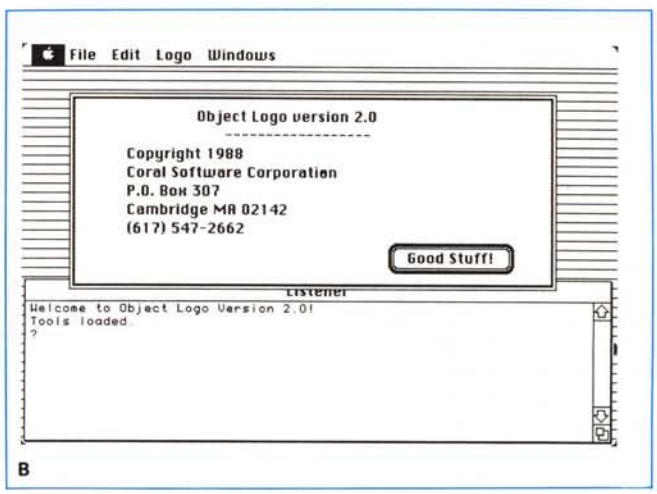
Il programmatore, d'altro canto, è divenuto sempre più smaliziato, da una parte, e dall'altra sempre meno disposto a trafficare con le locazioni di memoria; chiede al linguaggio routine sempre più potenti, efficienti, facili da usare, e sicure nel loro utilizzo.

Tutto ciò ha portato, come dicevamo, a linguaggi estremamente efficienti e sempre meno specializzati.

Questa della specializzazione, d'altro canto, è una storia che lascia abbondantemente il tempo che trova; non ho mai creduto, neppure dieci anni or sono, alla particolarità d'uso dei linguaggi; il problema è un altro, che detto in termini crudi si può esprimere in questi termini «Più il linguaggio è curato e di qualità, più cose può fare»; poiché sono sicuro che un buon Basic può essere, in un controllo di macchine, molto più efficiente di uno scalcinato Forth, sarebbe il caso di smetterla una volta per tutte con queste specializzazioni dei linguaggi, così come nessuno si sognerebbe di sostenere che il francese è più specializzato per la poesia o lo spagnolo per recitare la messa cantata.

Una dimostrazione di questo assunto è il pacchetto che analizziamo in questa puntata; Logo, per un non addetto ai lavori, significa, tout-court, giardino d'infanzia ed ABC dei programmatori in erba; mai come in questo caso si tratta di una ingiustizia delle più gravi e brucianti: Gary Bayers & C., autori di questo eccezionale linguaggio, non dormirebbero la notte pensando che un loro potenziale cliente ha tirato avanti guardando la parola «Logo» e pensando a giochetti di bambini.





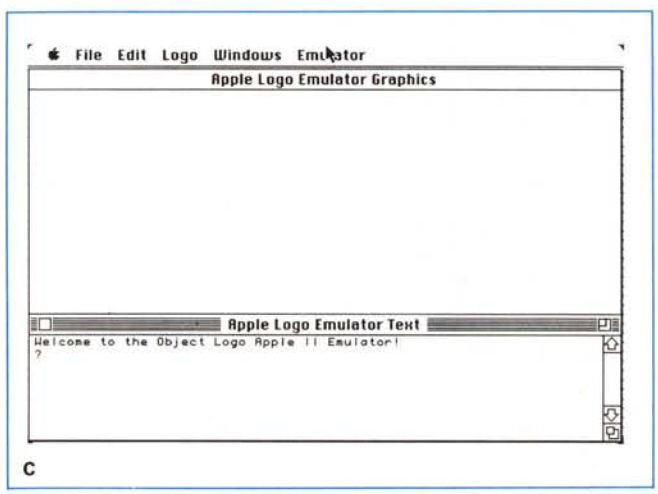


Figura A - Il contenuto dei dischetti.

Figura B - Il logo di apertura e presentazione.

Figura C - Le finestre principali di programma; superiormente quella di run del programma; l'inferiore è il listener, finestra interattiva per il debug e la visualizzazione delle primitive definite dall'utente.

Il pacchetto

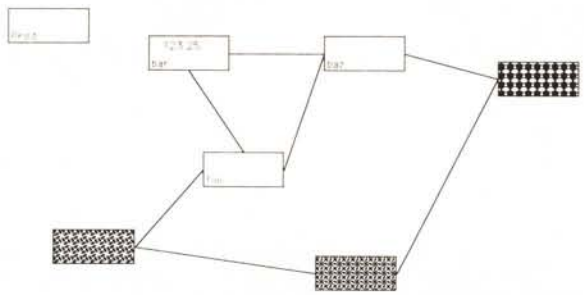
A livello di prestazioni e di potenza occorre immediatamente dire che Object Logo non teme confronti di alcun genere; indipendentemente dalla sintassi del linguaggio questo idioma non ha niente da invidiare ad alcun linguaggio titolato e, comunque, con questo pacchetto si possono produrre applicazioni degne di competere con quelle prodotte dal più blasonato Fortran, dal più sussiegoso Pascal o dal più facile ed interattivo Basic. Che gli autori non si siano fatti intimorire da alcuna falsa riverenza (che, oltre tutto, in un campo così aperto alla concorrenza è del tutto fuori luogo e falsa), lo dimostra la sbrigatività e l'essenzialità del manuale d'istruzioni che non spende neppure una parola in preamboli, presentazioni od altro. L'unica pagina di apertura (un mezzo foglietto) serve solo ad avvisare che il manuale, pesantissimo (più di 500 pagine) non è un tutorial; già in questo si dimostra la assoluta estraneità del linguaggio da tentativi di accattivarsi utenti alle prime armi; in altre parti lo stesso concetto viene riaffermato apertamente con frasi piuttosto esplicite, quasi a ricordare ogni tanto all'utente che ha tra le mani uno strumento di lavoro raffinato ed efficiente. Dopo di che, bando alle chiacchiere e si passa subito alla illustrazione delle caratteristiche del linguaggio.

Una introduzione informativa iniziale, cui è dedicato il primo capitolo, evidenzia e riassume la terminologia ed i formati usati nel manuale; tappa d'obbligo in tutti i trattati riferibili ai linguaggi, non si dilunga più di tre pagine. Immediatamente dopo si passa ad altri due capitoli introduttivi, ben più corposi; il primo dei due è dedicato alle finestre ed ai menu, peraltro, molto spogli in questo linguaggio; anche qui la maggior parte delle descrizioni si orienta, ovviamente, alla forma sintattica.

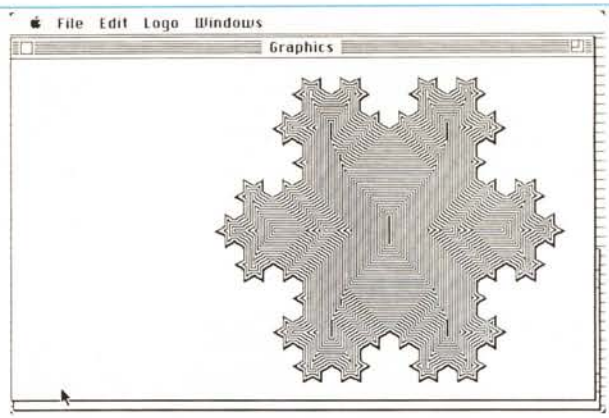
La introduzione III è la più interessante; insegna cioè il metodo per la migliore lettura del manuale stesso; innanzi tutto riferisce che le differenze tra Object Logo e i Logo più tradizionali vengono

Object Logo
versione 2.0

Coral Software
Corporation
P.O Box 307
Cambridge MA 02142
U.S.A.
Tel. (617) 547/2662



D1



D2

Figure D1, D2, D3 - Esempi di grafica, tra cui un frattale generato dal programma di cui vedete il listato.

evidenziati da un rigo in grigio sulla pagina; abbiamo provato a fare una statistica e, su cento pagine, oltre 75 mostrano questa simbologia (alla faccia della standardizzazione). In questo terzo capitolo vengono ulteriormente formalizzate alcune regole di comportamento e sintattiche circa l'uso di nomi, abbreviazioni, tipi procedurali, formati di linea, quotatura, tipologia e funzionalità degli operatori, variabili e loro tipologia, array, oggetti, strutture di controllo, ecc.

Da questo momento in poi si entra nel vero e proprio manuale operativo, da cui attingeremo a piene mani le parti migliori nella descrizione del pacchetto.

Tipi di dati

In Object Logo ci sono cinque tipi di dati: Word, Liste, Oggetti, Array e MacType. Una word consiste di una stringa di zero o più caratteri. Alcuni esempi di word sono Pippo, 42, 3, AN577 e così via. Le word possono essere di due sottotipi; numeri (word su possono agire operatori aritmetici) e simboli (word non numeriche).

Le prime due entità, word e liste, condividono numerose caratteristiche, essendo, alla fin fine, sequenze ordinate di elementi; per questo motivo hanno in comune un numeroso set di primitive (operatori di libreria).

Una word è una stringa di caratteri; ad esempio

hello

è una word contenente cinque caratteri. Allo stesso modo

an.577

ne contiene 6. Anche i numeri, come avevamo accennato precedentemente, sono word: 3.1416 è una word (6 caratteri). Ogni carattere in una word è consi-

```
;|
File: Fractal
Written in Object Logo
Last modified:  3/29/88

Copyright 1988 Coral Software Corp.
This program may be freely distributed.

This program runs in Version 2.0 (or later) of Object Logo.

This program draws an interesting fractal pattern that demonstrates the use
of turtle graphics and Quickdraw from Object Logo.

To load this file, select this window, choose 'Select All' from the Edit menu and
then 'Run Selection' from the Logo menu.  Or, you can load this file using
the Load primitive or the 'Load...' menu item in the Logo menu.

The 'Demo' procedure defined here draws a typical fractal pattern.  You can
also call Fractal.Pattern with your own inputs, for different patterns.
|;


; Draw a fractal of size :size with :level levels.
to fractal.pattern :level :inc
clearscreen
hideturtle
publicmake "window first turtlewindows
ask :window [setwsize [500 290]
_             wselect
_             startrgn]
clearscreen
fractal 260 :level
insetfractal ask :window [getrgn] :inc
end

to fractal :size :level
penup
back :size / 2
pendown
repeat 3 [fractal1 :size :level right 120]
end

to fractal1 :size :level
if :level = 0 [forward :size stop]
fractal1 :size / 3.0 :level - 1
left 60
fractal1 :size / 3.0 :level - 1
right 120
fractal1 :size / 3.0 :level - 1
left 60
fractal1 :size / 3.0 :level - 1
end

to insetfractal :rgn :inc
if emptyrgnp :rgn [stop]
ask :window [framergn :rgn]
insetfractal insetrgn :rgn :inc :inc :inc
end

; Sample invocation:
to demo
fractal.pattern 3 2
end
```

D3

Upgrade 2.0

Avevamo acquistato il pacchetto (vers. 1.3) circa 6 mesi or sono; abbiamo ricevuto, proprio alla fine della stesura di questo articolo (manca qualche giorno a Natale) l'upgrading alla versione 2.0; l'aggiornamento è stato gratuito (e non sollecitato); mi sono stati inviati i due dischetti che vedete nella foto d'apertura, ed un fascicolo aggiuntivo di un centinaio di pagine, oltre ad una serie di foglietti volanti evidenzianti le caratteristiche e le feature aggiunte all'ultima ora e non comprese nell'aggiornamento del manuale.

La versione 2.0 rappresenta un notevole passo avanti rispetto a quella precedente; l'ambiente Macintosh è stato notevolmente migliorato rispetto alla versione precedente.

È stato, innanzi tutto inserito un nuovo programma che emula integralmente l'ambiente di Apple Logo per Apple II; sebbene si tratti, alla fin fine, di una diminutio capitis, ciò ha la sua importanza se si considera che esistono sul mercato numerosi tutorial riservati a questo più anziano linguaggio, che, comunque, è referenziato abbondantemente in una dettagliata bibliografia.

L'interfaccia musicale è stata notevolmente migliorata, ed oggi supporta periferiche di tipo MIDI.

Aggiunte minori all'ambiente sono rappresentate da modifiche ai menu e dalla possibilità di aggiornare a piacimento la forma ed il profilo della «tartaruga».

Ancora è stato aggiunto un compilatore che, ovviamente, incrementa le prestazioni del linguaggio. Con esso è possibile creare delle applicazioni Mac del tipo stand-alone; in particolare il codice prodotto da queste operazioni risulta notevolmente compatto e senz'altro più ridotto di quello ottenibile con altri linguaggi, come Turbo Pascal o Atzec C (vero campione, quest'ultimo nella produzione di monumentali programmi oggetto).

Una quarantina di nuove primitive generiche, una ventina di primitive dedicate alla grafica e una completa possibilità di utilizzo di sintetizzatori di parola e di suono completano queste feature generali.

Chi programma Macintosh ad alto livello sa che una delle croci è il cosiddetto «Event Handling», vale a dire la possibilità del programma corrente di intercettare certe azioni dell'utente, come uso della tastiera o del mouse, schiacciamento di bottoni o srotolio di menu, ecc.

Anche qui esiste una notevole abbondanza di primitive, anche se il manuale è piuttosto sbrigativo nel trattarle, rimandando per ciò ad una serie di note del sempiterno Inside Macintosh.

Lo stesso vale per la gestione dei Trap e degli interrupt da parte dell'utente.

Driver e stream, e ancora una completa analisi del nuovo debugger (che tra l'altro consente una efficace operazione di Stepping e di Trace) completa l'enorme trattazione del nuovo linguaggio.

Si tratta di circa 400 nuovi operatori che, aggiunti ai seicento di cui si discute nell'articolo, portano ad un numero superiore a mille le primitive già esistenti nel programma; già così probabilmente, anzi sicuramente non esiste linguaggio in possesso di tale libreria; se aggiungiamo che il vero spirito di questo linguaggio è la customizzazione dei comandi ci si ritrova come Pizzarro di fronte alla foresta amazzonica o come un rapinatore davanti a Fort Knox.

E poi parlano del Logo da fare usare ai bambini delle elementari!

derato un elemento; le word sono individuate da limitatori, come spazi bianchi o RETURN, parentesi quadre, che hanno il compito di individuare una lista, oltre che, ovviamente di separare parti diverse; infine fungono da delimitatori segni particolari (come ad esempio gli operatori aritmetici) che forzano una separazione anche se non esiste materialmente un CR od uno spazio. I delimitatori sono, in Logo, 12 e sono:

( ) + - * / \ = < > ;

È possibile, al contrario, forzare l'unicità di una word; ad esempio il numero telefonico della Technimedia

06/4515524

sarebbe considerato come composto da 3 word; per rendere tutt'uno la parte viene utilizzato la barretta verticale: così per «unicizzare» il numero precedente avremo:

| 06/4515524 |

Le word che non sono numeri sono chiamate simboli. Le word simboliche possono essere utilizzate come variabili, procedure, funzioni. Ovviamente non esiste possibilità di utilizzare numeri assoluti per queste funzioni. Ai simboli ed alle liste è dedicato un pesante capitolo, che, nonostante la premessa iniziale, attraverso una lunga serie di esempi, introduce in maniera differenziata e particolareggiata a questo strano mondo, non comune in altri tipi di linguaggio. Il capitolo successivo è, per evoluzione di quello precedente, interamente dedicato all'analisi delle variabili, delle procedure, delle property list. Ampio spazio viene dato allo sviluppo delle variabili pubbliche; ma la vera potenza del linguaggio la si riconosce nelle procedure, mezzi eccezionali del linguaggio, e, in questo, molto più elastiche di quanto abbiamo mai visto in Pascal o delle funzioni del «C». Si tratta di mezzi estremamente efficienti, con possibilità di definire, nell'ambito della stessa procedura, input differenziati ed opzionali, e, ancora, con numero di input variabile nel corso del programma. Ogni primitiva, ancora, può essere ridefinita e, addirittura, la stessa procedura può essere analizzata da programma per verificare se essa è, in quel punto, accessibile o no.

La cosa più interessante, in questa parte del linguaggio, è rappresentata dalle cosiddette «property list», letteralmente liste di proprietà. Una property list può essere intesa come una variabile multipla che, invece di possedere un singolo valore, ne possiede una intera lista, ognuna associata con un suo nome. Per intendere ciò può essere utile un esempio; un record in una data base che raccolga, ammettiamo, generalità anagrafiche dei clienti (nome, cognome, data di nascita, residenza) è, a tutti gli effetti, una property list, riferita ad una variabile (record) singola. Le property list possono essere nidificate e, in ogni caso, differiscono dalle variabili per due motivi, non proprio ovvi. Il primo è che le property list non possono essere che globali, vale a dire non possono essere assegnate come argomento di una procedura, il secondo è che, anche se non inizializzate, esse sono già immediatamente definite all'atto della loro creazione, ancorché vuote.

Sempre a proposito di variabili, è opportuno spendere qualche ulteriore parola sulle array. La cosa, in OL, è particolarmente interessante in quanto, nel linguaggio, sono già contenute strutture simili alle array, come le liste. Esistono, però certe differenze tipologiche ed operative, che occorre considerare per utilizzare nella maniera più efficiente ambedue le strutture:

— una array è organizzata in memoria più efficientemente che una lista. I suoi elementi possono essere raggiunti e manipolati con maggiore efficienza e rapidità.

— Una primitiva che lavora su un'array è più efficace di quella che lavora su una lista; nel primo caso (anche se tutto ciò che diciamo ha scarso effetto sull'uso pratico) viene eseguita una vera e propria modifica dell'array, nel secondo la lista non viene effettivamente modificata.

— Un'array non ha, comunque, la flessibilità di una lista, poiché la lunghezza di un'array è predefinita.

Esistono altre minori differenze, di importanza ridotta, come ad esempio il limite inferiore di conteggio che nelle liste è 1 e nelle array è 0. Inoltre gli elementi di una lista vanno sempre dichiarati ed esplicitamente specificati; quelli di un'array, al contrario, devono solo essere definiti come numero e «riempiti» successivamente.

Object Logo e le operazioni numeriche

In linguaggi come i generici Logo presenti sul mercato, le possibilità nu-

meriche sono sempre trascurate in confronto ad altre possibilità. Object Logo, anche qui fa eccezione, ammettendo numeri in diversa precisione e, chi può affermare lo stesso, numeri complessi, definiti, secondo la filosofia del linguaggio, nel modo più semplice (vale a dire così come li scriveremmo con penna e carta). Viene rispettata la forma infissa degli operatori (il segno di operazione sta tra i due operandi) ed esistono più di cinquanta operatori diversi, dai più semplici analizzatori di numeri fino alle più complesse funzioni trigonometriche. Ma la cosa più interessante che ci è stata data di vedere in questa sezione è la presenza degli oggetti, che meritano una descrizione particolare.

Un oggetto (da cui proviene anche il nome del linguaggio), è qualcosa di particolare ed estremamente potente, presente finora solo in questo linguaggio; in breve esso è una collezione di procedure e variabili più una lista di procedure e variabili subordinate a queste più una lista, e così via. In altre parole un oggetto è un organismo organizzatore che colleziona procedure correlate e variabili concorrenti allo stesso fine. La potenza di tale meccanismo sta non solo nelle intrinseche potenzialità, ma in un meccanismo implicito negli oggetti stessi, chiamato «inheritance»; una grande parte del manuale è dedicata a queste potenzialità del linguaggio. In beve diremo che Object Logo possiede già oltre 600 primitive precostituite; per quanto appena detto, è possibile definire un numero illimitato di nuove. È ovvio che, in casi del genere, il problema della organizzazione di tale enorme numero di tool è immenso.

Ad esempio, se si desidera listare tutto quello che una turtle può fare, incluso, ad esempio primitive come «Forward» e «Setheading», come anche procedure definite dall'utente, come «Vai_a_destra» o «Cancella», i problemi di organizzazione delle informazioni ricevute soffriranno probabilmente di confusione. In Object Logo esiste una procedura precostituita, «Ask» che permette di individuare tutto quello che è associato ad una primitiva sia in fatto di procedure built-in, che di procedure costruite dall'utente. Naturalmente è possibile costruire oggetti a misura d'utente; ciò si ottiene attraverso il comando «KindOf». Un esempio, preso direttamente dal manuale, consente di costruire una «TinyTurtle», un oggetto che funziona come una Turtle ma a passi più piccoli; costruiremo, così, l'oggetto:

```
? Make "TinyTurtle KindOf Turtle
? Ask :TinyTurtle [To Forward :Dis]
> Usula.Forward :Dis / 5
> End
Forward defined
```

In pratica non esiste differenza tra la primitiva Turtle e l'oggetto TinyTurtle, tranne che per il fatto che le distanze operative della tartaruga (v. terzo rigo delle istruzioni) viene diviso per cinque.

Strutture di controllo, iterazioni, strutture condizionali. L'ambiente di sviluppo Object Logo dispone delle più efficienti e complesse strutture di controllo di-

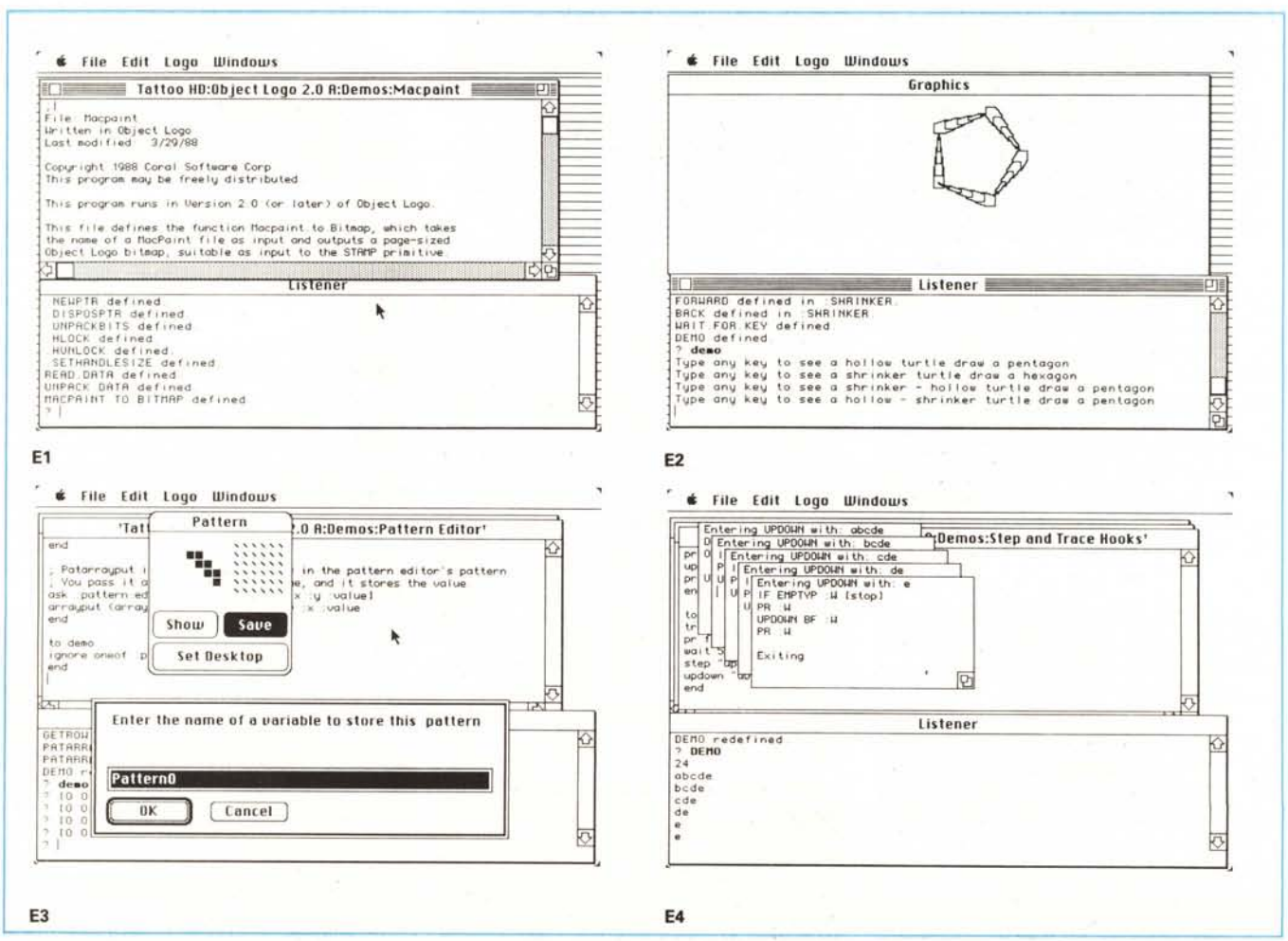


Figure E1, E2, E3, E4 - Esempi d'uso delle finestre di cui alla figura C.

sponibili negli altri linguaggi. Accanto a strutture come i solidi «If», «Repeat», «IfElse», «While», «DoUntil», ecc., compaiono strutture inusuali ed efficientissime come «Test», «Forever» (che ripete indefinitamente una istruzione), «Invoke» (che chiama una procedura con input differenziati), «IfTrue», «IfFalse», «CleanUp» (destinato a controllo degli interrupt direttamente da parte dell'utente).

Non mancano i soliti operatori logici, qui limitati solo all'AND, OR e NOT. Si tratta, come al solito, di una limitazione più formale che sostanziale, visto che la possibilità di costruire procedure ed oggetti consente di preparare, alla bisogna, tutto quello che non è presente.

Due grossi capitoli del manuale sono dedicati all'esame dell'ambiente di sviluppo Object Logo. Si tratta, probabilmente, della parte più interessante di tutto il linguaggio in quanto viene esaminato, in maniera formale e puntigliosa tutto il «workspace» di programmazione; a ciò si aggiunge l'esame di tutti i comandi destinati alle operazioni di I/O, di manipolazione delle memorie (si scopre tra le righe che la staticità delle array può essere superata brillantemente con un semplice artificio), di garbage collection, di esame degli spazi riservati alle parti di programma già predisposti, ad esempio esistono comandi che manipolano, esaminano e forniscono in output informazioni sul nome e sulla grandezza delle procedure, delle variabili, delle liste degli oggetti. Un grosso set di comandi è riservato alla manipolazione della memoria allocabile, degli heap, della memoria totale, di eventuali memorie tampone definibili, ecc.

Non manca ovviamente una estesa parte riservata alla manipolazione dei file, ma la cosa che davvero ci porta in ambiente Macintosh è quella che riguarda la manipolazione delle window (cap. 14), e una completa trattazione della sezione di QuickDraw. In quest'ultimo caso sono completamente accessibili tutte le routine del toolbox, rinunciando a tutte le macchinosità del Pascal per una più chiara e corretta gestione di tutti i comandi. Lo stesso si può dire della gestione dei menu (cap. 16) e di certe primitive particolari cui però il manuale fa espresso riferimento come pericolose da maneggiare. Termina il tutto una estesa ed articolata serie di appendici (ben nove) che rappresentano un luogo e un mezzo di riferimento estremamente facile da utilizzare.

Conclusioni

Object Logo, secondo me, ha un solo grande problema, il suo nome; molta gente, pensando al Logo sarà convinta di trovarsi davanti a poco più di un abbecedario per ragazzi, e rinuncerà ad approfondire questo linguaggio che (non ho mai usato questo vocabolo in tutti gli anni della mia collaborazione con MC) non ho tema di definire eccezionale. Sviluppato completamente in Lisp (anzi in Object Lisp della stessa casa), con solo una piccola routine in Assembler 68000 rappresenta un ambiente di sviluppo eccezionale, per la potenza dei mezzi a disposizione e per la congenita facilità di utilizzo del linguaggio stesso. Per un ammiratore del Basic e del «C» si tratta solo di una nuova affermazione che la semplicità non va necessariamente in disaccordo con la potenza e l'efficienza. Non mi meraviglierei pertanto che questo linguaggio divenisse tool di sviluppo ufficiale di qualche sofisticata software house; una sola cosa rimpiango: che con le capacità di chiarezza e potenza che la Coral è riuscita a portare in un linguaggio che finora era stato considerato sempre la Cenerentola dell'informatica, i progettisti di questo linguaggio non abbiano pensato a sviluppare qualche idioma ben più commerciale, come ad esempio proprio il Basic; non mi spiacerebbe vedere in circolazione qualcosa di quello implementato ad esempio sulle macchine HP; Basic, cioè di grande potenza ed efficienza, tanto per dimostrare, come abbiamo fatto adesso, che non è l'abito che fa il monaco!

Novità software

L'ultimo trimestre del 1988 è stato particolarmente largo di manica nel fornire nuovo software per il Macintosh: decine di pacchetti sono comparsi sul mercato dedicati alle più diverse specializzazioni; ne diamo qui un sunto molto generale, riservandoci di eseguire prove circostanziate appena saremo in possesso dei relativi pacchetti.

Disegno non geometrico

Oltre al Paint II, di cui abbiamo parlato in breve una paio di puntate or sono, e che ci riserviamo di analizzare insieme al blocco Draw II, Write II e Project, anch'esso II, Silicon Beach ha messo in circolazione il suo SuperPaint II. Dubl-Click Software presenta Wet-Paint, e Cricket Software ha finalmente realizzato una versione finale del suo Paint.

Circa la grafica a colori accanto all'attuale Pixel Paint compare Graphist Paint, prodotto dalla Aba e Photon Paint della Mediagenic (leggi Activision). Inoltre Letraset ha in progetto di lanciare la sua seconda versione di Image Studio.

Disegno geometrico

Accanto a Draw II ecco comparire la seconda versione di Canvas e quella, anch'essa seconda, di Draw it again, Sam, ambedue arricchite di nuove feature. Nel campo dei pacchetti di classe superiore (almeno per il prezzo), ecco comparire le remake di Illustrator (di Abode) e di FreeHand di Aldus. In particolare il primo pacchetto risulta notevolmente migliorato, tanto da aver superato il gap che lo separava dal cugino-rivale.

Software di presentazione (Presentation manager)

A distanza di qualche mese, ecco comparire la seconda versione di Power Point della Microsoft. Accanto a questo pacchetto dalle indubbie qualità ecco comparire il già ben testato Cricket Presents, Symantec di Living Videotext, e Manhattan Graphics di Ready-set-Go (alcuni di questi pacchetti sono realizzati anche per PC); come comprimari appaiono Management Graphics, Slidetek, MagiCorp (eccellente), Strade, VBS, Zenographics, ed altri a dimostrare la notevole vivacità del settore.

Image processing

Ancora sulla breccia Silicon Beach, col suo eccellente Digital Darkroom, dotato tra l'altro, di un driver dedicato a stampanti non PostScript. Quale concorrente troviamo ImageStudio 2.0, un po' più sotto di tono, ma che si differenzia per alcune caratteristiche che talora la rendono insostituibile, come la capacità di manipolare scale di grigio in parti selezionate dell'immagine.

CAD

Grosse novità in questo settore che finora, almeno in Italia, non riscuote il successo che merita; Paracomp, una compagnia di San Francisco, presenta due nuovi prodotti; ModelShop e Swiwel ambedue particolarmente versati nella modellazione tridimensionale. Ancora interessante appare Mac Architrion, destinato al mondo dell'architettura. Il sempre eccellente Super 3 D della Silicon Beach è adesso disponibile a colori.

Data Base

Grande attesa per il più volte annunciato (e mai visto) MS File 2.0; altrettanta attesa anima la nuova versione di 4th Dimension e la nuova release di File Maker; annunciata, ma per il prossimo anno, la nuova release del terribile OverVue.

Il nuovo System

Siamo già in possesso del nuovo Sistema operativo che, probabilmente, è la più grossa remake mai operata da Apple sul suo package; ne parleremo diffusamente nel prossimo numero.