IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER



ISTITUTO GEOGRAFICO DE AGOSTINI

IL MIO
COMPUTER

IL MIO COMPUTER

CORSO PRATICO DI INFORMATICA: MICRO, MINI E PERSONAL COMPUTER

volume quarto

ISTITUTO GEOGRAFICO DE AGOSTINI

Direzione: Achille Boroli - Adolfo Boroli
Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura di:
Edizioni Remo Sandron - Firenze
Redazione dell'edizione inglese:
Richard Pawson (*Editor*); David Tebbutt (*Consultant Editor*); Chris Cooper (*Executive Editor*); David Whelan (*Art Director*); Gareth Jefferson, Roger Ford (*Collaborators*)
Designed and Produced by Bunch Partworks Limited
Traduzione a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini

Titolo originale dell'opera: The home computer course
© Orbis Publishing Limited, Londra, 1983.
© Istituto Geografico De Agostini S.p.A., Novara, 1984.
Stampato in Italia - IGDA Officine Grafiche, Novara, 1985.

Sommario

721 Il Basicode
724 I file ad accesso casuale
726 Circuiti temporizzatori
728 Software in vetrina
729 Una gustosa albicocca
732 Che bestia è?
734 Alta e bassa risoluzione
737 Last In First Out
740 La Camputers
741 Divoratori di parole
744 Sottomarini ed esplosioni
746 A spasso per la memoria
748 Software in vetrina
749 L'Atmos, della Oric
752 Posizioni... chiave
754 I suoni del computer
755 Un gioco subacqueo
756 Somme e sottrazioni
760 La Quicksilva
761 Database e computer
764 Sprite... subacquei
766 Il quadrato magico
768 Software in vetrina
769 Le stampanti/plotter
772 L'unità logica
774 I file su cassette
776 Il Boogaboo
777 Moltiplicazioni e otazioni
780 La Olivetti
781 Scaccomatto al computer
784 La grafica su stampante
786 Gli Spreadsheet sono utili?
788 Software in vetrina
789 Il Colour Genie
792 Carovana del deserto
793 Il Minic Miner
794 Via ai sottomarini!
796 Le divisioni
800 La Dragon
801 Una rete di computer
804 Varietà di stampa
806 Corrispondenza automatizzata
808 Software in vetrina
809 TV contro monitor
812 Un mare di onde
814 I trucchi del mestiere
816 Twin Kingdom Valley
817 Pixel su pixel
820 La Bug-Byte
821 I computer... in volo
824 Prima impressione
826 Spectrum in rete
828 Software in vetrina
829 L'Advance 86a
832 Atterraggio lunare
834 Ci vuole stile...
836 Apocalypse
837 Gli sprite sullo Spectrum
840 La Motorola
841 Tecniche di marketing
844 Margherite per stampare
846 La tastiera dello Spectrum
848 Software in vetrina
849 L'unità Torch per BBC
852 Colori e numeri
854 Piano d'azione
856 Atic Atac
857 Sprite sul BBC
860 La LLamasoft
861 Programmare per vendere
864 Giocare via modem
866 Gli algoritmi
868 Software in vetrina
869 L'Adam
872 Grafica BBC ed Electron
875 Jet Pac
876 Finestre sullo Spectrum
879 Logica divertente
880 La Sharp
881 I "business games"
884 Campo visuale
886 La Brother EP-44
888 Software in vetrina
889 Dalla carta allo schermo
892 I programmi diagnostici
894 Cicli e iterazioni
895 Snooker
896 Piani inclinati
900 La Artic Computing
901 Il mondo in un computer
904 Mosse decisive
906 Il Prism VTX5000
908 Software in vetrina
909 L'Amstrad CPC 464
913 Il Sabre Wulf
914 Rotta di collisione
917 Prontezza di riflessi
918 Un cerchio di luce
920 La Texas Instruments
921 Il compressore di dati
924 Un bridgista d'eccezione
926 Scoppi e tuoni
928 Software in vetrina
929 L'interfaccia Acorn Plus 1
932 Cubi in rotazione
934 Programmazione modulare
936 Il Valhalla
938 I cerchi sul Commodore
940 La Audiogenic
941 Il computer ci controlla?
943 Un tocco di... tastiera
946 Spruzzi di colore
948 Software in vetrina
949 Il Machintosh
953 Il River Rescue
954 Percorsi circolari
956 Dai concetti alla pratica
958 Un attimo di tregua
960 La Prism

Indice per settore

Panorami



721 Il Basicode
728 Software in vetrina
741 Divoratori di parole
748 Software in vetrina
761 Database e computer
768 Software in vetrina
781 Scaccomatto al computer
788 Software in vetrina
801 Una rete di computer
808 Software in vetrina
821 I computer... in volo
828 Software in vetrina
841 Tecniche di marketing
848 Software in vetrina
864 Giocare via modem
868 Software in vetrina
881 I "business games"
888 Software in vetrina
901 Il mondo in un computer
908 Software in vetrina
924 Un bridgista d'eccezione
928 Software in vetrina
941 Il computer ci controlla?
948 Software in vetrina

Software



724 I file ad accesso casuale
752 Posizioni... chiave
755 Un gioco subacqueo
774 I file su cassette
776 Il Boogaboo
786 Gli Spreadsheet sono utili?
792 Carovana del deserto
793 Il Minic Miner
806 Corrispondenza automatizzata
816 Twin Kingdom Valley
836 Apocalypse
856 Atic Atac
861 Programmare per vendere
875 Jet Pac
895 Snooker
913 Il Sabre Wulf
936 Il Valhalla
943 Un tocco di... tastiera
953 Il River Rescue

Le chiavi dell'informatica



726 Circuiti temporizzatori
746 A spasso per la memoria
772 L'unità logica

Hardware



729 Una gustosa albicocca
749 L'Atmos, della Oric
769 Le stampanti/plotter
784 La grafica su stampante
789 Il Colour Genie
804 Varietà di stampa
809 TV contro monitor
824 Prima impressione

829 L'Advance 86a
844 Margherite per stampare
849 L'unità Torch per BBC
869 L'Adam
886 La Brother EP-44
889 Dalla carta allo schermo
906 Il Prism VTX5000
909 L'Amstrad CPC 464
921 Il compressore di dati
929 L'interfaccia Acorn Plus 1
949 Il Machintosh

Programmazione BASIC

732 Che bestia è?
734 Alta e bassa risoluzione
744 Sottomarini ed esplosioni
754 I suoni del computer
764 Sprite...subacquei
766 Il quadrato magico
794 Via ai sottomarini!
812 Un mare di onde
826 Spectrum in rete
832 Atterraggio lunare
846 La tastiera dello Spectrum
852 Colori e numeri
872 Grafica BBC ed Electron
879 Logica divertente
884 Campo visuale
892 I programmi diagnostici
914 Rotta di collisione
917 Prontezza di riflessi
926 Scoppi e tuoni
932 Cubi in rotazione
946 Spruzzi di colore
954 Percorsi circolari

Linguaggio macchina

737 Last In First Out
756 Somme e sottrazioni
777 Moltiplicazioni e otazioni
796 Le divisioni
817 Pixel su pixel
837 Gli sprite sullo Spectrum
857 Sprite sul BBC
876 Finestre sullo Spectrum
896 Piani inclinati
918 Un cerchio di luce
938 I cerchi sul Commodore
958 Un attimo di tregua

I pionieri dell'Informatica

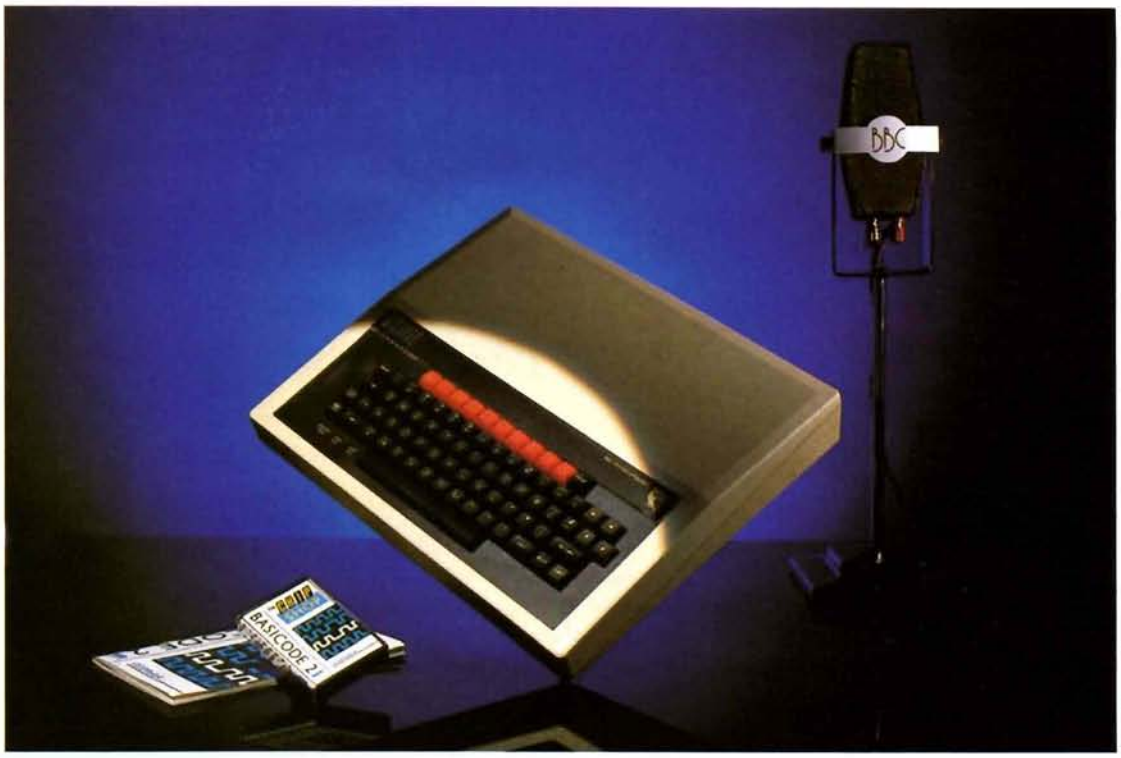
740 La Camputers
760 La Quicksilva
780 La Olivetti
800 La Dragon
820 La Bug-Byte
840 La Motorola
860 La LLamasoft
880 La Sharp
900 La Artic Computing
920 La Texas Instruments
940 La Audiogenic
960 La Prism

Tecniche di programmazione

956 Dai concetti alla pratica

Il BASICODE

Un sistema di codifica standard, per la trasmissione via etere dei programmi BASIC



IAN McKINNELL

Uno standard in comune

Il BASICODE, che consente ai micro di comunicare tra loro usando uno standard comune, possiede una serie ridotta di comandi BASIC ed un proprio formato di registrazione. Alcune stazioni radiofoniche trasmettono programmi BASICODE che, registrati dagli ascoltatori, possono poi essere caricati e fatti funzionare su apparecchi di diverse marche

Il BASIC è ormai divenuto il linguaggio standard degli home computer. Tuttavia, come ben sanno gli utenti di questi apparecchi, ci sono notevoli differenze tra i vari dialetti disponibili. Anche quando le macchine usano uno stesso dialetto, quale il Microsoft BASIC, ciò non garantisce che un programma, scritto per un particolare tipo di computer, debba necessariamente funzionare su un apparecchio differente.

Comunque, qualcosa sta cambiando anche in questo campo. Agli inizi del 1984, nel corso della trasmissione radiofonica "Chip Shop", la BBC ha iniziato a trasmettere programmi per computer via etere, in un dialetto BASIC chiamato BASICODE. Questi programmi sono stati registrati e fatti funzionare con successo su una grande varietà di computer. Il BASICODE rappresenta un nuovo modo di affrontare il problema della compatibilità. È stato sviluppato inizialmente nei Paesi Bassi per essere utilizzato nell'"Hobbyscoop", un progetto di carattere scientifico e tecnologico dalla Telac, l'equivalente olandese della Open University britannica. Quando, nel 1978, l'Hobbyscoop iniziò a trasmettere programmi, le trasmissioni erano destinate ai quattro computer allora più diffusi: l'Apple, l'Exidy Sorcerer, il Commodore PET ed il Tandy TRS-80. Ogni settimana veniva inviato, via etere, un programma per ciascuna macchina ma, poiché la velocità di trasferimento di due dei computer è estremamente bassa, gli ascoltatori erano costretti ad ascoltare fino ad otto minuti di trasmissione binaria, mettendo a dura prova la loro pazienza e il loro udito! Con la rapida comparsa sul mercato di nuovi apparecchi, ciascuno dei quali avrebbe richiesto una propria trasmissione, un simile sistema divenne chiaramente impossibile. Il primo che cercò di risolvere questo problema, fu Klaas Robers, un radioascoltatore appassionato d'informatica, che sviluppò la prima versione del BASICODE. Questo dialetto era basato su un comune sottogruppo di comandi BASIC, riconosciuti da tutti i tipi di computer. Il nuovo sistema causò qualche problema e il tentativo fallì perché, sebbene i comandi dei vari apparecchi fossero identici, diversa era la loro esecuzione. Klaas Robers, assieme a Jochem Erman, sviluppò dunque una versione perfezionata del linguaggio, che fu chiamata BASICODE-2.

I primi programmi radio del BASICODE-2 furono trasmessi il 1° gennaio 1983 e si dimostrarono un successo. Ascoltatori in Belgio, Francia, Gran Bretagna, Germania e Danimarca caricarono con successo i programmi trasmessi. L'attenzione crebbe ancor più quando la rete radiofonica olandese iniziò a trasmettere il BASICODE sulle gamme destinate all'estero.

Il BASICODE utilizza 42 parole chiave e 11 simboli comuni alla maggior parte delle macchine dotate del linguaggio BASIC. Le parole chiave non vengono immagazzinate sotto forma di caratteri, ma come codice (o "token") di un singolo byte (vedere a pag. 558). Ad esempio, sul Commodore 64 la parola LEFT$ viene codificata in un solo byte il cui valore è 200 e non da 5 byte contenenti i codici ASCII per L, E, F, T e $. Tuttavia, benché tutti i computer usino i token per immagazzinare ed interpretare un programma BASIC, ciascun apparecchio attribuisce ad essi valori diversi. Questo problema è stato risolto introducendo due programmi di conversione: il BASICODE-Save ed il BASICODE-Load. Pertanto, un programma scritto in BASIC viene memorizzato tramite il programma BASICODE-Save, che sostituisce i token del BASICODE a quelli del BASIC usato, producendo una versione su nastro in base allo standard BASICODE. A questo punto, il programma può essere riletto da un altro apparecchio, adoperando il BASICODE-Load, che riconverte i token nel formato adatto alla nuova macchina.

Sorge però un altro problema: come si può essere certi che i diversi tipi di computer leggano e scrivano sul nastro allo stesso modo? Infatti, sebbene tutte le macchine adottino gli stessi principi di caricamento e di "salvataggio" sul nastro dei programmi, le registrazioni effettuate da differenti apparecchi possono avere un formato molto diverso tra loro. Può variare la velocità di trasferimento sul nastro, ma anche i bit di start e di stop (ossia quei marcatori che segnalano al computer dove comincia e dove termina un'informazione) e le procedure di "checksum" (il sistema di verifica sulla correttezza dei dati trasmessi) potrebbero differire in modo radicale. Si è quindi deciso di ignorare le procedure individuali di gestione del nastro e di adottare un formato unico, detto "audiocode" (codice audio), per tali trasferimenti di dati.

In questo formato i dati vengono trasmessi a 1200 bit per secondo. Ciascun byte, preceduto dal bit di start (valore 0), viene trasmesso a partire dal bit meno significativo ed è seguito da due bit di stop (entrambi con valore 1). Ad esempio, il valore in codice ASCII di A è 65 (ossia 01000001 in binario): in codice audio verrà trasmesso come 01000001011. Una "coda" iniziale, composta da una serie di bit di stop trasmessi per 5 secondi, indica l'inizio di un programma BASICODE ed è seguita da un codice che indica "inizio del testo" (82 in esadecimale). Al programma BASICODE segue un byte di checksum, che consente al computer di verificare l'esattezza dei dati trasmessi. Una seconda sequenza di bit di stop, della durata di 5 secondi, indica la fine della trasmissione.

Il BASICODE si può adattare a quasi tutti i computer, senza intervenire sull'hardware. Fanno eccezione i modelli I e III del TRS-80 ed il Video Genie, per i quali occorre una piccola interfaccia in modo da poter leggere il nastro in maniera corretta. Il manuale del BASICODE-2, fornito assieme alla cassetta, dà tutte le istruzioni necessarie per costruire questa interfaccia. Per i meno portati al "fai da te", il "TRS-80 Users Group" olandese ha preparato un apposito circuito stampato.

Per scrivere un programma in dialetto BASICODE è necessario innanzi tutto caricare il programma BASICODE-Save. Questo non solo permette di salvare su cassetta, in formato standard, il programma BASIC appena redatto, ma fornisce anche un elenco di subroutine specifiche per il computer che stiamo usando. Queste subroutine possiedono numeri di linea compresi tra 0 e 999 ed occorre tener conto di ciò nello scrivere programmi che le utilizzino.

Le subroutine sono utili perché alcuni comandi comuni a molte macchine, come l'istruzione CLS per cancellare lo schermo, differiscono sostanzialmente a seconda dell'apparecchio usato. Così, invece di usare il normale comando CLS, il programmatore inserisce una GOSUB 100, che richiama la subroutine del BASICODE destinata a tale funzione.

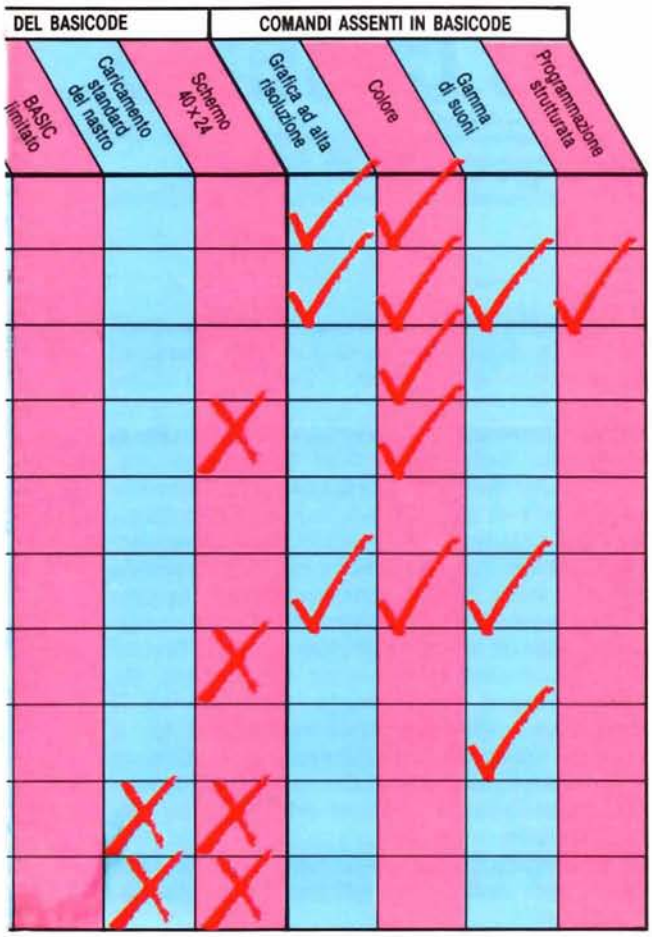
Questo prospetto dà un'idea dell'adattabilità di alcuni diffusi computer al formato BASICODE. Gli apparecchi che non appaiono all'altezza dello standard BASICODE sono indicati con una croce. Sono invece marcati con un "visto" i micro che presentano caratteristiche superiori a quelle richieste dal BASICODE



COMANDI OPERATIVI

Questa è una lista dei comandi e delle procedure previste dal BASICODE. Si noti che molti apparecchi possiedono, nel proprio BASIC, molti più comandi di quelli che il BASICODE è in grado di riconoscere

ABS	NEXT
AND	NOT
ASC	ON
ATN	OR
CHR$	PRINT
COS	READ
DATA	REM
DIM	RESTORE
END	RETURN
EXP	RIGHT$
FOR	RUN
GOSUB	SIGN
GOTO	SIN
IF	SQR
INPUT	STEP
INT	STOP
LEFT$	TAB
LEN	TAN
LET	THEN
LOG	TO
MID$	VAL
+	<
−	>
*	<>
/	<=
^	>=



La prima linea che il programmatore scrive dovrebbe avere il seguente formato:

```
1000 A=(valore): GOTO 20: REM nome del programma
```

dove (valore) indica il numero massimo di caratteri contenuti da tutte quante le stringhe. Da questo momento in poi, l'utente ha un'ampia libertà di programmazione. Tuttavia, esiste un certo numero di accorgimenti da osservare.
Le variabili, ad esempio, devono venire inizializzate prima di essere sottoposte ad una qualsiasi operazione, quindi, prima di un'istruzione del tipo LET T=T+1, la variabile T deve essere azzerata.
Altre restrizioni riguardano l'uso di numerose parole chiave BASIC. Per esempio:

```
5000 INPUT "NOMINATIVO?";A$
```

è un'istruzione vietata dal BASICODE-2. Il formato corretto di questa linea è:

```
5000 PRINT "NOMINATIVO?":INPUT A$
```

Inoltre, una linea di programma non può superare i 60 caratteri e si dà per scontato che lo schermo sia composto da 24 linee di 40 caratteri ciascuna.
Vediamo perché sono necessarie queste restrizioni. Per poter estendere il funzionamento del BASICODE al maggior numero possibile di apparecchi, è stato necessario riportare i vari stili di progettazione ad un unico denominatore comune. Inevitabilmente, è stato necessario un compromesso tra la sofisticatezza del BASICODE ed il gran numero di apparecchi in grado di usarlo. Ciò ha condotto ad un certo livellamento verso il basso, causato dagli apparecchi meno dotati tecnicamente, nei confronti dei modelli più avanzati. Per esempio, alcune delle caratteristiche richieste dagli utenti al momento dell'acquisto di un home computer non sono presenti nel formato BASICODE. Infatti, non è previsto un sistema per ottenere variazioni di tono e di durata dei suoni: il BASICODE riconosce ed esegue soltanto il primitivo comando BEEP. Similmente, l'unica grafica ottenibile è a bassa risoluzione e, per giunta, solo in bianco e nero.
È innegabile che, da quando il BASICODE è stato sviluppato, ci siano stati grandi progressi nelle tecniche di programmazione strutturata, ma ancora oggi il BASICODE non riconosce strutture del tipo WHILE...WEND o DEF FN. L'unica forma di strutturazione è rappresentata dal comando GOSUB.
Del resto, si pensi che molti degli apparecchi che adottano il BASICODE non sono neppure in grado di gestire i modesti standard forniti dal suo protocollo. Macchine come il VIC-20, lo ZX81, il TRS-80 ed il Video Genie hanno un display inferiore allo standard 40×24 del BASICODE.
Poiché le regole sono molto restrittive, il programmatore deve prestare molta attenzione per ottenere programmi trasferibili, limitarsi alle circa 50 parole chiave e operatori disponibili nel linguaggio ed usare il comando GOSUB in sostituzione di istruzioni non standard come CLS. Dovrà inoltre ricordare che alcune delle macchine provviste di BASICODE hanno limitate capacità di memoria e che i programmi molto lunghi, anche se scritti correttamente, possono superare tali capacità.
Utili indicazioni su particolari operazioni del programma, la cui esecuzione non è consentita dal BASICODE, possono essere inserite in linee di commento REM. Gli autori del BASICODE consigliano in questi casi l'uso delle linee comprese tra 20000 e 24999, cosicché specifiche funzioni per un particolare tipo di apparecchio usato possono essere aggiunte dopo che il programma è stato completato.
Tutte le istruzioni sull'uso del BASICODE-2 sono contenute in package, prodotto dalla BBC, da usarsi in combinazione con la serie di trasmissioni "Chip Shop": sul lato A la cassetta contiene i programmi di traduzione per i vari tipi di apparecchio. Alla maggior parte dei computer basta un solo programma di conversione per il BASICODE-2, mentre ai micro BBC ed al Vic-20 occorrono due programmi distinti LOAD e SAVE. Tutti i programmi della cassetta sono preceduti da messaggi che aiutano l'utente ad individuare il programma desiderato. Sulla facciata B sono registrati 18 programmi dimostrativi che permettono di farsi un'idea delle potenzialità del BASICODE.



IAN McKINNELL

Il BASICODE-2, corredato da un manuale e da una cassetta, può essere acquistato scrivendo alla:

Broadcasting Support Services
P.O. BOX 7
Londra
W3 6XY

I file ad accesso casuale

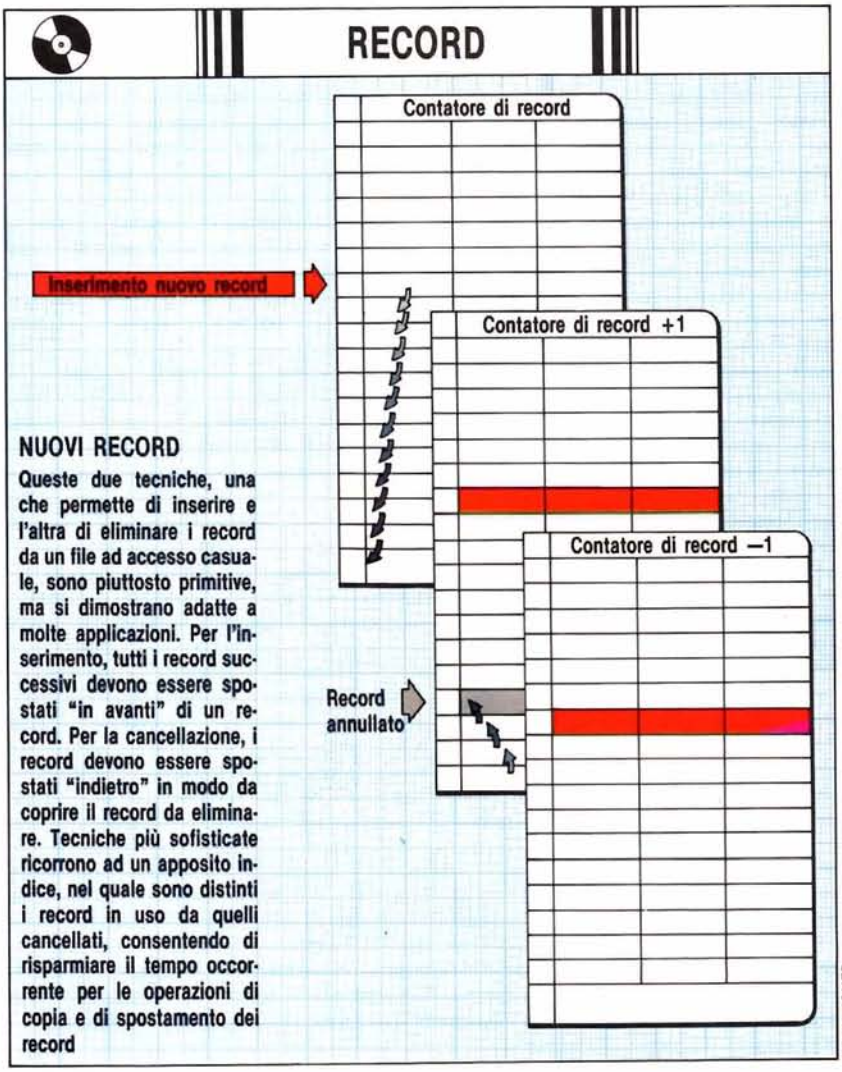
L'accesso ai dati è molto più rapido con questo tipo di file

Il principale limite dei file sequenziali è la necessità di dover sempre leggere in sequenza le informazioni memorizzate. Al contrario, i file ad accesso diretto o casuale (random) permettono di accedere ai record molto velocemente ed in qualsiasi ordine. (A scanso di equivoci, "casuale" non significa che il file sia strutturato in modo caotico, ma che ciascun record può essere scritto o letto senza dover esaminare tutti quelli precedenti).
Purtroppo i file registrati su cassetta non possono che essere sequenziali, poiché non è possibile posizionarsi su di un dato situato nella parte centrale del nastro magnetico, senza prima leggere tutta la parte di file che precede. In tal caso, l'unico modo per ottenere file ad accesso casuale sarebbe quello di caricare l'intero file in memoria, ma questa operazione è possibile solo con file di piccole dimensioni.
In pratica, i vantaggi dell'accesso casuale si possono avere soltanto disponendo di unità a dischetti.
I file ad accesso casuale implicano procedure di elaborazione molto più semplici di quelle necessarie per i file sequenziali e la loro suddivisione, in record e campi (vedere a pag. 706), riveste un'importanza determinante. Per accedere al file è necessario indicare il numero del record desiderato, che viene automaticamente ricopiato in un'area di transito (buffer) della memoria, dove i campi possono essere cancellati, corretti o inviati alla stampante.
Fortunatamente, è il sistema operativo che si fa carico delle operazioni più complesse, soprattutto quella di posizionarsi rapidamente all'inizio di un particolare record sul disco. Tale operazione è impossibile sui file sequenziali, perché il sistema deve necessariamente leggere tutti i dati, conteggiando ciascun codice separatore di campo. Nei file ad accesso casuale, la rapidità di ricerca viene favorita dall'identica dimensione dei record. Se tutti i record sono lunghi 100 byte ed il programma deve portarsi sul record 83, il sistema operativo posizionerà la testina di lettura sul byte 8300 del file, basandosi anche sul quantitativo di record contenuti in ciascun settore.
Questo modo di procedere risulta certamente più veloce dell'intera lettura di un file sequenziale.
Nel definire le dimensioni di un file, occorre tener conto della lunghezza del record più lungo che in esso verrà registrato: i record di dimensioni minori vengono automaticamente riempiti con spazi (32 in codice ASCII), anche se ciò significa un sensibile dispendio di memoria. I file ad accesso casuale vengono generalmente usati quando occorre elaborare velocemente piccole quantità d'informazioni, mentre quelli sequenziali se i dati sono molti e non è richiesta una grande rapidità d'accesso.
Anche la lunghezza dei campi che compongono un record dovrebbe essere costante. Nei sistemi dotati di funzioni per la ricerca casuale tanto dei campi che dei record, ciò è indispensabile, negli altri è soltanto questione di ordine e precisione. Il primo passo, nel fissare le dimensioni di un file random, consiste nell'elencare i vari campi e deciderne la lunghezza più adatta. Il campo che contiene il nome di una persona, ad esempio, potrebbe essere lungo venti caratteri, mentre per la registrazione dell'età sono sufficienti due caratteri.
L'organizzazione interna di un file deve sempre puntare alla massima economia di spazio, perché c'è un rapporto diretto tra quantità d'informazioni immagazzinate e numero di record impiegati. Conseguentemente, il sistema di codifica deve essere strutturato in modo da ridurre al minimo lo spazio occupato da ciascun dato: ad esempio, 1, 2 e 3 per i codici dei colori nero, rosso e verde oppure 841011 per la data 11 ottobre 1984. Sarà il

FILE A CONFRONTO

	FILE SEQUENZIALI	FILE AD ACCESSO CASUALE
VANTAGGI	• Rapido accesso ai record	• Risparmio di spazio • Uso delle cassette a nastro
SVANTAGGI	• Spreco di spazio • Necessità dei dischetti	• Lentezza e scomodità
APPLICAZIONI	• Dati con formato ben definito • Elaborazione di un piccolo numero di differenti record; ad esempio, in una libreria dove i clienti richiedono informazioni su particolari libri	• Gran numero di dati non strutturati • Quando moltissimi record in un file vengono elaborati durante un'unica esecuzione del programma; ad esempio, in un sistema di "gestione delle paghe"

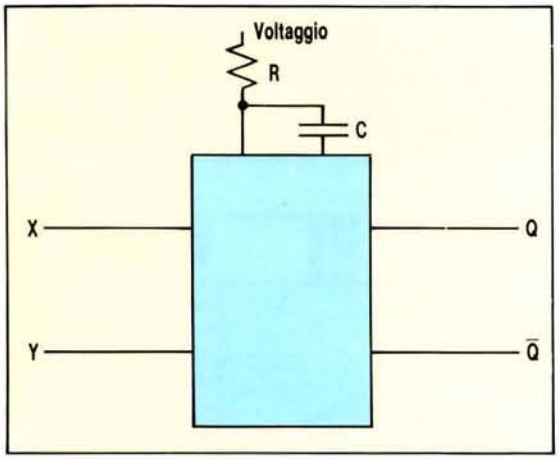
programma a convertire tale formato in qualcosa di più comprensibile al momento della visualizzazione dei dati.
Nel definire la lunghezza dei record, occorre tener presente altre due considerazioni. Moltissimi sistemi impongono un limite massimo alla lunghezza utilizzabile, che può variare tra 128 e 2048. Si otterranno risultati migliori adottando valori multipli della lunghezza del settore (i più comuni sono 64, 128, 256 e 512). In questo modo nessun record risulterà "a cavallo" tra due settori e si ridurrà anche il numero di accessi al disco, necessari per leggerlo.
Di solito, la gestione dei file ad accesso è più semplice di quella dei file sequenziali. In entrambi i sistemi è necessario tenere un conto aggiornato del numero di record presenti nel file.
Spesso, nei file ad accesso casuale, questo dato viene conservato nel primo record, assieme ad altre importanti informazioni (quali, ad esempio, la data di creazione dello stesso file). Ovviamente la struttura di questo record differirà da quella adottata per tutti gli altri.
Per variare il contenuto di un record è necessario leggerlo, modificarlo ed infine reinserirlo nella sua posizione originaria e per individuarlo si adopera il numero ad esso associato. Ovviamente, il programmatore non può ricordare a memoria tutti i numeri associati ai differenti record: ci sono quindi varie tecniche, simili a quelle usate in BASIC per le matrici, che permettono la loro ricerca e individuazione.
Talvolta, come chiave di accesso al file, viene usato uno degli stessi campi, ad esempio quello del nome. Un'apposita sezione del programma legge tutti i campi "chiave", creando un indice che permette di individuare il numero di record di ciascuna registrazione.
Senza un indice, i file ad accesso casuale vengono spesso letti record dopo record, come i file sequenziali. Se però i record vengono ordinati secondo un campo chiave, è possibile adottare metodi di ricerca veloce, quale, ad esempio, la "ricerca binaria" descritta a pag. 416.
In molti programmi, oltre a tecniche di ricerca molto sofisticate, si conservano in RAM i record più frequentemente consultati, rendendoli così rapidamente disponibili. Ciò consente un'efficiente gestione anche di file di grandi dimensioni.
Confrontata con la consultazione, la cancellazione e la registrazione di nuovi record può apparire relativamente lenta. Il metodo più semplice, per cancellare un record, è quello di scrivere su di esso il record che lo segue. Tutti i record seguenti verranno quindi trasferiti indietro di una posizione ed infine il numero dei record risulterà diminuito di uno.
In modo simile, per inserire un nuovo record, si dovrà spostare in avanti di una posizione l'ultimo record del file, quindi eseguire la stessa operazione su tutti i record che lo precedono, fino a quello immediatamente successivo al numero del nuovo record. Si libererà in questo modo uno spazio nella posizione voluta per la registrazione del nuovo record.
Queste tecniche non sono velocissime, ma certamente più efficienti del loro equivalente sui file sequenziali. Tuttavia, se abbiamo dotato il file di un indice, nel quale i record cancellati sono distinti da un particolare simbolo, allora cancellazioni e inserimenti saranno molto più rapidi. In questo caso, le informazioni contenute nel record restano inalterate: è solo l'indice a registrare l'operazione.
L'inserimento di nuovi record avviene consultando l'indice, alla ricerca del primo record vuoto o precedentemente usato ma poi cancellato. Al termine dell'operazione, l'indice viene aggiornato.
L'uso di indici, associati ai file random, è senz'altro conveniente. Per accedere ai record secondo un nuovo criterio logico, è sufficiente creare un nuovo indice. Inoltre, poiché l'annullamento di un record non provoca la sua effettiva eliminazione, ma solo una distinzione nell'indice, è possibile recuperare (almeno prima di un nuovo inserimento) le informazioni originali.



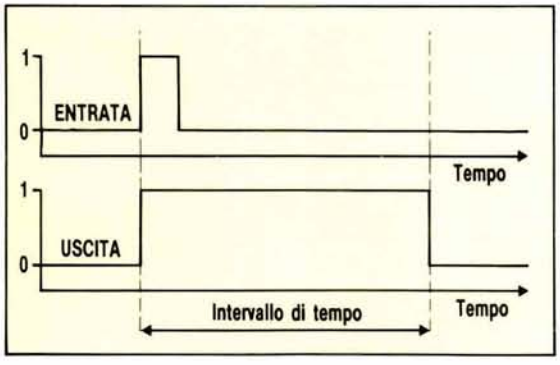
Circuiti temporizzatori

Il funzionamento dei circuiti monostabili, dei flip-flop tipo D e tipo J-K

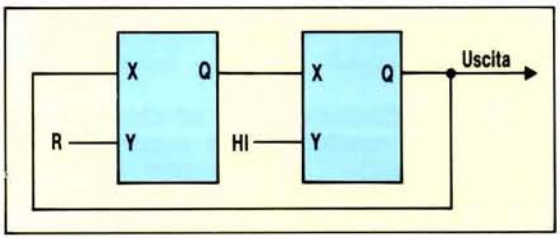
Un circuito "monostabile" permette di introdurre intervalli di tempo prefissati nelle operazioni eseguite da un circuito logico. Quando si applica un segnale ad un circuito monostabile, l'uscita viene posta a 1 (livello "alto" o HI) per un dato periodo di tempo, per poi tornare al normale valore 0 (livello "basso" o LO). La durata dell'intervallo di tempo viene determinato dai valori assegnati ad alcune componenti del circuito. Il seguente è un esempio di circuito monostabile:



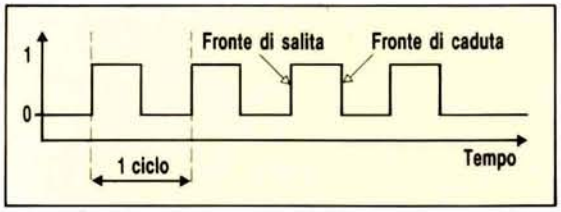
Un simile circuito si avvia cambiando il segnale applicato ad X da HI a LO, oppure quello applicato ad Y da LO a HI. La durata dell'intervallo si può variare modificando i valori della resistenza R e del condensatore C. Questo diagramma mostra le relazioni intercorrenti tra input e output:



L'uscita del circuito, così ottenuta, può essere utilizzata per controllare il motore a passi di un lettore di nastri o per ritardare di un dato tempo la trasmissione di un bit.
Collegando tra loro due circuiti monostabili è possibile realizzare un segnale di "clock", che oscilla fra HI e LO con una frequenza prefissata:



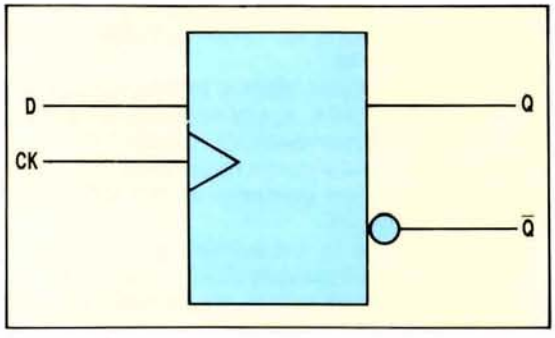
L'uscita ha un andamento tipico, detto ad "onda quadra" (come mostrato nel grafico seguente). L'intervallo di tempo compreso tra due successivi "scatti" di clock, prende il nome di ciclo. Nei clock dei computer, un ciclo dura generalmente un milionesimo di secondo. Il segnale di clock costituisce il cuore del computer e scandisce le numerose funzioni eseguite dalla CPU. Questo diagramma illustra la forma d'onda del segnale di clock:



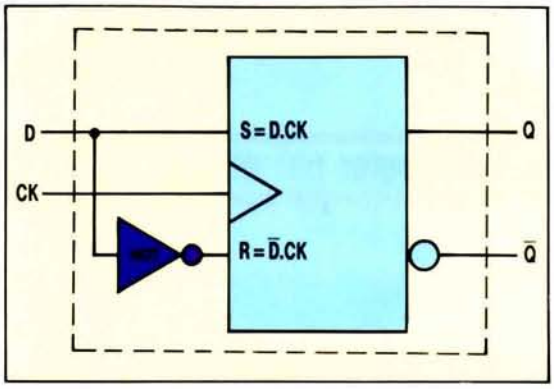
Vediamo ora due nuovi tipi di flip-flop, pilotati da un segnale di clock ad andamento regolare.

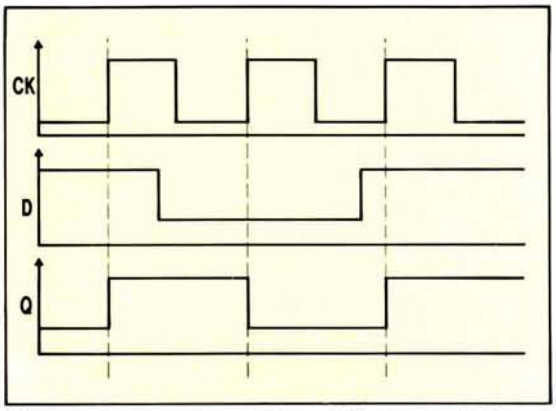
Il flip-flop tipo D

In un flip-flop tipo "D" esistono due ingressi: quello logico (D) e quello di clock (CK).



Il circuito è basato su un flip-flop "R-S": applicando in ingresso un segnale di clock, otteniamo un particolare tipo di funzionamento chiamato "latch" (catenaccio). Il livello dell'uscita Q viene determinato all'inizio di ciascun ciclo di clock. Se in quel momento l'ingresso in D è HI, l'output Q risulta essere HI e, viceversa, se l'ingresso D è LO, l'output Q sarà posto a LO.

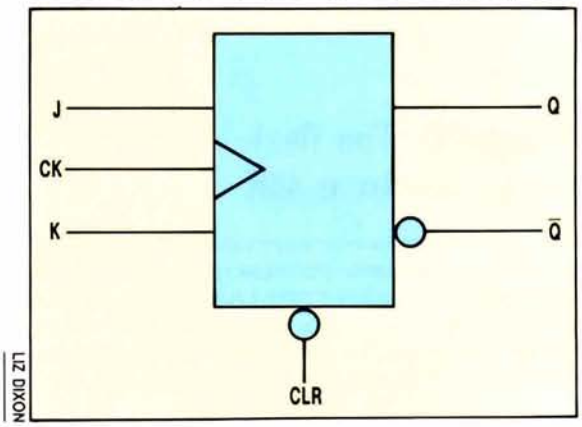




Questo meccanismo viene illustrato dal precedente diagramma: l'uscita Q può cambiare solo quando il segnale di clock passa da LO a HI. In termini più tecnici, si parla di flip-flop tipo "D", pilotato dal fronte di salita.

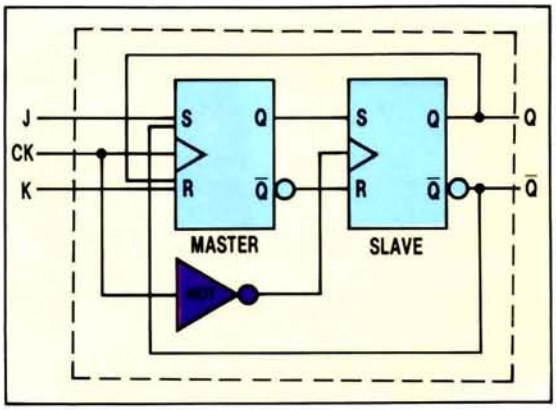
Il flip-flop J-K

Il flip-flop "J-K" è noto anche come "master/slave" (padrone/schiavo), perché composto da due flip-flop R-S, collegati in modo da formare una reciproca relazione di dominanza/schiavitù. Un dispositivo master/slave permette, durante un solo ciclo di clock, di immagazzinare il segnale d'ingresso in uno dei due flip-flop e, contemporaneamente, emettere un segnale di uscita dal secondo flip-flop, che dipende dal precedente segnale applicato. Un classico esempio è la procedura di "scorrimento", comune a molti microprocessori, grazie alla quale i bit all'interno del registro accumulatore vengono spostati a destra o a sinistra di una posizione. Il seguente è lo schema del circuito di un flip-flop J-K:



LIZ DIXON

Lo schema seguente mostra i collegamenti tra i due flip-flop R-S, l'uno "dominante" sull'altro. Supponiamo di applicare un segnale a J (oppure a K): se il segnale di clock è HI, l'input raggiunge il flip-flop "padrone", se invece è LO, l'input perviene al flip-flop "schiavo", perché i tipi R-S sono "circuiti di trigger a fronte di salita". Quindi, in un qualsiasi momento, solo uno dei due flip-flop R-S è attivo, mentre l'altro conserva al suo interno il segnale di input ricevuto in precedenza.



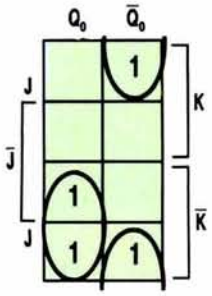
In margine è mostrata una "tavola di stato" per i flip-flop J-K. È molto simile ad una tavola della verità, ma contiene anche una variabile Q_0, che rappresenta l'output precedente. Si noti che due segnali HI, applicati simultaneamente a J e K, provocano un cambiamento di stato del flip-flop ad ogni segnale di clock. Questo effetto, chiamato "alternanza", è indotto dall'influenza delle uscite slave sugli ingressi master. Nei flip-flop R-S, una simile combinazione di ingressi non è consentita, producendo indeterminatezza nel segnale in uscita. Se consideriamo Q_0, J e K come ingressi, i risultati della tavola di stato si possono collocare in una k-mappa.

Q_0	J	K	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Dalla k-mappa possiamo ricavare la seguente espressione logica

$$Q = Q_0.J + Q_0.K$$

Questa equazione viene chiamata la "caratteristica" del flip-flop J-K.



Risposte all'Esercizio 7 a pag. 709

1) Un flip-flop è chiamato anche "bistabile" in quanto l'uscita si stabilizza in una sola delle due condizioni possibili (ossia quando Q = 1 e Q = 0 oppure quando Q = 0 e Q = 1).
2) **a)** Questa non è una condizione stabile.
   b) Il flip-flop passa in stato di RESET, se viene esaminata per prima la porta superiore, oppure in stato di SET, se la prima ad essere esaminata è la porta inferiore.
   c) Sì (vedere la risposta b).
   d) Per far sì che tutti i registri si trovino nello stato previsto al momento dell'accensione, devono essere riportati in stato di SET o di RESET durante l'inizializzazione del sistema.

Software in vetrina

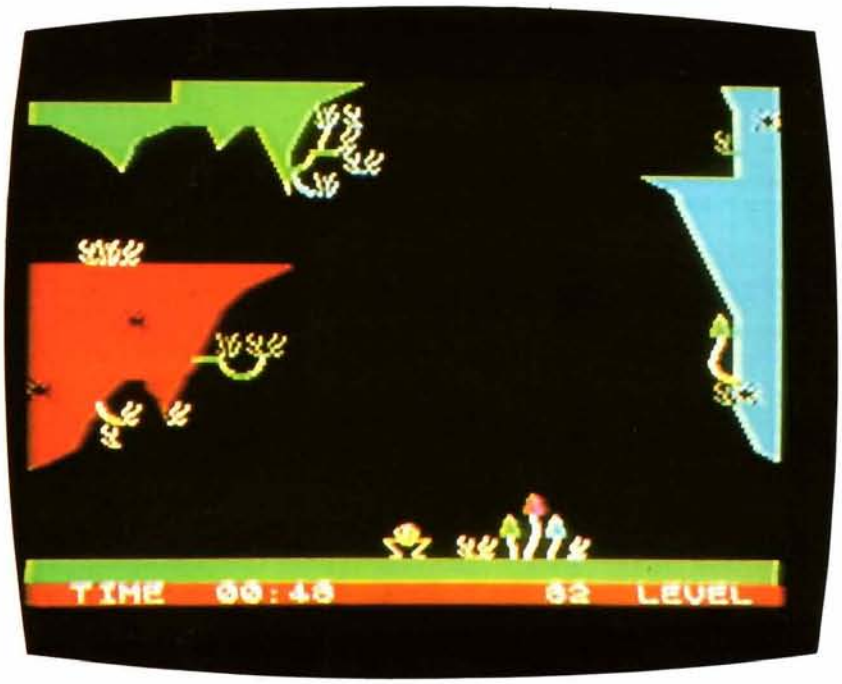
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

È questa la volta di due programmi per lo Spectrum: uno è del genere "war games", l'altro appartiene invece alla schiera dei giochi "arcade" più tradizionali.

Apocalypse per lo Spectrum 48K

Per gli appassionati dei giochi strategici, questo programma dal nome altisonante può rivelarsi una fonte di divertimento per ore e ore. Il "teatro" è l'Europa, tracciata sullo schermo con la grafica a blocchi. Il gioco è adatto a due giocatori e si svolge nel modo tipico dei "war games" da computer.





Boogaboo (The flea) per lo Spectrum 48K

Ciò che è veramente apprezzabile in questo gioco è il sapiente impiego della grafica. Il giocatore deve manovrare i movimenti di una piccola pulce, per farla sfuggire ad un vorace drago volante. Il percorso, naturalmente, non è dei più facili e gli ostacoli imprevisti non mancano.

Una gustosa albicocca

Un microcomputer dalle pregevoli prestazioni, adattissimo all'ufficio

Appositamente progettato per usi gestionali, l'Apricot viene venduto in due versioni: con due unità a dischetti da 3 1/2 pollici oppure con una sola unità da 3 1/2 pollici ed un disco rigido da 10 Mbyte. L'Apricot possiede numerose caratteristiche allettanti per gli operatori commerciali, ma non certo per gli appassionati di home computer: niente grafica a colori, quindi, né porte per il registratore, né comandi per giochi. È tuttavia dotato, nel modello standard, di un monitor monocromatico ad alta risoluzione, di una porta parallela per stampante, di una porta seriale RS232, di una certa quantità di software e di una tastiera di buona qualità.

Anzi, proprio la tastiera rappresenta una delle caratteristiche salienti dell'Apricot: nella sua parte superiore è dotata di un display a cristalli liquidi (LCD), battezzato "Microscreen", di due righe per 40 caratteri.

All'accensione dell'apparecchio, la riga superiore del Microscreen visualizza la data (giorno, mese ed anno) e l'ora. Entrambe possono essere variate usando uno dei programmi di utilità, mentre l'orologio, alimentato a pile, segna il tempo anche quando il computer è spento.

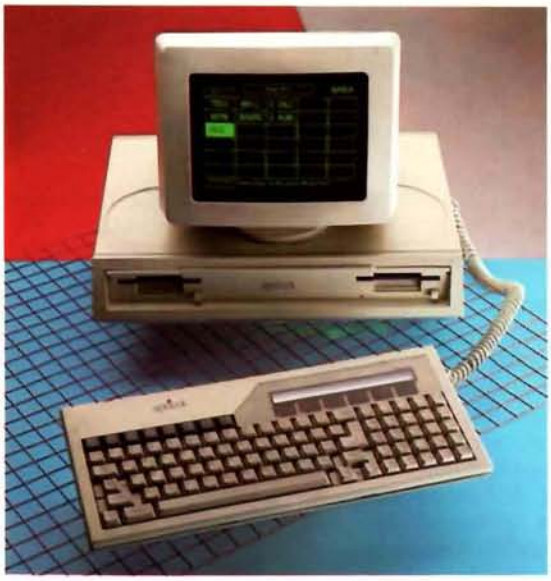
Automaticamente, quando si accende la macchina, viene avviato un programma di autoanalisi, che visualizza la quantità di memoria disponibile (lo standard è di 256 Kbyte ma può essere espansa fino a 768 Kbyte) e ricorda all'utente di inserire il dischetto contenente l'MS-DOS. Per coloro che non hanno familiarità con i sistemi operativi come il CP/M o l'MS-DOS, è presente un menù esplicativo, chiamato Manager, che permette una più facile selezione delle applicazioni software (ad esempio, il Supercalc, il Multiplan, il BASIC della Microsoft, ecc.) o dei programmi di utilità (per modificare le funzioni dei tasti o il set dei caratteri sullo schermo).

Il Microscreen è controllabile via software, quindi può svolgere molte altre funzioni oltre alla semplice visualizzazione della data e dell'ora. Sono disponibili 6 tasti programmabili e le funzioni che essi abilitano possono essere richiamate sul display in qualsiasi momento. Quindi, quando un programma invia sul video un particolare menu, la stessa immagine può essere duplicata sul Microscreen. Premere un particolare tasto di funzione equivale dunque a selezionare la stessa richiesta sul video principale usando i tasti Cursor e Return. L'unico "neo" della tastierina del display è la sua struttura a membrana (stile ZX81), piuttosto scomoda e poco funzionale rispetto ad una normale tastiera.

La tastiera principale possiede 8 normali tasti di funzione corredati di specifiche etichette (HELP, MENU, PRINT, FINISH, ecc.). Anch'essi, come tutti i tasti dell'Apricot, possono essere riprogrammati usando il programma "Keyedit", fornito con l'apparecchio. Il livello qualitativo della tastiera è certamente superiore al già alto standard che ci si aspetterebbe da una macchina per usi commerciali, ma i tasti Control e Escape si trovano in posizione poco funzionale. Il cavo di collegamento della tastiera si inserisce sul retro dell'unità principale e ciò consente una più comoda orientazione della macchina. Le notevoli dimensioni del monitor, tuttavia, impediscono all'apparecchio di poter essere compreso nella gamma dei computer trasportabili.

Il software che correda l'Apricot è composto da una completa gamma di programmi di utilità e dal Supercalc. Usando quest'ultimo, assieme al Superplanner, è possibile ottenere analisi "proiezioni" di carattere finanziario. Sia in questi programmi, sia in altre parti di software adattate all'Apricot, si avverte distintamente la fretta con la quale i progettisti si sono mossi nel mettere a punto i programmi per il lancio commerciale dell'apparecchio. L'Apricot dispone di due sistemi operativi: l'MS-DOS ed il CP/M-86 ed i fabbricanti garantiscono l'invio gratuito del Cuncurrent CP/M, non appena disponibile. La macchina dispone solo della prima versione del Supercalc: la seconda e la terza verranno commercializzate in seguito.

Una delle prime critiche mosse all'Apricot è la discutibile implementazione del sistema operativo MS-DOS e la conseguente lentezza operativa. Questo problema sembra essere



CHRIS STEVENS

Un'interessante offerta

Nel settore gestionale, l'Apricot della ATC è uno dei più interessanti computer sul mercato, ad un prezzo relativamente basso. L'apparecchio usa un processore a 16 bit, possiede 256 Kbyte di memoria standard ed un monitor di ottima qualità



I primi Microfloppy

L'Apricot è stato il primo micro gestionale ad adottare la nuova generazione di microdischetti. La ATC ha optato per le unità a microdischetti Sony che impiegano dischetti da soli 3 1/2 pollici, racchiusi in una custodia rigida. Le loro piccole dimensioni li rendono più robusti dei comuni dischi da 5 1/4 pollici. Una piccola finestrella a molla protegge la superficie magnetica dalla polvere



Apricot XI

Poiché la capacità dei microdischetti è piuttosto limitata, la ACT produce anche l'Apricot XI che incorpora un disco rigido da 10 Mbyte al posto di una delle due unità a dischetti

stato poi risolto. Il software applicativo si dimostra abbastanza efficiente ed i test di laboratorio sul BASIC confermano tale aspetto. Tuttavia l'impressione generale è che l'Apricot non riesca a sfruttare appieno la potenziale velocità di elaborazione che il processore 8086 gli consentirebbe.

La documentazione della macchina comprende un'introduzione per principianti, una esauriente guida all'MS-DOS e due manuali per il Wordstar ed il Multiplan.

La casa produttrice, la ATC, fornisce invece poche informazioni sull'hardware, anche se i programmi di utilità disponibili sono più che adeguati per la messa a punto del computer. Mancano le informazioni sulla mappa della memoria e sulle chiamate di sistema: documentazioni indispensabili per le software house intenzionate a sviluppare nuovi programmi per l'Apricot.

Del resto, l'Apricot è un computer espressamente orientato ad uso gestionale e non un sistema per progettisti software o per semplici appassionati. Se il successo di questa macchina eguaglierà quello del Sirius, anch'esso prodotto dalla ATC, vi saranno sicuramente dei fabbricanti indipendenti pronti a produrre schede ad innesto opzionali, mentre la stessa ATC ha intenzione di realizzare schede di memoria extra ed un modem.

Schermo

Il monitor a fosfori v
alta risoluzione fornisc
tima qualità dell'imma

Display LCD

Il display a cristalli liquidi può essere usato per messaggi, come calcolatrice oppure come orologio

Microfloppy Sony

I piccoli dischetti da 315K costituiscono una compatta e funzionale memoria di massa

Tasti di funzione

Questi tasti prevedono funzioni standard come HELP e REPEAT



Tasti di funzione a sfioramento

Le funzioni selezionate da questi 6 tasti vengono visualizzate sul display LCD posto sopra di essi

RAM 256K

Questa è la RA
dell'Apricot



La tastiera dell'Apricot

L'Apricot possiede un'ottima tastiera, oltre ad una tastierina a 6 tasti destinati a svolgere particolari funzioni in alcuni programmi. Poiché queste variano da programma a programma, quelle selezionate possono essere visualizzate, al di sopra di ciascun tasto, da un piccolo display a cristalli liquidi, dotato di due linee di 40 caratteri



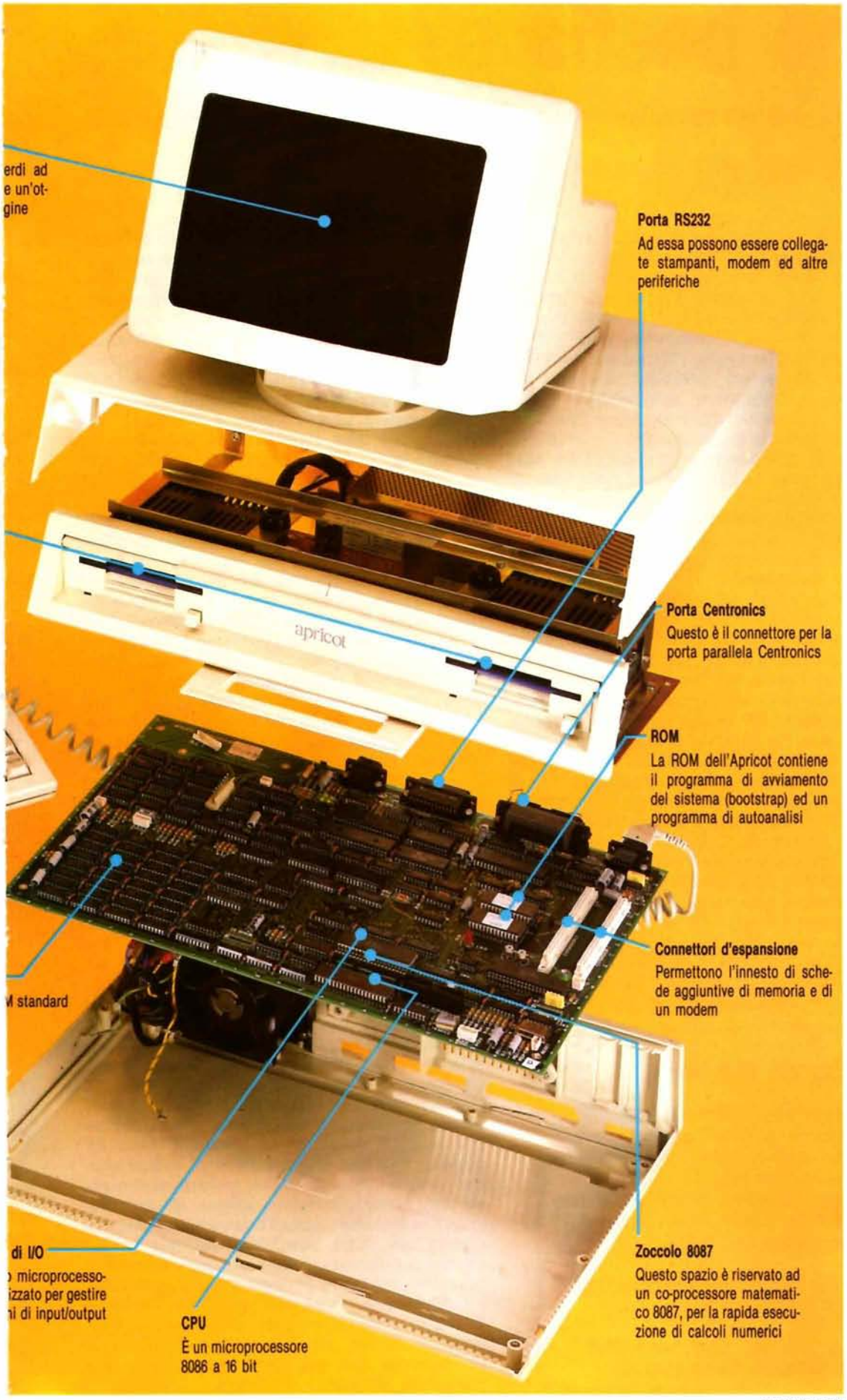
Il Monitor dell'Apricot

Il monitor ha uno schermo di soli 9 pollici, in grado di visualizzare 132 x 50 caratteri, anche se in genere si opta per il funzionamento con 80 x 25 caratteri. Per la comodità dell'operatore, lo schermo è antiriflesso.

Il peso del monitor limita molto la "trasportabilità" dell'Apricot: molti utenti tengono un monitor a casa ed uno in ufficio, limitandosi così a trasportare la sola unità centrale

Processore

Un second
re viene uti
le operazio



CHRIS STEVENS

APRICOT ACT

DIMENSIONI

mm. 488 x 413 x 313

CPU

8086 a 16 bit, con possibilità di aggiungere un coprocessore 8087 per calcoli numerici

MEMORIA

256 Kbyte di RAM, espansioni fino a 768 Kbyte

VIDEO

In modo testuale, si può scegliere tra la risoluzione a 132 x 50 caratteri o quella a 80 x 25. La grafica ha una risoluzione di 800 x 400 punti, monocroma

INTERFACCE

Seriale RS232, parallela Centronics, connettore per "mouse" e innesti per schede di espansione

UNITÀ A DISCHI

Una o due unità Sony a microdischetti da 3 e 1/4 pollici, con 315 Kbyte o 720 Kbyte di capacità. Il modello Apricot XI incorpora un disco rigido da 10 Mbyte ed un'unità a microdischetti

SISTEMA OPERATIVO

MS-DOS, CP/M-86 e Concurrent CP/M-86

TASTIERA

Tipo macchina da scrivere, con 90 tasti più 6 di funzione "a sfioramento", dotati di un piccolo visore a cristalli liquidi

DOCUMENTAZIONE

La documentazione è ampia e ben curata

PRO

È una macchina sulla quale è molto piacevole lavorare ed ha caratteristiche superiori ad altre macchine gestionali, ad un prezzo relativamente basso

CONTRO

Essendo orientata al settore commerciale, non è la macchina adatta per un semplice appassionato. Non è comodamente trasportabile e, per il momento, c'è carenza di software applicativo

Che bestia è?

Sviluppiamo un semplice gioco, che illustra i principi dell'Intelligenza Artificiale

Animals è un gioco nel quale il computer cerca di indovinare il nome dell'animale pensato dal giocatore, rivolgendogli domande del tipo: "Può volare?", "Ha la pelliccia?" e così via. Il giocatore risponde soltanto con un "sì" o un "no" ed il computer, passo dopo passo, trova il sentiero che gli permette di formulare un'ipotesi "pilotata". L'abilità dimostrata dal computer sorprenderà certamente quei giocatori che non hanno familiarità con apparecchi di questo tipo. I due principali motivi che rendono questo gioco così divertente sono la capacità del computer di usare un linguaggio sofisticato (anche se le risposte del giocatore si limitano ad un "sì" o un "no") e la vasta gamma di conoscenze mostrate dalla macchina nel tentativo di indovinare l'animale.

Animals è effettivamente un "programma euristico" molto semplice, capace di migliorare le proprie prestazioni col progredire del proprio funzionamento. Quando viene usato per la prima volta, conosce solo i nomi di due animali ed è in grado di rivolgere al giocatore una sola domanda. Se la risposta del giocatore è "sì", il programma può tentare di indovinare il nome dell'animale. Se il computer sbaglia (risultato molto probabile al primo tentativo), il programma chiede al giocatore di inserire il nome dell'animale pensato e di dotare il sistema di una nuova domanda per poterlo distinguere dall'animale ipotizzato dal computer. Questa nuova informazione viene quindi aggiunta all'archivio gestito dal programma ed entra a far parte di un "albero di conoscenza", che verrà usato nei successivi giochi. Le dimensioni dell'albero crescono ad ogni "partita" e dopo un certo tempo il programma sarà in grado di indovinare la maggior parte degli animali e solo occasionalmente si imbatterà in un nuovo nome.

È interessante ricordare che il programma, inizialmente, non ha nessuna "conoscenza": esso segue, infatti, un sentiero tracciato in base alla combinazione delle informazioni fornite dai vari giocatori. Il gioco può quindi riguardare i più svariati argomenti: i vari tipi di vino, le parti di una motocicletta, le malattie oppure i propri amici e familiari. Un programma che permetta al giocatore di definire la domanda iniziale e le due risposte possibili consente una vastissima gamma di variazioni.

In altre parole, il funzionamento del programma non dipende tanto dal tipo di informazioni inserite, ma dal modo in cui sono organizzate all'interno del "database".

È abbastanza facile scrivere un semplice programma BASIC capace di creare un albero di conoscenza. In genere le strutture di questo tipo vengono inserite in matrici BASIC: in questo caso si dovrà usare T$() per le domande e i nomi degli animali ed Y() e N() per i collegamenti tra le diverse registrazioni in T$. I collegamenti costituiscono i vari sentieri o "rami" dell'albero. Se la risposta ad una domanda è "sì", per ogni registrazione in T$ ci sarà una corrispondente registrazione in T() per indicare al programma quale sentiero percorrere. Ciò vale anche per le registrazioni in N() quando la risposta è "no". Alla fine del percorso, T$() non contiene il testo di una domanda, ma il nome di un animale. In questo caso tanto Y() che N() contengono il valore 0 ed il programma è costretto a fare un'ipotesi sul nome di animale pensato dal giocatore.

Il programma della pagina seguente è stato sviluppato, in modo breve e semplice, per agevolare la comprensione dei principi su cui si basa: lo si potrà tuttavia sviluppare, arricchendo la presentazione con grafica a colori, suoni e così via. Un significativo miglioramento potrebbe essere la possibilità d'immagazzinare il database su nastro o disco. Le migliori versioni di *Animals* sono quelle adoperate innumerevoli volte, perché contengono uno sviluppatissimo albero di animali, mostri mitologici, oggetti, personaggi famosi e così via, tutti mescolati assieme in un gigantesco database. Una versione ancor più completa potrebbe permettere al giocatore di selezionare la prima domanda e di cambiare le registrazioni contenute nell'albero o inserirne di nuove: ciò permetterà di usare il programma non più come passatempo, ma per scopi più seri.

Basic & Basic

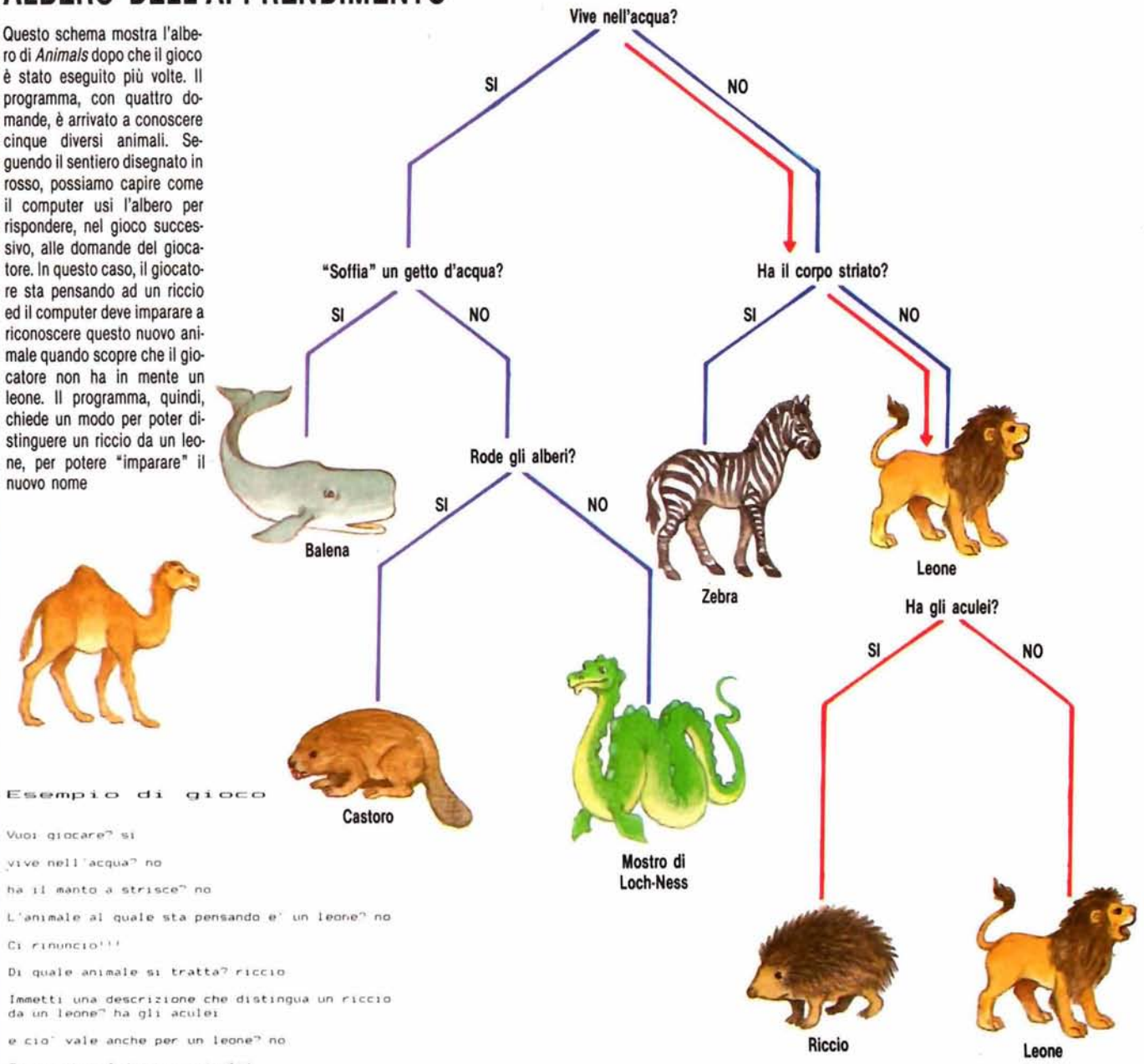
Questo programma è stato scritto in Microsoft BASIC per funzionare senza modifiche sul maggior numero possibile di macchine. Il solo cambiamento necessario riguarda il formato dei comandi PRINT.

Sullo Spectrum tutte le istruzioni di assegnazione devono cominciare con la parola chiave LET. Le seguenti linee dovranno essere così modificate:

```
 45 LET L=40: REM N. di caratteri in una domanda
 50 DIM Y(N): DIM N(N): DIM T$(N,L)
150 LET I$=A$(1): LET P$="A"
200 IF A=30 THEN PRINT: PRINT "CIAO": STOP
230 IF I(P)=0 AND N(P)=0 THEN GOTO 290
```

ALBERO DELL'APPRENDIMENTO

Questo schema mostra l'albero di *Animals* dopo che il gioco è stato eseguito più volte. Il programma, con quattro domande, è arrivato a conoscere cinque diversi animali. Seguendo il sentiero disegnato in rosso, possiamo capire come il computer usi l'albero per rispondere, nel gioco successivo, alle domande del giocatore. In questo caso, il giocatore sta pensando ad un riccio ed il computer deve imparare a riconoscere questo nuovo animale quando scopre che il giocatore non ha in mente un leone. Il programma, quindi, chiede un modo per poter distinguere un riccio da un leone, per potere "imparare" il nuovo nome



Esempio di gioco

```
Vuoi giocare? si

vive nell'acqua? no

ha il manto a strisce? no

L'animale al quale sta pensando e' un leone? no

Ci rinuncio!!!

Di quale animale si tratta? riccio

Immetti una descrizione che distingua un riccio
da un leone? ha gli aculei

e cio' vale anche per un leone? no

Ora conosco 4 diversi animali!

Vuoi giocare?
```

```
10 REM *** ANIMALS ***
20 REM
30 REM *** PREPARAZIONE ***
40 N=100: REM N. massimo di animali
50 DIM Y(N),N(N),T$(N)
60 C=3:FOR I=1 TO 3:READ Y(I),N(I),T$(I):NEXT I
70 PRINT CHR$(12):PRINT "A N I M A L S":PRINT
80 GOTO 210
90 REM *** Immissione risposta (SI/NO) ***
100 PRINT:PRINT O$;" ";:INPUT A$
110 IF A$="s" OR A$ = "S" OR A$ = "SI" OR A$ = "si" THEN A=1:RETURN
120 IF A$="n" OR A$ = "N" OR A$ = "NO" OR A$ = "no" THEN A=0:RETURN
130 PRINT:PRINT"Rispondi SI o NO!":GOTO 100
140 140 REM *** Aggiunta di UN o UNA ***
150 I$=RIGHT$(A$,1):P$=" un "
160 IF I$ = "A" OR I$="a" THEN P$=" una "
170 I$=LEFT$(A$,1)
180 IF (I$="A" OR I$ ="a") AND P$= " una " THEN P$ = " un'"
190 A$=P$+A$:RETURN
200 REM *** Inizio di un nuovo gioco ***
210 O$="Vuoi fare una partita":GOSUB 100
220 IF A=0 THEN PRINT:PRINT "Allora, ciao!":END
230 P=1
240 REM *** Svolgimento del gioco ***
250 IF Y(P)=0 AND N(P)=0 THEN 310
260 O$=T$(P):GOSUB 100
270 IF A=1 THEN P=Y(P)
280 IF A=0 THEN P=N(P)
290 GOTO 250
300 REM *** Tentativo di indovinare il nome ***
310 A$=T$(P):GOSUB 150:T$=A$
320 O$="L'animale al quale stai pensando e'"+A$:GOSUB 100
330 IF A=1 THEN PRINT:PRINT"Ho indovinato!!!":GOTO 450
340 REM *** Apprendimento di nuovo nome ***
350 PRINT:PRINT"Ci rinuncio !!!":PRINT "Di quale animale si
    tratta";:INPUT N$
360 A$=N$:GOSUB 150
370 PRINT:PRINT"Immetti una descrizione che distingua";A$;
    " da";T$:INPUT D$
380 O$="e cio' vale anche per "+T$:GOSUB 100
390 A$=T$(P):T$(P)=D$:T$(C+1)=A$:T$(C+2)=N$
400 IF A=1 THEN Y(P)=C+1:N(P)=C+2
410 IF A=0 THEN Y(P)=C+2:N(P)=C+1
420 Y(C+1)=0:N(C+1)=0:Y(C+2)=0:N(C+2)=0
430 C=C+2
440 REM *** Fine di un gioco ***
450 A=INT(C/2)+1
460 PRINT:PRINT"Adesso conosco ";A;" diversi animali !"
470 GOTO 210
480 REM *** Dati iniziali ***
490 DATA 2,3,"Vive nell'acqua"
500 DATA 0,0,"foca"
510 DATA 0,0,"leone"
```

DAVID HIGHAM

Alta e bassa risoluzione

Qualche suggerimento per sfruttare la grafica ad alta risoluzione del Commodore 64

Per le elaborazioni grafiche a bassa risoluzione lo schermo del Commodore 64 viene suddiviso in 25 righe di 40 caratteri ciascuna: 1000 celle in tutto. Come ben sappiamo, ogni carattere è costituito da una serie di piccoli punti, chiamati pixel, disposti su otto righe. Ciascuna cella di carattere contiene quindi 64 pixel. Nella grafica ad alta risoluzione, dobbiamo essere in grado di accendere o spegnere ciascun pixel individualmente, associando ad esso un solo bit della memoria del computer. Questa procedura viene chiamata "bit mapping". Poiché una locazione di memoria contiene 8 bit ed il video è composto da 64000 pixel, sono necessarie 8000 locazioni di memoria per immagazzinare tutte le informazioni.
Nel Commodore 64 si passa dalla bassa all'alta risoluzione ponendo a 1 il bit 5 della locazione 53265. Per selezionare questo bit senza toccare gli altri, occorre usare il seguente comando:

```
POKE 53256, PEEK (53256) OR 32
```

Abilitato il modo ad alta risoluzione, il video riceverà le informazioni da un blocco di memoria composto da 8000 byte. L'indirizzo del primo byte di questo blocco è contenuto nella locazione 53272, la stessa usata nell'ultima lezione (vedere a pag. 713), per la creazione di caratteri speciali.
L'area di memoria assegnata al video in bassa risoluzione, viene adesso sfruttata per contenere l'informazione sul colore di ciascuna cella 8×8 del video. Per selezionare i 16 colori disponibili bastano 4 bit, quindi i 4 bit più significativi di ogni locazione della memoria video vengono usati per memorizzare il colore dei pixel accesi in ciascuna cella, mentre gli altri 4 bit contengono il colore dei pixel spenti. Una cella del video può quindi avere due colori diversi: uno per il primo piano e l'altro per lo sfondo. Se volessimo uno sfondo violetto sull'intero schermo e un'immagine di primo piano ad alta risoluzione in nero, dovremmo usare i seguenti codici:

Il codice di colore per il nero è 0=0000 in binario
Il codice di colore per il viola è 4=0100 in binario

Riunendo i due valori binari in un solo byte, abbiamo 00000100, ossia 4 in decimale. Tramite una serie di POKE, possiamo ora depositare tale valore in ciascuna locazione di memoria video (da 1024 a 2023), ottenendo quanto desiderato.
Prima di cominciare a lavorare sul video ad alta risoluzione è bene "ripulire" l'area di memoria video azzerando tutte le locazioni: quest'operazione in BASIC richiede alcuni secondi, ma, se viene tralasciata, sul video apparirà solo un ammasso di punti confusi perché l'area di memoria in questione può contenere dei valori del tutto casuali.

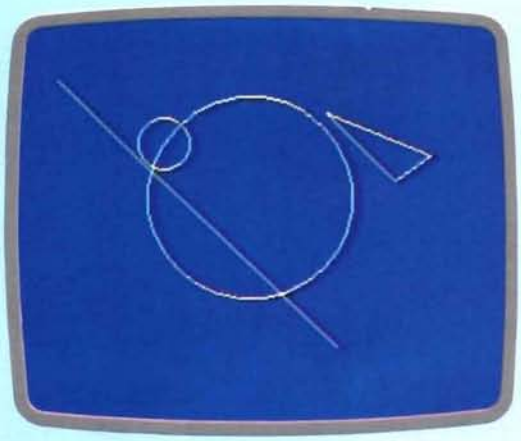
Uso della grafica

Un programma di grafica ad alta risoluzione deve essere in grado di accendere o spegnere singoli pixel sullo schermo. Se a ciascun punto vengono assegnate specifiche coordinate X e Y (X e Y sono compresi rispettivamente tra 0 e 319 e tra 0 e 199), allora il programma può individuare il corrisponden-

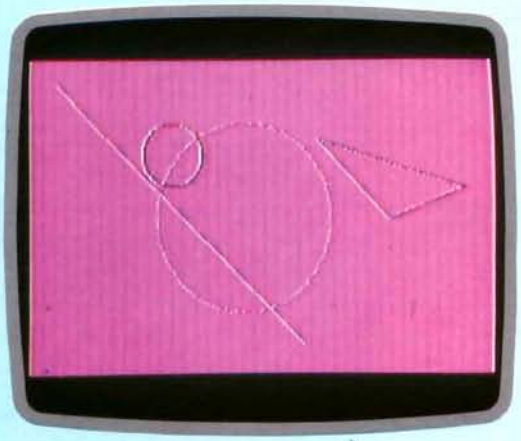
VELOCITÀ RELATIVE

Al Commodore 64 occorrono quasi 90 secondi e circa 50 linee di programma per realizzare quest'immagine. La creazione della stessa immagine sullo Spectrum richiede appena due secondi ed il seguente semplice programma:

```
4000 REM******ALTA RISOLUZIONE******
4050 LET M=18: DIM U(M): INK 6:
     PAPER 1: BORDER 1: RESTORE 4100
4100 DATA 20,160,160,-160
4120 DATA 8,123,15
4140 DATA 130,90,60
4160 DATA 175,140
4180 DATA 40,-40
4200 DATA 20,15
4220 DATA -60,25
4250 FOR K=1 TO M: READ U(K): NEXT K
4300 PLOT U(1), U(2): DRAW U(3), U(4)
4350 CIRCLE U(5), U(6), U(7)
4400 CIRCLE U(8), U(9), U(10)
4450 PLOT U(11), U(12): DRAW U(13),
     U(14)
4500 DRAW U(15), U(16): DRAW U(17),
     U(18)
4600 PAUSE 0
```

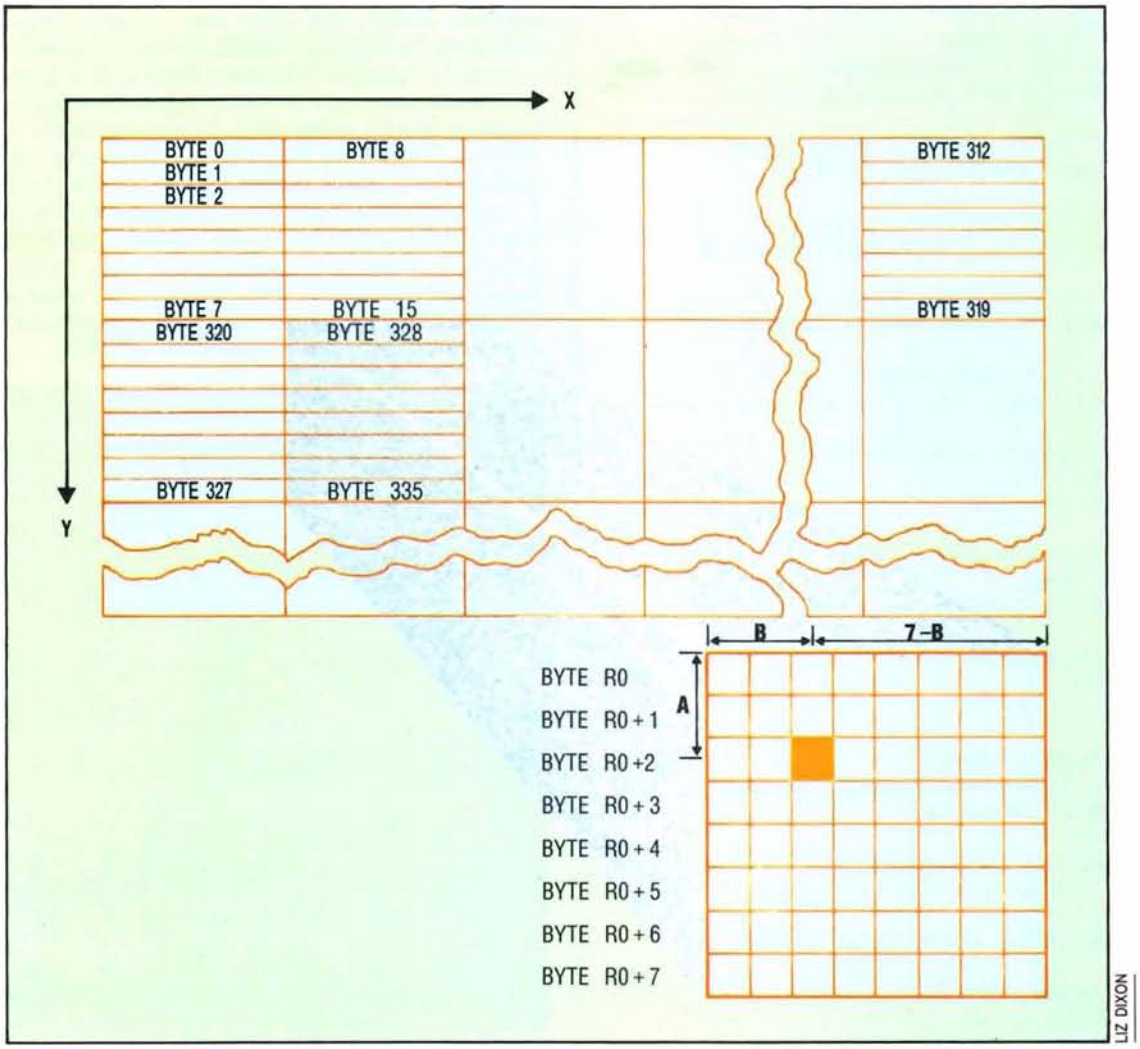


TEMPO D'ESECUZIONE (SPECTRUM) = 1,85 SEC.



TEMPO D'ESECUZIONE (COMMODORE 64) = 89,6 SEC.

LIZ HEANEY



Da punto a punto

I pixel che compongono lo schermo ad alta risoluzione del Commodore 64 non possono essere manipolati direttamente: la memoria del video occupa 40 x 25 = 8000 byte di RAM e ciascuna cella di carattere viene controllata da 8 byte. In alta risoluzione, un pixel viene individuato da X, che indica la sua distanza (in pixel) dal margine sinistro del video, e da Y, cioè la sua distanza dal margine superiore del video. Questi numeri devono poi essere convertiti nell'indirizzo del byte che contiene quel particolare pixel e nel numero dello specifico bit in quel byte

te bit della mappa di memoria di 8000 byte da porre a 1 oppure a 0.
La posizione orizzontale del byte può essere trovata in base alla coordinata X, usando il seguente comando:

```
HB=INT(X/8)
```

La posizione verticale del byte verrà invece individuata partendo dalla coordinata Y:

```
VB=INT(Y/8)
```

Il primo byte della serie, che contiene il bit cercato, può essere calcolato mediante HB e VB:

```
RO=VB*320+HB*8
```

Il byte che contiene il bit in questione si ottiene sommando a RO il resto ottenuto dalla divisione Y/8, che si può facilmente ricavare dai tre bit più a destra del valore di Y. Se A=Y AND 7 e BASE è l'indirizzo del primo byte del blocco di 8000 byte, allora è possibile calcolare l'indirizzo del byte, BY, che contiene il bit cercato:

```
BY=BASE+RO+A
```

Il bit all'interno del byte BY può essere individuato calcolando il resto della divisione X/8. Se B=XAND7, allora la seguente POKE pone ad 1 il bit che corrisponde al pixel con coordinate X e Y:

```
POKE BY, PEEK(BY) OR (2↑(7—B))
```

Dopo aver imparato questo metodo, possiamo scrivere delle routine per tracciare figure sul video. Il programma a pagina 736 traccia una linea retta tra due punti specificati (con coordinate X1, Y1 e X2, Y2) prima di disegnare un cerchio con centro (CX, CY) e raggio RA. Infine, una terza routine traccia un triangolo, in base alle coordinate dei tre angoli (XA, YA), (XB, YB) e (XC, YC). Si facciano degli esperimenti, modificando i valori delle coordinate nel programma.
È interessante notare che questo programma è composto da una serie di subroutine concatenate tra loro. La routine di livello più basso traccia un singolo punto sul video e viene ripetutamente usata da una routine di livello più alto, per disegnare una linea retta. Al livello più alto, la routine TRACCIA UN TRIANGOLO sfrutta la routine TRACCIA UNA LINEA tre volte per disegnare i tre lati del triangolo. Questo modello di programmazione offre numerosi vantaggi. È flessibile, in quanto è molto facile sviluppare una routine che disegni, ad esempio, esagoni regolari: basta richiamare la

```
65 REM  **** HI-RES DEMO ****
72 PRINT CHR$(147) : REM CLEAR SCREEN
80 POKE 53280,0 : REM COLOUR BORDER BLACK
90 :
100 REM **** COLOUR SCREEN MEMORY AREA ****
110 FOR I=1024 TO 2023: POKE I,4: NEXT I
120 :
130 REM **** SET BIT MAP POINTER ****
140 BASE =8192: POKE 53272. PEEK (53272) OR 8
150 :
160 REM **** CLEAR BIT MAP MODE ****
170 FOR I=BASE TO BASE + 7999:POKE I,0:NEXT I
180 :
190 REM **** SET BIT MAP MODE ****
200 POKE 53265, PEEK(53265) OR 32
210 :
220 REM **** DRAW STRAIGHT LINE ****
230 X1=20: X2=190: Y1=15: Y2=180
240 GOSUB 800: REM PLOT LINE
250 :
300 REM **** DRAW CIRCLE ****
310 CX=150: CY=100: RA=60
320 GOSUB 900: REM PLOT CIRCLE
330 :
370 **** ANOTHER CIRCLE ****
380 CX=100: CY=60: RA=20
390 GOSUB 900: PLOT CIRCLE
400:
410 REM ****DRAW TRIANGLE ****
420 XA=200:XB=250:XC=300:YA=50:YB=100:YC=80
430 GOSUB 600: REM PLOT TRIANGLE
440 :
450 GOTO 450: REM END OF MAIN PROGRAM
460 :
470 :
600 REM **** PLOT TRIANGLE SUBROUTINE ****
610 :
620 X1=XA: X2=XB: Y1=YA: Y2=YB
630 GOSUB 800: REM PLOT LINE
640 X1=XB: X2=XC: Y1=YB: Y2=YC
650 GOSUB 800: REM PLOT LINE
660 X1=XC: X2=XA: Y1=YC: Y2=YA
670 GOSUB 800: REM PLOT LINE
680 RETURN
690 :
800 REM **** PLOT LINE SUBROUTINE ****
810 S=1
820 IF X2<X1 THEN S=-1
830 FOR X=X1 TO X2  STEP S
840 Y=(Y2-Y1)*(X-X1)/(X2-X1)+Y1
850 GOSUB 1000:, REM PLOT POINT
860 NEXT X
870 RETURN.
880 :
900 REM **** PLOT CIRCLE SUBROUTINE ****
910 :
920 FOR ANGLE = 0 TO 2*PI STEP .04
930 X=INT (RA*COS(ANGLE) + CX)
940 Y=INT (CY-RA*SIN(ANGLE))
950 GOSUB 1000: REM PLOT POINT
960 NEXT ANGLE
970 RETURN
980 :
1000 REM **** PLOT POINT SUBROUTINE ****
1010 :
1020 IF X>319 OR X<0 OR Y>199 OR Y<0 THEN GOTO 1070
1030 HB=INT(X/8): VB=INT(Y/8)
1040 RO=VB*320+HB*8: A= Y AND 7: B=X AND 7
1050 BY=BASE+RO+A
1060 POKE BY, PEEK(BY)OR(2^(7-B))
1070 RETURN
```

routine TRACCIA UNA LINEA che, a sua volta, chiamerà la routine TRACCIA UN PUNTO. In alternativa, la routine TRACCIA UN TRIANGOLO potrebbe essere usata per costruire esagoni a partire da triangoli equilateri. In questo caso, la routine TRACCIA UN ESAGONO costituirebbe un quarto livello nella struttura del programma. È importante salvare il programma prima di dare il RUN perché un errato valore nella POKE potrebbe causare un arresto del tutto inatteso dell'apparecchio. Per tornare al video a bassa risoluzione, si devono premere contemporaneamente i tasti RUN/STOP e RESTORE.

PROGRAMMA "CACCIA AL SOTTOMARINO"

Un'importante sezione del gioco sottomarino che stiamo sviluppando è quella che aggiorna il punteggio. Nei giochi di questo tipo esistono molte soluzioni: il nostro sistema si attiene alle seguenti regole:

1) La profondità e la velocità del sottomarino sono i fattori principali. È più difficile colpire un sottomarino che si muove velocemente ed in profondità che non un sottomarino lento e meno profondo: il punteggio assegnato dopo ogni "centro" deve tenerne conto.
2) Il valore del sottomarino colpito viene sommato al punteggio del giocatore ma, se il sottomarino raggiunge il margine del video indenne, tale valore va sottratto dal punteggio. Tuttavia, non sono previsti punteggi negativi.

In una fase successiva del progetto, occorrerà sviluppare una routine che selezioni a caso la velocità e la profondità del sottomarino: per adesso ci interessa ricordare che la profondità e la velocità del sottomarino vengono memorizzate rispettivamente nelle variabili Y3 e DX. Il valore del sottomarino verrà calcolato in base ad esse. Per essere certi che il valore del sottomarino colpito venga correttamente conteggiato, dovremo usare la funzione INT nel seguente modo:

Valore del sottomarino = INT (Y3 + DX*30)

Il punteggio "corrente" del giocatore viene assegnato alla variabile SC. L'ultima operazione è quella di sommare o sottrarre il valore del sottomarino al valore della variabile SC, a seconda che si sia fatto "centro", o si sia mancato il sottomarino. La subroutine AGGIORNA IL PUNTEGGIO viene usata da due sezioni del programma:

1) Quando la posizione del sottomarino viene esaminata per controllare se ha raggiunto il margine del video
2) Durante la routine CENTRATO

In queste due sezioni del programma la "flag" DS verrà opportunamente modificata, per indicare quale sezione stia adoperando la subroutine AGGIORNA IL PUNTEGGIO. Ponendo DS = 1 nella routine CENTRATO e DS = —1 nella routine MARGINE DEL VIDEO, il punteggio verrà incrementato o diminuito nel modo seguente:

SC = SC + INT (Y3 + DX*30)*DS

Verificato che il punteggio non assuma valori negativi, possiamo visualizzare il nuovo valore di SC sulla prima linea del video. Aggiungiamo le seguenti linee al nostro programma e salviamo il tutto.

```
5500 REM ***AGGIORNA IL PUNTEGGIO***
5510 SC = SC + INT(Y3 + DX*30)*DS
5520 IFSC < OTHEN SC = 0
5530 PRINT  CHR$(19);  CHR$(144);  "PUNTEGGIO";  SC;
     CHR$(157);" "
5540 RETURN
```

Nella prossima parte vedremo come creare gli sprite da usare per la nave, il sottomarino, le bombe di profondità e l'esplosione.

Last In First Out

Esaminiamo il funzionamento dello "stack" e delle relative istruzioni

Una corretta gestione della memoria sta alla base di tutta la programmazione in linguaggio Assembly e molte delle istruzioni fin qui incontrate riguardano esclusivamente il caricamento o il recupero di un'informazione dalle locazioni di memoria. Abbiamo visto che in esse, nonostante gli svariati modi di indirizzamento usati per accedere alla memoria, deve sempre comparire uno specifico indirizzo di memoria. Tuttavia, esiste un particolare gruppo di istruzioni, che accedono ad una speciale area di memoria, pur non avendo un indirizzo come operando. L'area di memoria in questione è chiamata "stack". Lo stack è adoperato, sia dall'unità centrale del computer, sia dal programmatore, come area di lavoro provvisoria ed è paragonabile ad un blocco per appunti, sul quale scrivere, rileggere e cancellare dati ed indirizzi. Le operazioni di stack servono, appunto, per copiare i valori contenuti nei registri della CPU in aree libere dello stack e viceversa. Non occorre specificare alcun indirizzo, poiché un particolare registro della CPU, chiamato "puntatore allo stack" (stack pointer, o brevemente SP), contiene sempre l'indirizzo della successiva locazione di stack libera. Ne consegue che qualsiasi informazione trasferita sullo stack viene automaticamente scritta nel byte indicato dal puntatore allo stack e i dati richiamati da quest'ultimo vengono sempre copiati dall'ultima locazione di stack scritta. L'aggiornamento del puntatore allo stack viene automaticamente eseguito, come parte dell'istruzione.

Nel microprocessore 6502, lo stack si trova nei 256 byte di RAM compresi tra $0100 e $01FF. Nello Z80, invece, la collocazione dello stack viene assegnata dal sistema operativo, ma può essere cambiata dal programmatore. Questa differenza riflette le diversità di organizzazione interna dei due microprocessori (vedere lo schema a pag. 616): il 6502 ha un puntatore a singolo byte, mentre quello dello Z80 è a due byte.

La CPU del 6502 considera il contenuto del puntatore allo stack come il byte "basso" dell'indirizzo di stack. La parte "alta" è fissa e vale sempre $01. Ne deriva che gli indirizzi dello stack risiedono tutti nella pagina 1.

Al contrario, il puntatore allo stack dello Z80 è un registro a due byte capace di indirizzare una qualsiasi locazione compresa tra $0000 e $FFFF, cioè l'intera area di indirizzamento dello Z80. Lo stack può trovarsi in qualsiasi parte della RAM e la sua locazione può essere cambiata dal programmatore. Salvo sapere esattamente come intervenire, questa operazione è sconsigliata, perché il sistema operativo, dopo aver fissato la locazione dello stack, vi memorizza importanti informazioni, che si aspetta di ritrovare in seguito. Modificando la collocazione dello stack può dunque causare la perdita di tali informazioni ed una possibile caduta del sistema.

Per chiarire meglio l'uso dello stack, consideriamo la seguente routine, che scambia il contenuto delle due locazioni di memoria LOC1 e LOC2:

6502		Z80	
LDA	LOC1	LD	A,(LOC1)
PHA		PUSH	AF
LDA	LOC2	LD	A,(LOC2)
STA	LOC1	LD	(LOC1),A
PLA		POP	AF
STA	LOC2	LD	(LOC2),A

Il contenuto di LOC1 viene caricato nell'accumulatore e da qui trasferito o "appoggiato" sullo stack, mentre quello di LOC2, dopo essere stato caricato nell'accumulatore, è memorizzato in LOC1. A questo punto, il contenuto del primo byte dello stack (ossia il contenuto originale di LOC1) viene riportato nell'accumulatore e da questo, infine, in LOC2, completando l'operazione di scambio. Si noti che le operazioni di stack salvano in memoria il contenuto di LOC1 solo per il tempo necessario e senza che il programma specifichi esplicitamente una particolare locazione di memoria. Il precedente frammento di programma illustra molti aspetti delle operazioni di stack. Fondamentalmente, esse sono reciproche e sequenziali: l'ultimo dato trasferito nello stack, viene estratto dall'operazione di recupero immediatamente successiva. Una serie di operazioni di appoggio, non intercalate da recuperi, depositano informazioni in locazioni di stack adiacenti, procedendo verso l'alto. Viceversa, un'ininterrotta sequenza di operazioni di recupero estrae i valori dalle locazioni successive a quella indicata dallo stack pointer, ma procedendo verso il basso.

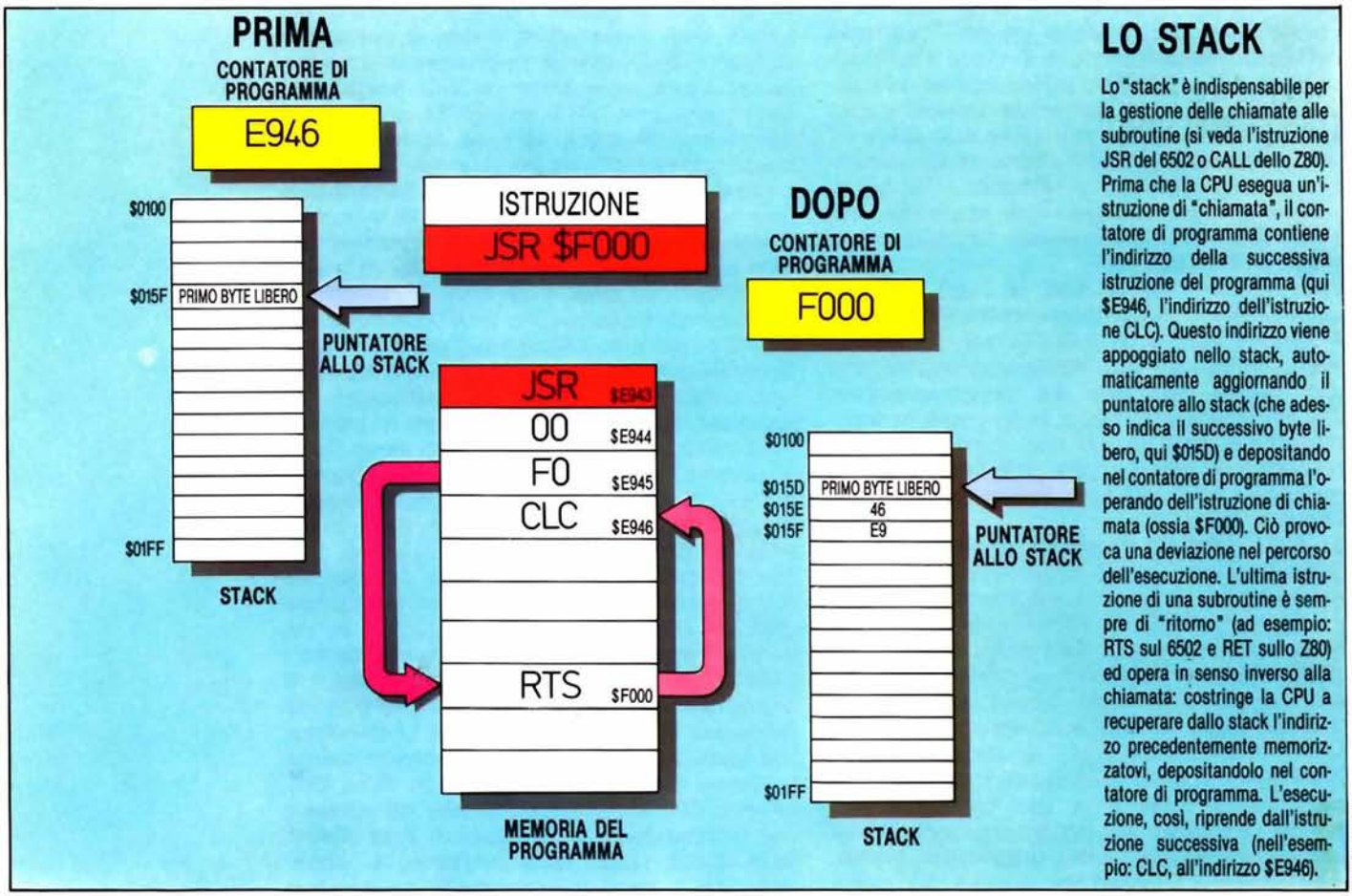
Per chiarirci le idee sullo stack, immaginiamo di prendere alcuni appunti su dei cartoncini, di disporli l'uno sopra l'altro in un contenitore ed, infine, di leggere e scartare i biglietti, finché ne restano nel contenitore. Il biglietto scritto per ultimo è sempre il primo ad essere letto, ossia il primo nel contenitore. Per questo motivo, lo stack è anche conosciuto come struttura LIFO (Last In First Out, ovvero: "Ultimo entrato, primo ad uscire"). Per convenzione, il "successivo byte libero" dello stack viene indicato come la "cima" dello stack, immaginando che lo stack cresca

verso il basso. Ma in realtà, sia nel 6502 che nello Z80, il puntatore di stack viene decrementato da un'istruzione di appoggio (PUSH), cosicché la "cima" dello stack corrisponde ad un indirizzo di memoria più basso, rispetto a ciò che viene considerato il "fondo". Questo concetto è più comprensibile ricorrendo ad un'illustrazione (vedi figura).

Il primo frammento di programma visto è tipico anche perché lo stack è adoperato in modo "bilanciato" (il numero delle istruzioni di appoggio è pari a quello delle istruzioni di recupero). Se tale equilibrio viene trascurato, nella struttura di una subroutine, può verificarsi un errore di ritorno all'uscita dalla subroutine e, di conseguenza, una "caduta" del programma. Questo è uno dei più comuni errori nei programmi in Assembly, ma può essere facilmente individuato verificando il numero delle istruzioni di appoggio e di recupero nelle varie sezioni del programma.

Lo stack dello Z80 presenta una sostanziale differenza da quello del 6502: i valori in esso depositati sono composti da due byte, mentre nel 6502 i valori sono di un singolo byte. Nel caso specifico, operando sull'accumulatore dello Z80, sullo stack finisce anche il contenuto del registro di stato del processore, poiché la CPU tratta questi due registri a singolo byte come un unico registro a due byte, chiamato registro AF (Accumulatore-Flag). La potenza di elaborazione dello Z80 deriva dalla capacità di gestire registri a due byte. È buona abitudine, all'inizio di una subroutine, trasferire nello stack i contenuti di tutti i registri della CPU, per poi recuperarli immediatamente dopo l'uscita da essa. In questo modo, dopo l'esecuzione della subroutine, si ripristinano le condizioni iniziali della CPU e la subroutine potrà usare qualsiasi registro senza rischiare di danneggiare informazioni essenziali per l'intero programma. Consideriamo, ad esempio, la seguente subroutine:

	6502		Z80	
	LDA	LOC1	LD	A,LOC1
SUM	ADC	#$6C	ADC	A,$6C
GSUB	JSR	SUBR0	CALL	SUBR0
TEST	BNE	SUM	JR	NZ,SUM
EXIT	RTS		RET	
SUBR0	PHP		PUSH	AF
	PHA		PUSH	HL
	TXA		PUSH	DE
	PHA		PUSH	BC
	TYA		PUSH	IX
SUBR1	PHA		PUSH	IY
SUBR2	STA	LOC2	LD	(LOC2),A
	LDA	#$00	LD	A,$00
SUBR3	PLA		POP	IY
	TAY		POP	IX
	PLA		POP	DE
	TAX		POP	BC
	PLA		POP	HL
SUBR4	PLP		POP	AF
	RTS		RET	



LO STACK

Lo "stack" è indispensabile per la gestione delle chiamate alle subroutine (si veda l'istruzione JSR del 6502 o CALL dello Z80). Prima che la CPU esegua un'istruzione di "chiamata", il contatore di programma contiene l'indirizzo della successiva istruzione del programma (qui $E946, l'indirizzo dell'istruzione CLC). Questo indirizzo viene appoggiato nello stack, automaticamente aggiornando il puntatore allo stack (che adesso indica il successivo byte libero, qui $015D) e depositando nel contatore di programma l'operando dell'istruzione di chiamata (ossia $F000). Ciò provoca una deviazione nel percorso dell'esecuzione. L'ultima istruzione di una subroutine è sempre di "ritorno" (ad esempio: RTS sul 6502 e RET sullo Z80) ed opera in senso inverso alla chiamata: costringe la CPU a recuperare dallo stack l'indirizzo precedentemente memorizzatovi, depositandolo nel contatore di programma. L'esecuzione, così, riprende dall'istruzione successiva (nell'esempio: CLC, all'indirizzo $E946).

KEVIN JONES

Il risultato delle istruzioni poste tra SUBR0 e SUBR1 è quello di appoggiare sullo stack il contenuto dei vari registri, mentre le istruzioni comprese tra SUBR3 e SUBR4 provvedono a ripristinare i valori originali. Delle due istruzioni che compongono la subroutine che inizia a SUBR2, la seconda non porta ad alcun risultato, poiché le istruzioni successive modificano completamente lo stato dell'accumulatore.
Nelle istruzioni PUSH e POP dello Z80, l'operando può essere una qualsiasi coppia di registri, mentre nel 6502 le istruzioni corrispondenti operano soltanto sull'accumulatore (PHA e PLA) o sul registro di stato del processore (PHP e PLP): da qui la necessità, sul 6502, di usare l'accumulatore come "tramite", mediante istruzioni di trasferimento (TXA, TAX, TYA, TAY). Si noti che abbiamo volontariamente commesso un errore, nella versione Z80, omettendo di recuperare il contenuto di tutti i registri in ordine inverso rispetto a quello di appoggio.
Ciò evidenzia quanto sia importante la precisione nelle operazioni di stack, ma dimostra anche come sia possibile appoggiare il contenuto di un registro sullo stack e da qui trasferirlo poi in un diverso registro. Questa operazione può sembrare contorta, ma è spesso molto efficace per scambiare dati tra registri.
Le funzioni e l'uso dei registri della CPU saranno l'argomento dell'ultima parte del corso, nella quale concluderemo l'analisi generale delle istruzioni del linguaggio Assembly.

ESERCIZI

1) Riscrivere la seconda routine presentata nelle risposte ai precedenti esercizi, per ottenere che il messaggio all'indirizzo LABL1 venga ribaltato, ossia:

LABL1 OIGGASSEM NU È OTSEUQ

Usare lo stack per ottenere il rovesciamento.

2) Sviluppare la routine precedente per ottenere che le parole del messaggio mantengano il loro ordine originale, ma risulti ribaltata ciascuna parola:

LABL1 OTSEUQ È NU OIGGASSEM

Risposte agli esercizi a pag. 717

1) Questa subroutine immagazzina, in ordine decrescente, i numeri da $0F a $00 all'interno di un blocco di $10 byte, riservati dallo pseudo-opcode DS dall'indirizzo LABL1 in poi.

6502			Z80		
ORIGIN	ORG	$7000	ORIGIN	ORG	$C000
LABL1	DS	$10	LABL1	DS	$10
LABL2	DW	$7100	LABL2	DW	$C100
			OFFST	EQU	$0F
BEGIN	LDY	#$FF	BEGIN	LD	IX,LABL1
	LDX	#$10		LD	B,OFFST
LOOP0	INY		LOOP0	LD	(IX+0),B
	DEX			INC	IX
	TXA		ENDLP0	DJNZ	LOOP0
	STA	LABL1,Y		LD	(IX+0),B
ENDLP0	BNE	LOOP0		RET	
	RTS				

La diversità tra lo Z80 ed il 6502, sia nel modo di procedere che nelle istruzioni, è palese. Il 6502 usa il registro Y come indice per l'indirizzo LABL1 ed il registro X come contatore di ciclo e fonte del dato da immagazzinare. Si noti che il registro X viene decrementato due istruzioni prima del test BNE (indirizzo ENDLP0), ma, poiché TXA (Trasferisci all'Accumulatore il contenuto di X) e STA non hanno alcun effetto sul registro di stato del processore, il test di verifica si riferisce al decremento del registro X.
La versione Z80 usa il modo di indirizzamento indiretto, adoperando il registro IX, per conservare l'indirizzo di memoria ed il registro B come contatore e fonte del dato da immagazzinare. All'indirizzo ENDLP0, nella versione Z80, troviamo DJNZ LOOP0, che significa "decrementa il registro B e esegui un salto relativo all'indirizzo LOOP0 se il risultato non è zero". Questa istruzione, che ci ricorda la struttura FOR... NEXT del BASIC, è molto comoda per creare cicli iterativi.

2) Questa routine trasferisce il messaggio immagazzinato all'indirizzo LABL1 nel blocco che inizia con l'indirizzo memorizzato in LABL2. Come codice di "fine messaggio" viene adoperato $0D (il codice ASCII per Return o Enter).

6502			Z80		
ORIGIN	ORG	$7000	ORIGIN	ORG	$C000
LABL1	DB	'THIS IS A MESSAGE'	LABL1	DB	'THIS IS A MESSAGE'
TERMN8	DB	$0D	TERMN8	DB	$0D
LABL2	DW	$7100	LABL2	DW	$C100
CR	EQU	$0D	CR	EQU	$0D
ZPL0	EQU	$FB			
BEGIN	LDA	LABL2	BEGIN	LD	IX,LABL1
	STA	ZPL0		LD	IY,(LABL2)
	LDA	LABL2+1	LOOP0	LD	A,(IX+0)
	STA	ZPL0+1		LD	(IY+0),A
	LDY	$FF		INC	IX
LOOP0	INY			INC	IY
	LDA	LABL1,Y		CP	CR
	STA	(ZPL0),Y	ENDLP0	JR	NZ,LOOP0
	CMP	CR		RET	
ENDLP0	BNE	LOOP0			
	RTS				

La versione 6502 usa il registro Y come indice per l'indirizzo indiretto ZPL0, nel modo indiretto post-indicizzato. Questo metodo è utilizzabile solamente con il registro Y e richiede che l'indirizzo dell'operando si riferisca alla pagina 0. Per questo motivo ZPL0 e ZPL0+1 sono inizializzati con l'indirizzo immagazzinato alla locazione LABL2. Il sistema operativo dei computer con microprocessore 6502 utilizza moltissime locazioni in pagina 0, ma non le locazioni comprese tra $FB e $FF del Commodore 64 e quelle comprese tra $70 e $8F del BBC Micro. Ecco perché ZPL0 è situato in una di queste locazioni. La versione Z80 impiega IX nel modo indicizzato e IY nel modo indicizzato indiretto.
Entrambe le routine contengono l'istruzione "confronta l'accumulatore" (CMP CR per il 6502 e CP CR per lo Z80) nella quale l'operando viene sottratto dal contenuto dell'accumulatore, influenzando così il registro di stato dell'accumulatore (PSR). Il contenuto dell'accumulatore viene quindi ripristinato, mentre PSR mostra il risultato del confronto. Quando l'accumulatore contiene $0D (il messaggio di fine), il confronto genera un risultato 0 e la successiva verifica, situata in ENDLP0, provoca l'esecuzione dell'istruzione di ritorno (RTS o RETURN).

La Camputers

La città di Cambridge, in Gran Bretagna, sta diventando una nuova Silicon Valley



John Shirreff

La Camputers è nata dalla mente di un solo uomo. Nel 1976, Dick Greenwood iniziò a lavorare come progettista di sistemi elettronici e collaboratore esterno di varie aziende, tra le quali la Pye Telecoms. Già alla fine del 1970 egli aveva fondato una propria attività e si era dedicato a lavori specialistici assai progrediti. Successivamente, la sua azienda si occupò anche dello sviluppo del software, creando un "Bar Management System", ossia un programma per la gestione delle scorte di magazzino, particolarmente destinato alle fabbriche di birra.

La compagnia si interessò anche alla progettazione di microcomputer basati sul chip Z80 della Zilog. Nel febbraio del 1981 Greenwood fondò la Camtronic Circuits (in seguito divenuta la Camputers) e, grazie ad alcune sovvenzioni governative, nell'estate del 1981 cominciò a lavorare sull'home computer Lynx (vedere a pag. 70). Il suo obiettivo era quello di "insegnare" allo Z80A "come aggirare i problemi piuttosto che scontrarsi frontalmente con essi".

Il compito di sviluppare gli aspetti hardware del progetto fu assegnato a John Shirreff, un laureato della Cambridge University, con precedenti esperienze nel settore industriale della musica rock. Lo sviluppo del software fu affidato a David Jansons, che redasse la versione BASIC usata dalla nuova macchina. Quando apparve, nel 1982, il Lynx fu giudicato un apparecchio dall'aspetto molto professionale. Caratterizzato da un'ottima linea estetica e da una tastiera di tipo QWERTY, il computer aveva una memoria standard di 48 Kbyte, espandibile fino a 192 Kbyte. L'apparecchio poteva visualizzare fino ad otto diversi colori, con una grafica ad alta risoluzione di 248 × 256 pixel. Inoltre, tramite un altoparlante incorporato, era capace di ottimi effetti audio.

Il Lynx non ha avuto molto successo in Gran Bretagna, ma il successo conseguito all'estero ha spinto la compagnia a proseguire nello sviluppo del progetto. Il Lynx 96, lanciato poco tempo dopo, possedeva più di 37 Kbyte di RAM utente e poteva funzionare con unità a dischetti di 5 e 1/4 pollici. In più, l'apparecchio era dotato di effetti sonori preselezionati su scheda ed era predisposto all'uso di interfacce per stampanti seriali e parallele.

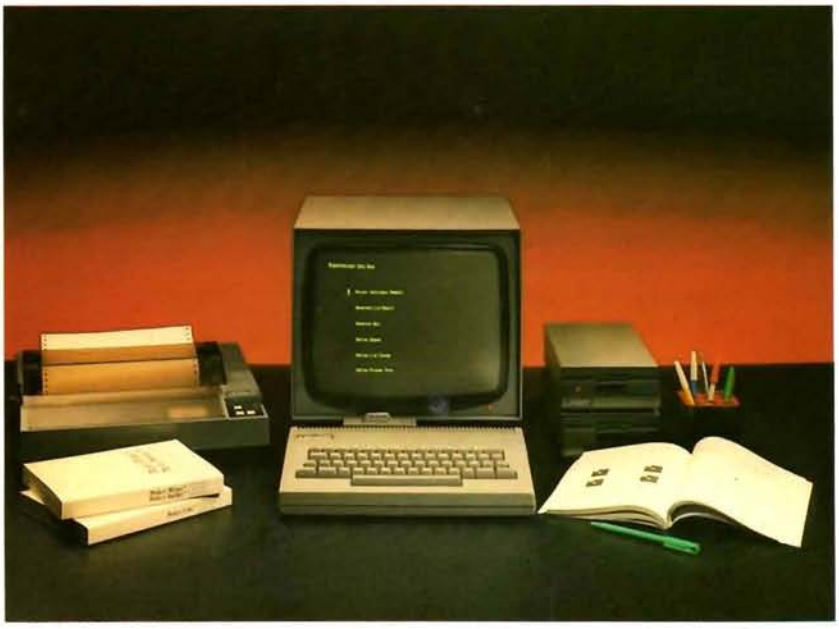
Recentemente, la Camputers ha lanciato una macchina per il mercato dei microcomputer, dedicata alla piccola gestione aziendale. Questo computer, anch'esso basato sul microprocessore Z80, è stato chiamato Lynx Laureate e può utilizzare il sistema operativo CP/M, il che permette all'utente di avvalersi della gran quantità di software CP/M scritto negli ultimi anni. Questo è uno dei punti di forza dell'apparecchio, in quanto pochi utenti sono oggi disposti ad acquistare un apparecchio per usi gestionali, non corredato da un'adeguata quantità di software. Sebbene progettato per usi gestionali, il Laureate è tuttavia compatibile con gli altri modelli Lynx e possiede un "bus" d'espansione a 40 canali, che lo mette in grado di controllare la vasta gamma di periferiche Lynx, compresa una stampante parallela, un joystick ed il software su ROM.

Nonostante questa diversificata produzione di micro, la Camputers sta continuando a sviluppare nuovi modelli. Nell'immediato futuro la compagnia pensa di lanciare una versione alternativa del Laureate. Questo apparecchio sarà disponibile come sistema integrato, diversamente dalla versione modulare del precedente progetto.

La Camputers si propone inoltre di rilanciare sul mercato interno il Lynx 48, col nuovo nome di Leisure. Questo apparecchio punterà al mercato degli home computer e dei computer per giochi, un settore nel quale la compagnia ritiene di non aver riscosso il successo meritato. Infine, la Camputers sta lavorando al progetto di un apparecchio che dovrebbe divenire il concorrente più agguerrito del Sinclair QL.

Sistema gestionale

Il Lynx Laureate è un sistema modulare progettato per usi gestionali. Una memoria su scheda da 128K permette a questo apparecchio di utilizzare il CP/M



per concessione della CAMPUTERS

Divoratori di parole

I computer vengono adoperati sempre di più nell'elaborazione dei testi

In realtà, i word processor non sono che una versione computerizzata delle macchine da scrivere tradizionali: i testi, immessi tramite una tastiera, vengono memorizzati e compaiono su uno schermo, dove apportare modifiche e correzioni è un processo veramente facile. Si passa alla stampa del documento così redatto soltanto quando si è raggiunta una versione definitiva e corretta dello scritto.

Inizialmente, l'unico disagio che si prova con l'uso di un word processor, oltre al solito timore nei confronti di un computer, è l'impiego della tastiera.

Eppure, per un principiante è molto più facile ottenere buoni risultati usando un word processor, che non una macchina da scrivere tradizionale: gli inevitabili pasticci che si combinano con quest'ultima, infatti, sul word processor vengono corretti in pochi istanti, e si ottiene così una copia su carta pulita, priva di errori e di scarabocchi.

Anche quasi tutti gli home computer si possono usare come word processor, ma alcuni sono più adatti di altri. In certi casi ciò è dovuto alla mancanza di buoni programmi, in altri all'inadeguatezza delle periferiche disponibili. La spesa per un sistema di word processing può anche essere piuttosto elevata, poiché gli accessori indispensabili spesso vengono a costare fino al doppio dello stesso computer. In genere, l'acquisto che incide maggiormente è quello per la stampante: se questa non è di buona qualità, l'utilità e l'efficacia dell'intero sistema vengono meno. Infatti, nonostante il proliferare di reti informative computerizzate e di sistemi di posta elettronica, il mezzo cartaceo resta (e resterà per lungo tempo) l'unico metodo veramente valido per lo scambio di informazioni.

Anche le stampanti meno dotate, purtroppo, non sono molto economiche, nonostante la bassa qualità dei risultati. In molti casi, invece, quanto prodotto con il word processor deve essere presentato in una forma gradevole alla vista e perfettamente leggibile. In quanto a questo le stampanti a matrice di aghi non sono le più adatte, mentre le più costose stampanti a margherita costituiscono, sia pure con la loro lentezza di stampa, gli apparecchi che possono dare i migliori risultati.

Alternativamente, esistono macchine da scrivere elettroniche collegabili al computer mediante speciali interfacce. Ciò può rappresentare un notevole risparmio per chi, passando dalla dattilografia tradizionale a quella computerizzata, possieda già una macchina da scrivere elettronica.

Una soluzione spesso adottata da gruppi di amici o da soci di "computer club" è quella di contribuire tutti all'acquisto di una buona stampante. Ogni possessore di computer provvede poi a procurarsi l'interfaccia adatta oppure, se questa è già incorporata, il semplice cavo di connessione. Da queste forme di

Il micro BBC

Questo computer, affiancato dall'espansione per dischi "Torch", consente l'uso di software commerciale di ottima qualità, compreso il notissimo Wordstar della Micropro. Tuttavia, il costo totale (incluse le varie espansioni) può risultare superiore a quello di molti altri personal computer. Ovviamente, ci si può accontentare di una stampante di media qualità e del "Perfect Writer" (il word processor fornito con l'espansione "Torch")



IAN McKINNELL

utilizzo misto sono però esclusi i computer Commodore ed Atari, in quanto adoperano soltanto stampanti della propria marca. Per lo Spectrum della Sinclair è necessario l'acquisto di una speciale interfaccia, poiché non ne ha una incorporata.
Disponendo di una stampante adeguata, il successivo problema da affrontare riguarda la scelta del programma di word processing. Per i microcomputer più diffusi la scelta è molto ampia, ma si riduce sensibilmente nel caso di apparecchi meno venduti. Inoltre, anche la qualità del software varia notevolmente da caso a caso: ci sono programmi dotati di appena qualche funzione rudimentale ed altri, invece, forniti di un'ampia gamma di possibilità (una tra le più apprezzabili è il controllo di "giustezza" automatico, ossia l'inserimento automatico, da parte del programma, di opportune spaziature per dare margini uniformi a tutto il testo).
Un'altra comoda funzione è la ricerca di particolari parole nel corpo del testo (talvolta accompagnata dalla possibilità di operare una sostituzione automatica con una parola nuova).
In aggiunta, esistono dei programmi "accessori" (per adesso adatti soltanto alla lingua inglese, però) che controllano l'esatta compitazione di ciascuna parola del testo, segnalando eventuali errori.
Poiché alcune stampanti a matrice di aghi dispongono di più corpi di stampa (normale, corsivo, espanso, ecc.), alcuni dei word processor più sofisticati prevedono comandi per selezionarne più d'uno all'interno dello stesso brano di testo. Addirittura, con certi programmi di word processing è possibile anche sfruttare le capacità grafiche della stampante, consentendo di creare ed usare caratteri di forma completamente diversa.
In genere, le stampanti a margherita permettono anche un metodo di stampa detto "a spaziatura proporzionale", nel quale la larghezza occupata da ciascun carattere non è fissa, ma varia a seconda della lettera o del numero stampati. Anche in questo caso, alcuni word processor sono capaci di sfruttare questa possibilità, producendo una stampa molto più gradevole all'occhio, senza peraltro rinunciare alla corretta giustezza dei due margini. Questa capacità è ideale per produrre lettere o anche piccoli manuali di ottima qualità senza dover ricorrere a mezzi tipografici tradizionali. Tuttavia, il progresso tecnologico ha investito anche il settore tipografico e già alcune ditte di fotocomposizione accettano dischetti magnetici da computer, sui quali si trovano lavori preparati da un word processor. Ciò risparmia il tempo e la fatica di dover ribattere l'intero testo per ottenere la composizione definitiva.
Il software per word processing viene venduto sotto le più varie forme: nastri, dischetti, cartucce ROM, e via dicendo, ma il fatto più importante riguarda come viene memorizzato il testo prodotto. Generalmente ciò avviene su nastro o su dischetto. Benché economico l'uso di nastri è però lento e laborioso oltre a limitare, generalmente, la lunghezza dei testi da elaborare. L'impiego di dischi risulta più efficace, perché non soffre di simili limitazioni. Nell'ambito degli home computer, altri sistemi (quali i Microdrive della Sinclair) costituiscono un buon compromesso tra costo e prestazioni, sebbene il software attualmente disponibile non preveda ancora l'uso di tali unità.
Interessante è il sistema a nastri adottato nel Coleco Adam, un home computer dotato di stampante a margherita, nel cui progetto è stato evidentemente tenuto conto del word processing.
La possibilità di memorizzare lunghi testi su nastro o su disco ne permette la stesura e l'elaborazione anche a più riprese ed in momenti diversi, come può accadere per lettere o documenti generici nei quali basta

Lo Spectrum della Sinclair
Questo rappresenta il più economico sistema di word processing di una discreta qualità. Si devono fare i conti, tuttavia, con una tastiera non particolarmente adatta a questo impiego e con l'uso di nastri a cassetta che rallentano molto il lavoro. Esiste anche un programma, il "Tasword Two", che prevede l'impiego dei Microdrive



IAN McKINNELL

variare pochi elementi. In tal caso è sempre bene mantenere una copia d'archivio degli scritti, per evitarne la perdita accidentale durante l'uso.
I comandi di alcuni word processor sono semplicissimi, altri sono invece costituiti da particolari combinazioni di tasti più difficili da tenere a mente. In certi apparecchi (come ad esempio lo Spectrum) la qualità della tastiera non è proprio ideale per un impiego orientato all'elaborazione di testi. In genere, le tastiere dotate di più tasti di funzione facilitano il lavoro, poiché ad essi possono essere associati i comandi adoperati più frequentemente. Anche il tipo di display influisce molto: certi computer visualizzano soltanto parti esigue di testo in una sola volta. Il Commodore Vic-20, ad esempio, visualizza solo 22 caratteri per riga, mentre nella maggior parte delle macchine professionali le righe sono composte da 80 caratteri. I risultati migliori, nel word processing, si hanno con quegli apparecchi che adoperano monitor sui quali visualizzare almeno 25 righe di 80 colonne.
Anche la forma e l'aspetto dei singoli caratteri sono fattori importanti, che variano da un microcomputer all'altro. Più sono i pixel (o punti) adoperati per definire ciascun carattere, più nitido e leggibile risulta il testo riportato sullo schermo, affaticando meno la vista dell'operatore.
In genere la larghezza della griglia di punti è di 8 pixel, ma esistono anche apparecchi professionali in cui la griglia misura ben 16 × 16 punti.
Finché le applicazioni di word processing sono di entità limitata, anche un piccolo home computer può bastare, ma in caso contrario un sistema tipico è composto da: un'unità centrale, un buon monitor, una stampante di ottima qualità, due unità a dischetti e una buona tastiera.
Naturalmente, occorre anche un word processor adatto al lavoro che intendiamo svolgere. Ma raggiungere un simile livello, aggiungendo accessori di vario genere ad un home computer, può rivelarsi spesso più dispendioso dell'acquisto di un apparecchio professionale.
I microcomputer per uso professionale hanno, indiscutibilmente, maggiori pregi: oltre ad un hardware di qualità superiore, esiste una vastissima scelta nel software. Ciò è possibile perché viene generalmente adottato uno stesso sistema operativo anche su apparecchi di marca diversa.
Il word processor di maggior successo, in questo settore, è il Wordstar (prodotto dalla Micropro), disponibile per i sistemi operativi CP/M, CP/M-86 ed MS-DOS. Il programma è molto sofisticato e flessibile, ma, ovviamente, il suo prezzo è piuttosto alto (fino a 20 volte quello dei word processor per home computer).
La spesa per un sistema professionale è notevole, specialmente per una piccola ditta commerciale o uno scrittore, ma orientarsi verso un home computer quando la quantità di testi da elaborare è ingente, costituisce un risparmio fittizio: il tempo e le energie sprecate su di un sistema limitato si fanno immediatamente sentire.
Del resto, la spesa per dotarsi di un sistema professionale cala continuamente e la possibilità di adoperare sistemi operativi standard (quale il CP/M) risulta veramente valida. Per di più, molti personal computer vengono oggi venduti già corredati di word processor di buona qualità (talvolta anche dello stesso Wordstar), senza ulteriore aggravio nel prezzo.
Infine, la più recente innovazione è il word processor "da viaggio", ossia montato sui piccoli e compatti computer da valigetta. È dunque prossima l'estinzione non solo della macchina da scrivere, ma anche della classica "carta e penna"?

Il Commodore 64

Per questo home computer è disponibile un word processor su disco dal prezzo più che ragionevole. Tuttavia, qualche limitazione è dovuta allo schermo, che può visualizzare soltanto 40 caratteri per riga. È possibile adoperare una stampante economica, a matrice di aghi, ma la qualità di stampa è decisamente scarsa per usi commerciali



IAN McKINNELL

Sottomarini ed esplosioni

Dalla creazione degli sprite ad un gioco completo per il Commodore 64

Gli sprite sono delle sagome grafiche mobili sullo schermo e vengono definiti in modo analogo ai caratteri UDG (caratteri definiti dall'utente), di cui si è parlato a pag. 713, sia pure usando una griglia di maggiori dimensioni. Nel Commodore 64, una volta definito uno sprite, il colore e la posizione sullo schermo ad esso associati sono controllati da speciali registri del chip VIC.

La griglia di uno sprite è composta da 21 righe di 24 pixel. I pixel di ciascuna riga vengono raggruppati in segmenti di otto bit; ogni riga occupa pertanto tre byte di memoria ed occorrono 63 byte per l'intera definizione. Come già detto per gli UDG, anche negli sprite ogni singolo pixel viene illuminato se il bit corrispondente vale 1 o spento se vale 0. Possiamo quindi calcolare i valori decimali equivalenti ad ogni byte di ciascuna riga dello sprite. I diagrammi qui illustrati mostrano quattro sprite, utilizzati nel gioco della "Caccia al Sottomarino". I numeri riportati nelle rispettive colonne, a fianco delle griglie, costituiscono gli equivalenti decimali dei tre byte di ogni riga e vengono usati nelle frasi DATA del programma (si vedano le linee da 6000 a 6370).

Bersaglio mobile

Uno sprite è definito da 21 righe di tre byte ciascuna, che rappresentano la configurazione di bit occorrenti. I valori usati nei programmi BASIC vengono però espressi in notazione decimale. Nelle seguenti illustrazioni, a fianco della sagoma, sono riportati i valori in decimale relativi a ciascuna riga. Il programma, mediante delle POKE, deposita queste sequenze in una particolare zona della memoria, dove vengono poi interpretate dal chip VIC

La memorizzazione degli sprite

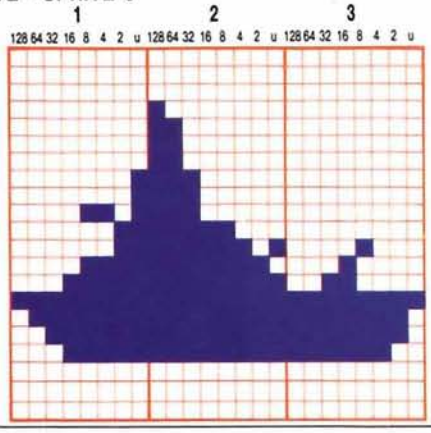
Dopo aver definito uno sprite ed inseriti i valori in frasi DATA, queste devono essere lette (READ) ed i valori depositati nella memoria con delle POKE. A tal scopo si possono usare diverse aree di memoria. Ad esempio, si potrebbe partire dalla locazione 12288, all'interno della parte normalmente adoperata per conservare i programmi BASIC (dalla locazione 2048 alla 40960). Poiché l'occupazione di questa zona cresce con l'aumentare delle dimensioni del programma, quest'ultimo dovrebbe essere lungo più di 10 Kbyte, prima di interferire con la definizione dello sprite. Tuttavia, poiché l'esecuzione di un programma implica l'uso di variabili e queste vengono memorizzate proprio nella zona di memoria che và dalla fine del codice relativo al programma al margine superiore della memoria, il rischio che la definizione dello sprite venga "sporcata" da valori estranei è molto alto.

Per risolvere questo problema basta accorciare il margine superiore della memoria riservata al programma BASIC, lasciando sufficiente spazio per le definizioni. Il computer, a questo punto, crederà di aver meno memoria a propria disposizione e le definizioni si troveranno in una posizione protetta. Il puntatore al limite superiore della memoria è situato nelle locazioni 55 (byte basso) e 56 (byte alto). Normalmente, queste due locazioni contengono, rispettivamente, i valori 0 e 160, che assieme indicano l'indirizzo 40960. In notazione basso-alto, l'indirizzo 12288 è espresso come 0 e 48 e possiamo depositare questa coppia di valori (POKE) nella locazione riservata al puntatore (si veda la linea 90 del programma).

I puntatori agli sprite

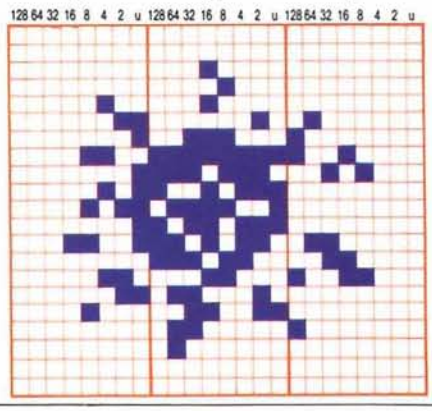
Siccome la definizione di uno sprite può risiedere ovunque in memoria, al computer occorre un puntatore per sapere da dove inizi ciascuna definizione. Il Commodore 64 dispone di otto puntatori riservati a tal scopo, situati nelle locazioni da 2040 (per lo sprite 0) a 2047 (sprite 7). Il valore di ogni puntatore è legato alla posizione in memoria della definizione secondo questa formula: indirizzo ini-

NAVE · SPRITE 0



1	2	3
0	0	0
0	0	0
0	0	0
0	128	0
0	192	0
0	192	0
0	192	0
1	224	0
1	224	0
13	224	0
3	248	128
3	253	8
15	254	16
31	255	48
255	255	255
127	255	254
63	255	254
31	255	252
0	0	0
0	0	0
0	0	0

ESPLOSIONE · SPRITE 1



1	2	3
0	0	0
0	0	0
0	16	0
0	8	0
4	16	0
3	2	64
1	56	128
12	255	144
1	238	40
5	151	0
11	121	0
1	183	0
25	214	96
0	236	48
6	24	152
3	98	0
8	51	0
0	96	128
0	64	0
0	0	0
0	0	0

ziale dei 63 byte di dati = (puntatore allo sprite) × 64. Nel programma, i dati per la nave (sprite 0) iniziano dalla locazione 12288, cosicché il relativo puntatore (nella locazione 2040) vale 192 (= 12288/64). Il blocco di dati successivo si riferisce all'esplosione (sprite 1). Se poniamo il valore del puntatore contenuto nella locazione 2041 a 193, allora i dati devono iniziare dalla locazione 12352. Ecco i valori da noi impiegati:

Numero Sprite	0	1	2	3
Puntatore	192	193	194	195
Area per la definizione (63 byte)	da 12288 a 12350	da 12352 a 12414	da 12416 a 12478	da 12480 a 12542

Si noti che un byte, alla fine di ciascun blocco, non viene usato. La parte del programma relativa alla lettura dei dati ed all'assegnazione dei puntatori si trova tra le linee 2000 e 2210.

Manipolazione degli sprite

Il chip VIC possiede diversi registri, usati per il controllo degli sprite. La prima locazione del VIC è la 53248 ed è più comodo riferirsi a tutte le altre posizioni dei registri, con un valore relativo a questo indirizzo. Se poniamo V = 53248, infatti, la locazione successiva (la 53249) viene espressa come V+1 e via dicendo. V dovrebbe esser definita all'inizio del programma (vedere la linea 100).
Il colore di ciascun sprite è assegnato depositando (con una POKE) il corrispondente codice di colore, da 0 a 15, in uno speciale registro, tra quelli contenuti nelle locazioni da V+39 a V+46 e destinati agli otto sprite. Ad esempio, per colorare la nave di nero, basta depositare il valore 0 in V+39. Altrettanto vale per gli altri sprite (vedere le linee da 2220 a 2250).
Il posizionamento degli sprite sullo schermo viene ampiamente spiegato nella lezione successiva. Per adesso basti sapere che la coordinata x dello sprite 0 è contenuta nella locazione V, la coordinata y nella locazione V+1, le coordinate dello sprite 1 nelle locazioni V+2 e V+3, e così via fino alla locazione V+15 (vedere le linee da 2260 a 2280).

Le dimensioni degli sprite possono essere raddoppiate orizzontalmente, verticalmente o in entrambe le direzioni ed è ciò che facciamo nel programma con gli sprite della nave e dei sottomarini per allungare orizzontalmente le sagome. La locazione del VIC che si occupa dell'espansione orizzontale è la V+29, di uso estremamente facile. Anziché adoperare otto diversi registri, per definire gli attributi di ciascuno sprite, la funzione viene semplicemente abilitata o disabilitata, ponendo ad 1 oppure a 0 i bit del registro. La seguente tabella mostra le POKE necessarie per ingrandire i quattro sprite del programma:

Numero sprite	7	6	5	4	3	2	1	0
Contenuto di V+29	0	0	0	0	1	1	1	1

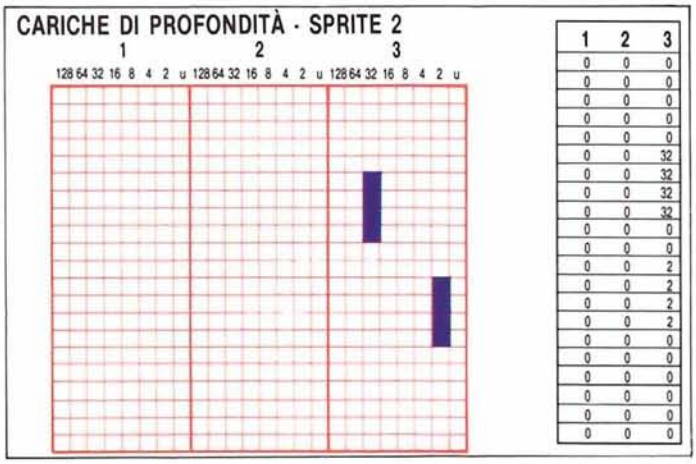
(= −15 decimale)

L'espansione in verticale è controllata mediante il registro V+23: lo sprite dell'esplosione viene raddoppiato in entrambe le direzioni, come si nota nelle linee da 2290 a 2310.
Per l'ultimo compito, che riguarda l'accensione dei vari sprite, basta agire sul bit di un singolo registro: il V+21. Inizialmente, nel programma, vengono attivati soltanto gli sprite della nave e del sottomarino (linee da 2310 a 2360).
Trascritto il programma, possiamo verificare l'esatta lettura dei dati per gli sprite, interrompendo l'esecuzione con RUN/STOP, non appena compare il contatempo sullo schermo, e digitando le seguenti istruzioni (senza però inserire alcun numero di linea), che servono a visualizzare gli sprite:

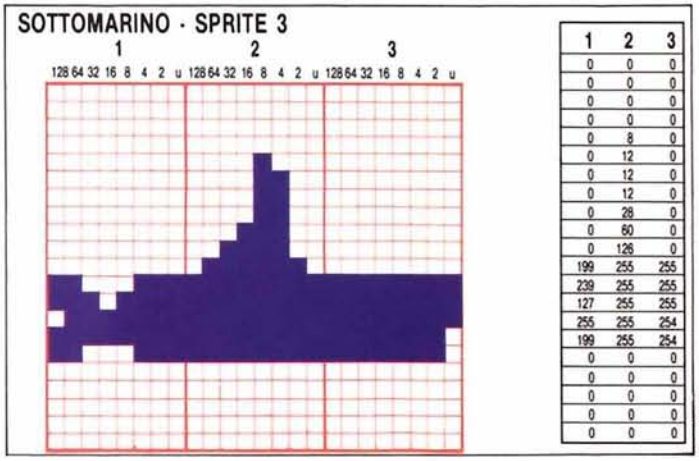
POKEV,160	(Coordinata della nave)
POKEV+2,240:POKEV+3,100	(Coordinate x,y dell'esplosione)
POKEV+4,160:POKEV+5,100	(Coordinate x,y dei siluri)
POKEV+6,100:POKEV+7,100	(Coordinate x,y del sottomarino)
POKEV+21,15	(Azionamento degli sprite 0 — 3)

Prima di lanciare il programma, è opportuno memorizzarlo su nastro. Se il programma si interrompe con un messaggio "OUT OF DATA ERROR", si controlli di aver trascritto accuratamente le frasi DATA. Devono esserci 63 dati per ciascun sprite. Accertarsi, inoltre, di aver inserito l'assegnamento di V (linea 100).

```
1 REM *** GRAFICA DEL C64 ***
90 POKE 55,0:POKE 56,48:CLR:REM ACCORCIA
   MEMORIA
100 V=53248:FL=0:SC=0
110 GOSUB 1000:REM PREPARAZIONE SCHERMO
120 GOSUB 2000:REM CREAZIONE SPRITE
2000 REM *** CREAZIONE SPRITE ***
2020 REM ** LETTURA DATI NAVE **
2030 FOR I=12288 TO 12350
2040 READ A:POKE I,A:NEXT I
2060 REM *** LETTURA DATI ESPLOSIONE ***
2070 FOR I=12352 TO 12414
2080 READ A:POKE I,A:NEXT I
2100 REM * LETTURA DATI CARICHE *
2110 FOR I=12416 TO 12478
2120 READ A:POKE I,A:NEXT I
2140 REM * LETTURA DATI SOTTOMARINO *
2150 FOR I=12480 TO 12542
2160 READ A:POKE I,A:NEXT I
2180 REM ** INIZIALIZ. PUNTATORI **
2190 POKE 2040,192:POKE 2041,193:
     POKE 2042,194:POKE 2043,195
2220 REM ** INIZIALIZ. COLORI **
2230 POKE V+39,0:POKE V+40,1:POKE V+41,0:
     POKE V+42,0
2260 REM ** COORDINATE NAVE **
2270 POKE V+1,80:X0=160
2290 REM ** ESPANSIONE SPRITE **
2300 POKE V+29,15:POKE V+23,2
2320 REM ** ACCENSIONE SPRITE **
2330 POKE V+21,9
2340 RETURN
2350 :
6000 REM ** DATI NAVE **
6010 DATA 0,0,0,0,0,0,0,0,0
6020 DATA 0,128,0,0,192,0,0,192,0
6030 DATA 0,192,0,1,224,0,1,224,0
6040 DATA 13,224,0,3,248,128,3,253,8
6050 DATA 15,254,16,31,255,48,255,255,255
6060 DATA 127,225,254,63,255,254,31,255,
     252
6070 DATA 0,0,0,0,0,0,0,0,0
6100 REM ** DATI ESPLOSIONE **
6110 0,0,0,0,0,0,0,16,0,0,8,0,4,16
6120 DATA 0,3,2,64,1,56,128,12,255,144
6130 DATA 1,238,40,5,151,0,1,12,0,1
6140 DATA 183,0,25,214,96,0,236,48,6,24
6150 DATA 152,3,98,0,8,51,0,0,96,128,0
6160 DATA 64,0,0,0,0,0,0,0,0
6200 REM * DATI CARICHE PROFONDITA' *
6210 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6220 DATA 0,0,0,32,0,0,32,0,0,32,0,0,32,0
6230 DATA 0,0,0,0,0,0,0
6240 DATA 2,0,0,2,0,0,2,0,0,2,0,0
6250 DATA 0,0,0,0,0,0,0,0,0
6260 DATA 0,0,0,0,0,0,0,0,0
6300 REM ** DATI SOTTOMARINO **
6310 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6320 DATA 0,8,0,0,12,0,0,12,0
6330 DATA 0,12,0,0,28,0,0,60,0
6340 DATA 0,126,0,199,255,255
6350 DATA 239,255,255,127,255,255
6360 DATA 255,255,254,199,255,254
6370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```



1	2	3
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	32
0	0	32
0	0	32
0	0	32
0	0	0
0	0	0
0	0	2
0	0	2
0	0	2
0	0	2
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0



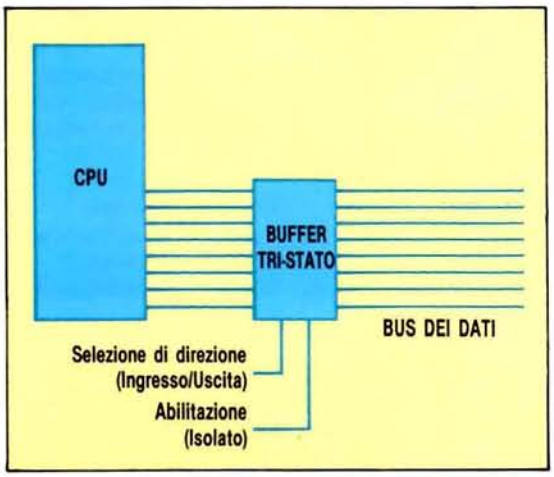
1	2	3
0	0	0
0	0	0
0	0	0
0	0	0
0	8	0
0	12	0
0	12	0
0	12	0
0	28	0
0	60	0
0	126	0
199	255	255
239	255	255
127	255	255
255	255	254
199	255	254
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

LIZ DIXON

A spasso per la memoria

L'esecuzione di un'istruzione in codice macchina richiede più fasi distinte

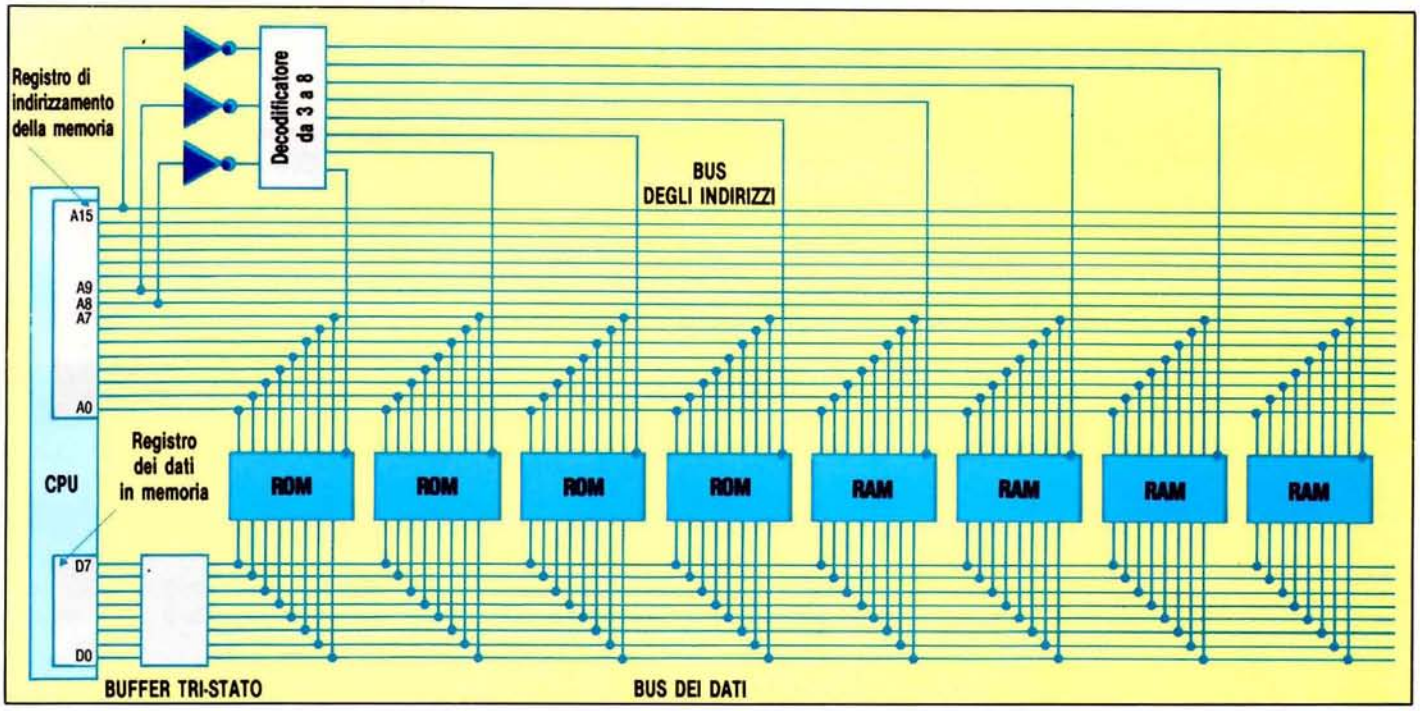
Ogni locazione della memoria di un computer è generalmente formata da otto bit. I dati vengono trasferiti, otto bit alla volta, nella CPU, dove sono elaborati e manipolati in base alle istruzioni dei programmi. I dati possono anche percorrere il cammino inverso, onde memorizzare il risultato di un'operazione. L'istruzione Assembly LDA $1234 è un esempio del primo tipo di trasferimento, mentre STA $1234 è un esempio del secondo.
Il bus dei dati che è il canale attraverso il quale scorrono i dati consente il flusso nelle due direzioni. In certe occasioni, però, è utile isolare la CPU dal bus dei dati; pertanto ci sono tre possibili condizioni per ogni linea del bus: in ingresso, in uscita o isolata. Per selezionare una tra queste condizioni, ogni linea del bus è dotata di un piccolo circuito, chiamato dispositivo "tri-stato".



Per comodità, otto dispositivi "tri-stato" vengono racchiusi in un solo circuito integrato. Il diagramma qui sopra mostra come il circuito integrato si innesta tra il bus dei dati e la CPU. Nell'illustrazione sono anche indicate le linee di "abilitazione" e di "selezione di direzione", grazie alle quali si mette il dispositivo in grado di funzionare a seconda delle necessità. Simili circuiti si possono anche impiegare per collegare periferiche al bus dei dati.
Ogni volta che desideriamo estrarre il contenuto di una locazione di memoria, adoperiamo il suo "indirizzo", ossia il numero che identifica la particolare cella di memoria ROM o RAM. Vediamo come, a livello di hardware, sia possibile selezionare una particolare locazione, onde effettuare un trasferimento di dati.
I microprocessori hanno un secondo canale di comunicazione tra CPU e memoria, il cosiddetto "bus degli indirizzi", che, normalmente, contiene 16 linee. Ciò significa che le combinazioni binarie su tali linee sono 65.536 (2^{16} = 65.536). In altre parole, con un bus di 16 linee possiamo indirizzare 64 Kbyte di memoria. La memoria può essere considerata come composta da più blocchi, ognuno di 256 byte e la parte "bassa" dell'indirizzo (gli otto bit meno significativi) possono servire ad individuare una particolare cella di ciascun blocco. Il blocco viene selezionato in base al valore dei primi otto bit (parte "alta") dell'indirizzo. Se supponiamo di avere un microprocessore dotato di una memoria di 2 Kbyte, è facile vedere come avvenga l'indirizzamento. Poiché ogni blocco è composto da 256 locazioni, i blocchi saranno 8. Ai fini dell'esempio, supponiamo anche che la memoria sia distribuita equamente tra RAM e ROM.
L'indirizzo di una particolare locazione cui accedere è contenuto in uno speciale registro a 16 bit della CPU, chiamato, appunto, "registro di indirizzamento della memoria". Siccome gli otto bit meno significativi dell'indirizzo selezionano una locazione all'interno di un blocco, le corrispondenti otto linee vengono collegate a tutti i blocchi di memoria. Per la selezione del blocco appropriato, bastano altri tre bit (2^3 = 8). Questo codice binario a tre cifre va convertito in modo da selezionare una tra otto possibili linee (una per blocco di memoria).
Il secondo diagramma mostra come le connessioni tra la CPU ed i vari blocchi di memoria vengano realizzate mediante il bus degli indirizzi e quello dei dati. A ciascun blocco è collegata un'uscita del circuito per la decodificazione (da tre bit a otto linee). Se volessimo aggiungere altri blocchi di memoria, sarebbe sufficiente sfruttare i restanti bit della parte alta dell'indirizzo.

Istruzioni in codice macchina

Adesso vediamo come la CPU esegue le istruzioni in codice macchina, che, di norma, occupano locazioni consecutive della memoria. Un'istruzione può occupare da una a quattro locazioni. Prendiamo ad esempio l'istruzione ADD $13FF, che significa "somma il contenuto della locazione con indirizzo esadecimale $13FF all'accumulatore". Per questa istruzione occorrono tre byte: uno per contenere il codice distintivo dell'operazione somma, e due per l'indirizzo $13FF, di 16 bit. Supponiamo che

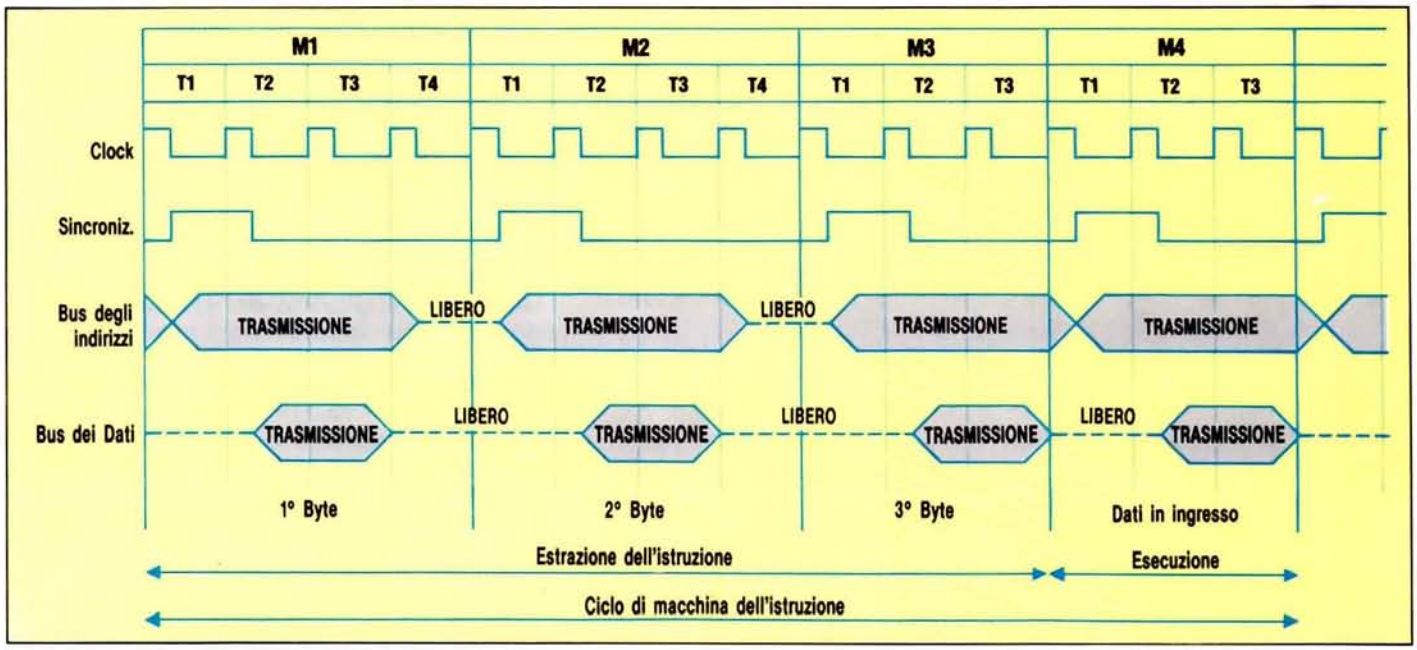


le locazioni siano $1000, $1001 e $1002; prima di poter eseguire l'istruzione, essa va estratta dalla memoria (in inglese è adoperato il termine "fetch", molto ricorrente nella letteratura tecnica sui computer). Tale estrazione è composta da tre distinte fasi di accesso ai bus, al termine delle quali l'istruzione viene depositata in uno speciale registro della CPU, pronta ad essere eseguita. La particolare istruzione presa ad esempio, implica però una seconda operazione di accesso alla memoria, per estrarre il contenuto di $13FF, in modo da poterlo sommare all'accumulatore.

Tutti i fabbricanti di computer pubblicano le caratteristiche dei propri processori, sotto forma di diagrammi di temporizzazione, che illustrano il susseguirsi degli eventi necessari alle varie operazioni. Proviamo a tracciare un simile diagramma relativo ai cicli di "fetch" e di esecuzione di un'istruzione in codice macchina. Tutte le funzioni sono scandite dal clock di sistema (vedere a pag. 726) ed il nostro diagramma mostra come il fronte di salita del segnale di sincronizzazione abiliti il bus degli indirizzi, mentre il fronte di discesa abiliti il bus dei dati. A sua volta, il segnale di sincronizzazione è innescato dal fronte di discesa del primo impulso di clock nella fase operativa, o "ciclo di macchina". Tali cicli sono di diversa durata, poiché il processore impiega più tempo a decodificare il byte dell'istruzione (op-code), che non a trattare i byte che formano l'operando. Inoltre, l'op-code và decodificato immediatamente, per conoscere l'esatto numero di byte che fanno parte della stessa istruzione.

Estrazione ed esecuzione

Un'istruzione in codice macchina che consista in un op-code, seguito da due byte di operando viene trattata in un "ciclo di macchina" che comprende le fasi di estrazione (fetch) e di esecuzione (execute). Durante la prima fase, il bus degli indirizzi accede alle locazioni di memoria che contengono l'istruzione, ed il bus dei dati trasferisce i byte che la compongono nella CPU. Subito dopo, i due bus sono di nuovo impegnati ad accedere alla memoria per estrarre il contenuto della locazione specificata nella stessa istruzione



LIZ DIXON

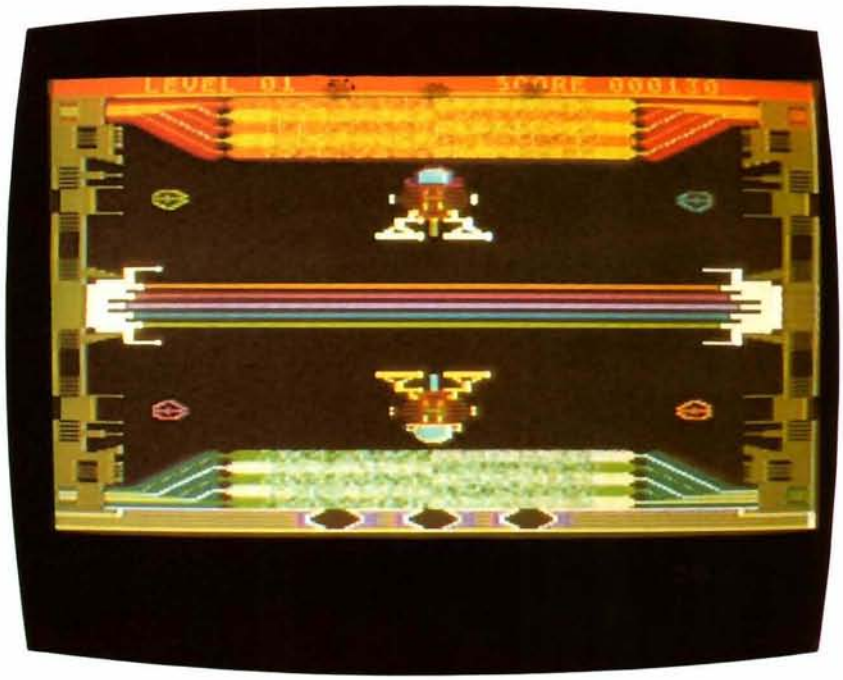
Software in vetrina

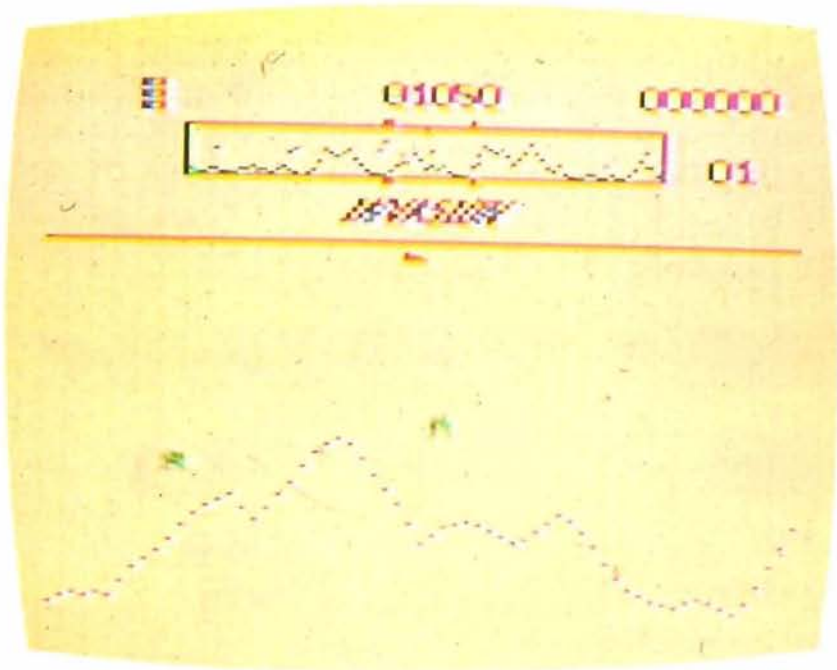
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

I giochi di tipo "arcade" riscuotono sempre un gran successo, soprattutto perché sono, in genere, ricchi d'azione e di imprevisti ed offrono un po' di brivido al giocatore.

Wal war per l'Atari 48K

In questo gioco, la cui grafica è ricca di brillanti colori, due "Microbot" (incrocio tra micro e robot) si affrontano, sparandosi attraverso un muro. Gradualmente il muro si sgretola, ed i colpi riescono finalmente a raggiungere la "stanza" dell'avversario. Il giocatore controlla uno dei due Microbot (è previsto anche l'impiego di joystick). Prodotto dalla "Sierra-Online", questo programma è fornito su dischetto.





Planet Invasion per il Dragon

Il gioco è uno dei "classici" da sala giochi: un'astronave, pilotata dal giocatore, sorvola un terreno accidentato e gli alieni tentano in tutti i modi di abbatterla. Fortunatamente, il giocatore dispone di numerose risorse: cannoni a lungo raggio, bombe ad altissimo potenziale e perfino la possibilità di tuffarsi nell'iperspazio (se le cose si mettono proprio male!).
Il gioco è fornito in due versioni: a colori oppure in bianco e nero (in cui la grafica è molto più sofisticata). Prodotto dalla Microdeal, questo gustosissimo gioco richiede l'impiego di joystick.

L'Atmos della Oric

Dopo il limitato successo dell'Oric-1, ecco una nuova versione più aggiornata

Dotato di un poderoso BASIC (stile Microsoft), un'interfaccia Centronics incorporata ed una presa monitor RGB, l'Oric-1 (vedere a pag. 50), rappresentava già, ai tempi del suo lancio, un buon investimento. Tuttavia, l'apparecchio non ha ottenuto il successo sperato, sia per il software disponibile, sia a causa di alcuni irritanti aspetti dei programmi in ROM.

La Oric Products International ha ora eliminato gli inconvenienti nelle ROM, fornendo il computer di un nuovo involucro e di un nuovo nome: Oric Atmos. La vecchia tastiera (tipo calcolatrice) è stata sostituita con una di tipo professionale, molto più comoda e funzionale e l'intero design è ora più attraente. Ai soliti tasti, che conservano la disposizione dell'Oric-1, è stato aggiunto un tasto di funzione, per adesso non utilizzato ma disponibile per "future espansioni".

Il microprocessore adoperato per l'Atmos è il notissimo chip 6502 e la memoria RAM disponibile per i programmi BASIC nella configurazione di base è di 37 Kbyte. La grafica è ad otto colori, con una risoluzione di 240 × 200 pixel. Le definizioni dei vari caratteri e simboli sono mantenute in RAM, consentendo al programmatore di modificarle a piacimento. Esiste anche un secondo insieme di caratteri, coi quali è possibile creare una grafica a blocchi, del genere teletext. Anziché conservare gli attributi relativi alla grafica in RAM (come ad esempio accade nello Spectrum), l'Atmos mantiene una sequenza di "attributi seriali".

Questo sistema consente un risparmio di memoria, ma occorre prestar attenzione nella definizione di videate sullo schermo, poiché gli attributi vengono visualizzati come caratteri spazio.

L'Atmos può emettere quattro diversi suoni, adatti a giochi tipo "arcade": ZAP, PING, SHOOT e EXPLODE sono i relativi comandi. Per la riproduzione di brani musicali, sono disponibili i comandi MUSIC, PLAY e SOUND, che agiscono sul sofisticato chip generatore di suoni. Il volume



IAN McKINNELL

Il sistema Atmos

L'Atmos della Oric è un home computer dal prezzo molto ragionevole, pur essendo dotato di 48 Kbyte di memoria, di una grafica a colori e di un sofisticato generatore di effetti audio. La Oric produce due accessori per l'Atmos: un'unità a microdischetti da 3 pollici ed una stampante/plotter a 4 colori



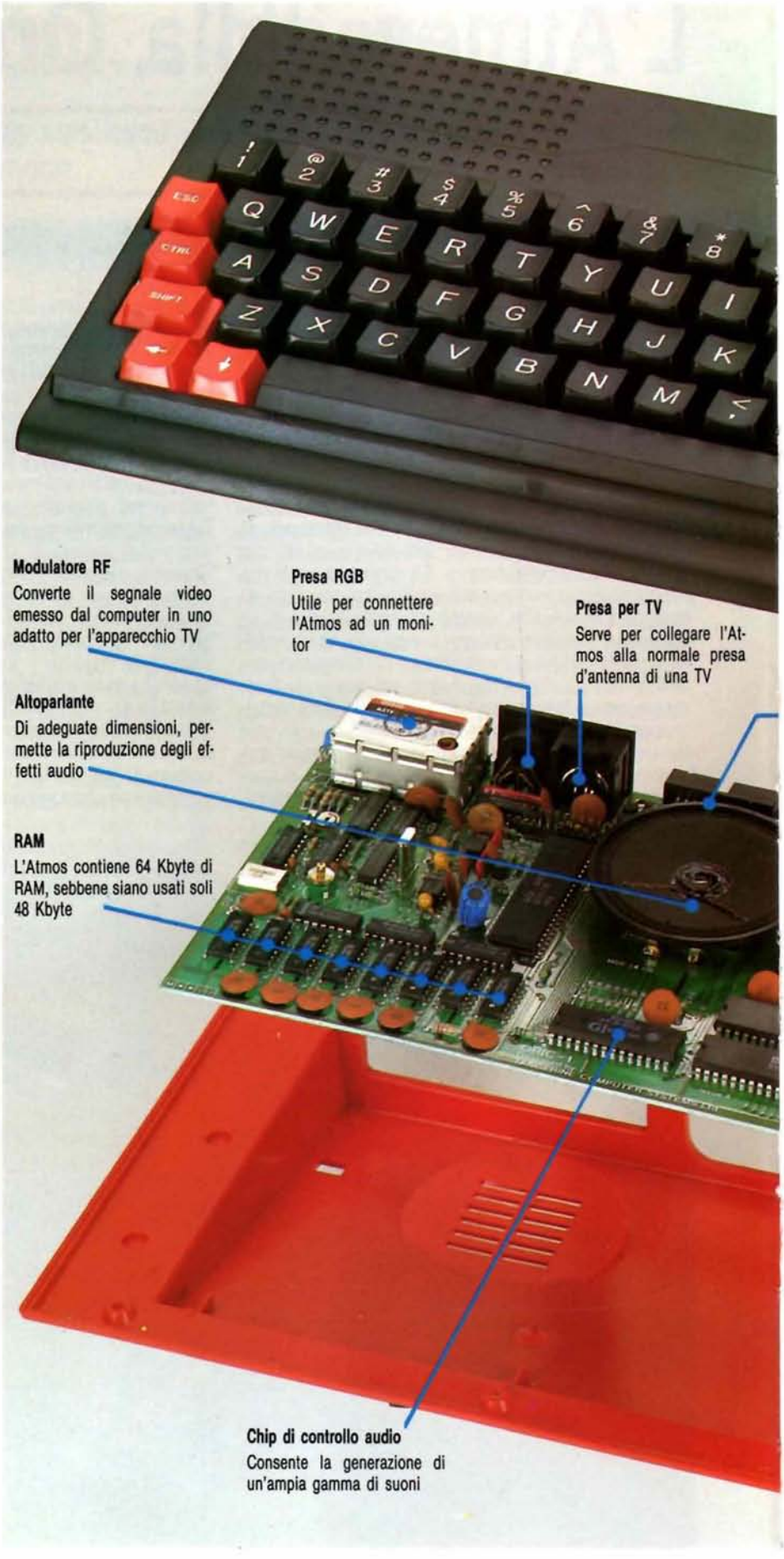
L'Oric-1
L'Atmos è un'edizione rivista e corretta del vecchio Oric-1, del quale abbiamo parlato a pag. 50. La scheda principale è la stessa, ma i programmi nelle ROM sono stati notevolmente migliorati, per eliminare alcuni inconvenienti. I possessori dell'Oric-1, con una speciale espansione, possono ottenere le medesime prestazioni dell'Atmos, benché parte dei programmi non siano compatibili tra i due apparecchi

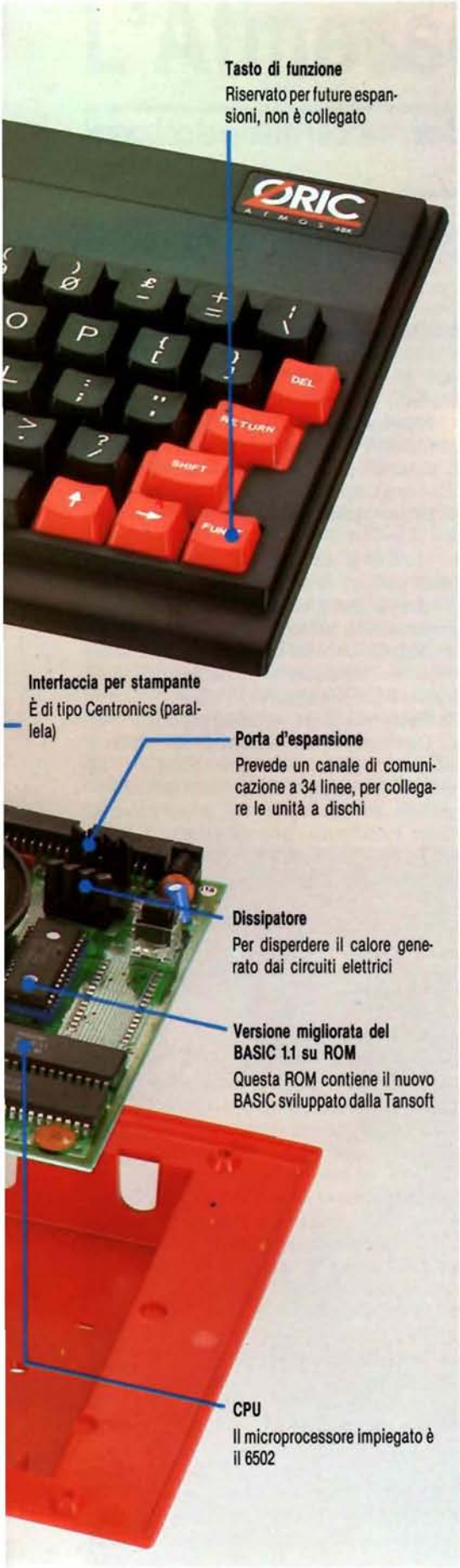
varia da zero a molto intenso, mentre sia i tre generatori di note che quello di rumore possono spaziare su ben sette ottave. La versione di BASIC adottata per l'Oric-1 presentava alcuni piccoli inconvenienti, ad esempio il comando TAB non funzionava in modo corretto e quanto visualizzato sullo schermo poteva venir danneggiato dall'uso dei comandi audio. Inoltre venivano generati dei segnali spuri durante l'esecuzione dell'istruzione STR$. Anche VAL e LEN soffrivano di qualche inconveniente. Nella ROM dell'Atmos questi problemi sono stati eliminati, ma a conseguenza di ciò molti dei programmi scritti per l'Oric-1 si rifiutano di funzionare con l'Atmos. Il BASIC è una versione ampliata, del dialetto Microsoft, sviluppata dalla Tansoft sulla base del Tangerine BASIC originale. Sono previste strutture dei tipi IF...THEN...ELSE e REPEAT...UNTIL. Una caratteristica insolita è la presenza delle istruzioni POP e PULL, che consentono di uscire dalle subroutine e dai cicli REPEAT...UNTIL, senza per questo generare un messaggio d'errore.

Sulle prime versioni dell'Atmos era stato inserito un sottoprogramma di verifica dei programmi registrati su nastro talmente sensibile, che venivano segnalati errori durante quasi tutte le letture dal nastro. Questo inconveniente è stato poi eliminato ed ora il caricamento dei programmi non genera più problemi.

Per adeguarsi al nuovo stile rosso/nero dell'Atmos, anche l'involucro della stampante/plotter è stato riprogettato. Questo apparecchio è dotato di quattro penne a sfera colorate, montate su di un carrello di scrittura mobile. I colori (nero, rosso, verde e blu) vengono selezionati via software. La stampante/plotter è molto lenta nella stampa dei caratteri (12 caratteri al secondo), ma ha il pregio di adoperare carta di tipo normale.

Anche la tanto sospirata unità a microdischetti è stata riprogettata per adeguarsi al nuovo stile dell'Atmos. I dischetti impiegati sono i Sony da 3 pollici, racchiusi in un rigido





CHRIS STEVENS



Unità a microdischetti

Vengono impiegati i microdischetti Sony da 3 pollici, con una capacità di 160 Kbyte per facciata. Sfilando e reinserendo il dischetto capovolto, si può sfruttare anche l'altra facciata, con un totale di 320 Kbyte. Il sistema prevede soltanto l'impiego di file di tipo sequenziale

involucro di plastica. All'Atmos si possono collegare fino a quattro unità a microdischetti, su una delle quali è montata la speciale interfaccia necessaria anche alle altre unità. L'alimentatore per dette unità è dimensionato in modo da poter fornire corrente sufficiente per due unità a dischetti e per lo stesso computer. Nelle prime versioni del sistema operativo sorgeva un grave inconveniente se le unità a dischi e la stampante erano attive allo stesso tempo: qualsiasi tentativo di correggere una linea causava l'eliminazione di tutte le altre linee dal listato. La Oric assicura che anche questo problema è stato eliminato nelle successive versioni del sistema operativo.

I progettisti della Oric hanno saputo sfruttare positivamente questo avvio piuttosto incerto del nuovo prodotto, apportando le necessarie modifiche per assicurare un perfetto funzionamento del sistema. Per ovviare alla scarsità di software, la Oric ha incaricato la Tansoft di produrre una gamma abbastanza estesa di programmi applicativi. Se il livello di questi ultimi sarà sufficientemente buono, l'Atmos della Oric dovrebbe presto incontrare il favore del pubblico.



La stampante/plotter

Un'eccellente periferica per l'Atmos è la stampante/plotter a quattro colori. Impiega quattro penne a sfera, montate su di un supporto di scrittura mobile. Ha il vantaggio di usare carta comune, ma la massima larghezza dei fogli è di circa 12 cm., le penne sono di tipo speciale e la scrittura è lenta rispetto ad una normale stampante

CHRIS STEVENS

ATMOS

DIMENSIONI

mm. 278 x 178 x 50

CPU

6502

MEMORIA

48 Kbyte di RAM, 16 Kbyte di ROM

VIDEO

26 righe per 40 caratteri in modo testuale, 200 x 240 pixel in modo grafico ad 8 colori

INTERFACCE

Per registratore a cassette, interfaccia parallela Centronics, connettore RGB

LINGUAGGI FORNITI

BASIC esteso e FORTH

TASTIERA

Tipo macchina da scrivere, con 58 tasti. Il tasto di funzione non è collegato

DOCUMENTAZIONE

Il manuale, in inglese, è scritto con chiarezza ed in uno stile scorrevole, ed è inteso, evidentemente, a portare i principianti ad una graduale comprensione della programmazione in BASIC. Per i più esperti, contiene anche diversi capitoli dedicati al codice macchina ed ai sistemi di ingresso/uscita. Le appendici trattano molti dettagli tecnici

PRO

L'Atmos offre una vasta gamma di prestazioni raramente trovate anche in apparecchi molto più costosi. Il BASIC è una versione molto facile da usare ed è dotato di una serie di comandi che semplificano la programmazione

CONTRO

Il funzionamento grafico dello schermo è un poco laborioso e le unità a microdischetti, purtroppo, sono in grado di gestire soltanto file sequenziali

Posizioni... chiave

L'uso di indici e la tecnica "hashing": due metodi per velocizzare l'accesso ai file random

Nella lezione precedente è stato introdotto l'uso dei file ad accesso causale, sottolineando quanto questi differiscano in modo sostanziale dai file sequenziali.
Le principali differenze riguardano la quantità di informazioni trattabili e la velocità di accesso ad esse.
Vediamo adesso come organizzare efficientemente le informazioni nei file random, usando le tecniche "ad indice" e quelle "hashing".
Dopo un po' di esercitazioni con i file ad accesso casuale (o random), ci si accorge quanto questi facilitino la gestione degli archivi. Basta specificare il numero di record e si accede immediatamente ai dati, senza dover ricorrere a quelle laboriose procedure di lettura e scrittura, necessarie con i file di tipo sequenziale. Tuttavia, i metodi di inserimento e di annullamento, esposti a pag. 724, non sono tra i più efficienti; infatti adoperando un file di indice si ottengono risultati migliori.
Per la creazione di un file simile, occorre scegliere un particolare campo del record ed usarlo come "chiave". Il valore di tale chiave viene successivamente impiegato per accedere ai record dell'archivio. In definitiva, si tratta di riunire in un solo indice sia la chiave sia il numero di record corrispondente a ciascun record. Ad esempio, se i dati relativi a Giuliano Moschi sono archiviati nel record n. 16 e Moschi risulta ottavo in ordine alfabetico, allora l'indice conterrà il nome Moschi ed il numero 16 nella sua ottava posizione. Se il file di indice viene frequentemente riordinato alfabeticamente, la ricerca di un record può diventare veramente rapida.
Per accelerare ancor più il processo, il file di indice può essere trasferito nella RAM, oppure può anche esser creato lì per lì, operando una lettura integrale del file e depositando chiavi e numeri di record in due matrici del BASIC. Una volta riordinato alfabeticamente, l'indice è pronto per l'uso. Questo metodo, però, può anche richiedere diverso tempo.
Alternativamente, l'indice può essere conservato su disco, assieme ai dati. Ciò consente anche di creare più indici, anziché uno solo, in base a più campi del record (ad esempio: uno per il cognome, uno per la data di nascita, ecc.).
Come mantenere un file di indice è presto detto. Ad un indice occorrono soltanto due campi (quello della chiave e quello del numero di record). Esso viene letto dal disco quando si avvia il programma e poi riscritto soltanto se si sono apportate variazioni. Questa è un'applicazione ideale per i file sequenziali, perché negli indici l'ordine degli elementi corrisponde a quello di memorizzazione. Abbiamo dunque una felice combinazione tra l'uso dei due tipi di file.
L'annullamento di un record, a questo punto, si riduce semplicemente a marcare il record come "annullato", facendolo scomparire dall'indice. Il metodo più comune consiste nello scrivere un carattere speciale (ad esempio un asterisco) nella prima posizione del primo campo.
La chiave relativa al record viene quindi rimossa dall'indice, o l'annullamento viene segnalato con un numero di record pari a −1.
Qualsiasi metodo si adotti, è necessario indicare chiaramente nel file l'avvenuto annullamento, cosicché successivi inserimenti di nuovi record possano occupare lo spazio di record annullati, anziché nuovo spazio sul dischetto. Contemporaneamente, la chiave corrispondente viene inserita nell'indice, al posto di quella annullata. In un momento opportuno per lo svolgimento del programma, l'indice viene riordinato per riflettere il cambiamento. Un simile modo di procedere può permettere, con qualche accorgimento, il recupero di record annullati per errore (purché nel frattempo non ne siano stati inseriti di nuovi).
Conviene dotare il programma di una buona procedura di riordino dell'indice, perché il metodo consente la possibilità d'inserire record senza una particolare sequenza logica e con l'andar del tempo, oltre a crearsi dei vuoti, la lunghezza dell'indice può diventare eccessiva, rallentando l'esecuzione del programma. Tale operazione può essere inserita come opzione a parte nel menu, oppure eseguita periodicamente, in maniera automatica.
Il metodo appena descritto non è l'unico che renda più rapida la gestione di archivi: un'altra tecnica validissima, ma leggermente più complessa, è nota col termine "hashing" ed è particolarmente adatta per archivi di dimensioni molto ampie (viene spesso usata su sistemi a disco rigido ad alta capacità e anche all'interno di molti sistemi operativi).
Questa tecnica sostituisce all'indice una formula o algoritmo di "hash". In sostanza, dalla stessa chiave viene estratto un codice, che rappresenta il numero del record nel quale viene scritto (o letto) il record abbinato alla chiave. L'algoritmo da adottare dipende dalle caratteristiche della chiave. Ad esempio, se la chiave è una data, si può usare il numero del mese, moltiplicato per le ultime due cifre del-

l'anno, cui si somma il numero del giorno. Viceversa, se la chiave è un nome, il codice si può ricavare manipolando opportunamente i codici ASCII che lo compongono.
Supponiamo di voler creare, con questa tecnica, un archivio di dati concernenti i dipendenti di una ditta, usando come chiave il cognome. L'algoritmo potrebbe essere: prendere i primi quattro codici ASCII del nome, sottraendo 65 da ciascuno di essi, e trattare il risultato come un numero di quattro cifre, elevare tale numero al quadrato ed utilizzare, infine, le prime quattro cifre a destra della virgola nel risultato come codice di hash. Il nome CASELLI darebbe un hash di 703, mentre CASTELLI darebbe 2425.
La tecnica hashing è molto diversa da quella ad indice: permette una sola chiave di ricerca (ed un solo algoritmo di hash) per ciascun file, da usare all'inserimento dei dati. È molto rapida nella ricerca, poiché, fornita la chiave, si ottiene immediatamente il numero di record. Col metodo precedente, invece, si possono creare più indici per un file, in qualsiasi momento sia necessario.
Un problema, con la tecnica hashing, si potrebbe avere se due chiavi producono il medesimo codice di hash. L'algoritmo, dunque, deve essere tale da non incorrere in simili eventi.
Il procedimento di hash può essere quindi così riassunto: quando viene scritto un record, dalla sua chiave si estrae un codice, equivalente al numero di record; se tale record dovesse risultare occupato, il sistema cerca sequenzialmente il record libero più vicino. Tale ricerca, in genere prosegue per altri cinque record "di riserva". Nella fase di ricerca, viene estratto il codice di hash dalla chiave e, se la chiave nel record risultante non coincide con quella di ricerca, la scansione avviene di nuovo in modo sequenziale.
Questo metodo può sembrare lungo, ma in realtà la ricerca coinvolge appena 5 o 6 record, anche se i record totali sono qualche migliaio.
Può accadere, però, che i record di riserva siano esauriti. In tal caso, una soluzione, la più ovvia, potrebbe essere quella di avvertire con un messaggio che "il file è pieno". Un'altra soluzione potrebbe esser quella di appoggiare i record in eccedenza in un apposito file a parte.
L'annullamento di un record, col sistema hashing, è altrettanto veloce: dopo aver individuato il record da eliminare, lo si marca con un codice particolare, cosicché un successivo inserimento può sfruttare lo spazio libero venutosi a creare.
Fatto sta, comunque, che per garantire efficienza a questo metodo si cerca di mantenere i file occupati all'80%.
Tutto ciò evidenzia il fatto che la tecnica hashing può portare ad una gestione molto rapida dei dati, ma con grosso dispendio di spazio, maggiore di quello necessario per un sistema ad indice.
Nella prossima lezione verranno trattati i comandi per la gestione dei file su cassette.

I FILE DI INDICE

Il metodo più comune per accedere ad un file random è l'impiego di un file di indice, dove vengono conservati (spesso nella stessa RAM) i valori delle chiavi e del numero di ciascun record dell'archivio principale. Per ritrovare un record, basta scorrere l'indice ed estrarre il numero di record. Anche l'annullamento risulta in un processo molto semplice: basta segnalare il fatto nell'indice, con un valore opportuno, ad esempio –1

Trovare Davitti

CHIAVE	N.
Andreoli	1
Bastiani	–1
Berotti	5
Cerasi	–1
Davitti	7
Desmoni	23
Falteri	15
Gustavi	28
Isidori	37
Lemuzzi	25
Martocchi	11
Niccodei	10

File di indice

Archivio principale

	Nome	Tel. (ab.)	Tel. (uff.)	Professione
1	Andreoli	242 079	727 942	Designer
2	Piccoli	636 241	221 940	Contabile
3	Sansi	631 083	286 170	Giornalista
4	Record annullato			
5	Berotti	729 821	236 190	Dentista
6	Pietri	836 662	298 310	Imbianchino
	Davitti	743 721	450 692	Giardiniere
8	Record annullato			
9	Record annullato			
10	Niccodei	730 631	429 579	Meccanico
11	Martocchi	493 899	455 842	Avvocato
12	Zucchi	736 770	693 045	Parrucchiere

CODICE HASH

Trovare Davitti

ALGORITMO DI HASH

Nella tecnica hashing, è necessario un algoritmo per ricavare il numero di record direttamente dalla stessa chiave di ricerca. I record con codice di hash identico vengono raggruppati assieme

Lo spazio non usato, tra i blocchi di record, viene conservato per i successivi inserimenti

Nome	Tel. (ab.)	Tel. (uff.)	Professione
Daboni	876 294	277 039	Idraulico
Dadolini	662 092	389 220	Direttore Banca
Dalmansi	728 492	372 499	Cameriere
Darrani	926 491	430 696	Orologiaio
Davitti	743 721	450 692	Giardiniere
Demasti	830 883	384 384	Infermiere
Estelli	783 103	459 039	Autista
Ettori	831 829	839 388	Ristoratore

I file ad accesso casuale, gestiti col metodo hashing, offrono un'alta velocità di accesso ai record, ma la struttura richiede un attento impegno di programmazione.
La chiave, elaborata da un apposito algoritmo, potrebbe produrre un codice già impegnato: per questo motivo è previsto un blocco di record, sul quale viene eseguita una ricerca sequenziale

KEVIN JONES

I suoni del computer

Aggiungiamo una nuova dimensione ai giochi, sfruttando la generazione dei suoni

```
10 REM ***** AUDIOGAME ******
20 PRINT "UN SEMPLICE AUDIOGAME":
   PRINT:PRINT"ATTRACCARE ALLA BASE
   PRIMA CHE FINISCA IL CARBURANTE"
30 PRINT:PRINT"PIU' VICINI SI E',PIU'
   ALTO DIVENTA IL TONO DEL SEGNALE"
40 PRINT:PRINT"I CONTROLLI SONO:":
   PRINT"I = SU  M = GIU'
   J = SINISTRA  K = DESTRA"
60 PRINT:PRINT"**** PREMERE QUALSIASI
   TASTO PER CONTINUARE ****"
70 A$=INKEY$(0):IF A$="" THEN GOTO 70
80 P=INT(RND(1)*500)+1:Q=INT(RND(1)*
   500+1):F=3*(P+Q)/2:I=0:Y=0
90 ID=0:YD=0
100 PRINT "CARBURANTE=";F; DISTANZA=";
110 IF (ABS(P-I)<2 AND ABS(Q-Y)<2)
    THEN PRINT"CE L'HAI FATTA, E TI
    SONO RIMASTE";F;" UNITA' DI
    CARBURANTE":END
120 A$=INKEY$(0):IF A$="" THEN
    GOTO 120
130 IF (A$="I" AND YD<3) THEN YD=YD+1
140 IF (A$="M" AND YD>-3) THEN YD=YD-1
150 IF (A$="J" AND ID>-3) THEN ID=ID-1
160 IF (A$="K" AND ID<3) THEN ID=ID+1
170 I=I+ID:Y=Y+YD
180 D=SQR((P-I)*(P-I)+(Q-Y)*(Q-Y))
190 PRINT D
200 SOUND 1,-8,(255-D),2
210 F=F-ABS(ID)-ABS(YD):IF F > 0
    THEN GOTO 90
230 PRINT TAB(10)"************ CRASH
    ************":PRINT TAB(11)"****
    ** CARBURANTE FINITO ! ******"
240 PRINT:PRINT"LA POSIZIONE DISTA
    DALLA BASE";D;" UNITA'"
260 END
```

Nei giochi di tipo "arcade" viene fatto largo uso di effetti audio per accompagnare l'ottima grafica sullo schermo. Queste colonne sonore rendono i giochi più realistici ed emozionanti, ma raramente costituiscono l'obiettivo del gioco stesso. Tuttavia, se possiamo usare il senso della vista come base per una gran quantità di giochi, perché non creare programmi incentrati proprio sulle capacità auditive del giocatore?

Il gioco qui illustrato è dunque un "audiogame". Forse non si tratta del più entusiasmante gioco in circolazione, ma è senz'altro divertente e utile per imparare qualcosa in più sulla generazione sonora dei computer. Ciò che sembra inizialmente un compito molto facile, diviene una sfida avvincente, principalmente per il fatto che dobbiamo agire in base ad un senso diverso da quello cui siamo abituati negli altri giochi, imperniati sulla grafica.

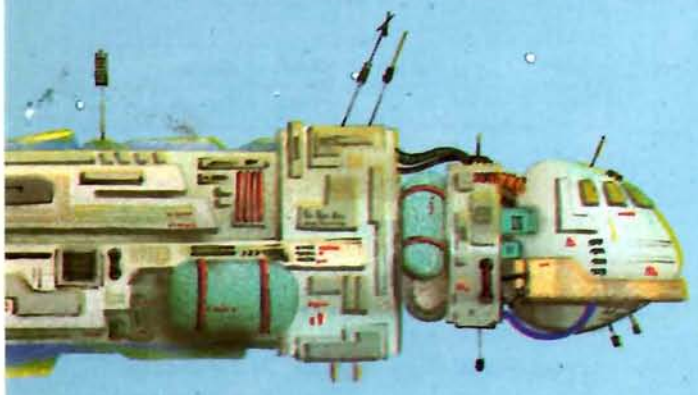
Nel gioco siamo al comando di un'astronave che sta finendo il carburante. L'unica speranza di sopravvivere è quella di attraccare ad una base poco lontana. Sfortunatamente, a causa di un guasto nel sistema principale, non possiamo valerci di un contatto visivo con la base: l'unica maniera di orientarsi è servirsi di un segnale audio, che funge da "faro". Più alto è il tono del segnale, più siamo vicini alla base, quindi le manovre dipendono dalla nostra bravura ad interpretare correttamente i segnali.

Come in una vera astronave, la navigazione prosegue in una direzione, finché non interveniamo sui comandi per mutare rotta. Se premiamo due volte il tasto (I) per salire, allora occorre premere due volte il tasto (M) per scendere nella posizione originaria e ciò rende il gioco più arduo di quanto non si creda.

Ci vorrà un po' di pratica prima di riuscire ad attraccare felicemente, usando come guida soltanto il segnale audio. Se volessimo facilitare il gioco, potremmo visualizzare sullo schermo la posizione dell'astronave, rispetto alla base: la variabile che contiene la distanza è D. Oppure, si potrebbe suggerire la direzione da intraprendere (se SGN(P—X) = —1, allora occorre virare a sinistra, per esempio).

Una volta trascritto e provato il gioco, verrà voglia di includere una simile novità nei propri giochi: l'uso del suono per localizzare o evitare oggetti può senza dubbio arricchire qualsiasi programma di gioco: un sottomarino guidato da un sonar, oppure la ricerca di un tesoro con un "metal detector" non sono che alcune idee.

Naturalmente, gli effetti più stravaganti si ottengono con i computer dotati di sofisticati generatori di suoni (quali i Micro BBC od i Commodore) sui quali è possibile adoperare più voci o più timbri, ciascuno associato ad un particolare oggetto od ostacolo. Anche il controllo sul volume del suono può servire a migliorare l'effetto generale.



DAVE COOPER-SMITH

Basic & Basic

Questo programma è stato scritto per il Micro BBC, nel MODE 7, ed è quindi in un BASIC molto simile al dialetto Microsoft. I comandi PRINT presuppongono uno schermo a 40 colonne, per cui è necessario adeguarli alle dimensioni del proprio schermo. Il comando SOUND (linea 200) è proprio del Micro BBC: il valore del parametro (255-D) influisce sul tono del segnale emesso; gli altri parametri riguardano il volume, la durata ed il canale da usare. Sia l'uso di INKEY$(0), alle linee 70 e 120, che di RND (linea 80) varia da una versione di BASIC all'altra:

Spectrum

Inserire una LET in tutte le assegnazioni di variabili. Cambiare INKEY$(0) in INKEY$. Sostituire RND a RND(1). La linea 200 diventa:

200 BEEP 0.4, (255—D)

inserire: 15 RANDOMIZE
195 IF D>254 THEN LET D=D—254

Commodore 64 / Vic-20

Cambiare A$=INKEY$(0) in GET A$. Per i comandi di suono, consultare la sezione "Suoni e Luci".

Inserire: 75 X=RND(—TI)

Dragon

Cambiare INKEY$(0) in INKEY$. Sostituire RND(1) con RND(0). La linea 200 diventa:

200 SOUND (255—D),10

Inserire: 195 IF D>254 THEN D=D—254

Atmos

Cambiare INKEY$(0) in KEY$. La linea 200 diventa:

200 SOUND 1,(255—D),9:WAIT 40:PLAY0,0,0,0

Inserire: 195 IF D>254 THEN D=D—254

Un gioco subacqueo

Ecco finalmente un gioco per computer diverso dalle solite battaglie spaziali

In un mercato di software invaso da una miriade di giochi del tipo "Space Invaders", è confortante trovare un gioco totalmente nuovo: lo Scuba Dive (letteralmente: immersione col respiratore) della Durrell Software. Ne esistono tre versioni: una per lo Spectrum (scritta da Mike Richardson), una per l'Oric-1 (scritta da Ron Jeffs) ed una per il Commodore 64 (scritta da Nigel Dewdney). La versione Oric sarà presto disponibile anche per l'Atmos.

Il giocatore impersona un sub alla ricerca di tesori nascosti sul fondo del mare, rischiando la propria vita ad ogni... bracciata. L'obiettivo è quello di raccogliere il maggior numero di perle possibile, estraendole da ostriche e da bivalve di ogni tipo. Ad un livello più impegnativo, i tesori sono racchiusi in forzieri, situati in profondi antri marini.

Ogni immersione è resa pericolosa dalla presenza di meduse, piovre, lamprede e, nella versione Spectrum, anche di squali! Se capita di toccare una di queste creature (non attaccano mai di spontanea volontà) si perde una vita, basta quindi evitarle accuratamente. Un altro pericolo è costituito dalle ostriche giganti: possono richiudersi improvvisamente ed intrappolare il malcapitato sub.

Gli stretti pertugi che si aprono sulle caverne più profonde sono custoditi da piovre giganti e passarvi è veramente rischioso: basta un nulla per urtare i lunghi tentacoli, sempre in movimento. Comunque, qualche volta si riesce nell'intento. Nella versione Commodore al posto delle piovre ci sono delle porte a trabocchetto che si aprono e si chiudono in continuazione ed occorre oltrepassarle nel momento giusto, per non essere "accoppati".

Il programma elargisce ben tre vite per ogni partita, che può svolgersi con quattro diversi livelli di difficoltà. Una vita viene meno se si tocca una delle creature sottomarine o se si esaurisce l'ossigeno.

Ogni partita inizia presentando una vista della superficie del mare e buona parte dello scenario sottostante, oltre alla nave dalla quale si tuffa il sub. Ci sono imprevisti anche nei tuffi, poiché può capitare di rimanere impigliati nella chiglia della nave.

Nella versione Spectrum è utile possedere un buon senso dell'orientamento perché la nave può allontanarsi troppo durante l'immersione. Nella versione Oric, al contrario, se la nave esce dallo schermo a destra, ricompare poco dopo dal lato sinistro, cosicché non c'è il rischio di perderla di vista quando emerge.

La qualità grafica della versione Spectrum (migliore tra le tre) è eccellente. È stato fatto un ottimo uso del colore e le creature marine sono molto realistiche. Il movimento del sub si controlla tramite i tasti X e Z (per ruotarlo in senso orario ed antiorario), mentre i tasti "spazio" e "shift" muovono il sub in avanti. I possessori di Spectrum possono utilizzare anche i joystick, per controllare i movimenti, tuttavia il programma non è predisposto per l'impiego dell'interfaccia per joystick della Kempston. Anche la versione per il Commodore consente l'uso di joystick, mentre chi possiede l'Oric deve accontentarsi del controllo da tastiera.

La versione Oric, in effetti, è la meno suggestiva ed avvincente, specie se paragonata a quella per lo Spectrum: il movimento del sub non è continuo, ma a scatti, e la grafica molto meno accurata.

Controllo di qualità

Le figure mostrano la diversa grafica impiegata nelle tre versioni del programma



Spectrum



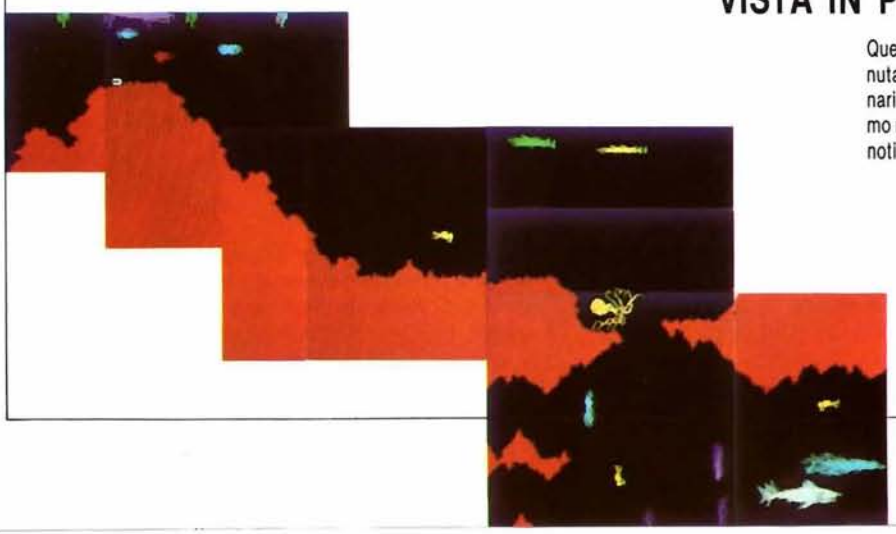
Oric-1



Commodore 64

VISTA IN PROSPETTIVA

Questa immagine è stata ottenuta sovrapponendo più scenari che appaiono sullo schermo nella versione Spectrum. Si noti l'accuratezza della grafica



IAN McKINNELL

Somme e sottrazioni

Come eseguire queste semplici operazioni in codice macchina

Le diversità di funzionamento, in semplici operazioni di calcolo, tra i microprocessori Z80 e 6502, rivelano la differente impostazione dell'architettura interna dei chip. Nello Z80 l'abbondanza di registri e la versatilità delle operazioni possibili caratterizzano un processore elegante, complesso e potente. L'architettura ed il funzionamento più elementari del 6502, suggeriscono di aver a che fare con un processore robusto e pratico, ma non appartenente alla stessa classe dello Z80. Tuttavia, anche se fondamentalmente questa impressione è esatta, il 6502 ha una discreta versatilità, dovuta al particolare sistema di indirizzamento indiretto e di un sagace uso della pagina zero come registro indice addizionale. Queste doti faranno sopravvivere il 6502 per molto tempo ancora.
L'enorme vantaggio offerto dai registri dello Z80 è la loro flessibilità d'uso: sia come registri doppi a 16 bit, che come registri singoli ad 8 bit, permettono una vasta scelta sul modo d'indirizzamento più adatto. Il 6502, al contrario, non è dotato di registri a 16 bit, ma è tuttavia capace di sfruttare la pagina zero come una tabella di valori sia a singolo che a doppio byte.

Base aritmetica

Abbiamo visto che i registri della CPU consentono molte possibilità sul modo di accedere alla memoria, ma un'efficiente manipolazione della memoria richiede molto più di semplici operazioni di trasferimento o di confronto tra più valori. La capacità di eseguire le quattro operazioni fondamentali è di capitale importanza, tuttavia sia lo Z80 che il 6502 non sanno che sommare e sottrarre due valori. La moltiplicazione e la divisione sono possibili soltanto ricorrendo a più istruzioni di programma (come del resto accade per somme e sottrazioni di numeri maggiori di $FF). Questa è una grave limitazione, benché la programmazione di queste operazioni assuma un valore didattico enorme per il programmatore principiante. Comunque, sui microprocessori a 16 bit (della generazione successiva allo Z80 ed al 6502) sono previste anche queste due operazioni.
Per adesso dobbiamo usare le istruzioni ADC (somma con riporto) e INC (incrementa), per operare su singoli byte. Ecco due maniere per sommare il contenuto di due locazioni di memoria:

6502			Z80		
INDR1	DW	$7E60	INDR1	DW	$7E60
INDR2	DW	$4A51	INDR2	DW	$4A51
SOMMA	DS	$03	SOMMA	DS	$03
INIZI	CLC		INIZI	LD	A,$00
	LDA	INDR1		AND	A
	ADC	INDR2		LD	HL,(INDR1)
	STA	SOMMA		LD	DE,(INDR2)
	LDA	INDR1+1		ADD	HL,DE
	ADC	INDR2+1		LD	(SOMMA),HL
	STA	SOMMA+1		ADC	A,$00
	LDA	$00		LD	(SOMMA+2),A
	ADC	$00		RET	
	STA	SOMMA+2			
	RTS				

Il metodo a singolo byte, usato nel 6502, può essere applicato anche allo Z80, mentre non è vero il contrario poiché mancano registri a 16 byte sul 6502. Si noti come viene usato il flag di riporto: le istruzioni CLC (6502) e AND A (Z80) azzerano detto flag prima della somma, poi, al termine del programma, esso viene copiato in SOMMA+2. È sempre opportuno tener conto dei valori massimi di un risultato.
La sottrazione può essere trattata in modo analogo, poiché entrambi i processori sono dotati di istruzioni SBC (sottrazione con riporto), benché lo Z80 ne possieda anche una versione per i registri a 16 bit. Siccome il risultato può essere negativo, occorre pensare a come rappresentare in binario l'eventuale segno algebrico.
Tanto per cominciare, ecco un concetto di facile apprendimento:

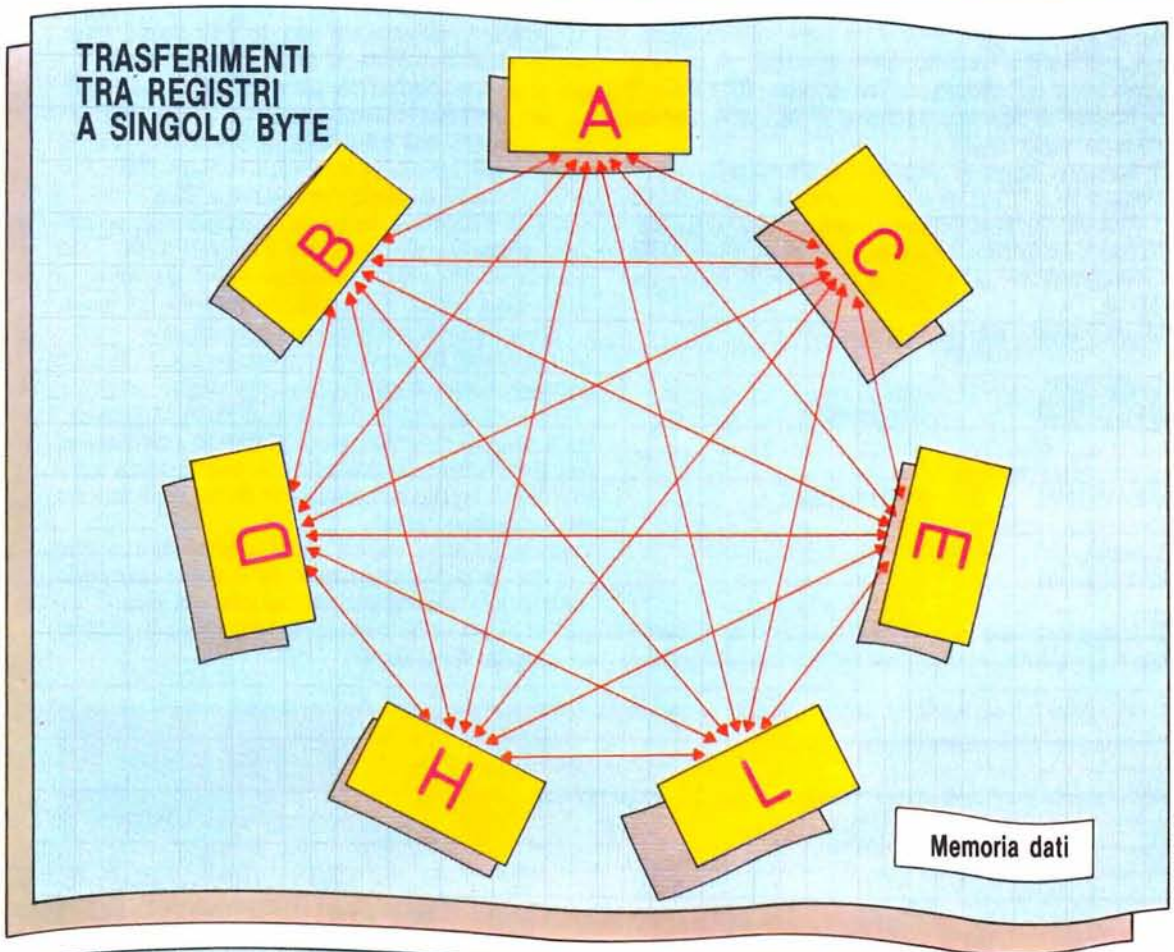
se A+B = 0, ne consegue che A = −B

In altre parole: se A è un numero positivo, allora la sua "negazione" o "complemento" è quel numero che, sommato ad A, dà risultato zero. Per esempio, se A è il numero a singolo byte $04, allora il suo complemento (sempre per un singolo byte) è $FC:

$04 + $FC = $100.

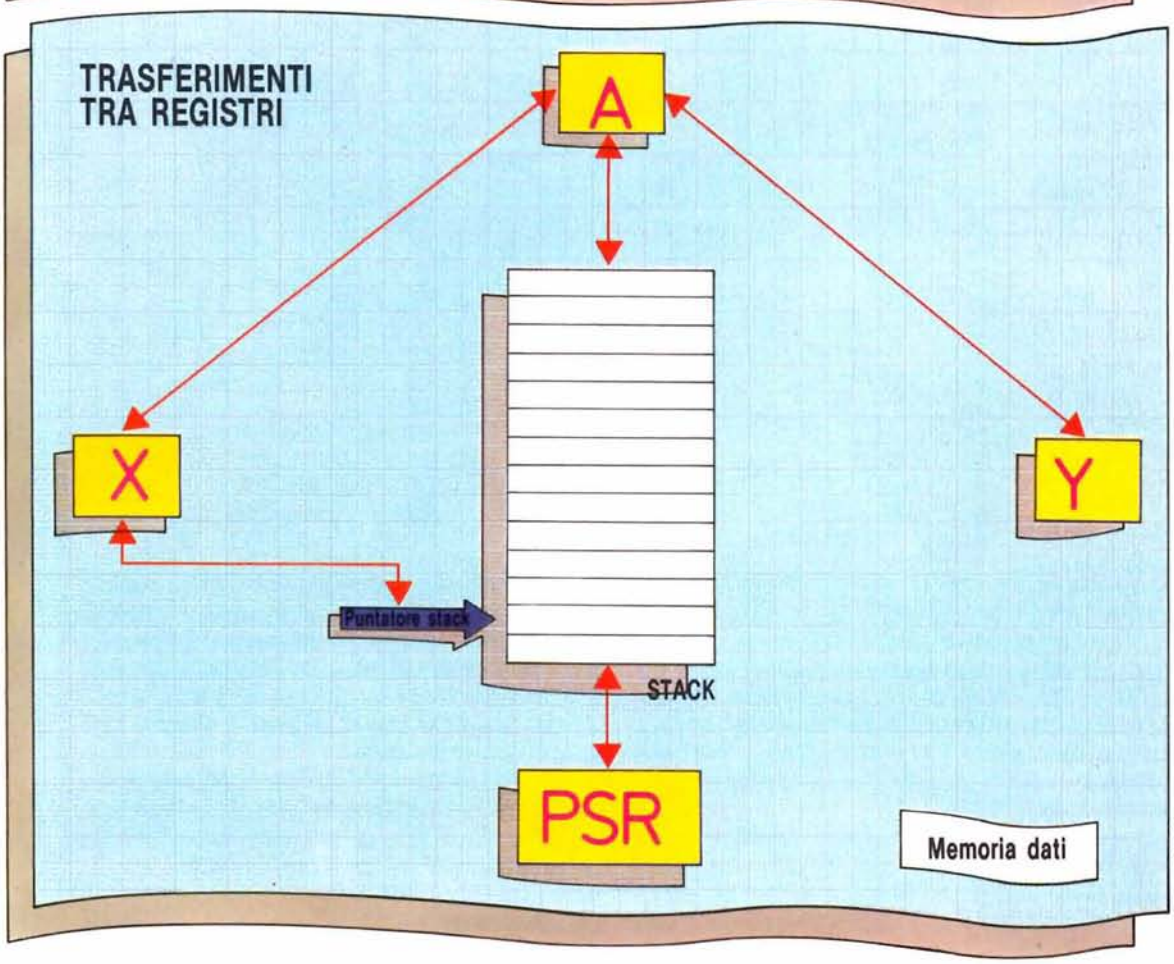
Si rammenti che $100 = $00, se abbiamo un solo byte a disposizione per il risultato. Ne consegue che si può operare una differenza anche tramite un'operazione di somma con un numero negativo, ossia:

A−B equivale a A + (−B).



Doppia identità

I registri dello Z80 possono comunicare tra loro come registri a singolo byte ed ognuno di loro può anche comunicare con la memoria nei modi: diretto, indiretto, assoluto ed indicizzato. Se adoperati come BC, DE ed HL (come coppie di registri, a doppio byte) i registri possono trasferire valori a 16 bit nella memoria o nello stack. Alcuni di essi possono servire anche come accumulatori a 16 bit per somme e sottrazioni. Questa ampia varietà di utilizzazioni ha contribuito al grande successo del chip Z80



Puro e semplice

Le comunicazioni interne del chip 6502 sono estremamente lineari e limitate a trasferimenti di valori a soli 8 bit. Soltanto l'accumulatore può comunicare direttamente coi registri X ed Y, soltanto X comunica con lo stack e soltanto l'accumulatore ed il registro PSR (condizione del processore) accedono allo stack. I modi utilizzabili per i trasferimenti in memoria sono: assoluto, diretto, indiretto, indicizzato, immediato, e in pagina zero. Il fatto di poter sfruttare convenientemente la pagina zero sopperisce alla scarsità di registri nel 6502. Infatti, la pagina zero può essere usata come una sequenza di 128 registri a 16 bit

Pertanto, $08–$05 equivale a $08+(–$05) e siccome (–$05)=$FB (infatti: $FB+$05=$100), il nostro problema si riduce alla forma $08+$FB. Il risultato di questa somma è $103, ma, in un singolo byte, è $03.
Il metodo appena esposto è chiamato "complemento a 2": il complemento di un numero a singolo byte si ottiene sottraendolo da $100. Esiste anche una rappresentazione chiamata "complemento ad 1", in rapporto con la precedente.
Si consideri, ad esempio:

```
$05 = 00000101          binario
$FA = 11111010          complemento a 1
          +1
-----------------------
$FB = 11111011          complemento a 2
-----------------------
$05 + $FA = $FF
$05 + $FB = $00
```

Il complemento a 1 di un numero a singolo byte si ottiene semplicemente completando o "negando" ciascun bit che lo compone. Sommando 1 al risultato, si ha il complemento a 2 del numero originario. La somma di un numero col proprio complemento a 1 dà sempre $FF, mentre col proprio complemento a 2 dà sempre $00 (anche se in realtà sarebbe $100). Per convenzione, quindi, i numeri da $00 a $7F (da 0 a 127 in decimale) vengono considerati positivi, mentre quelli da $80 a $FF negativi (da –128 a –1 in decimale). Confrontando i due gruppi di numeri binari, si noterà che i numeri negativi hanno sempre il bit più significativo posto ad 1, mentre in quelli positivi tale bit è zero. Di conseguenza, il bit 7 viene chiamato anche il "bit di segno" di un numero binario "segnato", ed il flag di riporto, nel registro di condizione del processore, rispecchia la condizione 0 o 1 del bit di segno nel risultato di un'operazione aritmetica o logica.
Purtroppo non esiste alcuna scorciatoia che faciliti la comprensione di questo procedimento, ma esercitandosi coi numeri binari negativi si può afferrare presto il metodo, di per sé non tanto difficile.

Risposte agli esercizi di pag. 739

1) Il seguente programma inverte l'ordine dei caratteri nella stringa conservata a LABL1:

6502

```
;
ORIGIN   ORG   $7000
LAST1    EQU   $0D
LABL1    DB    'QUESTO E' UN MESSAGGIO'
TERMN8   DB    LAST1
;
BEGIN    LDX   #$FF
         LDA   #LAST1
         PHA
LOOP0    INX
         LDA   LABL1,X
         PHA
         CMP   #LAST1
ENDLP0   BNE   LOOP0
CLRSTK   PLA
;
BEGIN1   LDX   #$FF
LOOP1    INX
         PLA
         STA   LABL1,X
         CMP   #LAST1
ENDLP1   BNE   LOOP1
         RTS
```

Z80

```
         ORG   $C000
LAST1    EQU   $0D
LABL1    DB    'QUESTO E' UN MESSAGGIO'
TERMN8   DB    LAST1
;
BEGIN    LD    IX,LABL1-1
         LD    A,LAST1
         PUSH  AF
LOOP0    INC   IX
         LD    A,(IX+0)
         PUSH  AF
         CP    LAST1
ENDLP0   JR    NZ,LOOP0
CLRSTK   POP   AF
;
BEGIN1   LD    IX,LABL1-1
LOOP1    INC   IX
         POP   AF
         LD    (IX+0),A
         CP    LAST1
ENDLP1   JR    NZ,LOOP1
         RET
```

Nella versione 6502, i codici compresi tra LOOP0 e ENDLOOP0 usano l'indirizzamento indicizzato con X, all'interno di un ciclo, per estrarre ciascun carattere della stringa, depositandolo sullo stack, non senza averci prima depositato il codice ASCII di "fine stringa". Anche l'ultimo carattere ad essere appoggiato sullo stack è un codice di "fine stringa", ma qui si tratta di quello estratto alla fine della sequenza di caratteri. Il programma provvede poi ad eliminare questo secondo carattere dallo stack (CLRSTK).
La versione Z80 adopera l'indirizzamento indiretto con IX per caricare l'accumulatore con i valori da LABL1 in poi. Sullo stack viene depositato non solo l'accumulatore, ma anche i flag. Ciò significa che, nello stack, tra i vari caratteri sono interposti i successivi valori dei flag di condizione. Le istruzioni comprese tra BEGIN1 e ENDLP1, in entrambe le versioni, sono analoghe a quelle del ciclo precedente, ma questa volta recuperano, in ordine inverso, la serie di caratteri depositata nello stack, trasferendo i caratteri a partire da LABL1 in poi. Il ciclo si conclude quando si raggiunge il codice di "fine stringa" in fondo allo stack.
Si noti come sia importante equilibrare l'uso dello stack e come la parte più complessa sia quella che concerne i casi limite (che fare all'inizio dei cicli, come terminarli e come gestire un'eventuale "pulizia" di caratteri residui).
Le istruzioni Z80 a BEGIN e BEGIN1 (ossia LD IX,LABL1–1) dimostrano l'utilità di possedere un Assembler, che converte

l'espressione (LABL1—1) nell'indirizzo del byte che precede la definizione di LABL1, inserendo tale indirizzo nel programma in codice macchina. La maggior parte degli Assembler offrono la possibilità di adoperare espressioni complesse, anziché obbligare a definire esplicitamente gli indirizzi necessari alle varie istruzioni.

2) Questo programma inverte l'ordine dei caratteri in ciascuna parola della stringa che inizia a LABL1, pur mantenendo intatto l'ordine delle parole:

6502

```
;
ORIGIN    ORG    $7000
LAST1     EQU    $0D
SPACE     EQU    $20
LABL1     DB     'QUESTO E' UN MESSAGGIO'
TERMN8    DB     LAST1
;
BEGIN     LDX    #$FF
LOOP0     JSR    RVSWRD
          CMP    #LAST1
ENDLP0    BNE    LOOP0
          RTS
;
;****S/R INVERSIONE PAROLA****
LASTCH    DB     $00
LASTX     DB     $00
RVSWRD    TXA
          TAY
          INY
RVSLP0    INX
          LDA    LABL1,X
          PHA
          CMP    #SPACE
          BEQ    CLRSTK
          CMP    #LAST1
ENDRV0    BNE    RVSLP0
CLRSTK    PLA
          STA    LASTCH
          STX    LASTX
RVSLP1    PLA
          STA    LABL1,Y
          INY
          CPY    LASTX
ENDLP1    BNE    RVSLP1
          LDA    LASTCH
          RTS
```

Z80

```
          ORG    $C000
LAST1     EQU    $0D
SPACE     EQU    $20
LABL1     DB     'QUESTO E' UN MESSAGGIO'
TERMN8    DB     LAST1
;
BEGIN     LD     DE,LABL1-1
LOOP0     CALL   RVSWRD
          CP     LAST1
ENDLP0    JR     NZ,LOOP0
          RET
;
;***S/R INVERSIONE PAROLA***
LASTCH    DB     $00
RVSWRD    PUSH   DE
          POP    HL
          INC    HL
RVSLP0    INC    DE
          LD     A,(DE)
          PUSH   AF
          CP     SPACE
          JR     Z,CLRSTK
          CP     LAST1
ENDRV0    JR     NZ,RVSLP0
CLRSTK    POP    AF
          LD     (LASTCH),A
;
RVSLP1    POP    AF
          LD     (HL),A
          INC    HL
          LD     A,L
          CP     E
          JR     NZ,RVSLP1
          LD     A,H
          CP     D
ENDRV1    JR     NZ,RVSLP1
          LD     A,(LASTCH)
          RET
```

Ci sono più punti degni d'interesse in questo programma: l'uso delle istruzioni JSR e CALL, ad esempio. La struttura della subroutine RVSWRD è simile a quella del programma dato nell'esercizio 1, ma in questo caso l'inversione viene operata sui caratteri di una sola parola per volta. In entrambe le versioni, il registro indice (X oppure IX) viene usato per "passare" l'indirizzo iniziale della parola alla subroutine, mentre l'accumulatore viene impiegato per restituire al programma chiamante il carattere che ha terminato l'inversione (il codice dello spazio o di "fine stringa"). Il "passaggio" di valori è una tecnica frequentemente usata nel linguaggio Assembly, ma va adoperata con attenzione, specialmente se si ha l'abitudine di appoggiare tutti i registri della CPU sullo stack all'inizio di ogni subroutine (come esemplificato a pag. 738). Un altro fatto da notare è l'impiego del registro Y nella versione 6502. Inizialmente esso contiene l'indirizzo di partenza della parola, mentre X è usato come indice nel ciclo di appoggio sullo stack. In un secondo momento, Y è impiegato come indice nella subroutine di recupero dallo stack, mentre X contiene l'indirizzo della parola. Il termine indirizzo è qui adoperato impropriamente, poiché sia X che Y sono registri ad un solo byte e pertanto incapaci di contenere un indirizzo completo. Effettivamente, X e Y contengono soltanto lo scarto rispetto all'indirizzo base LABL1. Al contrario, i registri IX ed IY dello Z80 contengono realmente indirizzi completi, essendo registri a 16 bit.

Nella versione Z80, non vengono affatto usati IX ed IY, ma le coppie HL e DE, che, come nella versione 6502, contengono l'indirizzo iniziale e finale della parola. Tuttavia, anziché essere usati come indici, essi sono adoperati per un indirizzamento indiretto (l'istruzione LD A,(DE) significa "copia nell'accumulatore il contenuto del byte il cui indirizzo si trova in DE"). Tutti i registri dello Z80 possono essere impiegati in modo analogo. L'unico vero neo, nell'insieme delle istruzioni per lo Z80, riguarda la mancanza di istruzioni per eseguire confronti sui registri a doppio byte: per confrontare DE con HL, occorre infatti confrontare prima E con L, poi D con H. Similmente, nella versione 6502, i registri X e Y vengono confrontati indirettamente, ricorrendo ad una locazione di memoria, poiché non esiste un'istruzione per un confronto diretto tra X e Y.

La Quicksilva

Il cammino di una software house che sui giochi ha creato un piccolo impero

Quasi ogni settimana compare sulla scena una nuova software house, ma ben poche riescono ad affermarsi stabilmente. Un'eccezione è costituita dalla Quicksilva che, dal 1981, è sempre stata in testa alle classifiche di vendita.

La Quicksilva ha tratto il proprio impeto iniziale dal lancio sul mercato dello ZX80 della Sinclair. Il successo di questo primo computer a prezzo "stracciato" stimolò Nick Lambert, un ingegnere inglese, a progettare un'espansione di memoria da 3 Kbyte, da applicare allo ZX80. Il prodotto ottenne un gran successo e quando la Sinclair, l'anno dopo, produsse lo ZX81, Lambert formò la Quicksilva, assieme a John Hollis e Mark Eyles, per la produzione di accessori per il nuovo apparecchio. Nel 1981, Lambert redasse anche un programma, Defender, che rappresentò il primo ingresso della Quicksilva nel settore software.

Defender ottenne un successo strepitoso, incoraggiando la Quicksilva a proseguire nella creazione di giochi del genere "arcade", tralasciando il settore hardware.

Il secondo grande successo commerciale della società è rappresentato dal programma *Timegate*, appartenente al genere "adventure". Durante le feste natalizie del 1982, la Quicksilva aveva 10 diversi titoli in catalogo e una grande catena di negozi di cartoleria ordinò ben 10.000 copie di *Timegate*. Grazie a prodotti di qualità (ed alla fama creatasi) la distribuzione raggiunse presto una capillarità impressionante, confermando il successo della software house: partita con un capitale negativo (oltre 200 sterline "in rosso" nel conto in banca), nel suo primo anno di attività ha prodotto un fatturato di oltre 150.000.000 milioni di lire. I prodotti della Quicksilva sono da tempo venduti in molti paesi del mondo, compresa l'Italia.

Un'ascesa così rapida ha obbligato ad ampliare il proprio organico e sulla stampa britannica appaiono spesso annunci rivolti ad autori di software intenzionati a collaborare con la Quicksilva. Gli autori di programmi originali possono aspettarsi fino al 15 per cento di diritti sul prezzo di vendita dei nastri a cassetta.

Oggi la Quicksilva non si orienta più sui soli prodotti Sinclair, ma ha in catalogo giochi anche per il Commodore 64, per il Vic-20, per il Micro della BBC e per il Dragon e viene considerata una delle software house più importanti del Regno Unito.

Un altro importante successo è stato ottenuto col programma *Ant Attack* (vedere a pag. 486), un gioco insolito in cui la grafica viene usata molto appropriatamente. Nel 1983, per dedicarsi a progetti maggiormente creativi, Lambert ha lasciato la guida della Quicksilva a Rod Cousens, dopo esser stato nominato vice presidente dalla "Guild of Software Houses" (l'associazione delle software house britanniche) e "Personaggio dell'anno" dall'associazione per il commercio di computer (la Computer Trade Association).

La Quicksilva è alla continua ricerca di settori nei quali diversificare la propria attività e, a questo proposito, ha prodotto un nuovo gioco di genere "non violento" (in contrapposizione alla maggior parte di quelli attualmente in commercio), chiamato *L'omino di neve* e basato su di una favola per bambini. Una delle ambizioni della Quicksilva è quella di sfondare nel mercato del software nel Nord America.

Ultime novità

Questi sono alcuni dei titoli più recenti della Quicksilva, che offre i propri prodotti in più versioni adatte ai vari home computer



IAN McKINNELL



Rod Cousens

Il direttore commerciale della Quicksilva



Mark Eyles

Il direttore del settore pubblicitario ed uno dei fondatori della Quicksilva

per concessione della QUICKSILVA

Database e computer

La gestione organizzata delle informazioni è una delle applicazioni più frequenti sui computer

Le informazioni che si possono conservare ordinatamente in un "database" possono riguardare gli argomenti più disparati. In genere, però, si preferisce costruire un database per ogni particolare applicazione. Nel caso di un'associazione, ad esempio, i dati potrebbero essere: i nomi e gli indirizzi dei membri di un club, le spese sostenute per l'organizzazione di convegni, il programma delle riunioni, la situazione contabile delle quote associative. Un buon database può facilmente contenere tutte queste informazioni in un unico file di notevoli proporzioni.

Un file è composto da una serie di record, ciascuno dei quali è a sua volta suddiviso in campi. In alcune applicazioni, come ad esempio nei programmi di contabilità, la struttura del file viene fissata una volta per tutte dal programmatore, mentre nei database essa può essere variata (talvolta anche dallo stesso utilizzatore), per adattarsi agli obiettivi che si intendono raggiungere. Le dimensioni dei campi ed il tipo di informazioni immagazzinate in un particolare database, dunque, possono differire notevolmente da quelle di un altro.

In un file di nomi ed indirizzi, ad esempio, le informazioni su ciascun individuo si trovano tutte in un unico record: tanto il nome che ciascuna riga dell'indirizzo occupano un "campo alfanumerico" (composto da lettere), la cui lunghezza è solitamente di 30 caratteri. L'alternativa al "campo alfanumerico" è il "campo numerico" (riservato, appunto, ai soli numeri), che consente al programma di eseguire calcoli sui dati immagazzinati e di produrre utili tabulati. Nel database per un'associazione, preso come esempio, il programma potrebbe calcolare quanti membri hanno pagato la quota associativa, l'incasso totale fino ad una particolare data e quanto ancora deve essere riscosso. Tuttavia, non sempre è obbligatorio o conveniente immagazzinare numeri in campi numerici. I numeri telefonici, ad esempio, possono essere registrati in campi alfanumerici, poiché su di essi non verranno mai eseguiti calcoli e, inoltre, ciò consente di frapporre opportuni simboli

Raccogliamo le idee!

Molte persone hanno l'abitudine di scrivere brevi appunti in modo caotico su piccoli pezzi di carta. Un database per microcomputer potrebbe essere utile per ricordare le scadenze di un'assicurazione, il numero di conto bancario ed altre informazioni personali. Inoltre lo si potrà adoperare per catalogare la collezione di francobolli, di monete, di dischi o di qualsiasi altra raccolta che richieda un'archiviazione sistematica



MIKE BROWNLOW

Un solo collegamento

I database riscuotono grande successo nelle società tecnologiche, la cui sopravvivenza dipende sempre di più dall'accesso alle informazioni. Nel passato i database, sotto forma di archivi tenuti a mano, erano isolati tra loro dallo spazio e dal tempo. Oggi molti archivi pubblici e privati sono computerizzati e la possibilità che essi hanno di comunicare tra loro può minacciare la "privacy" e la libertà di tutti. L'illustrazione mostra un quadro approssimativo dei vari archivi adottati in passato, dal "Libro del Catasto" (stilato nel 1086 su commissione del Re Guglielmo I, in Gran Bretagna) fino a quelli di più recente applicazione

alle cifre, per facilitarne la lettura a colpo d'occhio.

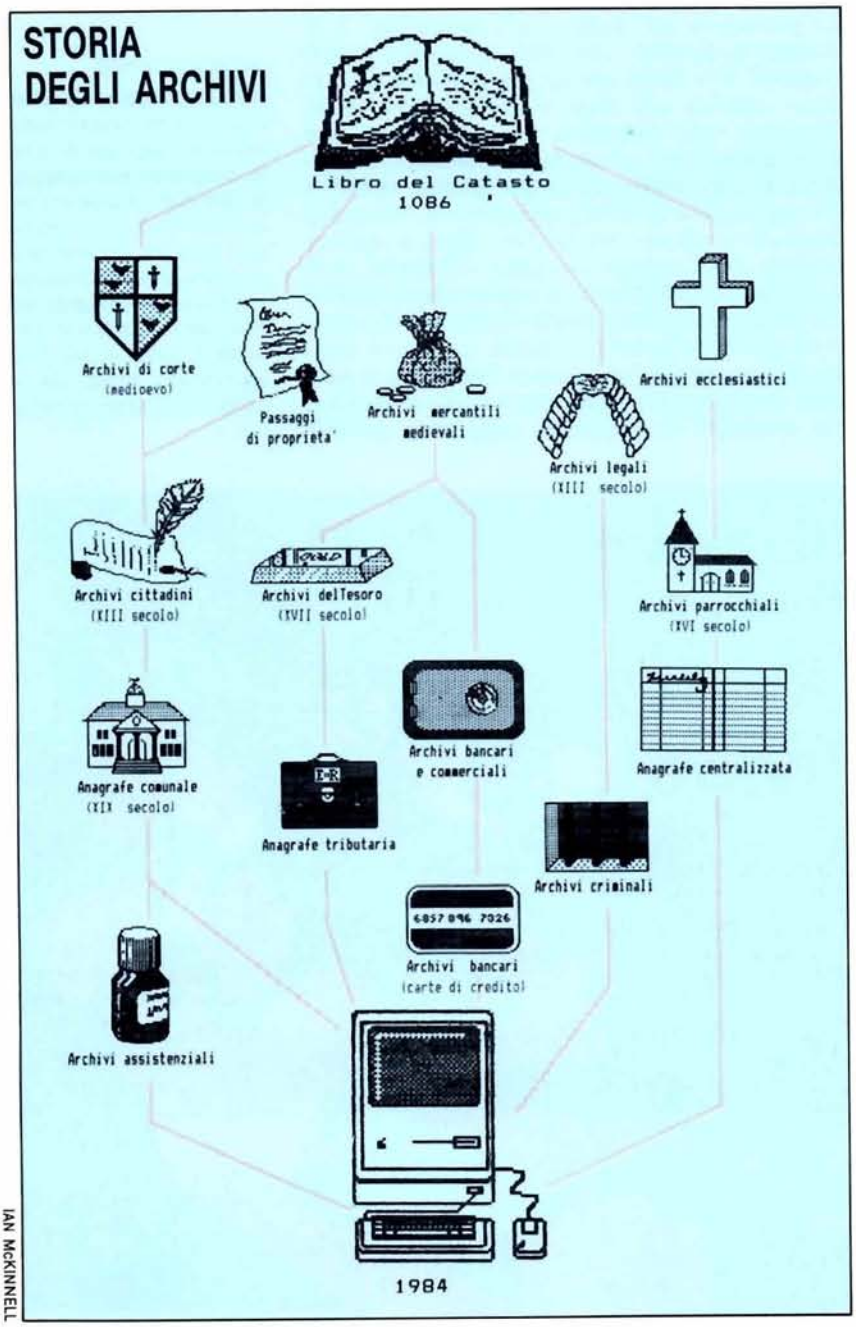
Prima di affermare se un file possa essere usato senza problemi sul nostro computer, è necessario prevederne la lunghezza massima (cioè quanti record dovrà contenere), calcolare la memoria occupata da ciascun record ed infine verificare se la memoria disponibile sul computer è sufficiente per immagazzinare l'intero file. Un record composto da un nome, da tre campi per l'indirizzo di 30 caratteri ciascuno, e da un numero telefonico di 10 caratteri, occupa complessivamente 130 byte. Usando dischetti da 200 Kbyte, si potranno inserire, su ciascun dischetto, fino a 1500 record. Al contrario, in un sistema privo di unità a dischetti, con soli 48 Kbyte di memoria RAM, si potranno immagazzinare soltanto 300 record (lasciando 10 Kbyte per il programma e per il sistema operativo). In realtà, per riordinare il file o eseguire altre operazioni, è necessario uno spazio di lavoro a parte, limitando le dimensioni del file praticamente a metà dell'area di memoria disponibile. Questi due sistemi sono totalmente diversi: nel secondo caso i file conservati su nastro, per essere elaborati, vanno trasferiti per intero in memoria; nel primo invece, grazie alla velocità degli accessi al file su disco, l'elaborazione avviene in memoria, ma il file viene tenuto sul dischetto.

Alcuni programmi per la gestione di dati più sofisticati, indicati col nome "Database manager", consentono di estrarre un dato da un file (supponiamo il totale delle spese di un anno) e collegarlo o confrontarlo con un secondo dato proveniente da un altro file (ad esempio l'incasso totale delle quote associative). Questa operazione, di per sé piuttosto semplice, permette di inviare una circolare ai membri del club per informarli periodicamente sulla situazione di cassa. Lo stesso programma si occupa di stampare nomi ed indirizzi sulle buste (o su apposite etichette). Alcuni programmi per database, addirittura, si possono collegare ad un word processor, per produrre lettere o tabulati. Teoricamente, su di un database si possono eseguire un gran numero di operazioni, benché all'atto pratico vi siano delle limitazioni, soprattutto se il computer usato ha una memoria poco estesa ed impiega nastri magnetici anziché dischetti. I programmi per database meno sofisticati spesso operano su un solo archivio alla volta e, pur potendo contenere tutte le informazioni viste in precedenza, non possono in realtà collegarle o confrontarle efficacemente.

Il maggior pericolo cui si va incontro durante le elaborazioni è l'accidentale perdita dei dati immagazzinati in memoria. Conviene sempre possedere una o più "copie di sicurezza" delle informazioni importanti ed eseguire frequentemente delle stampe, in modo che nulla vada irrimediabilmente perduto.

I programmi database meno sofisticati gestiscono un solo archivio alla volta. Tuttavia, la procedura d'immissione e di richiamo delle informazioni è sempre molto semplice e paragonabile alla compilazione manuale di schede o moduli. I risultati delle ricerche possono essere riportati sia sullo schermo che sulla stampante. I singoli record possono venir richiamati in base a particolari criteri di ricerca e di ordinamento ottenendo, ad esempio, un elenco di tutti i membri in ritardo nel pagamento delle quote annuali e che hanno un particolare codice postale. Il sistema permetterà anche di stampare solo alcuni campi di un record, come il nome ed i campi d'indirizzo nel caso della preparazione di etichette per l'invio di lettere. Pacchetti software di questo tipo sono disponibili per la maggior parte di home computer con memoria a nastro, ma per i sistemi a disco la gamma è molto più ampia.

I sistemi di database su home computer vengono spesso usati per catalogare raccolte



IAN McKINNELL

di dischi, libri e francobolli, riferimenti bibliografici oppure per registrare i risultati di calcio o di altre partite e competizioni.
Pacchetti più sofisticati permettono di conformare la visualizzazione dei dati sullo schermo ad eventuali sistemi manuali usati in precedenza. Altri pacchetti consentono di estrarre i dati dal database confrontando i campi numerici (ad esempio, tutti i membri del club con arretrati di pagamento superiori ad una certa cifra). È quindi importante vagliare le caratteristiche del programma, per trovare quale risponda meglio alle nostre particolari esigenze.
Un computer dotato di un programma per database si dimostra assai utile nelle operazioni ripetitive o quando si ha bisogno di memorizzare o recuperare rapidamente una serie di informazioni. Esistono, tuttavia, alcune applicazioni che mal si adattano a questi sistemi. Ad esempio, la ricerca di un numero telefonico, per effettuare una chiamata, è scarsamente utile: nel tempo impiegato ad accendere il computer, caricare il programma di database, richiamare il file interessato e richiedere un particolare numero si potrebbero fare 6 telefonate, cercando manualmente ciascun numero sull'elenco o su una rubrica. Un file contenente nomi e numeri telefonici potrebbe essere impiegato per altri scopi, come la stampa di etichette o di lettere standard, ma è insensato usare un sistema di elaborazione quando già esistono sistemi manuali sperimentati e perfettamente adeguati.
I pacchetti più sofisticati, oltre a possedere tutte le capacità dei precedenti, permettono di ottenere informazioni quali l'incasso totale delle quote associative del club oppure il servizio più usato tra quelli offerti dal club. È possibile anche collegare differenti file di dati, in modo da integrare informazioni di diverso tipo, ma riferite ad uno stesso soggetto, per esempio tutte le notizie riguardanti una particolare data. Per far funzionare un programma di così vaste capacità, occorre in genere un computer gestionale o da ufficio con un minimo di 64 Kbyte di memoria e dotato di unità a dischetti.
Negli ultimi anni l'uso dei sistemi informativi è notevolmente aumentato. I medici usano database per avere informazioni complete sulla "storia clinica" dei loro pazienti; i ricercatori se ne servono per classificazioni specialistiche e per confronti incrociati di dati; i commercianti tengono su di essi informazioni aggiornate sui clienti da cui estrarre elenchi di indirizzi per l'invio di circolari (i cosiddetti "mailing list"). Maggiore è la quantità di dati da elaborare, più comune è l'uso di questi sistemi. I database possono oggi gestire qualsiasi tipo d'informazione, dai semplici elenchi ai più complessi sistemi di file multipli, e sono in grado di produrre elaborati resoconti e di riorganizzare i dati in loro possesso in modo da poterli poi trasferire ad altri sistemi software come, ad esempio, un word processor.
La capacità di comunicare con servizi di database pubblici (quali il Prestel ed il Micronet 800), permette ai possessori di home computer di accedere ad un'enorme quantità di informazioni. In un futuro ormai prossimo anche i microcomputer domestici potranno trasformarsi in terminali collegati ad unità di elaborazione centralizzate.

DATABASE A CONFRONTO

In questa tavola vengono paragonate le caratteristiche dei programmi per database di maggior successo, funzionanti con il sistema operativo CP/M della Digital Research

Caratteristiche	DBASE II	DATASTAR	CONDOR	SUPERFILE
Records per file	65536	(file da 512 K)	32767	(mem. max.)
Caratteri per record	1000	(mem. max.)	1-2K	20.000 ca.
Caratteri per campo	254	128	77	(mem. max.)
N. di tipi di informazione	32	255	127	250
N. campi per record	32	255	127	nessun limite
Lunghezza variabile dei campi	NO	NO	NO	SI
Possibilità di allungare i record	NO	SI	SI	SI
N. chiavi per record	7	(Indice da 64K)	(mem. max.)	5000 ca.
Ricerca di sottostringhe	SI	NO	SI	SI
Ricerca fonetica	NO	NO	NO	SI
Collegamento con normali linguaggi di programmazione	NO	NO	NO	SI
Linguaggio speciale di programmazione	SI	SI	SI	NO

mem. max. = quantità massima di memoria RAM disponibile sull'apparecchio

Sprite... subacquei

Dopo aver esaminato la definizione degli sprite, ecco come muoverli sullo schermo

Per posizionare uno sprite sul video è necessario definire una coppia di coordinate X e Y. Ciascuno sprite viene realizzato su di una griglia di 21 × 24 pixel e le coordinate, la cui origine è nell'angolo superiore a sinistra della griglia stessa, sono poi conservate in registri interni al chip VIC. L'assegnamento delle coordinate avviene in questo modo:

Numero dello sprite	0		1		2		3		7	
Coordinata	X	Y	X	Y	X	Y	X	Y	X	Y
Registro VIC	V	V+1	V+2	V+3	V+4	V+5	V+6	V+7	V+14	V+15

Ciascuna locazione può accogliere valori compresi tra 0 e 255, più che sufficienti per specificare uno dei 200 pixel nella direzione verticale (asse delle Y): i valori in sovrappiù vengono usati per permettere allo sprite di scomparire e riapparire dal margine superiore o inferiore del video. Viceversa, per individuare i 320 pixel nella direzione orizzontale (asse delle X) non è sufficiente un solo byte, perciò ad ogni sprite viene associato un ulteriore bit. Questi bit aggiuntivi si trovano raggruppati in un unico registro con indirizzo V+16. La figura in basso mostra i confini del video entro i quali gli sprite non espansi sono interamente visibili.

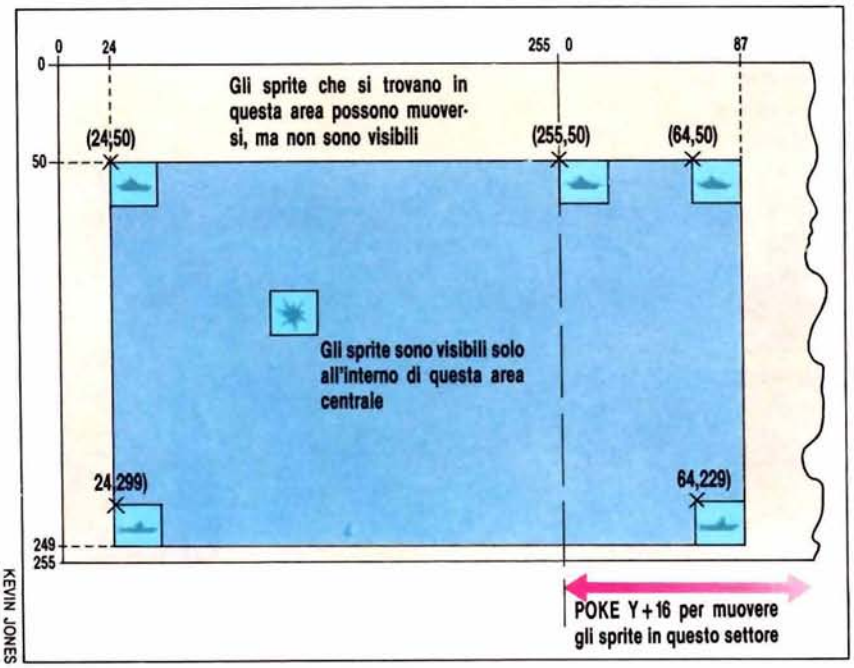
Il movimento della nave è controllato dalle linee di programma 230-250 e 270-290. Le coordinate iniziali della nave vengono definite nella linea 2270 della routine di creazione dello sprite (vedere a pag. 745). La nave si muove solo orizzontalmente in quanto la coordinata Y viene tenuta fissa e assegnando ad essa il valore 80, la nave appare nella corretta posizione: sulla superficie del mare, realizzata nella routine di preparazione dello schermo. Il valore iniziale della coordinata X della nave, X0, è 160: quindi all'inizio del gioco essa compare in posizione centrale.

La nave viene guidata da tastiera, battendo i tasti "Z" e "X" rispettivamente per gli spostamenti verso sinistra e destra. Le linee 230-250 del ciclo principale ricevono (istruzione GET) il carattere battuto sulla tastiera. Se non viene premuto alcun tasto, l'esecuzione del programma continua, altrimenti (comando di INPUT) il programma si ferma, finché il tasto viene tenuto abbassato. Il valore ottenuto mediante la GET viene quindi assegnato ad A$ ed utilizzato per modificare la coordinata X della nave. Se il tasto battuto è "Z", la nave esegue un piccolo spostamento verso sinistra, facendo diminuire la coordinata X. Viceversa, col tasto "X" si ottiene uno spostamento a destra, facendo aumentare la coordinata X. La seconda parte delle linee 240 e 250 contiene un test per verificare se la nave ha raggiunto i suoi limiti di spostamento. In BASIC, se la condizione di verifica in una frase IF...THEN è falsa, il resto della linea viene ignorato. Nel nostro programma, l'analisi della condizione è strutturata in modo tale da eliminare gli sprechi di tempo per calcoli non necessari. Ad esempio, se viene battuto il tasto "X" ed il valore di X0 cresce, non è necessario esaminare il valore limite minimo di X0. Dopo la verifica, il corrente valore di X0 viene inserito (con una POKE) nella locazione V, che rappresenta il registro della coordinata X per lo sprite 0.

Gli spostamenti del sottomarino sono controllati dalle linee 300-350 e 2500-2750, ma vengono operati all'interno del corpo principale del programma, come mostrato nel diagramma di flusso alla pagina seguente. La subroutine per la ridefinizione delle coordinate del sottomarino, che inizia alla linea 2500, utilizza la funzione RND per definire la profondità (Y3) e la velocità (DX) del sottomarino. Y3 è sempre un numero intero compreso tra 110 e 214, quindi lo sprite del sottomarino non apparirà mai al di sopra del mare, né al di sotto del fondale. Il valore della velocità, DX (compreso tra 1 e 3), definisce il numero di pixel che, di volta in volta, dovranno essere sommati o sottratti alla coordinata X. Quest'ultima ed il bit di registro V+16, che controlla le coordinate X

La posizione degli sprite

Il disegno presenta i parametri per il posizionamento degli sprite sul video del Commodore 64. Gli sprite sono visibili soltanto se si trovano nella parte centrale dello schermo, ma scompaiono, invece, se fuoriescono nella cornice esterna. Gli sprite posti ai quattro angoli dell'area centrale indicano il limite massimo di spostamento entro il quale restano ancora visibili per intero. I numeri tra parentesi rappresentano le coordinate dell'angolo superiore sinistro dei vari sprite



KEVIN JONES

dello sprite 3 superiori a 255, devono essere entrambi posti a 0.
Nelle linee 300-350 del ciclo principale, il valore della variabile DX viene sommato alla coordinata X del sottomarino. Il risultato ottenuto viene quindi verificato, per scoprire se il sottomarino ha raggiunto il margine dello schermo. Le linee 340 e 350 servono quando X3 assume valori superiori a 255. Ad esempio, se la coordinata X cresce fino a 256, il bit corrispondente allo sprite 3 (nel registro V+16), viene posto ad 1 ed il registro della coordinata X riparte nuovamente da 0. La seguente tabella mostra ciò che accade nei registri quando il sottomarino incrocia il margine X=255:

X3	V+16								V+6							
254	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
255	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
256	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
257	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Se il valore di X3 supera 255, la variabile H3 assume il valore 1 e, di conseguenza, L3 viene riportato a 0. I valori di L3 ed H3 possono quindi essere inseriti nei registri V+6 e V+16.

Lancio delle cariche di profondità

In qualsiasi momento, durante il gioco, è possibile lanciare cariche di profondità. Per facilitare la stesura del programma stabiliremo che, dopo il lancio di una carica, è possibile lanciarne una seconda soltanto se:
a) il sottomarino è stato colpito; oppure
b) le cariche hanno mancato il sottomarino e si trovano ad una profondità superiore alla sua; oppure
c) le cariche hanno mancato il sottomarino ed hanno raggiunto il fondale.

Il corpo principale del programma deve occuparsi di due operazioni, connesse alle cariche di profondità: scoprire quando viene battuto il tasto "M" e controllare il movimento verticale della carica lanciata. Inoltre deve garantire che non vengano lanciate nuove cariche, se ve ne sono altre in fase di discesa. Questa verifica può essere effettuata usando un "flag". Una tale tecnica viene spesso usata nel controllo dei programmi, poiché permette di segnalare il verificarsi o meno di un particolare evento. In questo caso, per segnalare la discesa di una carica di profondità, si utilizza la variabile FL: il suo valore sarà 1 se c'è una carica in fase di discesa, oppure 0 negli altri casi. All'inizio del gioco, nella linea 100 del programma, la variabile FL viene inizializzata a 0. La linea 260 richiama la subroutine per la messa a punto delle cariche di profondità (linea 3000) qualora venga premuto il tasto "M" e, contemporaneamente (grazie ad un AND logico), il flag FL contenga il valore 0. La linea 380 passa il controllo del programma ad una seconda subroutine, situata alla linea 400, che guida il movimento della carica di profondità lanciata.

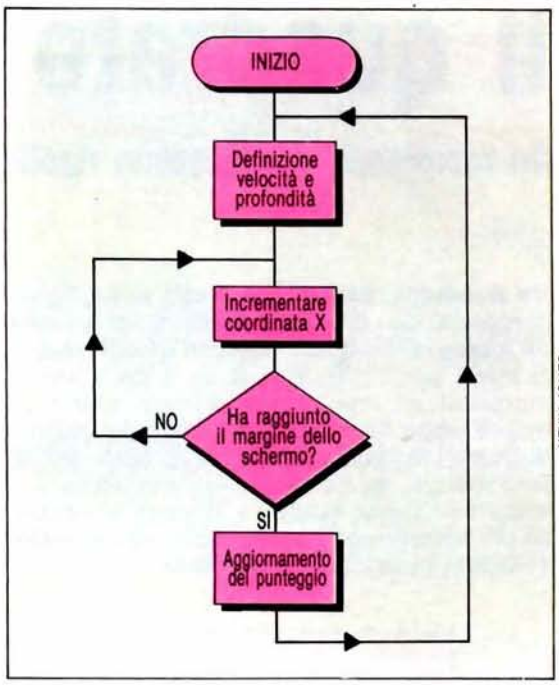


Diagramma di flusso
Questo semplice diagramma di flusso chiarisce come viene controllato il movimento del sottomarino: il programma assegna di volta in volta valori casuali alla profondità ed alla velocità del sottomarino, verificando che esso si trovi sott'acqua, ma non al di sotto del fondale. Lo sprite del sottomarino attraversa lo schermo in linea retta fino a raggiungere il margine opposto

KEVIN JONES

La subroutine di messa a punto delle cariche di profondità esegue tre operazioni:
1) Pone ad 1 il valore di FL, per segnalare il lancio di una carica.
2) Definisce le coordinate di partenza: la coordinata X è identica a quella della nave, mentre la coordinata Y viene fin dall'inizio posta a 95. Quindi la carica di profondità viene a trovarsi esattamente al livello del mare.
3) Accende lo sprite della carica di profondità.

La subroutine per il movimento della carica di profondità controlla lo spostamento verso il margine inferiore dello schermo. Inoltre, vanno eseguite le seguenti verifiche:
1) Se la carica di profondità ha superato il sottomarino oppure (OR logico) ha raggiunto il fondale.
2) Se la carica di profondità ha colpito il sottomarino.

Nel primo caso, lo sprite della carica viene spento ed il flag FL riportato a 0, consentendo così il lancio di una seconda carica. Il secondo caso viene verificato sfruttando una specifica caratteristica degli sprite del Commodore 64: il registro di collisione degli sprite. Come accade per altri registri del chip VIC, questo registro (V+30) possiede un bit per ogni sprite. Se due sprite entrano in collisione, i bit corrispondenti vengono posti a 1. Dunque, se la carica di profondità (sprite 2) colpisce il sottomarino (sprite 3), il contenuto del registro V+30 sarà 12 (00001100=12). L'esame del contenuto del registro V+30 ci permette quindi di stabilire se la carica è andata a segno. In questo caso, il controllo del programma passa alla subroutine "Bersaglio colpito", alla linea 5000. L'ultima parte del nostro lavoro riguarderà quest'ultima subroutine, le istruzioni per aggiornare il punteggio e quelle per far ripartire il gioco.

SUBROUTINE PER IL MOVIMENTO DEL SOTTOMARINO

```
100 GOSUB 2500: REM SELEZIONE COORDINATE
    SOTTOMARINO
200 GET A$
240 IF A$="Z" THEN I0=I0-1.5: IF I0<24
    THEN I0=24
250 IF A$="I" THEN I0=I0+1.5: IF I0>245
    THEN I0=245
260 IF A$="M" AND FL=0 THEN GOSUB 3000:
    REM SELEZIONA LE CARICHE DI PROFONDITA'
265 :
270 REM MOVIMENTO DELLA NAVE
290 POKE V,I0
295 :
300 REM MOVIMENTO DEL SOTTOMARINO
310 X3=X3+DX
315 :
320 REM IL SOTTOMARINO HA RAGGIUNTO IL
    MARGINE?
330 IF X3>360 THEN DX=-1: GOSUB 5500:
    GOSUB 2500
340 H3=INT (X3/256): L3=X3-256*H3
350 POKE V+6, L3: POKE V+16, PEEK (V+16)
    OR (8*H3)
360 IF FL=1 THEN GOSUB 4000: REM MOVIMENTO
    DELLE CARICHE DI PROFONDITA'
370 GOTO 200: REM RIDA' INIZIO AL CICLO
    PRINCIPALE
380 :
390 :
2500 REM RIDEFINISCE LE COORDINATE DEL
     SOTTOMARINO
2510 Y3=110+INT(RND(TI)*105)
2520 X3=0: DX=RND(TI)*3+1
2530 POKE V+7,Y3: POKE V+6,X3
2540 POKE V+16: PEEK(V+16) AND 247
2550 RETURN
2560 :
2570 :
3000 REM ABILITA LE CARICHE DI PROFONDITA'
3010 :
3020 REM AGGIUSTA IL VALORE DEL FLAG
3030 FL=1
3040 :
3050 REM SELEZIONA LE COORDINATE
3060 Y2=95: X2=I0
3070 POKE V+4, X2: POKE V+5,Y2
3080 :
3090 REM ACCENDE LO SPRITE 2
3100 POKE V+21: PEEK (V+21) OR 4
3110 RETURN
3120 :
3130 :
4000 REM MUOVE LA CARICA DI PROFONDITA'
4010 :
4020 REM INCREMENTA LA COORDINATA Y
4030 Y2=Y2+2
4040 POKE V+5, Y2
4050 :
4060 REM VERIFICA FONDALE & SCOMPARSA
     DELLA CARICA
4070 IF Y2=Y3+25 OR Y2>216 THEN PKE V+21,
     PEEK (V+21)  AND 251: FL=0
4080 :.
4090 REM VERIFICA PER BERSAGLO COLPITO
4100 IF PEEK (V+30)=12 THEN GOSUB 5000:
     REM ROUTINE  PER BERSAGLIO COLPITO
4110 RETURN
4120 :
4130 :
```

Il quadrato magico

Un rompicapo antichissimo risolto con un mezzo recentissimo



Un quadrato magico non è che una griglia, composta da un certo numero di caselle all'interno delle quali vengono inseriti numeri interi positivi (1, 2, 3, 4, 5,...). Lo 0 non è ammesso ed uno stesso numero non può essere usato due volte. L'obiettivo del gioco è disporre i numeri sulla griglia in modo che la loro somma, su ciascuna riga e colonna, dia sempre lo stesso risultato. Il quadrato magico più semplice è di 3 × 3 caselle ed una delle soluzioni possibili è la seguente:

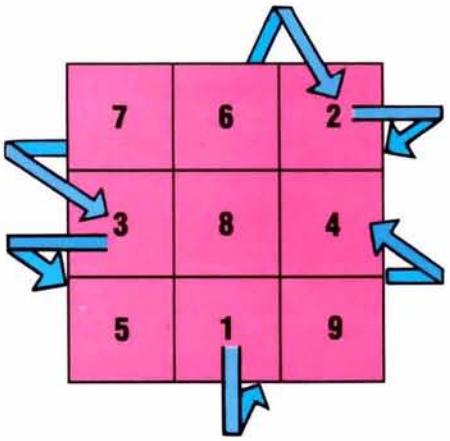
7	6	2	15
3	8	4	15
5	1	9	15
15	15	15	

In questo esempio, tutte le righe e le colonne danno come risultato 15. Consideriamo adesso una griglia 5 × 5 e proviamo a disporre su di essa i numeri compresi tra 1 e 25. Ci accorgeremo che l'operazione non è affatto facile come sembra e si commettono certamente molti errori prima di trovare la giusta soluzione.

Fortunatamente, il nostro computer può rendere le cose più semplici: basta scrivere un programma che inserisca i numeri nelle caselle e che verifichi, individualmente, il risultato di ciascuna riga e colonna. Tuttavia, un simile metodo potrebbe impiegare anche molte ore nella ricerca di una soluzione per quadrati di modeste dimensioni, larghi ad esempio 7 o 9 caselle. Ciò che ci occorre è una procedura per la generazione dei numeri. Il procedimento qui descritto funziona in qualsiasi circostanza:

1) Iniziamo inserendo il numero 1 nella casella centrale dell'ultima riga della griglia.
2) Dopo ogni inserimento, ci spostiamo di una posizione in basso e a destra prima di inserire il numero successivo. Se lo spostamento a destra ci porta fuori dal perimetro della griglia, si passa alla prima colonna del lato opposto. Analogamente, se uno spostamento verso il basso provoca una fuoriuscita dalla base della griglia, si passa alla prima riga in alto. Questo procedimento è simile all'effetto di "sequenza ciclica", che si ha talvolta sullo schermo.
3) Se la successiva casella è già occupata, ci si sposta di una posizione a sinistra rispetto all'ultima casella usata.
4) Il processo continua fino a raggiungere l'ultima casella libera.

Applicando questa procedura al nostro esempio, otterremo il seguente risultato:





IAN McKINNELL

PICCOLE E GRANDI... MAGIE!

Questi quadrati sono stati generati e quindi verificati dal programma, come dichiarato dal messaggio sul video. Nel quadrato 15 × 15, come conseguenza dell'algoritmo di costruzione scelto, sono presenti numeri a due e tre cifre

52	42	32	22	12	2	73	72	62
63	53	43	33	23	13	3	74	64
65	55	54	44	34	24	14	4	75
76	66	56	46	45	35	25	15	5
6	77	67	57	47	37	36	26	16
17	7	78	68	58	48	38	28	27
19	18	8	79	69	59	49	39	29
30	20	10	9	80	70	60	50	40
41	31	21	11	1	81	71	61	51

È abbastanza semplice scrivere un programma che esegua queste operazioni su quadrati di qualsiasi dimensione. Dovremo creare una matrice bidimensionale, vuota, della giusta misura per il quadrato magico desiderato ed usare un ciclo iterativo per l'inserimento dei numeri, attenendosi alle regole viste. Si noti che questo sistema è applicabile solo a quadrati magici con un numero di caselle dispari e il programma rifiuterà di produrre griglie con numero di caselle pari.
La visualizzazione del quadrato dovrà essere molto accurata. Per maggior chiarezza, i numeri dovranno essere correttamente allineati in righe e colonne. Se il nostro computer dispone del comando PRINT USING, un simile risultato è facile da ottenere, altrimenti conviene inserire i numeri da visualizzare in una stringa, separandoli tra loro con opportune spaziature in modo da ottenere sempre serie di numeri con identica lunghezza. La subroutine che esegue l'operazione descritta è la seguente:

```
1000 REM Converti A in A$ e allinea
1010 A$ = STR$(A)
1020 IF LEN(A$)<3 THEN A$ = " "+A$: GOTO 1020
1030 RETURN
```

Naturalmente, la procedura va adattata al tipo di computer usato.
Un altro problema riguarda il formato dello schermo: infatti molti micro non sono in grado di visualizzare quadrati magici di grandi dimensioni. Uno schermo a 40 colonne non può contenere più di 13 colonne di numeri a due cifre. Ma, dato che un quadrato di 13 × 13 presenta anche numeri a tre cifre, uno schermo di queste dimensioni potrà contenere al massimo un quadrato magico di 9 × 9. Al contrario, l'uso di una stampante permette di visualizzare quadrati molto più grandi: le stampanti, in genere, hanno 80 o 132 colonne. Le stampe di quadrati ancora più grandi potranno essere eseguite in fasi successive e quindi combinate assieme.
L'obiettivo principale è quello di creare quadrati magici quanto più grandi possibile e presentarli poi in modo chiaro ed ordinato. Come esperimento, si provi a scrivere un programma per tentativi successivi, e verifichiamo il tempo impiegato a trovare una risposta (ricordiamo però che ci vorrà molto tempo prima di ottenere un risultato positivo). In alternativa, cerchiamo di trovare un metodo più rapido per generare i quadrati magici.

130	114	98	82	66	50	34	18	2	211	210	194	178	162	146
147	131	115	99	83	67	51	35	19	3	212	196	195	179	163
164	148	132	116	100	84	68	52	36	20	4	213	197	181	180
166	165	149	133	117	101	85	69	53	37	21	5	214	198	182
183	167	151	150	134	118	102	86	70	54	38	22	6	215	199
200	184	168	152	136	135	119	103	87	71	55	39	23	7	216
217	201	185	169	153	137	121	120	104	88	72	56	40	24	8
9	218	202	186	170	154	138	122	106	105	89	73	57	41	25
26	10	219	203	187	171	155	139	123	107	91	90	74	58	42
43	27	11	220	204	188	172	156	140	124	108	92	76	75	59
60	44	28	12	221	205	189	173	157	141	125	109	93	77	61
62	46	45	29	13	222	206	190	174	158	142	126	110	94	78
79	63	47	31	30	14	223	207	191	175	159	143	127	111	95
96	80	64	48	32	16	15	224	208	192	176	160	144	128	112
113	97	81	65	49	33	17	1	225	209	193	177	161	145	129

```
10 REM ******************************************
15 REM **          QUADRATI MAGICI              **
20 REM ************* DEFINIZIONE ******************
30 M=19: DIM A(M,M)
40 PRINT: PRINT"Quadrati Magici"
50 PRINT: PRINT"Quante righe (da 1 a 19)": INPUT S
60 IF S<0 OR S<>INT(S) THEN PRINT"ERRORE": GOTO 50
70 IF S>M THEN PRINT"ERRORE": GOTO 50
80 IF S/2=INT(S/2) THEN PRINT"ERRORE - Solo numeri dispari": GOTO 50
90 REM ********** GENERA IL QUADRATO ***************
100 X=INT(S/2)+1: Y=S: C=1
110 A(X,Y)=C
120 C=C+1: IF C>S*S THEN GOTO 200
130 X=X+1: IF X>S THEN X=1
140 Y=Y+1: IF Y>S THEN Y=1
150 IF A(X,Y)<>0 THEN X=X-2: Y=Y-1
160 IF Y=0 THEN Y=S
170 IF X=0 THEN X=S
180 IF X=-1 THEN X=S-1
190 GOTO 110
200 REM ****** VISUALIZZA IL QUADRATO **************
210 PRINT: PRINT
220 FOR Y=1 TO S: FOR X=1 TO S
230 A=A(X,Y): GOSUB 380: PRINT" ";A$;" ";
240 NEXT X: PRINT: NEXT Y

250 REM ****** VERIFICA LINEE & COLONNE ************
260 F=0
270 FOR Y=1 TO S: T=0
280 FOR X=1 TO S: T=T+A(X,Y): NEXT X
290 IF F=0 THEN U=T: F=1
300 IF T<>U THEN PRINT"ERRORE - la riga  1  e la riga";Y;" danno
    totali diversi": STOP
310 U=T: NEXT Y
320 FOR X=1 TO S: T=0
330 FOR Y=1 TO S: T=T+A(X,Y): NEXT Y
340 IF T<>U THEN PRINT"ERRORE - la riga  1 e la colonna";X;" danno
    totali diversi": STOP
350 U=T: NEXT X
360 PRINT: PRINT"Tutte le righe e le colonne danno come somma";T
370 STOP
380 REM ***** CONVERS. STRINGA-NUMERO **************
390 A$=STR$(A)
400 IF LEN(A$)<3 THEN A$=" "+A$: GOTO 400
410 RETURN
```



Basic & Basic

Questo programma è stato scritto in Microsoft BASIC e può funzionare senza modifiche su quasi tutti i computer. I possessori di Spectrum devono inserire una LET in ciascuna istruzione di assegnazione. Il programma chiede quante righe (e, quindi, colonne) deve avere il quadrato magico, verificando poi che il numero introdotto sia positivo, intero e dispari. In seguito, elabora e visualizza il quadrato magico sul video e, a partire dalla linea 250, verifica l'esattezza del risultato ottenuto.

Software in vetrina

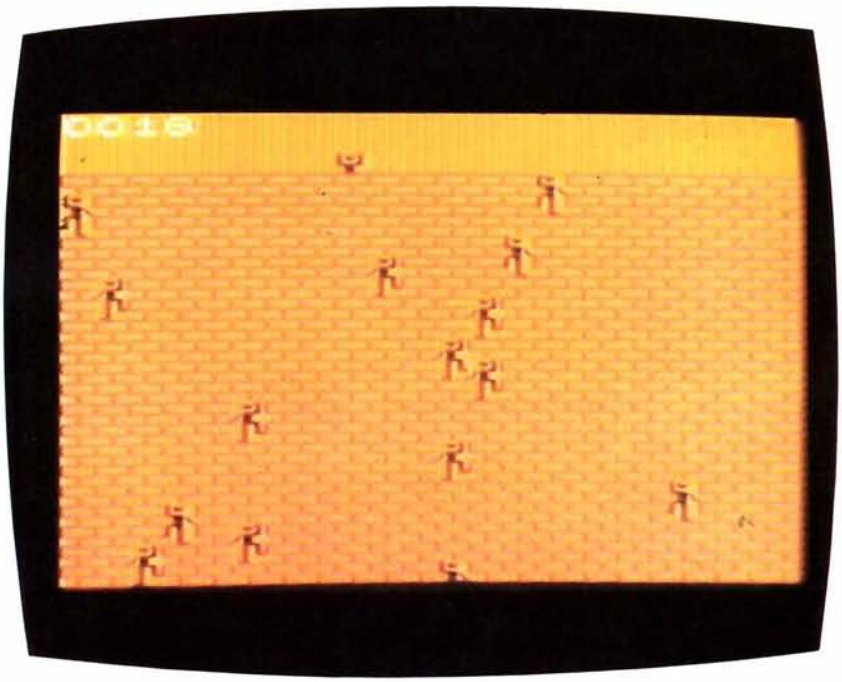
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Ecco altri due giochi, appartenenti al genere "arcade". Il primo di essi, *Zaxxon*, è diventato ormai un vero classico per appassionati e viene prodotto in più versioni, adatte ai vari home computer.



Zaxxon per Atari, Apple e Tandy Color

Ciò che maggiormente colpisce in questo gioco è la splendida grafica con effetto tridimensionale: un ottimo esempio di cosa si riesca ad ottenere anche con una limitata memoria a disposizione. L'obiettivo del giocatore, alla guida di un'astronave, è quello di abbattere quanti più robot, chiamati Zaxxon, possibile. L'impresa non è facile, perché ogni Zaxxon è barricato in una fortezza spaziale, protetta da una flotta di bellicose astronavi.

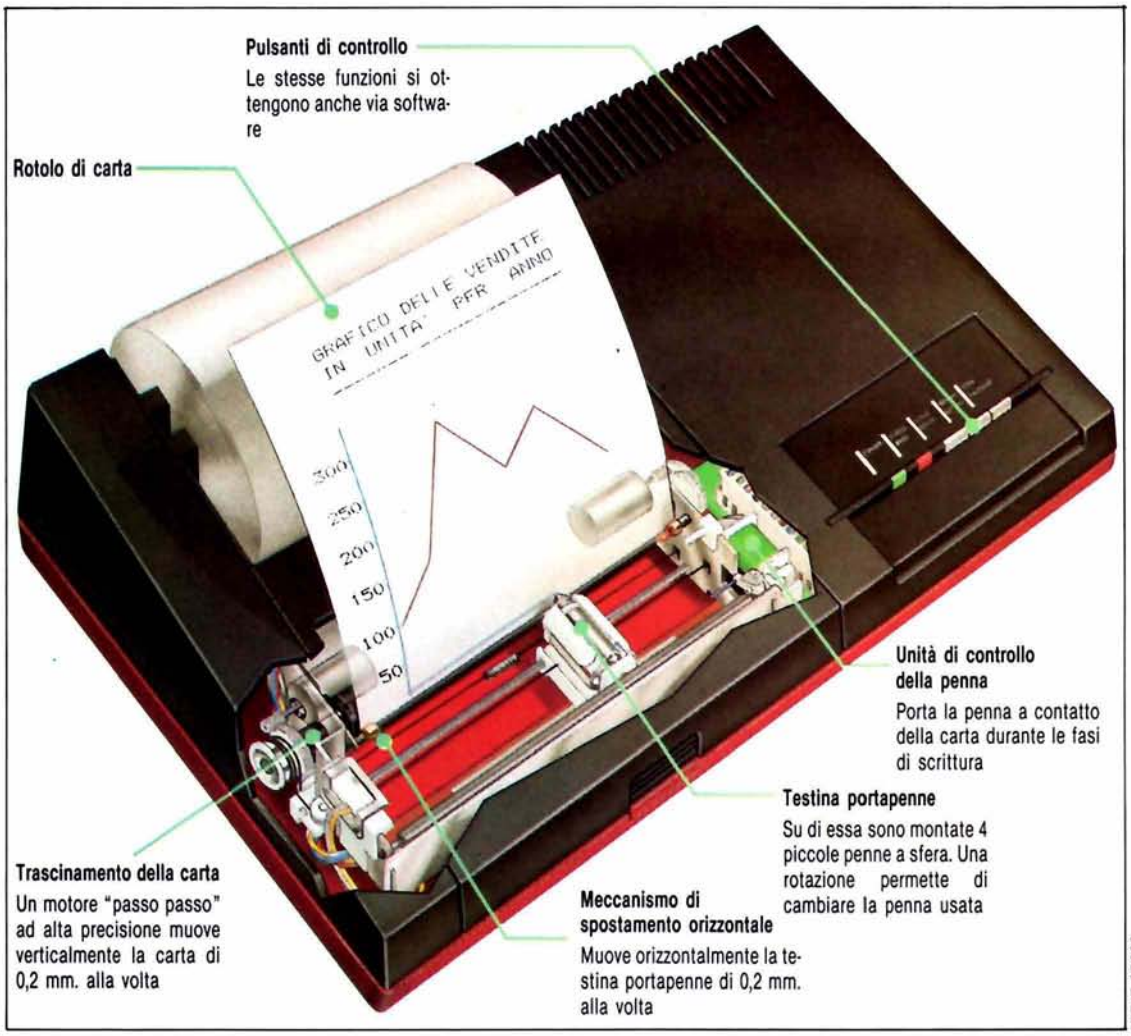


Siege per il Vic-20

Anche questo gioco si rifà ad un classico da sala giochi: la "scalata". Il giocatore, situato in cima all'alto muro di un castello, deve difendersi dall'attacco di un'orda di nemici. Costoro tentano la scalata e l'unica arma di difesa a disposizione del giocatore è costituita da barili di birra: con questi deve cercare di far cadere gli assalitori. Se a prima vista il gioco può apparire puerile, in realtà esso diventa presto avvincente. Una partita si conclude se anche un solo nemico riesce a scavalcare la cima del muro.

Le stampanti/plotter

I plotter professionali sono generalmente molto cari, ma esiste un'economica alternativa



STEVE CROSS

Un plotter da famiglia

La stampante/plotter possiede una risoluzione di 500 x 2000 pixel nel modo grafico e svariati "corpi" tipografici. Il colore della penna viene selezionato con una rotazione della testina, quindi le operazioni di scrittura o di disegno coinvolgono tanto lo spostamento orizzontale della testina in entrambe le direzioni, quanto lo scorrimento della carta verso l'alto o verso il basso. La penna può essere appoggiata o distaccata dalla carta: la prima posizione serve per disegnare, la seconda per posizionare la testina, senza lasciare alcuna traccia sul foglio. La meccanica dell'apparecchio è di alta precisione, ma il continuo movimento della carta e del cilindro possono causare, a lungo andare, piccole imperfezioni

Atari 1020, Commodore 1510 ed Oric MCP-40 sono solo alcune delle sigle sotto le quali viene venduto un identico plotter di basso costo per microcomputer. Sotto l'apparente diversità di marca si nasconde il medesimo apparecchio, fabbricato da un'industria giapponese e successivamente adattato alle esigenze delle varie marche (una tecnica di marketing oggigiorno molto diffusa).

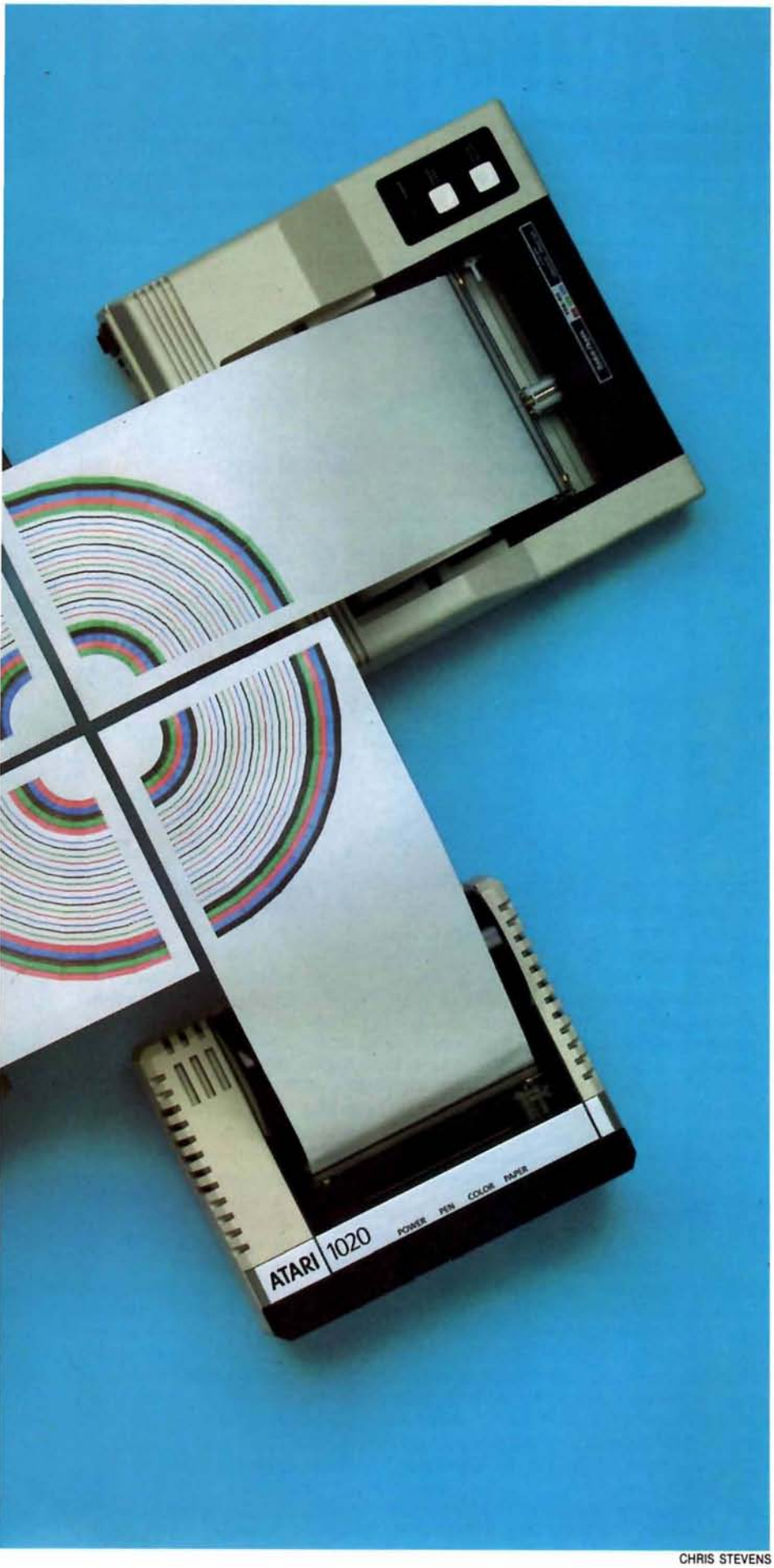
Questo plotter è in grado di produrre, oltre alla grafica, testi scritti quindi, proprio per questa combinazione, è più appropriato indicarlo col termine "stampante/plotter". L'apparecchio è dotato di una testina rotante, che contiene quattro piccole penne a sfera. Per tracciare una linea, la testina viene prima fatta ruotare per selezionare il colore desiderato (i colori standard sono: rosso, nero, verde e blu), quindi si sposta mantenendo la punta della penna aderente al foglio. Per tracciare linee orizzontali, la testina si sposta da destra a sinistra e viceversa mentre, per le verticali, è il foglio di carta che viene mosso verso l'alto o verso il basso. I caratteri alfanumerici per i testi vengono riprodotti in modo analogo, solo che i modelli delle lettere e dei simboli sono già presenti nella memoria della stampante/plotter e vengono richiamati in base ai codici inviati dal computer. La qualità della stampa è più che buona (sicuramente migliore di quella che si ottiene sulla maggioranza delle stampanti ad aghi di basso costo). Quando si trova in "modo testuale", la stampante/plotter funziona esattamente come una qualsiasi altra stampante. Anche se le dimensioni della carta sono di soli 11,5 cm (4,5 pollici), l'apparecchio può stampare 40 o 80 caratteri per riga. Le ridotte dimensioni della carta rendono il testo ad 80 caratteri piuttosto compresso, ma la definizione dei caratteri

resta ottima. Il testo può essere stampato in uno dei quattro colori a disposizione, con una velocità di stampa di circa 10 caratteri per secondo. La velocità è piuttosto ridotta, se paragonata a quella delle stampanti ad aghi, ma è pressappoco uguale a quella delle stampanti "a margherita".
Abilitando invece il "modo grafico", la stampante/plotter viene messa in condizioni di tracciare linee, curve e lettere di grandi dimensioni. Quando l'apparecchio, così predisposto, riceve dal computer dei caratteri, li interpreta come comandi e provvede immediatamente alla stampa. In BASIC, si usa il seguente formato per le istruzioni necessarie a tracciare linee:

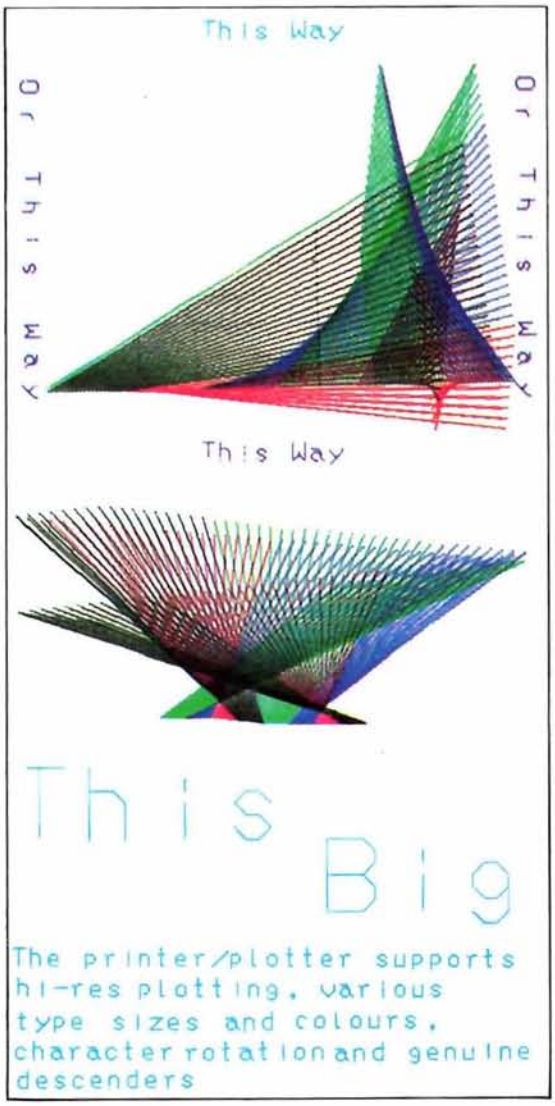
```
LPRINT "D 0,100,100,100,100,0,0,0"
```

dove D è il comando per tracciare le linee, mentre i numeri che seguono indicano la posizione dei vari punti di riferimento. In questo caso (presumendo che le coordinate iniziali della testina siano 0,0) viene disegnato un quadrato. Gli altri comandi adottano un analogo formato.
Nella stampante/plotter la larghezza della carta è idealmente suddivisa in 480 sezioni, usate per posizionare con precisione la testina di stampa. Ciò può far pensare che non ci sia alcun limite alle dimensioni verticali della grafica ma, di fatto, un prolungato scorrimento della carta può portare ad imprecisioni nel disegno, quindi la stampante/plotter non permetterà che la carta venga "trascinata" per più di una certa lunghezza.
Con questo apparecchio è possibile produrre immagini grafiche anche molto complesse, ma i programmi per la loro realizzazione richiedono molto tempo e lavoro: non è infatti possibile copiare un'immagine direttamente dallo schermo del computer. Disegnare delle linee è abbastanza facile ma la mancanza di comandi come PAINT o FILL, costringe a tracciare molte linee sottili e fitte per ottenere una colorazione uniforme delle immagini (non certo una procedura rapida né divertente!).
Inoltre, la stampante/plotter, non essendo in grado di lavorare su tutta la larghezza della carta, lascia un piccolo margine bianco su entrambi i lati. Tuttavia, è particolarmente adatta per disegnare grafici e possiede anche un comando che ne traccia automaticamente gli assi, con i parametri di misura inseriti ad opportuni intervalli.
Purtroppo, le penne a sfera si consumano molto velocemente e, se lasciate sulla testina di stampa per un certo tempo, l'inchiostro tende a seccarsi. Quando la macchina viene accesa, viene automaticamente eseguita una verifica di funzionamento su tutte le penne tracciando dei minuscoli quadratini in ciascun colore.
I plotter professionali sono estremamente precisi e producono una grafica a colori di alta qualità, ma hanno un prezzo notevolmente superiore: sarebbe assurdo, quindi, aspettarsi prestazioni analoghe da un apparecchio tutto sommato molto economico.





CHRIS STEVENS



Interfacce per il plotter

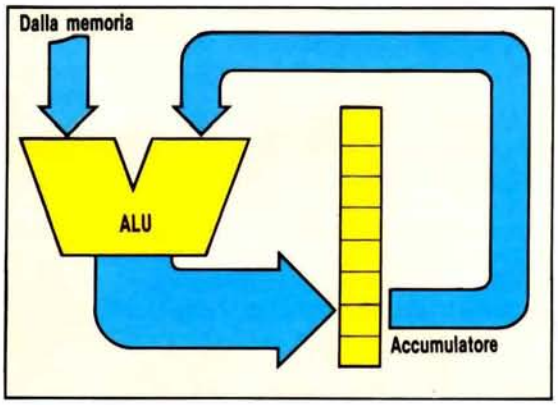
La tabella mostra quali stampanti/plotter possono essere usate con i microcomputer più diffusi. L'unico problema è quello di individuare il giusto sistema di collegamento. L'Oric, lo Sharp, il Commodore e l'Atari possiedono canali specifici. Lo Spectrum è privo di un'interfaccia adeguata

	Atari 1020	Commodore 1520	MCP-40	Oric MCP-40	Sharp MZ-80P5(K)	Tandy CGP-115
Atari 400/600/800	●					
BBC Micro			●	●		●
CGL/Sord M5			●	●		●
Colour Genie						●
Commodore Vic-20		●				
Commodore 64		●				
Dragon 32/64			●	●		●
Memotech MTX			●	●		●
Oric-1/Atmos			●	●		●
Sharp MZ-700					●	
Tandy TRS-80 Colour						●

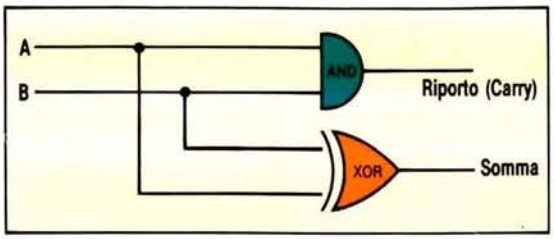
L'unità logica

Le funzioni dell'ALU, l'unità logica e aritmetica, contenuta nella CPU

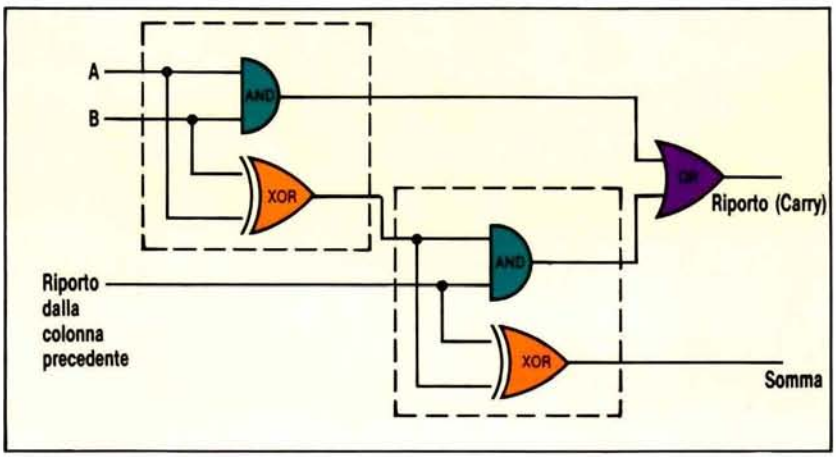
Le operazioni svolte dall'unità logico-aritmetica si distinguono in due categorie: quelle aritmetiche (addizione, sottrazione e incremento) e le tre operazioni logiche (OR Esclusivo, OR Inclusivo e AND).
Per alcune di queste funzioni, ad esempio l'addizione, occorrono due operandi. Per altre, come l'incremento, ne basta uno solo. In quest'ultimo caso, l'operando si trova in genere nel particolare registro della CPU, detto accumulatore. Quando sono invece necessari due operandi, il secondo viene preso dalla memoria principale.
I due numeri, sotto forma di impulsi elettrici, vengono inviati al circuito dell'ALU per l'esecuzione dell'operazione richiesta. Il risultato finale viene quindi trasferito nuovamente nell'accumulatore.
Il seguente schema mostra il flusso di un dato all'interno del chip ALU:



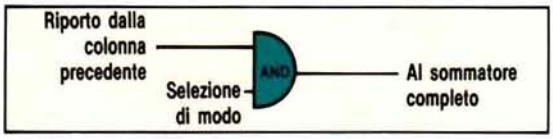
I valori trattati, sia nell'accumulatore che nella memoria, sono lunghi un byte e le operazioni all'interno dell'ALU vengono eseguite simultaneamente (ossia in parallelo) sugli otto bit che li compongono.
Per meglio comprendere la struttura elettronica di una ALU, consideriamo uno solo degli 8 bit e lo schema di un circuito che ci permetta di eseguire le 6 diverse operazioni nominate in precedenza. Chiameremo rispettivamente A e B i bit del primo e del secondo operando.
Alla base del funzionamento dell'ALU che ci accingiamo ad esaminare c'è un circuito, chiamato sommatore completo, che noi abbiamo già incontrato a pag. 526. Questo circuito utilizza due semisommatori composti da porte AND, OR e NOT.
Per semplificare il circuito, i due semisommatori possono anche prevedere l'impiego di una porta OR Esclusiva:



Per un circuito sommatore completo, i semisommatori vengono così collegati:

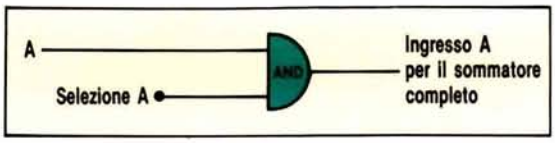


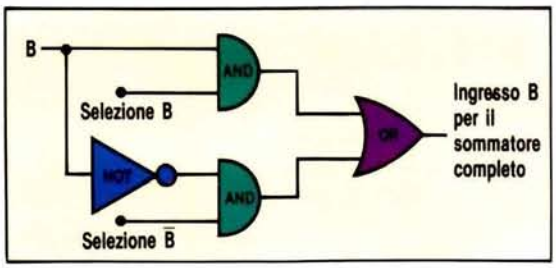
Aggiungendo piccoli circuiti, si possono sfruttare le porte del sommatore per eseguire le altre funzioni dell'ALU: basta introdurre una serie di segnali di controllo, il primo dei quali è detto "segnale di selezione di modo". L'ingresso indicato come "riporto dalla colonna precedente" viene usato nelle operazioni aritmetiche, ma non è necessario in quelle logiche. Il segnale di selezione del modo abilita o disabilita l'ingresso di riporto per mezzo di una porta AND.



Nelle operazioni aritmetiche, per le quali è necessario il valore di riporto, occorre che il segnale di selezione del modo venga posto ad 1, permettendo così al riporto di attraversare la porta AND. Altrimenti il segnale di selezione del modo viene posto a 0. Analogamente, possiamo aggiungere porte AND agli altri due ingressi. Quest'operazione ci permette di selezionare il bit A, il bit B oppure entrambi i bit assieme.
Durante il processo di sottrazione, per mezzo di addizioni (vedere a pag. 758), viene calcola-

to il "complemento a due" del numero che deve essere sottratto. Quest'operazione richiede il cambiamento di tutti gli 1 in 0 e viceversa. Per le sottrazioni dovremo aggiungere alcuni circuiti che permettano di selezionare la negazione del bit B. Dovremo quindi trasmettere il bit B attraverso una porta NOT, ed il relativo segnale di selezione verrà aggiunto tramite una nuova porta AND. Questo schema rappresenta il circuito finale:

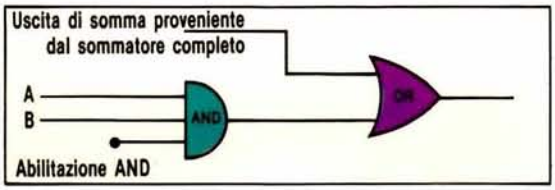




Con questi quattro segnali di controllo si eseguono tutte le operazioni aritmetiche. La seguente tabella mostra le varie combinazioni:

Funzione	Tipo di selezione	Selezione A	Selezione B	Selezione $\overline{B}$
Addizione	1	1	1	0
Sottrazione	1	1	0	1
Incremento di A (Seleziona a 1 il primo ingresso di riporto)	1	1	0	0
Incremento di B	1	0	1	0

Per tutte le operazioni logiche, alle quali non occorre l'ingresso "riporto dalla colonna precedente", possiamo porre a 0 il segnale di selezione del modo. Ciò significa che si può ottenere la funzione logica OR Esclusivo sull'uscita di somma ponendo a HI (= alto, ossia ad 1) le selezioni A e B e ponendo a LO (= basso, ossia 0) il segnale di selezione del modo.
Per ottenere la funzione AND dal circuito esistente occorre introdurre ingressi separati per A e B ed un segnale di selezione AND.



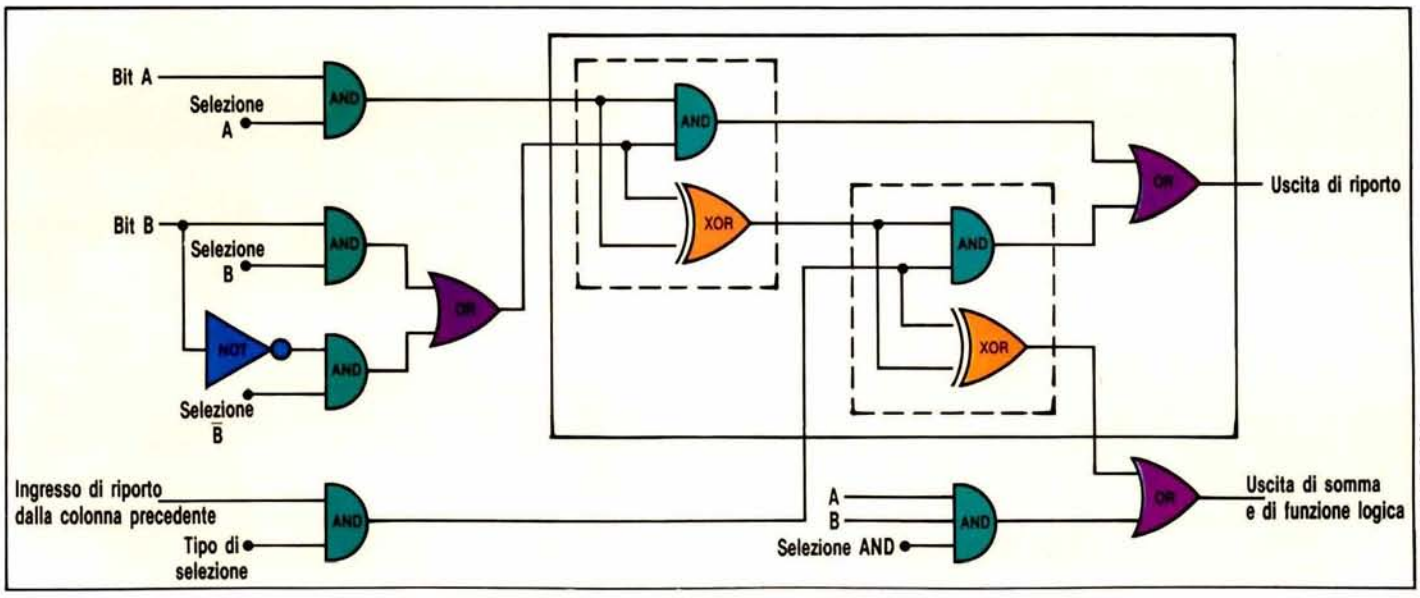
Infine, la funzione OR può essere realizzata associando su una porta OR le uscite XOR e AND. La seguente tavola della verità lo dimostra:

A	B	Uscita	Descrizione
0	0	0	
0	1	1	Funzione XOR
1	0	1	
1	1	1	Funzione AND

La tavola che segue mostra come sia possibile ottenere le tre funzioni logiche usando diverse combinazioni dei segnali di controllo:

Funzione	Tipo di selezione	Selezione A	Selezione B	Selezione $\overline{B}$	Selezione AND
XOR	0	1	1	0	0
AND	0	0	0	0	1
OR	0	1	1	0	1

Lo schema finale mostra lo "stadio" di un singolo bit in un circuito ALU: sono presenti il circuito sommatore completo e gli altri circuiti addizionali per i segnali di controllo. Il circuito completo, ovviamente, possiede otto circuiti di questo tipo posti in parallelo. L'uscita di riporto proveniente dall'ottava colonna viene usata come "flag" nell'apposito registro "di condizione" del processore.
Questa lezione conclude la serie sulla logica dedicata ai circuiti logici.



diagrammi di LIZ DIXON

I file su cassette

La conservazione dei dati in forma sequenziale avviene secondo modalità diverse

Ogni micro possiede le proprie tecniche di gestione dei file, pertanto la versione standard di un programma deve essere modificata per poter funzionare su un particolare apparecchio. Per questo motivo è importante conoscere quali siano le caratteristiche ed i comandi del nostro sistema, che fanno riferimento ai metodi standard di cui stiamo parlando. Esaminiamo, ad esempio, l'immagazzinamento dei file dati su un micro dotato di registratore a cassette. La prima cosa da notare è che i sistemi a cassetta, per loro propria natura, non possono trattare file ad accesso casuale ma solo quelli sequenziali (vedere a pag. 724).

La gestione di un file sequenziale comporta la lettura dell'informazione, la sua elaborazione ed infine il trasferimento dell'informazione modificata su un secondo file: quindi i file in uso (aperti) simultaneamente devono essere due. Poiché un registratore a cassetta non può spostarsi direttamente e con precisione da un punto all'altro del nastro, la maggior parte dei micro a cassette non sono in grado di gestire sistemi "a due file". Fanno eccezione quei pochi micro, come il Newbrain e la serie PET della Commodore, che possiedono porte per due unità a nastri: una per la lettura e l'altra per la scrittura.

Nella maggior parte dei micro, quindi, per apportare le modifiche necessarie occorre trasferire per intero il file nella memoria RAM del computer e, una volta concluse queste modifiche, il tutto viene nuovamente registrato sulla cassetta. Un simile procedimento provoca delle pause durante o alla fine dell'esecuzione del programma. Inoltre, i file dei dati devono avere dimensioni tali da poter essere contenuti in quella parte di RAM rimasta libera dopo il caricamento del programma applicativo.

Molti sono i metodi per registrare un'informazione su cassetta. Il più semplice non usa neppure file dati separati: tutte le variabili in uso vengono immagazzinate assieme allo stesso programma usando il comando SAVE. Tanto lo ZX81 che lo Spectrum Sinclair adottano questo metodo. Per creare un nuovo file dati, si usa una copia "fresca" del programma e la si salva assieme ai nuovi dati. Quando, in seguito, questa versione viene caricata in memoria, i dati vengono automaticamente assegnati alle rispettive variabili. Il pregio di questo sistema è la semplicità: l'unica operazione da fare è una verifica che il programma sia stato "salvato" su nastro e riletto in modo corretto.

Un sistema appena più sofisticato necessita di un BASIC capace di memorizzare su nastro e successivamente rileggere determinate matrici. Sull'Oric, ad esempio, il comando STORE A$, "NOME", registra su nastro il contenuto della matrice A$, mentre RECALL A$, "NOME" lo rilegge. Viene dunque salvata l'intera matrice (A$(1),A$(2),ecc...) senza che i comandi STORE e RECALL facciano specifico riferimento alle sue dimensioni (automaticamente definite quando la matrice viene DIMensionata all'inizio.

Questo metodo, però, può far perdere il filo di tutte le registrazioni utilizzate nel programma per la matrice. Una soluzione può essere quella di memorizzare il numero delle registrazioni nella matrice prima di salvare anch'essa su nastro. La maggior parte dei BASIC prevede l'uso, nelle matrici, anche dell'indice 0, permettendo di specificare, ad esempio, l'elemento A$(0,0), in cui depositare il conteggio delle registrazioni. Questo sarà una variabile numerica (R nel programma a pagina seguente) ma, se usiamo una matrice a stringa, la variabile dovrà essere convertita in una stringa mediante una linea simile a questa: A$(0,0)=STR$(R). Una volta ricaricata la matrice in memoria, R viene azzerato nel seguente modo: R=VAL(A$(0,0)).

Molti micro non possiedono queste sofisticate procedure per la gestione dei file ma esse possono venire simulate (vedere alla pagina seguente). Dopo aver caricato il file in memoria, è facile servirsi delle matrici BASIC per trattarlo come un file ad accesso casuale.

Supponiamo di usare, per la memorizzazione di informazioni, una matrice a stringa bidimensionale, preparata con l'istruzione DIM A$(100,3). Il primo indice della matrice può riferirsi ad una particolare registrazione, il secondo punterà invece a uno dei 4 campi. Ciò permette di immagazzinare l'informazione nella matrice seguendo una procedura simile a quella dei file ad accesso casuale.

Un'altra utile istruzione, presente in molti apparecchi, è il comando APPEND, comodo per "accodare" nuovi dati alla fine di un file sequenziale senza doverlo leggere per intero e crearne quindi una nuova versione. Alcuni BASIC possiedono speciali comandi che, permettendo di saltare un certo numero di campi all'interno di un file sequenziale, forniscono facilitazioni per l'accesso casuale.

In questa serie di lezioni sono stati approfonditi gli aspetti fondamentali di un complesso argomento: la gestione dei file varia molto da macchina a macchina e tutte le idee fin qui esposte dovranno essere adattate al tipo di computer usato. Tuttavia, i principi di base sono gli stessi e molto del materiale trattato in queste pagine si dimostrerà utile per la stesura dei propri programmi.

LA SCRITTURA E LA RILETTURA DI MATRICI

Variabili salvate assieme al programma

Lo Spectrum salva tutte le variabili, ogni volta che il programma viene memorizzato su nastro, ma le matrici possono essere anche "salvate" in modo autonomo. Il seguente programma inserisce i dati nella matrice A$( ) ed il comando SAVE trasferisce la matrice sul nastro assieme al programma. Quando il programma viene nuovamente caricato, l'esecuzione inizia automaticamente alla linea 700 perché l'istruzione SAVE fa specifico riferimento a questa linea. Non si usi assolutamente il comando RUM, perché provocherebbe la perdita di tutte le variabili

Spectrum

```
100 REM**DIMOSTRAZIONE PER LO SPECTRUM**
200 DIM A$(100,20)
300 FOR K=1 TO 100
400 LET A$(K)="N. Record"+STR$(K)
500 NEXT K
600 STOP
700 PRINT "LA MATRICE CONTIENE..."
800 FOR K=1 TO 300
900 PRINT A$(K),
1000 NEXT K
1100 STOP

SAVE "DIMOPROG" LINE 700

LOAD "DIMOPROG"
```

Salvataggio di matrici specifiche

L'Atmos della Oric possiede i comandi STORE e RECALL per salvare e ricaricare dal nastro particolari matrici: questi comandi facilitano l'immagazzinamen-

IMMAGAZZINAMENTO DI RECORD E CAMPI IN UNA MATRICE BASIC

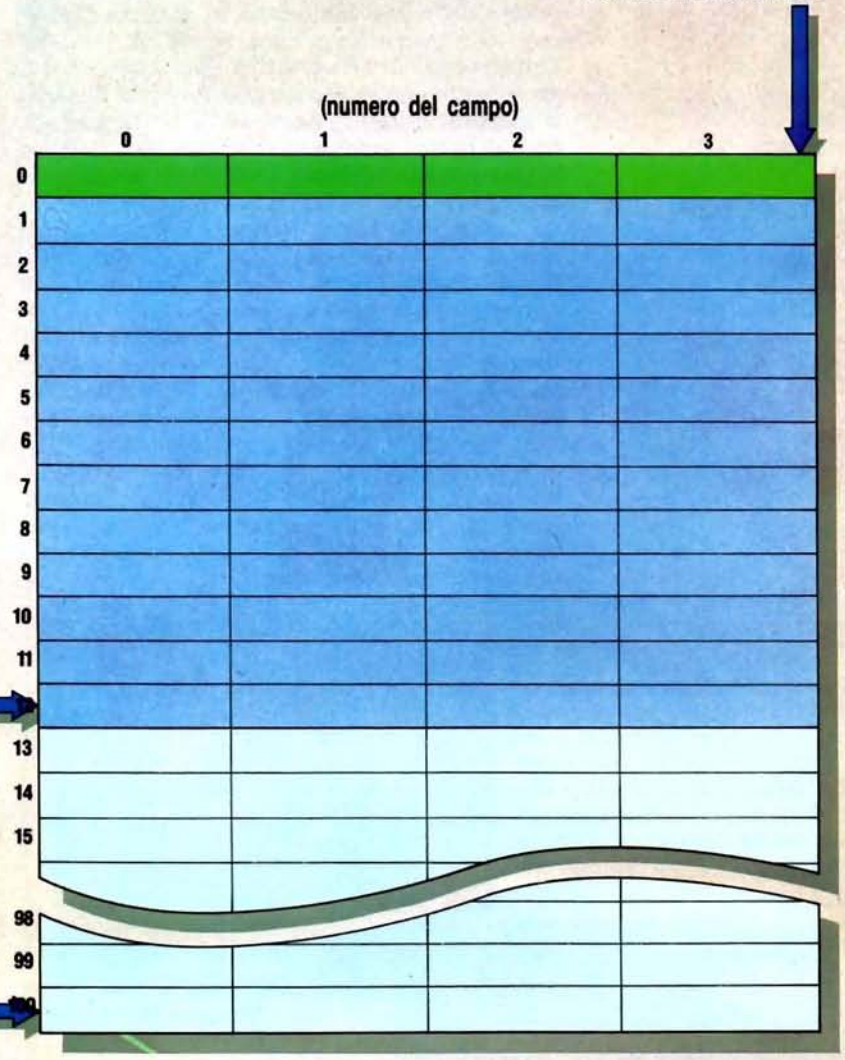
Una matrice a stringa bidimensionale può essere usata per simulare un file ad accesso casuale. Il primo indice in parentesi viene usato per indicare un particolare record, il secondo per identificare i campi all'interno del record

Numero dei campi

È comodo conservare questo numero in una variabile all'inizio del programma. Ciò permette di cambiarlo più facilmente qualora si introducano record di dimensioni maggiori: in questo caso sarà sufficiente modificare una sola istruzione

"Coda" iniziale

A$(0,0) viene usato per conservare il numero totale dei record. Quando salviamo su nastro un nuovo record, il contatore viene incrementato di 1



Numero dei record

Una variabile contiene il numero dei record attualmente inseriti nel file

Numero massimo di record

Una seconda variabile viene adoperata per indicare il numero massimo di record che possono essere immagazzinati nella matrice. Questo numero viene definito nell'istruzione DIM usata per creare la matrice

to di un file contenuto in una matrice

Atmos Oric

```
100 REM**Salva la matrice su nastro**
110 A$(0,0)=STR$(R)
120 PRINT "Inserire il nastro, premere i
    tasti PLAY e RECORD, quindi battere
    RETURN"
130 A$=KEY$: IF A$="" THEN 130
140 STORE A$, "FILEA", S
150 PRINT "TERMINATO"
160 RETURN

200 REM**Carica la matrice dal nastro**
210 PRINT "Inserire il nastro e premere
    il tasto PLAY, quindi battere RETURN"
220 A$=KEY$: IF A$="" THEN 220
230 RECALL A$, "FILEA", S
240 R=VAL(A$(0,0))
250 PRINT "TERMINATO"
260 RETURN
```

Uso dei file sequenziali

Il Micro BBC è uno dei molti home computer che mantengono su cassette veri file sequenziali. Le seguenti subroutine immagazzinano e ricaricano la matrice creando un file sequenziale. Il primo campo contiene il contatore di record

Micro BBC

```
100 REM**Salva la matrice su nastro**
105 PRINT "Inserire il nastro"
110 X=OPENOUT ("FILEA")
120 PRINT#X,R
130 FOR I=1 TO R
140 FOR J=1 TO F
150 PRINT#X, A$(I,J)
160 NEXT J: NEXT I
170 CLOSE#X
180 PRINT "TERMINATO"
190 RETURN

200 REM**Carica la matrice dal nastro**
205 PRINT "Inserire il nastro, premere il
    tasto PLAY, quindi battere RETURN"
210 IF INKEY(0)=-1 THEN 210
215 X=OPENIN ("FILEA")
220 INPUT#X,R
230 FOR I=1 TO R
240 FOR J=1 TO F
250 INPUT#X, A$(I,J)
260 NEXT J: NEXT I
270 CLOSE#X
280 PRINT "TERMINATO"
290 RETURN
```

Inserimento di un record nella matrice

Il seguente frammento permette di inserire un nuovo record nella matrice, alla posizione N. Tutti i record successivi al punto d'inserimento vengono spostati in avanti per creare spazio per il nuovo record immagazzinato in N$, C$, D$ ed E$

```
100 LET R=R+1
110 IF R>M THEN PRINT "MATRICE PIENA": RETURN
120 FOR I=R TO N+1 STEP -1
130 FOR J=0 TO F
140 LET A$(I,J)=A$(I-1,J)
150 NEXT J: NEXT I
170 LET A$(N,0)=A$: LET A$(N,1)=C$
180 LET A$(N,2)=D$: LET A$(N,3)=D$
190 RETURN
```

Cancellazione di un record dalla matrice

Questo frammento di programma permette di cancellare il record di numero N dalla matrice, usando il metodo visto in precedenza. Tutti i record che seguono N vengono spostati indietro di una posizione, andando così a coprire l'informazione precedentemente registrata nella posizione N

```
200 FOR I=N TO R-1
210 FOR J=0 TO F
220 LET A$(I,J)=A$(I+1,J)
230 NEXT J: NEXT I
240 LET R=R-1
250 RETURN
```

KEVIN JONES

Il Boogaboo

Le peripezie della piccola pulce, nel gioco Boogaboo, hanno decretato un enorme successo di questo programma

Boogaboo, il fantasioso titolo del gioco creato dalla Quicksilva, inizia con l'avvertire il giocatore della sua vicinanza al pianeta Cebella-7, che presenta alcuni segni di vita. Se l'atterraggio avviene senza danni, il giocatore si muove nella tenebrosa flora che ricopre il pianeta, prima di perdere l'orientamento e cadere in una profonda voragine che termina in una grande caverna. L'obiettivo del gioco è riuscire a fuggire dalla caverna.

Le cenge e le rocce, vistosamente colorate, ricordano i paesaggi dipinti da Hieronymus Bosch: ci sono funghi e fiori coloratissimi, ma anche giganteschi ragni, aggrappati alle pareti della caverna. Il giocatore può tentare di fuggire risalendo di roccia in roccia, ma deve tenersi lontano da un drago "mangiapulci". Nella versione per il Commodore, ci sono dei fiori venusiani volanti, che rendono la fuga ancor più rischiosa.

In entrambe le versioni, l'immagine si muove come una "finestra" sull'area di gioco e, via via che il giocatore si sposta verso l'alto, appaiono nuovi anfratti della caverna. Le dimensioni totali sono di circa tre schermi di larghezza e cinque d'altezza, quindi il giocatore impiegherà un certo tempo prima di scoprire tutti i pericoli che lo minacciano. Egli può imboccare diversi cammini e le cenge sono disposte in modo da rendere più difficile la risalita.

Lo scenario è assai originale, ma il gioco viene in un certo senso "menomato" dall'impostazione data al programma: il giocatore ha infatti una sola vita e le istruzioni introduttive vengono visualizzate tutte le volte che viene ucciso. Ciò accade spesso perché è molto difficile sfuggire al drago.

Col procedere del gioco, le cenge diventano sempre più difficili da raggiungere ma, verso la sommità della caverna i pericoli scompaiono: anche questo può esser considerato un punto a sfavore: sarebbe più emozionante se nelle ultime fasi comparissero nuovi mostri. Del resto, la memorizzazione dei molti schermi grafici occupa una discreta quantità di memoria, lasciandone ben poca per le complicate regole del gioco e per l'azione; altri programmi simili (ad esempio il *Jet Set Hill*) sono molto meno estesi. La versione per il Commodore 64 ha una grafica più ricca rispetto a quella dello Spectrum, ma occupa ovviamente una maggior quantità di memoria.

La versione per lo Spectrum è controllata da tastiera: i tasti "1" e "0" provocano salti a sinistra e a destra e la lunghezza del salto è direttamente proporzionale alla durata della pressione sul tasto. La versione Commodore prevede invece un joystick, il cui pulsante di fuoco controlla i salti. Questo sistema si dimostra migliore del precedente, purché il joystick sia di buona qualità.

Nonostante le critiche fatte, il livello qualitativo del *Boogaboo* resta più che apprezzabile. Le procedure di controllo sono estremamente semplici e permettono di iniziare immediatamente il gioco senza consultare le istruzioni. Infine la grafica è stupenda e ricca di fantasia.

SALTO IN ALTO

La caratteristica del Boogaboo che più colpisce è l'accuratezza della grafica. Viene mostrata una profonda caverna con pareti ricche di sporgenze che il giocatore (impersonato da una pulce) scala a salti cercando di riconquistare la libertà. Le immagini del Commodore sono leggermente più grandi di quelle dello Spectrum



Bugaboo (la pulce) sullo Spectrum



Bugaboo (la pulce) sullo Spectrum



Booga-boo (la pulce) sul Commodore 64

IAN McKINNELL

Bugaboo ("la pulce"): versioni per lo Spectrum 48K e per il Commodore 64 (+ joystick)

Prodotto da: Quicksilva, (G.B.)

Autori: Indescomp & Microbyte

Joystick: solo la versione Commodore

Formato: Cassette

Moltiplicazioni e rotazioni

Il curioso metodo per eseguire le moltiplicazioni in codice macchina

Sia lo Z80 che il 6502 possiedono l'istruzione SBC (Sottrazione con riporto), ma i loro usi sono piuttosto diversi. Sul 6502, il flag di riporto (Carry) viene usato per gestire la funzione "resto" in una sottrazione, che equivale alla funzione "riporto" in un'addizione. Nell'Assembler dello Z80, l'istruzione SBC svolge esattamente lo stesso compito di ADC: il flag di riporto viene posto a 1 o azzerato per indicare il risultato dell'operazione.
Supponiamo di sommare $E4 a $5F usando il comando ADC (dopo aver azzerato il flag di riporto). Nell'accumulatore avremo come risultato $43. Il flag di riporto viene posto a 1, indicando che il vero risultato è $0143. Il flag di riporto riflette la condizione di overflow (traboccamento), perché l'accumulatore non è in grado di contenere per intero il risultato.
Supponiamo adesso di azzerare il flag di riporto sullo Z80 e di sottrarre $E4 da $5F: il risultato sull'accumulatore sarà $7B e il flag di riporto verrà posto a 1. Se adesso sommiamo $7B a $E4 (dopo aver nuovamente azzerato il flag), troviamo come risultato nell'accumulatore $5F e vediamo che il flag di riporto è stato ancora una volta posto a 1. Quindi:

$5F−$E4 = $7B Flag di riporto a 1
$5F = $E4 + $7B Flag di riporto a 1

Se consideriamo lo stato del flag di riporto come indicatore di un risultato negativo, allora possiamo interpretare $7B come un numero in "complemento a due":

$7B in binario = 01111011
meno uno −1
dà, come complemento a uno: 01111010
negazione
dà come complemento a due 10000101 = $85

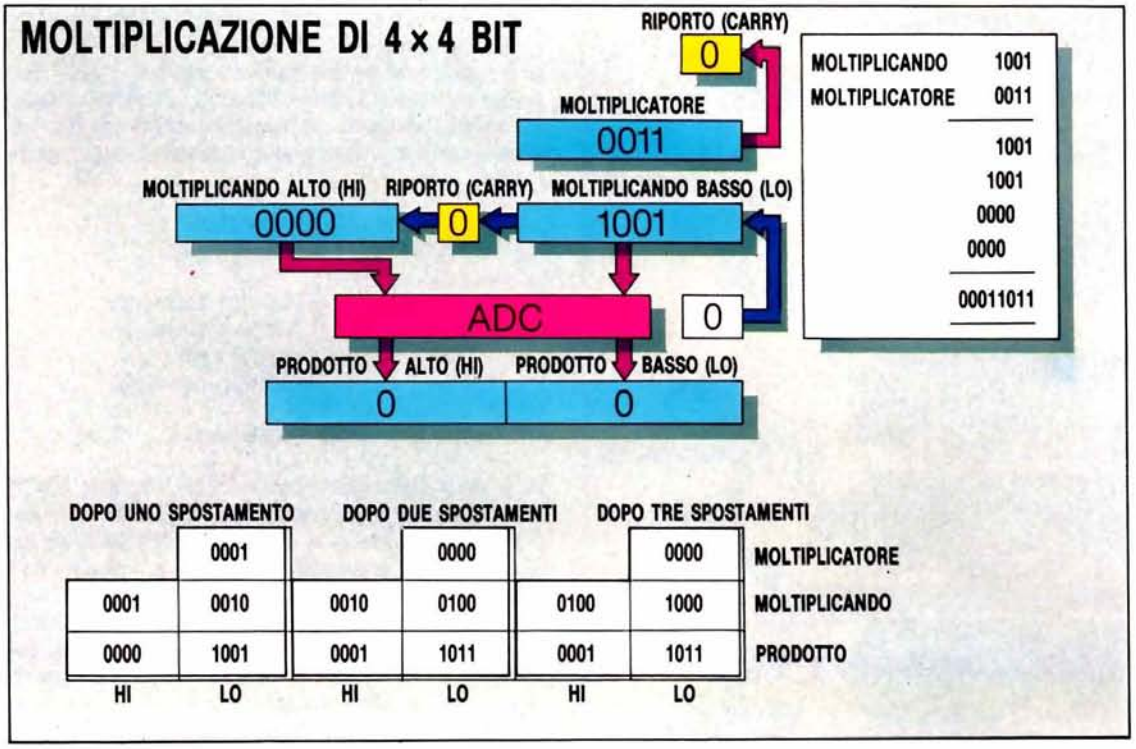
Dovremmo quindi aspettarci di trovare che $5F−$E4 dà come risultato il numero negativo −$85. Verifichiamo questo risultato in decimale:

$5F = 95 in decimale
−$E4 = 228 in decimale
$85 = −133 in decimale

Ovviamente, tutto ciò ha un valore limitato. Supponiamo ora che la sottrazione presa in esame fosse: $375F−$21E4.

HI	LO	
$37	5F	= 14175 in decimale
−$21	E4	= −8676 in decimale
$15	7B	= 5499 in decimale

Sappiamo che se facciamo per prima la sottrazione del byte basso (LO), il risultato è $7B e un riporto. Quest'ultimo viene quindi sommato, dall'istruzione SBC, a $21 dando $15.



Tempo di spostamenti

Questo esempio presenta lo sviluppo di una moltiplicazione a 4 bit: il numero dei bit, comunque, non modifica l'algoritmo. L'esempio mostra come il prodotto venga ottenuto dall'addizione di 0 oppure da versioni "ruotate" del moltiplicando, a seconda che il bit del moltiplicatore sia 0 o 1. I bit del moltiplicatore vengono spostati a destra attraverso il flag di riporto, mentre i bit del moltiplicando vengono spostati a sinistra dal byte basso al byte alto passando attraverso il flag di riporto

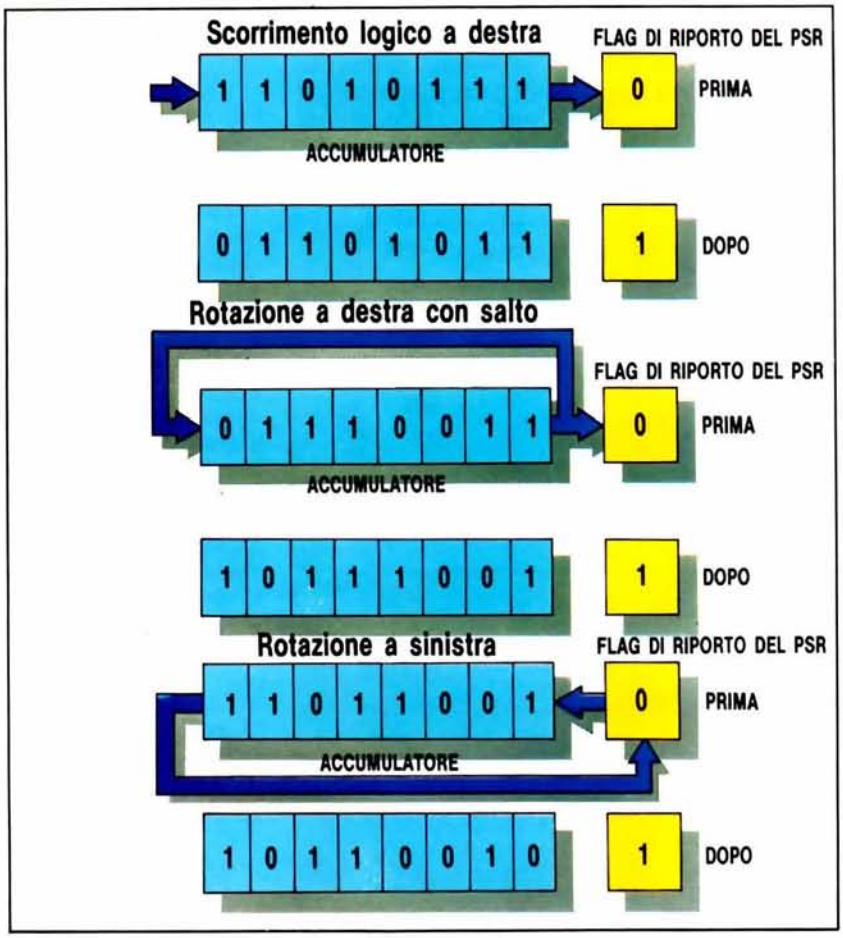
Meccanismo di spostamento

Le istruzioni Shift e Rotate vengono utilizzate soprattutto per esaminare, un bit alla volta, il contenuto di un registro. Ad ogni spostamento, il bit iniziale (o finale) del registro viene trasferito nel flag di riporto del PSR. Lo stato del flag di riporto può essere quindi usato dall'istruzione di salto BRANCH per determinare il percorso del programma (come abbiamo visto nella subroutine di moltiplicazione descritta in questa lezione). Le istruzioni di rotazione possono essere impiegate per conservare i contenuti del registro, mentre le istruzioni di scorrimento logico inseriscono degli 0 nel registro via via che ne vengono espulsi dei bit. Uno scorrimento a sinistra moltiplica quindi i contenuti del registro per due, mentre uno scorrimento a destra li divide per due

Si può vedere che la risposta, $157B, è corretta, verificandola in forma decimale.
Quindi, nell'aritmetica a due byte dello Z80 si segue questa semplice procedura:

1) Azzeramento del flag di riporto.
2) Sottrazione dei byte bassi con riporto.
3) Sottrazione dei byte alti con riporto.

La versione 6502 di questa sequenza differisce nel primo passaggio: il flag di riporto deve essere posto a 1 per permettere un "resto" dei byte bassi, proveniente dai byte alti. Se non c'è resto, la sottrazione procede normalmente ed il flag di riporto resta posto a 1 per la sottrazione dei byte alti, che dovrebbe anch'essa procedere normalmente. Se, tuttavia, si verifica una condizione di "underflow" (traboccamento verso la parte bassa) nella sottrazione del byte basso, il flag di riporto si comporta come "nono bit" dell'accumulatore. Ciò permette di ottenere ugualmente un risultato corretto e il flag di riporto viene azzerato. Quando sottraiamo i byte alti con un flag di riporto azzerato, si ha lo stesso effetto che si ottiene sullo Z80 nella sottrazione del byte alto con flag di riporto posto a 1: il numero da sottrarre viene decrementato prima che venga operata la sottrazione. Entrambi i metodi esposti hanno un loro equivalente nei tradizionali sistemi aritmetici di "riporto" (nel secondo caso) e di "restituzione" (nel primo caso). Esaminiamo ora in modo più approfondito la versione 6502.



Se azzeriamo il flag di riporto e sottraiamo $E4 da $5F, il risultato nell'accumulatore sarà $7A ed il flag rimarrà azzerato. Abbiamo visto, nell'esempio Z80, che un risultato "vero" è $7B con il flag di riporto che indica un numero negativo. $7B rappresenta il complemento a due della risposta "reale" (–$85). Possiamo verificare che $7A è il complemento a uno di questo numero e che lo stato del flag di riporto, quindi, rappresenta una specie di interruttore riferito allo stato dell'accumulatore: vale a dire, viene posto a 1 per il complemento a due e a 0 per il complemento a uno.
Eseguendo la sottrazione sul 6502 con il flag posto a 1, l'accumulatore conterrà $7B ed il flag di riporto risulterà azzerato. Se questa è una sottrazione a due byte, l'azzeramento del flag di riporto assicurerà che il risultato della sottrazione del byte alto venga decrementato, facendo quindi attenzione al "resto" dei byte alti.

Moltiplicazione

Consideriamo la seguente moltiplicazione decimale:

```
   174   Moltiplicando
  ×209   Moltiplicatore
---------
  1566   1° Prodotto parziale
   000   2° Prodotto parziale
  +384   3° Prodotto parziale
---------
 36366   Prodotto finale
```

Non c'è bisogno di conoscere la notazione posizionale per usare questo metodo: si devono solo seguire le semplici procedure per fare una moltiplicazione a singola cifra. Il nucleo del procedimento è scrivere ciascun prodotto parziale una posizione più a sinistra rispetto al precedente.
Nel nostro caso, c'è solo un prodotto effettivo nella moltiplicazione binaria, cioè 1×1. Tutti gli altri prodotti a singola cifra danno 0. Consideriamo la seguente moltiplicazione binaria:

```
  1101  =  13 in decimale
  1001  =   9 in decimale
-------
  1101    1° Prodotto parziale
 0000     2° Prodotto parziale
 0000     3° Prodotto parziale
1101      4° Prodotto parziale
-------
1110101 = 117 in decimale
```

In questo esempio appare chiaramente sia lo spostamento dei prodotti parziali, che la semplicità complessiva della moltiplicazione in binario. Un prodotto parziale è uguale a 0 oppure al moltiplicando spostato verso sinistra, a seconda che il moltiplicatore corrispondente sia 0 o 1. Questa procedura ha molti punti in comune con alcune strutture di controllo del linguaggio Assembler. Per eseguire una moltiplicazione binaria, si esami-

nano in successione i bit del moltiplicatore ed aggiungeremo al totale il valore 0 (se il bit è 0) oppure il moltiplicando spostato a sinistra (se il bit è 1). Come si deve procedere nell'esame dei singoli bit del moltiplicatore e nello spostamento del moltiplicando?

Tanto sullo Z80 quanto sul 6502, è possibile esaminare il contenuto di un particolare bit in un byte usando l'istruzione BIT. Sullo Z80, questo comando richiede come operandi un indirizzo ed un numero di bit: se il bit esaminato è 0, il flag di zero viene posto a 1, altrimenti esso viene azzerato. Sul 6502, l'operando è un indirizzo; il contenuto di questo indirizzo viene posto in AND rispetto all'accumulatore ed il flag zero viene posto a 1 se il risultato è falso oppure azzerato se è vero.

Queste istruzioni consentono ingegnose procedure di programmazione, ma nessuna di esse è particolarmente utile nel nostro caso. Sarebbe assai più utile poter usare il bit in questione come un flag zero o di riporto, così da indirizzare automaticamente il flusso del programma a seconda del valore contenuto in ciascun bit preso in esame. Ovviamente in entrambi i processori è possibile ottenere questo risultato usando le istruzioni di "scorrimento" (shift), che risolvono brillantemente il problema dello spostamento a sinistra del moltiplicando.

Entrambi i processori possiedono numerose istruzioni di scorrimento e di "rotazione", ma quelle dello Z80 sono molto più complesse. Generalmente producono uno spostamento a destra o a sinistra dei bit di un registro, ma differiscono nella manipolazione dei bit estremi del registro stesso: quando un bit viene spostato fuori dal registro ad un'estremità, un altro bit viene inserito all'estremità opposta. Se il bit 7 viene spostato fuori dal registro ed immediatamente reinserito nel bit 0, l'operazione viene detta "rotazione a sinistra"; se, al contrario, è il bit 0 che viene trasferito in posizione 7, allora parleremo di "rotazione a destra". Questa operazione cambia l'ordine dei contenuti del registro, ma non ne inserisce di nuovi e, dopo 8 rotazioni, il registro torna al suo valore originario.

Se non si usa la rotazione, è necessario assegnare una destinazione al bit "espulso" dal registro e trovare un'origine per il bit che viene inserito. Entrambe vengono spesso fornite dai diversi flag condizionali del registro di stato del processore (PSR), in particolare dal flag di riporto. Quando scriviamo una subroutine per moltiplicare due numeri a singolo byte, è necessario spostare il moltiplicando a sinistra ed il moltiplicatore a destra. I bit espulsi dal moltiplicando devono essere inseriti nel byte alto del moltiplicatore mentre gli zero vengono inseriti nei bit non occupati.

I bit del moltiplicatore devono passare attraverso un flag del PSR per una verifica, mentre la loro destinazione e lo stato dei bit inseriti nel moltiplicatore non è importante, a meno che non si abbia bisogno di conservare il contenuto del moltiplicatore.

Tutto ciò che ci interessa sapere del moltiplicatore durante l'operazione è se il bit "espulso" è 1 o 0.

Dato che il moltiplicatore viene immagazzinato all'indirizzo MPR, il moltiplicando a MPDLO ed il prodotto agli indirizzi PRODLO e PRODHI, possiamo scrivere queste subroutine nel seguente modo:

MOLTIPLICAZIONE A OTTO BIT

6502			Z80		
	ORG	$C100		ORG	$D000
START	LDA	#$00	START	LD	BC,(MPR)
	STA	PRODLO		LD	B,$08
	STA	PRODHI		LD	DE,(MPDLO)
		MPDHI		LD	D,$00
	LDX	#8		LD	HL,$00
	CLC		LOOP0	SRL	C
LOOP0	ROR	MPR		JR	NC,CONT0
	BBC	CONT0		CALL	ADDIT
	JSR	ADDIT	CONT0	SLA	E
CONT0	ASL	MPDLO	ENDLP0	DJNZ	LOOP0
	ROL	MPDHI		LD	PRODLO
	DEX			RTS	
ENDLP0	BNE	LOOP0	MPR	DB	$E2
	RTS		MPDLO	DB	$7A
MPR	DB	$E2	MPDHI	DB	$00
MPDLO	DB	$7A	PRODLO	DW	$0000
MPDHI	DB	$00	ADDIT	ADD	HL,DE
PRODLO	DB	$00		RET	
PRODHI	DB	$00			
ADDIT	CLC				
	LDA	PRODLO			
	ADC	MPDLO			
	STA	PRODLO			
	LDA	PRODHI			
	ADC	MPDHI			
	STA	PRODHI			
	RTS				

Come si può vedere in questo esempio, è molto più facile programmare lo Z80 usando i suoi registri a 16 bit e le istruzioni associate (in particolare, si confronti la subroutine SOMMA nei due programmi precedenti). La versione 6502 usa il comando ROR per ruotare a destra il moltiplicatore passando attraverso il riporto, e le istruzioni ASL e ROL per spostare, attraverso il riporto, il moltiplicando a sinistra, da MPDLO in MPDHI. Il ciclo iterativo viene controllato dal registro X usato come contatore.

La versione Z80 utilizza SRL per spostare il moltiplicatore a destra attraverso il riporto e SLA ed RL per spostare il moltiplicando a sinistra in DE sempre attraverso il riporto. Il ciclo viene qui controllato dal registro B. Notare che l'istruzione ADD prevede operazioni aritmetiche sul registro a 16 bit e, diversamente dal comando ADC, non influenza il flag di riporto.

Nella prossima lezione, verranno affrontate l'operazione di divisione e varie procedure di controllo dello schermo. Quest'ultima parte completa gli elementi didattici del corso ed è seguita da esercizi ed esempi per il 6502 e lo Z80.

Esercizio 15

1) Scrivere una subroutine di moltiplicazione usando un moltiplicando a 16 bit ed un moltiplicatore a 8 bit di propria scelta.
2) Una moltiplicazione è in realtà un'addizione ripetuta più volte: scrivere una subroutine di moltiplicazione di 8 x 8 bit che non implichi l'uso delle istruzioni di scorrimento e di rotazione.

La Olivetti

La ben nota casa produttrice di macchine da scrivere italiana, che ha imposto la propria presenza anche nel campo dell'informatica



Progetto e design

L'elegante personal computer M20 è una macchina a 16 bit lanciata dalla Olivetti nel 1981. Inizialmente, l'apparecchio si basava su un microprocessore Zilog Z8000 ed adottava il sistema operativo PCOS Olivetti. Recentemente ne è stata lanciata una versione, compatibile con il PC della IBM, in cui viene adoperato il chip 8086

Eleganza architettonica

La Olivetti si è sempre distinta per la linea moderna dei propri prodotti. Ma questa particolare cura estetica è stata applicata anche all'architettura dei propri uffici. La costruzione in figura, eretta nel 1959, presenta alcuni tratti strutturali che furono in seguito adottati in altri edifici

Camillo Olivetti fondò la propria industria nel 1908. Con uno staff di 20 persone egli aprì un negozio ad Ivrea, in Piemonte, e dette avvio alla produzione della sua prima macchina da scrivere, la M1. In quegli anni, l'economia del nostro paese era ancora prevalentemente agricola, in quanto l'industria pesante non aveva ancora raggiunto uno sviluppo pari a quello di altri paesi come la Germania, la Gran Bretagna e gli Stati Uniti. Nonostante ciò, le capacità produttive della Olivetti si mantennero in costante ascesa, da quattro macchine al giorno nel 1914 a 50 macchine nel 1929.

Negli anni '30, Adriano Olivetti, figlio di Camillo, avviò la riorganizzazione produttiva dell'industria, assumendo studenti che uscivano da una scuola fondata dalla stessa Olivetti nel 1924 e mettendo a disposizione dei propri dipendenti abitazioni e servizi sociali. Adottò quindi, per primo, una politica industriale seguita più tardi con grande successo da molte altre nazioni. La Olivetti, al contrario di molte altre industrie, continuò a svilupparsi anche nel periodo bellico e, nel 1933, aveva già venduto 15 milioni di unità per ufficio. Nel 1937 furono lanciate sul mercato le prime telescriventi Olivetti, seguite, nel 1940, dal primo calcolatore.

La Seconda Guerra Mondiale provocò una momentanea flessione nell'espansione della Olivetti ma, nel periodo post-bellico, la società si lanciò alla conquista di nuovi mercati. Il suo successo si fondava su un design elegante e sulla qualità dei prodotti: anche alcuni dirigenti della IBM furono costretti ad ammettere che "l'intera gamma dei prodotti Olivetti possedeva uno standard qualitativo più che apprezzabile".

Negli anni '50 e '60 la Olivetti orientò la propria ricerca nel settore dei computer per ufficio. Questo processo dette i suoi primi risultati nel 1955 con lo sviluppo di una macchina contabile e della prima unità centrale, l'Elea, progettata pochi anni dopo.

La Olivetti continuò a diversificare la propria gamma di prodotti, spostando i propri interessi dagli apparecchi meccanici per ufficio ai prodotti elettronici, da essa considerati il mercato del futuro nel campo dell'automazione per ufficio. Introdusse quindi una nuova serie di microcomputer gestionali, assieme a terminali bancari e ad apparecchiature per telecomunicazioni.

Oggi la Olivetti produce una vasta gamma di apparecchi elettronici commerciali e investe cospicui capitali nello sviluppo di software per i propri apparecchi. Nel 1982 la Olivetti era la seconda produttrice europea di computer (superata solo dalla IBM), grazie al buon successo del portatile M10 e del personal computer M20.

Il portatile M10 pesa circa 1 chilo e 700 grammi ed è dotato di uno schermo testuale di 8 × 40. La macchina viene alimentata a pile e possiede 8 Kbyte di RAM interna espandibile fino a 32 Kbyte. L'M20 è una macchina per ufficio a 16 bit basata su un processore Z8001, un chip che non ha avuto molta fortuna presso altri fabbricanti di apparecchi a 16 bit.

Ultimamente, è stato affiancato anche da un processore 8086, che permette di utilizzare il CP/M-86 e l'MS-DOS.

L'Olivetti ha anche prodotto una nuova macchina, compatibile con l'IBM-PC, ma dal costo inferiore. L'apparecchio, chiamato M24, adotta il processore 8086-2 e presenta come opzione la scheda Z8001 che lo rende compatibile con l'M20. Questa compatibilità ha comportato il parziale abbandono del proprio sistema operativo PCOS.

Recentemente, la Olivetti ha sottoscritto un accordo con la AT&T (l'industria di telecomunicazioni più grande del mondo) per collaborare ad uno sviluppo del sistema operativo Unix. Non c'è quindi dubbio che in futuro la vasta rete commerciale della Olivetti, le sue capacità di ricerca ed il suo settore di sviluppo manterranno ancora alto il nome della società per quanto riguarda lo stile e la qualità dei prodotti.





STRATEGIE NEL SOFTWARE

Nel tentativo di dare una valutazione ad alcuni dei più diffusi programmi di scacchi, è stato organizzato un piccolo torneo tra i seguenti prodotti: Sargon III della Hayden Software, che gira sull'Apple IIe (su dischetto, programmato da Dan e Kathe Spracklen); Cyrus IS Chess della Sinclair Software, per lo Spectrum a 48K (su cassetta, programmato dalla Intelligence Software); Colossus 2.0 della CDS Micro System, per il Commodore 64 (su dischetto, programmato da Martin Bryant); Gran Master 64 della Audiogenic, anch'esso per il Commodore 64 (su cassetta, programmato dalla Kingsoft). Questi programmi si sono più volte "incontrati" in tornei internazionali di scacchi per computer, ma qui la valutazione è molto meno formale, riguardando essenzialmente la "giocabilità" e l'abilità. Il minitorneo non era orientato a determinare un vincitore finale, ma prevedeva un minimo di due partite per ciascun programma: la prima al livello più semplice e la seconda ad un livello medio.

Uno dei principali problemi negli scontri tra due computer è la determinazione di un livello di gioco che ponga i due apparecchi sullo stesso piano. I livelli sono di solito contraddistinti dal tempo concesso per l'effettuazione di una mossa ma, a seconda dei casi, la selezione di uno stesso livello non garantisce un'effettiva parità qualitativa. Ad esempio, il Sargon III "ruba" il tempo ai suoi avversari e, al contrario di tutti gli altri programmi, mantiene operativo il suo generatore di mosse anche quando tocca all'avversario muovere. Nonostante ciò, si è fatto ogni sforzo per mettere tutti i programmi nelle stesse condizioni di gioco.

QUALITÀ DEL GIOCO

In generale tutti i programmi hanno giocato delle buone partite (impiegando circa 10 secondi per ogni mossa) al livello più basso e tutti hanno eseguito qualche mossa strana, apparentemente inutile, nel finale di partita. Ciò dipende probabilmente da una posizione d'attesa nella quale il computer "aspetta" che accada qualcosa di interessante. Ad un livello più alto, da competizione (circa 10 minuti per mossa), tutti e 4 i programmi hanno mostrato un gioco tattico intelligente e, a volte, brillante. La seguente tabella mostra i risultati del torneo.

sviluppi tecnologici, la qualità di questi programmi dipende soprattutto dal software.
Poiché i computer sono capaci di grande velocità di calcolo, gli scacchi per computer si basano proprio su procedure di calcolo numerico, usate per valutare i due elementi chiave del gioco: il "materiale" e la "mobilità". Per materiale si intende il numero e l'efficacia dei pezzi presenti sulla scacchiera. Il programma assegna ad ogni pezzo un valore numerico. Il Re ne avrà uno altissimo (ad esempio, 10000) oppure infinito perché il "matto" al Re determina la fine del gioco; la Regina ha di solito valore 9, le Torri 5, Alfieri e Cavalli 3 e Pedoni 1. Quando il programma "riflette" se sia il caso di sacrificare un pezzo per catturarne uno dell'avversario, innanzi tutto confronta il valore dei pezzi. Una grande importanza viene dunque attribuita ai valori relativi e raramente il programma accetterà uno "scambio" svantaggioso, a meno che non ne consegua un evidente vantaggio tattico.
La mobilità, di enorme importanza negli scacchi, determina il valore relativo dei singoli pezzi: hanno maggior valore i pezzi capaci di tenere sotto controllo più caselle della scacchiera contemporaneamente. I programmi per scacchi quindi devono poter valutare tanto la mobilità quanto il materiale. Inoltre, devono essere capaci di prevedere lo sviluppo del gioco, decidendo la sequenza di mosse più favorevole in base alla posizione di ciascun pezzo. In questo campo i programmi per scacchi possono eccellere, sfruttando la velocità del computer per esaminare in breve tempo il maggior numero di mosse possibili. Moltissimi programmi di scacchi adottano una "tecnica d'urto", esaminando quante più mosse possibili nel tempo consentito. Il tempo utile per ciascuna mossa dipende dal "livello" di gioco selezionato all'inizio della partita e può variare da pochi secondi ad alcune ore: più tempo si dà al computer per analizzare la situazione del gioco, migliore sarà la strategia d'attacco adottata in base alla corrente posizione dei pezzi.
Dopo ogni mossa il computer verifica se il Re è "sotto scacco", esamina quali pezzi verranno probabilmente conquistati o persi, se è possibile occupare caselle chiave e molti altri problemi simili. Più situazioni il programma esamina, migliore sarà il risultato. L'obiettivo, ovviamente, è scoprire se sia possibile costringere il Re avversario in posizione di "scacco matto".
Nelle partite tra uomini e computer, questi ultimi sono avvantaggiati dalla velocità e dalla possibilità di sofisticate analisi. Tuttavia, un discreto giocatore dovrebbe riuscire a battere un buon programma di scacchi, poiché, in definitiva, l'uomo può ideare e sviluppare nuove aperture e varianti di gioco. I computer possiedono una splendida tattica

Scaccomatto al computer

Qualche accenno sulle tecniche impiegate dai computer per giocare a scacchi



Campione di scacchi

David Levy è un Maestro Internazionale di scacchi, ritiratosi dalle competizioni nel 1978. Nel 1968 scommise una forte somma sulla propria abilità, dichiarando che nessun programma di scacchi sarebbe riuscito a batterlo nei successivi 10 anni: e così è stato, anche per un periodo più lungo! Oggi, Levy dirige la Intelligence Software, una società che produce programmi di scacchi per microcomputer e fornisce software dedicato a molti computer scacchistici. Levy ritiene che i micro stiano ormai raggiungendo le prestazioni di gioco dei grandi elaboratori e pensa che entro 5-8 anni un micro, possibilmente usando processori paralleli per velocizzare le funzioni di ricerca, sarà in grado di battere Bell (un computer unicamente dedicato al gioco degli scacchi) ed il grosso elaboratore Cray Blitz (vincitore nel 1983 del Campionato Mondiale di Scacchi per Computer)

Pochi giochi hanno riscosso, nei secoli, un successo pari a quello degli scacchi: è stato giocato da milioni di persone in tutto il mondo per migliaia di anni e le attuali regole hanno subito solo lievi variazioni dal XVII secolo. Alcuni appassionati hanno dedicato tutta una vita allo studio ed alla padronanza di questo gioco di strategia, basato sul rigore intellettuale e sull'elasticità mentale. Del gioco sono state create molte variazioni, nel tentativo di accrescerne il già alto livello di complessità. Gli scacchi tridimensionali, ad esempio, implicano l'uso di diverse scacchiere disposte l'una sull'altra nello spazio e richiedono moltissima concentrazione. Esiste poi una versione per tre giocatori, in cui la scacchiera ha la forma di una Y: nell'area centrale, d'intersezione, vengono adottate particolari regole per il movimento dei pezzi. In questa versione due giocatori si alleano contro il terzo, quindi si disputano la vittoria tra loro. Tuttavia la versione di maggior successo resta quella tradizionale nella quale i due giocatori si affrontano su una scacchiera di 64 caselle.

Negli scacchi le variazioni possibili sono pressoché infinite. Nel 1949, il matematico Claude Shannon scrisse un articolo intitolato: *Programmazione di un Computer per il gioco degli Scacchi* in cui affermava che ci sono 10^{120} possibili variazioni in una partita di 40 mosse: una persona che giocasse a scacchi 24 ore al giorno per 7 giorni alla settimana, concludendo ogni partita in un'ora (un tempo piuttosto ridotto per 40 mosse), impiegherebbe 10^{17} anni per giocare tutte le partite possibili! Oggi, tuttavia, gli approfonditi studi di tattica scacchistica hanno ridotto le possibili variazioni di una certa misura, che dipende dalle capacità del giocatore.

Per queste ragioni, la programmazione dei computer per giocare a scacchi ha richiesto molto tempo e un duro lavoro. Inoltre, per molti anni, erano disponibili soltanto programmi adatti ad apparecchi di grosse dimensioni e solo ultimamente sono state realizzate delle versioni per microcomputer. La realizzazione di programmi di scacchi ad alto livello, per microcomputer, è strettamente connessa allo sviluppo dell'hardware: occorre molta memoria ed una discreta velocità di elaborazione. Oggi, grazie ai più recenti

Cyrus IS: ha pareggiato con il Colossus ed ha battuto Gran Master al livello più basso, mentre ha pareggiato col Sargon III al livello da competizione.

Colossus: ha pareggiato con il Cyrus IS Chess al livello più basso, ha vinto con il Gran Master ed ha pareggiato col Sargon III al livello di competizione.

Sargon III: ha perso col Gran Master al livello più basso ed ha pareggiato con il Cyrus e col Colossus al livello di competizione.

Gran Master: ha battuto il Sargon III ed ha perso col Cyrus al livello più basso; ha nuovamente perso col Colossus al livello di competizione.

CARATTERISTICHE

Tutti i buoni programmi di scacchi prevedono la possibilità di "arroccare", portare un pedone a Regina, catturare "en passant" e conoscono la "patta" e lo "stallo". Alcuni possono avere caratteristiche addizionali assai interessanti. Il Sargon III è il programma con più "extra": in un secondo dischetto ci sono 107 partite di scacchi "classiche" e 45 problemi di scacchi. Ha una documentazione completa, raccolta in un manuale di 75 pagine. Naturalmente, il Sargon III, utilizzabile con l'Apple IIe, è circa tre volte più costoso degli altri programmi. A parità di prezzo, il Cyrus IS Chess ed il Colossus offrono alcune caratteristiche interessanti, come si può vedere nella seguente tabella.

Caratteristiche	Gran Master 64	Colossus	Cyrus IS	Sargon III
Gioco invisibile	NO	SI	NO	NO
Elenco mosse possibili	NO	SI	NO	NO
Gioca la "mossa migliore"	NO	NO	SI	NO
Ripetizione di una partita	NO	SI	SI	SI
Elenco delle mosse giocate	NO	SI	NO	SI
Stampa dell'elenco	NO	NO	SI	SI
Ripetizione delle mosse	SI	SI	SI	SI
Regole del gioco	SI	NO	NO	SI
Promozione (non solo Regina)	NO	SI	SI	SI
Partita tra uomini	NO	SI	SI	SI
Scambio delle parti	SI	SI	SI	SI
Analisi dei problemi	NO	SI	SI	SI
Mostra la ricerca	UNA MOSSA	SI	NO	SI
Inversione scacchiera	SI	SI	SI	SI
Offerta di patta	NO	NO	NO	SI
Stampa la scacchiera	NO	NO	SI	SI
Salva la partita	NO	SI	SI	SI
Disattivazione archivio	NO	NO	NO	SI
Autogioco	NO	SI	SI	SI
Cronometro in tempo reale	SI	SI	NO	SI

CONCLUSIONI

Per quanto riguarda la giocabilità, il Cyrus IS Chess ed il Colossus sono i più facili da usare, perché le mosse vengono inserite usando il cursore, mentre nel Sargon III e nel Gran Master si devono scrivere le mosse in notazione algebrica: ad esempio, E2-E4 per pedone in Re 4. Il Colossus ed il Sargon III hanno la migliore grafica video. Il Gran Master è un eccellente programma di scacchi ad un prezzo assai basso.

di gioco ma, come accade anche tra avversari umani, un buon giocatore posizionale dovrebbe poter sconfiggere un giocatore tattico di pari capacità. I creatori dei programmi di scacchi puntano sulla tattica perché, per un computer, il gioco tattico implica semplici elaborazioni numeriche. Se l'avversario umano esegue una variazione inattesa, il computer può anche sbagliare la mossa in risposta. È per questa impostazione che i computer hanno difficoltà a gestire le situazioni "d'attesa", frequenti, ad esempio, nel finale della partita, in cui nessuna mossa offre particolari vantaggi tattici.

Spesso, il programma si limita a muovere stancamente i pezzi senza prevedere chiusure vincenti.

Una recente tecnica di programmazione prevede la "ricerca selettiva", in cui il computer simula il comportamento di un giocatore umano, analizzando molto più attentamente un ristretto numero di possibili mosse. Hegener e Glaser, in Germania, hanno introdotto la ricerca selettiva nel loro programma *Mephisto III*, che esamina ogni possibile movimento per due mosse consecutive, quindi restringe la ricerca, vagliando con più attenzione un ristretto numero di mosse. In *Mephisto III* esiste anche una distinzione tra le posizioni d'attesa e tattiche. Simili tecniche permetteranno ben presto ai computer di divenire agguerriti avversari dei giocatori umani.

CUORE CONTRO CERVELLO

La possibilità di prevedere fino a 9 mosse garantisce ai programmi di scacchi una supremazia tattica. La dote principale dei Maestri di scacchi è la capacità di individuare un ristretto numero di mosse chiave e su di esse concentrarsi, arrivando a prevedere fino a 30 mosse. Presentiamo il finale di una partita giocata nel 1922 da Moritz (bianco) contro Emmerich (nero). Il nero può dare scacco matto sacrificando la Regina ed eseguendo tre eleganti mosse di Cavallo: la maggior parte dei giocatori opterebbe per questo finale. Il grande Moritz dimenticò questa chiusura e non riuscì a vincere. Tutti i pacchetti qui esaminati hanno trovato il "matto", ma nessuno ha adottato le mosse di Cavallo, anche se qualche programma le ha prese in considerazione. L'incapacità del computer di interpretare questo finale come il migliore sembra offrire agli uomini ancora una possibilità di difesa contro la crescente abilità del computer per scacchi.



Questa è la sequenza delle mosse di Cavallo

1		H5—H2 scacco
2	G1—H2	E5—G4 scacco
3	H2—G1	F4—H3 scacco
4	G1—F1	G4—H2 matto



ROY INGRAM

La grafica su stampante

La definizione dei caratteri e delle immagini



Stampare è un'arte
Queste illustrazioni mostrano i risultati grafici ottenibili con alcune stampanti a matrice d'aghi. Ciascun ago della testina viene controllato individualmente e ciò permette di ottenere figure complesse e di grande effetto. Queste immagini sono state create con una stampante Paintbox della Print'n' Plotter Products

Il gran numero di stampanti disponibili sul mercato può veramente sconcertare: quasi ogni fabbrica di home computer produce almeno una stampante per i propri apparecchi. Il potenziale acquirente deve innanzi tutto decidere quale tipo sia il più adatto: se a matrice d'aghi o a "margherita", per quanto il mercato offra anche altri tipi come le stampanti termiche o a getto d'inchiostro. I modelli a margherita danno risultati di più alta qualità (sono anche i più costosi) e rappresentano quindi il meglio per le operazioni di word processing. Le stampanti a matrice d'aghi sono di solito più economiche, più lente nella stampa, ma si dimostrano lo strumento ideale per stampare listati o generici lavori di programmazione. Prendiamo ora in esame le stampanti a matrice d'aghi.

Il prezzo di questi apparecchi varia da meno di mezzo milione a oltre 2 milioni di lire, per i modelli più sofisticati. Le due principali caratteristiche da valutare sono la velocità di stampa e la qualità del testo prodotto. I modelli più costosi sono capaci di produrre una spaziatura proporzionale (ossia i caratteri stretti, come la "i", occupano meno spazio di quelli più larghi, come la "m") e caratteri tipografici di vario tipo.

La velocità di stampa è importante, perché l'uso della stampante vincola l'attività del computer: infatti il testo rimane in memoria, finché la stampante non lo avrà riprodotto sulla carta ed in questo lasso di tempo il computer non potrà eseguire altre operazioni. La velocità di stampa viene misurata in "caratteri per secondo" (cps), quindi un costoso apparecchio, che stampi 200 caratteri al secondo, può impiegare un solo minuto a stampare un lungo listato, mentre ad un modello più economico, con velocità di stampa di 30 cps, occorreranno più di 6 minuti. Questi tempi morti possono essere eliminati adottando un'apposito "buffer" per stampante, ossia una speciale scheda di memoria RAM, inserita tra la stampante ed il computer, nella quale viene "scaricata" una notevole quantità di testo, lasciando libero il computer per altre operazioni. Le stampanti più costose possiedono, incorporati, buffer di grande capacità.

Raramente la velocità di stampa dichiarata dalle case produttrici corrisponde alla realtà

dell'uso pratico, poiché le prove di velocità vengono eseguite in condizioni ideali (come accade per le prove di consumo delle automobili), magari sulla stampa di un'unica linea di testo, composta da uno stesso carattere. Con un testo normale, composto da vari caratteri, spaziature, avanzamenti di interlinea e ritorni di carrello, la velocità di stampa può risultare notevolmente minore: una stampante con velocità dichiarata di 160 cps, probabilmente raggiungerà appena i 100 cps quando stampa un listato di programma.

La qualità dei caratteri prodotti varia notevolmente da stampante a stampante e dipende soprattutto dal numero di aghi nella testina di stampa. Nei modelli più economici la testina ha solo 7 aghi, mentre in quelli più costosi gli aghi possono essere più di 16. La stampante della Commodore, che ha solo 7 aghi, produce i caratteri all'interno di una matrice a punti di 7 × 6 mentre la Canon PW1080, apparecchio ben più caro, adotta una matrice di 16 × 23. In questo caso, i caratteri, formati da punti molto ravvicinati tra loro, appaiono molto meglio definiti e più compatti. La qualità della stampa riveste grande importanza nelle procedure di word processing, ma è irrilevante nella stampa di un semplice listato di programma.

Una stampante a matrice d'aghi è in realtà un microcomputer dedicato: possiede un proprio microprocessore e vari chip di memoria ROM e RAM. Come tale, può essere programmata per svolgere mansioni anche diverse dalla stampa di un testo. In questi casi, occorre che il computer invii alla stampante speciali codici di controllo, oppure intervenga sulla disposizione di piccoli interruttori, detti DIP (da "Dual Inline Package", montaggio a duplice fila), posti all'interno dell'apparecchio. Ad esempio, la serie standard di caratteri ASCII, registrata nella memoria della stampante, può essere modificata per ottenere caratteri alfabetici differenti.

Altri effetti speciali riguardano la stampa di caratteri a doppia larghezza o evidenziati (di colore più scuro) e la modifica delle spaziature d'interlinea. La Epson FX80 è una delle più versatili stampanti a matrice d'aghi e possiede più di 70 di queste opzioni. Può stampare in caratteri "italici", sottolineare automaticamente il testo ed usare la spaziatura proporzionale.

La gamma di stampanti Epson si è ormai affermata come uno "standard industriale": ciò significa che gran parte del software per stampanti oggi disponibile (pacchetti di word processing, programmi di fatturazione, ecc...) presuppone il possesso di una stampante Epson, perché le altre marche di stampanti non possiedono i requisiti necessari.

La scelta della stampante può essere influenzata da altre importanti considerazioni: ad esempio, l'affidabilità e la rumorosità dell'apparecchio. Una stampante poco costosa può andar bene per l'occasionale stampa di listati, ma è del tutto inadeguata per un pesante e quotidiano lavoro d'ufficio. Analogamente, l'eccessiva rumorosità può risultare fastidiosa agli altri occupanti dell'ufficio o dell'abitazione. Tutte le stampanti a matrice d'aghi adottano un meccanismo di trascinamento che lavora solo con pile di carta continua perforata su entrambi i lati (i cosiddetti "moduli continui"). Per l'uso di fogli singoli, la stampante deve possedere un meccanismo di trascinamento ad attrito, simile a quello delle normali macchine da scrivere.

Un'ultima considerazione riguarda il tipo di microcomputer che si intende usare: quasi tutte le stampanti sono corredate di una porta parallela Centronics o di un'interfaccia seriale RS232. Se la stampante non possiede una porta adatta al nostro computer, può darsi che sia reperibile, ad un prezzo ragionevole, un'apposita interfaccia ed un appropriato cavo di connessione per collegare la stampante al computer.

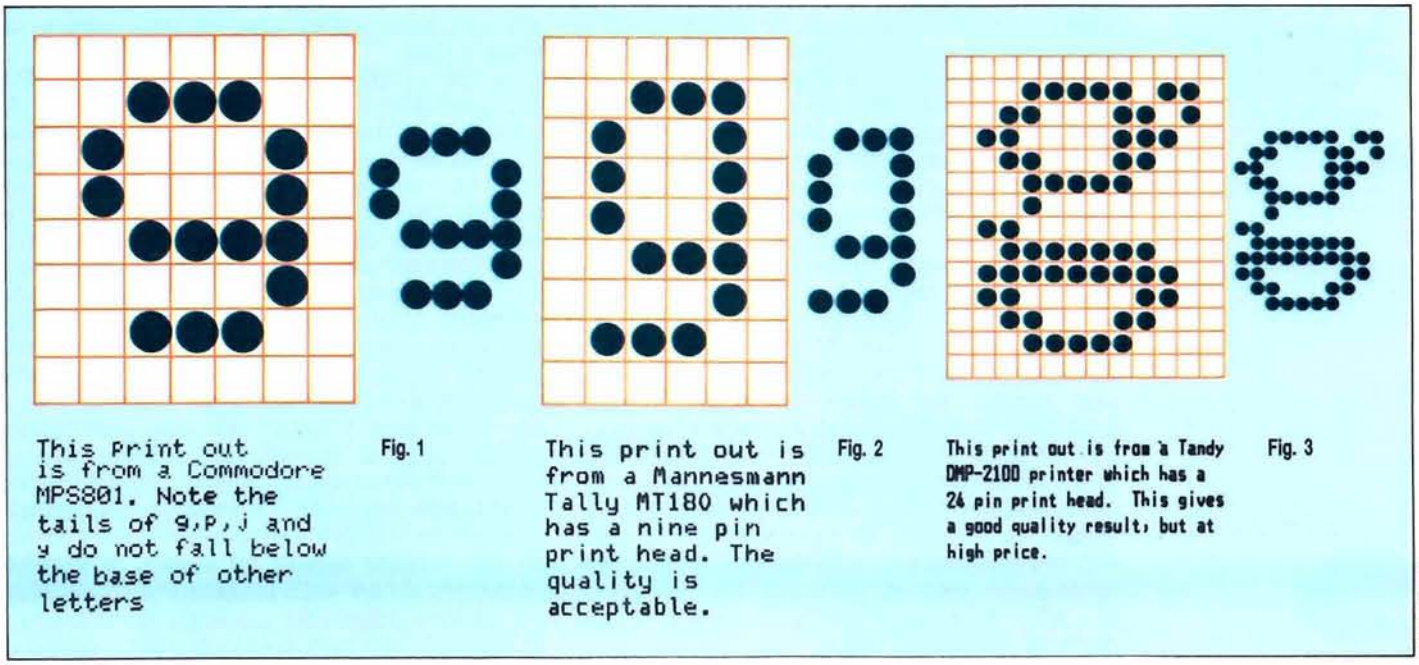
Dettagli...puntigliosi

Questi esempi evidenziano la diversa qualità di stampa ottenibile su alcune stampanti a matrice d'aghi. Le differenze dipendono dal numero di aghi presenti sulla testina di stampa: quelle con più aghi producono caratteri meglio definiti

Fig. 1. Stampante MP8801 della Commodore. La matrice è a soli 7 aghi, insufficienti per stampare le gambette "discendenti" delle lettere g, p, q e y (mancano cioè i "discendenti veri")

Fig. 2. Questo carattere è stato ottenuto con una stampante Mannesmann Tally MT80, dotata di testina di stampa a 9 aghi. La qualità è decisamente accettabile

Fig. 3. Questo carattere è stato stampato con la Tandy DMP-2100, dotata di una testina con 24 aghi. Il risultato è ottimo, ma la stampante è molto costosa



Gli Spreadsheet sono utili?

Le prestazioni delle lavagne elettroniche vengono spesso sottovalutate

Come spesso accade per un word processor o un database, lo "spreadsheet" o lavagna elettronica possiede molte risorse che gli utenti raramente adoperano in modo adeguato. Nei sistemi di word processing, i comandi più sofisticati vengono usati solo di rado, mentre i programmi di database sono utilizzati quasi esclusivamente come indici o per la gestione dei file, trascurando le loro capacità di elaborazione dei dati. La maggior parte dei possessori di home e di personal computer di solito trascura l'utilità del programma di spreadsheet considerato uno strumento difficile da usare, oppure destinato principalmente ad applicazioni finanziarie e commerciali. Raramente viene detto che il programma può essere proficuamente adoperato, ad esempio, nella pianificazione del bilancio familiare, sfruttando così le notevoli potenzialità di elaboratore di cifre e (perché no?) di idee nel campo della contabilità. In effetti, lo spreadsheet può dare risultati paragonabili a quelli ottenuti nella redazione di un testo con un word processor.

In sostanza, una lavagna elettronica abbina le funzioni di un text editor a quelle di un calcolatore. Il nome spreadsheet deriva dalla ripartizione in righe e colonne dell'immagine sul video, che somiglia ad un tabulato quadrettato di contabilità. Le celle della lavagna elettronica possono essere usate in molti modi: vi si possono inserire messaggi o valori numerici da elaborare e visualizzare, oppure formule o procedure di calcolo che operano sui valori contenuti in altre celle. Dopo l'introduzione di tali formule, lo spreadsheet si trasforma in un programma di calcolo progettato "su misura" da e per lo stesso utente: basta immettere i dati da elaborare. Ad ogni nuova registrazione di dati, siano essi messaggi, valori numerici o algebrici, i valori di tutte le celle coinvolte dalle formule vengono ricalcolate in rapida successione e la lavagna risulta quindi costantemente aggiornata. Si può impiegare uno spreadsheet per semplici operazioni di impaginazione video/stampa, semplificando le procedure di formattazione e di stampa non solo di quei calcoli che si potrebbero fare manualmente (se non fossero così noiosi), ma anche di quelli che non sarebbe possibile eseguire in altro modo.

Per ottenere simili risultati con appositi programmi, sarebbero necessari una buona conoscenza pratica del BASIC ed un bel po' di tempo. Gran parte di questo trascorrerebbe nell'impostazione e nella correzione di un'interminabile serie di istruzioni PRINT TAB, PRINT AT e INPUT, necessarie per la creazione di una corretta immagine video.

Uno spreadsheet offre il grande vantaggio di poter registrare i messaggi, i dati numerici ed i risultati direttamente nella posizione che assumeranno poi sullo schermo o nel tabulato di stampa.

Gli spreadsheet possiedono alcuni comandi che facilitano le operazioni d'impaginazione: è possibile copiare, spostare o cancellare blocchi di celle, inserire e cancellare intere righe e colonne, modificare le dimensioni di una cella o di un blocco, il suo allineamento rispetto agli altri dati della stessa colonna e la posizione del punto decimale. Sono proprio questi dettagli d'impaginazione, difficili da trattare con la maggior parte dei dialetti BASIC, ad avere grande importanza per la chiarezza grafica e la comprensibilità di un qualsiasi tabaulato.

Oltre a queste funzioni di formattazione, una lavagna elettronica ne possiede alcune di calcolo. Con un solo comando è possibile calcolare la media dei valori numerici di una riga o di una colonna, oppure conteggiare le registrazioni diverse da zero in una tabella, calcolare la somma di un insieme di numeri, trovare il valore minimo e quello massimo di una serie e via dicendo. Nelle espressioni matematiche, queste ed altre operazioni vengono indicate con una simbologia molto semplice o perlomeno familiare a chi usa il BASIC ad esempio "+" e "/", SQR e ABS. Non tutti gli spreadsheet dispongono però di queste opzioni: il numero e la qualità delle funzioni disponibili dipendono dalla capacità di memoria del computer e da quanto si è disposti a spendere per un programma di questo tipo.

Uno dei comandi più utili è forse REPLICATE, che permette di duplicare un'operazione di calcolo o un dato, registrato in una particolare cella, in un qualsiasi numero di altre celle. Quest'istruzione permette di realizzare, con appena una dozzina di battute, tabulati che hanno come oggetto serie di dati ricorrenti.

La programmazione di una lavagna elettronica può essere considerata una naturale estensione delle capacità aritmetiche del BASIC e permette addirittura di calcolare complicate espressioni matematiche più facilmente di quanto lo consenta il BASIC.

Le definizioni relative ad una particolare applicazione di spreadsheet possono essere memorizzate su disco o nastro, per essere poi riutilizzate più volte. Le versioni più costose prevedono anche il salvataggio di testi o dati in un formato adatto ad essere in seguito elaborato da un word processor o un database. Questa possibilità consente di "riversare in blocco" i risultati di calcoli e proiezioni nel

TEMPO D'ESAMI

Formato
Qui il comando FORMAT è usato per definire la larghezza della colonna D, per allineare a sinistra le celle dei nominativi e per assegnare a tutti i numeri due cifre decimali

Copiatura di celle
Con il comando COPY è possibile ricopiare un particolare blocco di celle in qualsiasi altra parte della lavagna

Fattore di scala
Per ottenere una graduazione "normalizzata", ogni voto viene automaticamente moltiplicato per una costante

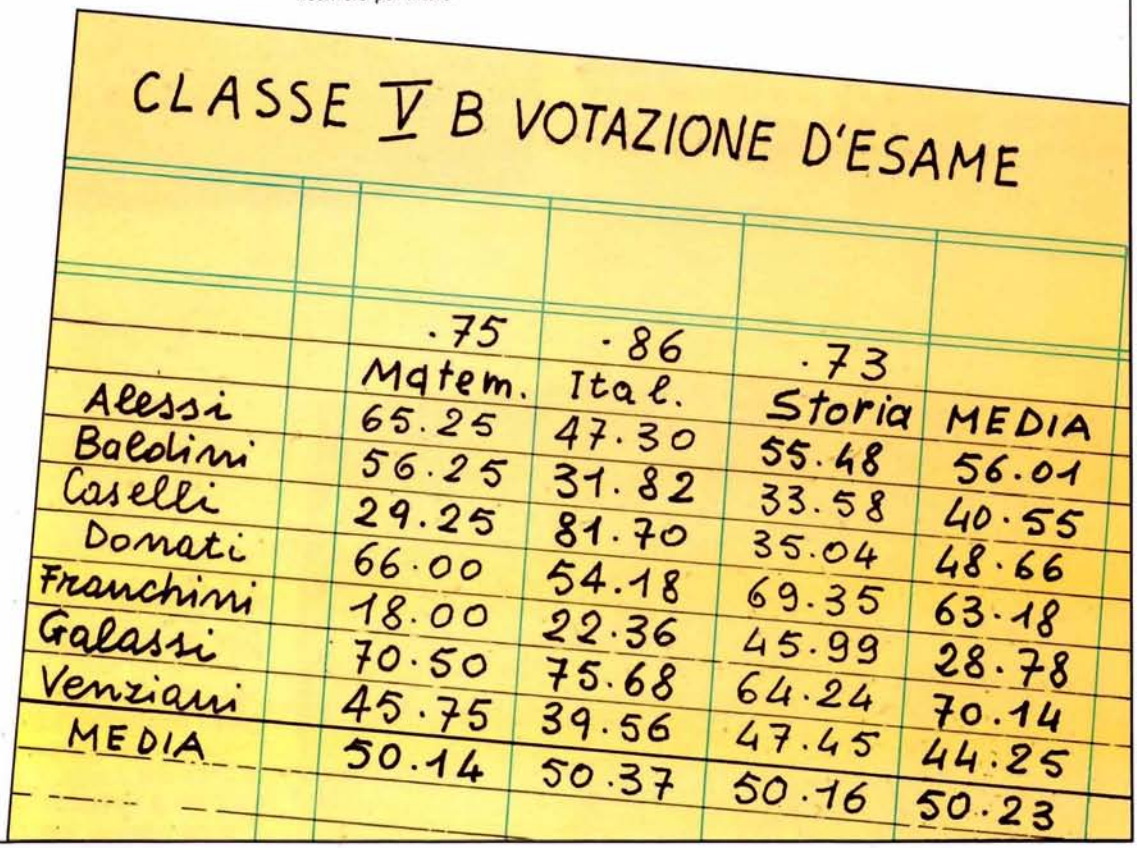
	C	D	E	F	G	H	J	K	L	M
1	NORMALIZZAZIONE DELLE VOTAZIONI									
2	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********
3			VOTAZIONI			*	VOTAZIONI NORMALIZZATE			
4	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********
5						*	.75	.86	.73	
6			Matem.	Ital.	Storia	MEDIA	* Matem.	Ital.	Storia	MEDIA
7	Alessi	:	87.00	55.00	76.00	72.67	* 65.25	47.30	55.48	56.01
8	Baldini	:	75.00	37.00	46.00	52.67	* 56.25	31.82	33.58	40.55
9	Caselli	:	39.00	95.00	48.00	60.67	* 29.25	81.70	35.04	48.66
10	Donati	:	88.00	63.00	95.00	82.00	* 66.00	54.18	69.35	63.18
11	Franchini	:	24.00	26.00	63.00	37.67	* 18.00	22.36	45.99	28.78
12	Galassi	:	94.00	88.00	88.00	90.00	* 70.50	75.68	64.24	70.14
13	Venziani	:	61.00	46.00	65.00	57.33	* 45.75	39.56	47.45	44.25
14			==========	==========	==========	==========	* ==========	==========	==========	==========
15	MEDIA	:	66.86	58.57	68.71	64.71	* 50.14	50.37	50.16	50.23
16										
17	**********	**********	**********	**********	**********	**********	**********	**********	**********	**********
18										

Ripetizione di carattere
L'istruzione REPEAT TEXT consente di riempire un'intera riga digitando un solo carattere (ad esempio l'asterisco *) nella prima cella

Calcolo automatico
La formula inserita in una delle celle, grazie al comando REPLICATE, può essere copiata automaticamente in altre celle senza doverla riscrivere per intero

Media
Viene calcolata dal comando AVERAGE (cell#1:cell#2)

Per avere un quadro generale dei risultati conseguiti dagli allievi, nelle varie materie di studio, un insegnante può cercare di mettere in scala i voti d'esame, ottenendo un voto medio per ogni materia. Supponiamo che la votazione sia in "centesimi": andranno adottati fattori di scala diversi, per ottenere un quadro omogeneo di tutte le materie, calcolando i voti a più riprese. Questa operazione, noiosa e facilmente soggetta ad errori, può essere completata in pochi minuti da uno spreadsheet, che esegue automaticamente i calcoli necessari. Cambiando il fattore di scala, ad esempio, si otterrà in pochi secondi una nuova colonna di risultati, per una particolare materia, calcolati in base all'ultimo valore di scala introdotto



corpo di un testo o di una relazione e rappresenta un significativo passo in avanti verso il software integrato.
I limiti di una lavagna elettronica, dotata di una buona serie di comandi, dipendono quasi esclusivamente dalla fantasia dell'utente o dalla capacità di memoria del computer. Alcune particolari applicazioni contengono talvolta tabulati di grandi dimensioni e sofisticate procedure di elaborazione dati, che esauriscono ben presto la memoria disponibile. Inoltre, procedure di calcolo molto complesse possono rallentare in modo sensibile la velocità di risposta del programma.

Software in vetrina

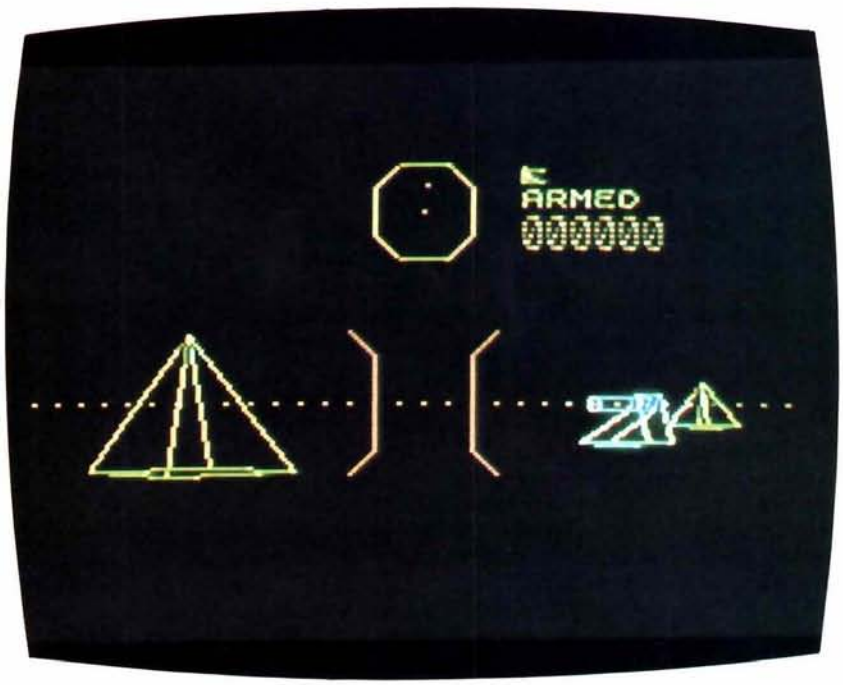
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

È questa la volta di due programmi alquanto diversi tra loro, ma con un denominatore comune: sono giochi "combattivi", un genere molto diffuso nelle sale giochi.

Venus Voyager 2 per l'Atari

Questo gioco è un misto tra il genere degli atterraggi lunari e quello dei salvataggi di umanoidi su pianeti alieni. Ci sono sette astronavi in volo e varie basi di atterraggio. Lo scopo del gioco è di riuscire a far atterrare il maggior numero di navi, recuperando gli uomini bloccati sul pianeta. Il tutto, naturalmente, evitando pericolose pareti sporgenti, mine spaziali vaganti, ed un'astronave "madre", dalla quale, eventualmente, succhiare carburante. Il gioco, che si avvale del joystick, ha una buona grafica, ma è molto impegnativo. È prodotto dalla "English Software".





3D Combat Zone per lo Spectrum 48K

Agli assidui giocatori di videogame questo gioco è familiare: lo schermo rappresenta la visuale data da un periscopio su di un paesaggio stilizzato. L'obiettivo è quello di abbattere il maggior numero possibile di carri armati nemici. Per aiutarci, il gioco visualizza anche lo schermo di un "radar", col quale individuare gli avversari. I movimenti ed il lancio di ordigni, con un discreto effetto tridimensionale, sono comandati da tastiera. Se però è il nemico a colpirci, il nostro schermo (quello simulato, ovviamente!) si frantuma. Il programma è prodotto dalla Artic.

Il Colour Genie

Un compatto home computer dai numerosi pregi

Questo microcomputer, basato sul noto microprocessore Z80, possiede un'ottima tastiera con 62 tasti: 4 di questi sono "di funzione", 2 di Reset (da premere simultaneamente) ed uno è il tasto Select Mode, che permette di ottenere direttamente da tastiera caratteri grafici predefiniti.
L'apparecchio ha 32 Kbyte di memoria, due dei quali riservati alle operazioni di sistema, mentre altri 4 sono dedicati alla grafica ad alta risoluzione. I 16 Kbyte della ROM contengono una versione estesa del BASIC Microsoft, priva però di alcune caratteristiche di programmazione strutturata, presenti invece nei più recenti dialetti BASIC. Tuttavia, è consentito l'uso di variabili intere, a singola e a doppia precisione, di matrici multidimensionali relative a qualsiasi tipo di variabile, oltre a prevedere numerose facilitazioni per il trattamento delle stringhe. Esistono anche molti utili comandi per il controllo del suono e della grafica ad alta risoluzione.
La generazione di suoni è relativamente sofisticata: sono previsti tre canali (che permettono di suonare accordi) ed un'uscita audio applicabile ad un televisore. I comandi BASIC che controllano l'emissione dei suoni sono due: PLAY produce un suono predeterminato tipo "carillon", mentre SOUND permette di generare sonorità d'altro genere.
Le possibilità grafiche del Colour Genie, per quanto estese e di buona resa qualitativa, appaiono piuttosto superate, se confrontate a quelle di apparecchi più recenti. Lo schermo viene considerato come composto da "due pagine" (ed in realtà ci sono due diverse aree di memoria video), una delle quali immagazzina e visualizza testi, stringhe di caratteri grafici e grafica definita dall'utente, mentre l'altra è dedicata alla grafica ad alta risoluzione. Nel modo testuale, il Genie può visualizzare fino a 25 righe di 40 caratteri; nel modo grafico, la dimensione dello schermo è di 160 × 121 pixel.

Uso della grafica

Mediante il tasto MOD SEL si accede alla pagina grafica ad alta risoluzione, riservandole 4 Kbyte di memoria. Il BASIC possiede numerosi comandi per la gestione della grafica, che consentono di tracciare linee, colorare in modo omogeneo estese aree del video, disegnare e cancellare figure. Il comando FGR, se inserito in un programma BASIC, abilita la



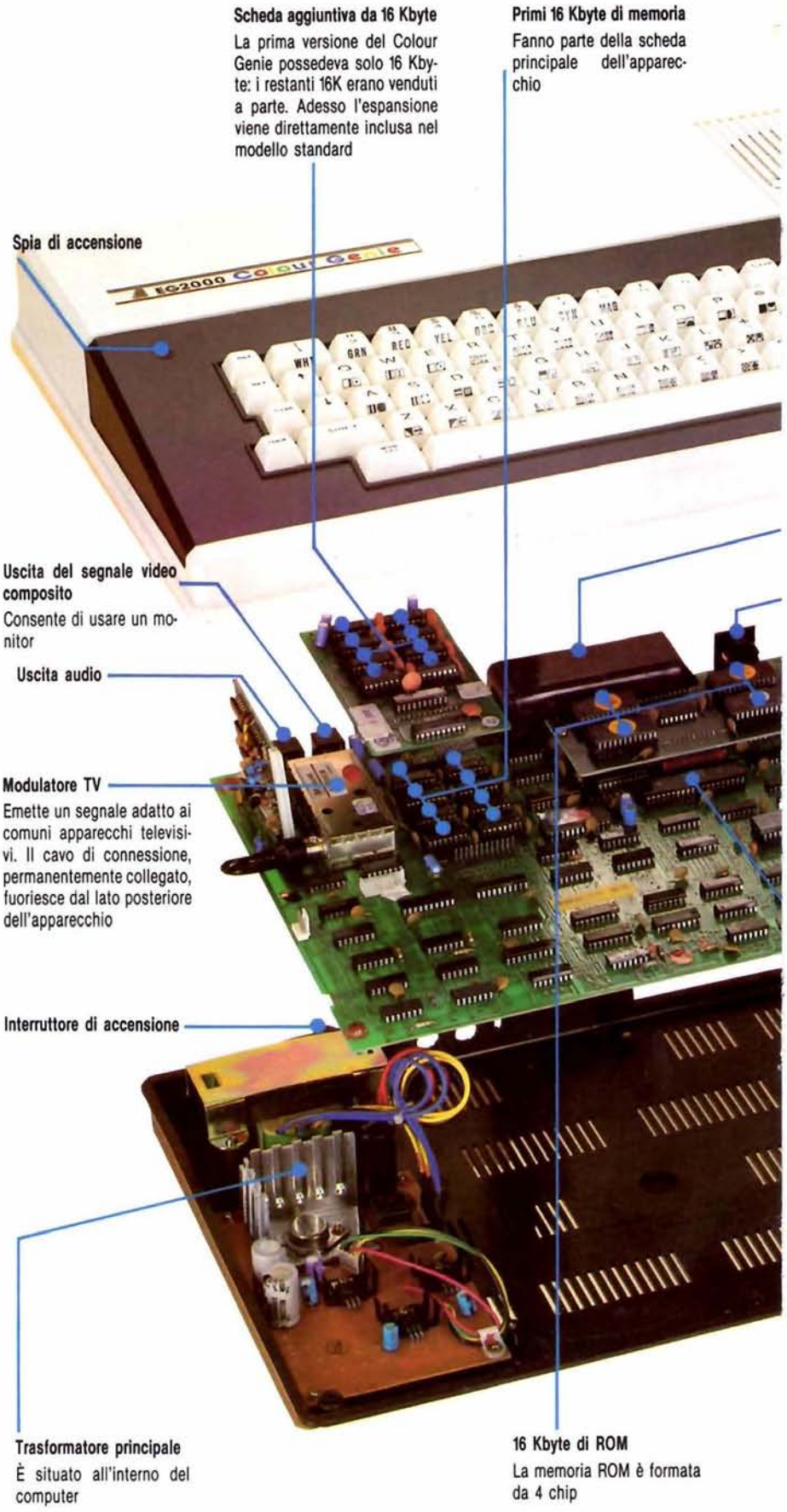
CHRIS STEVENS

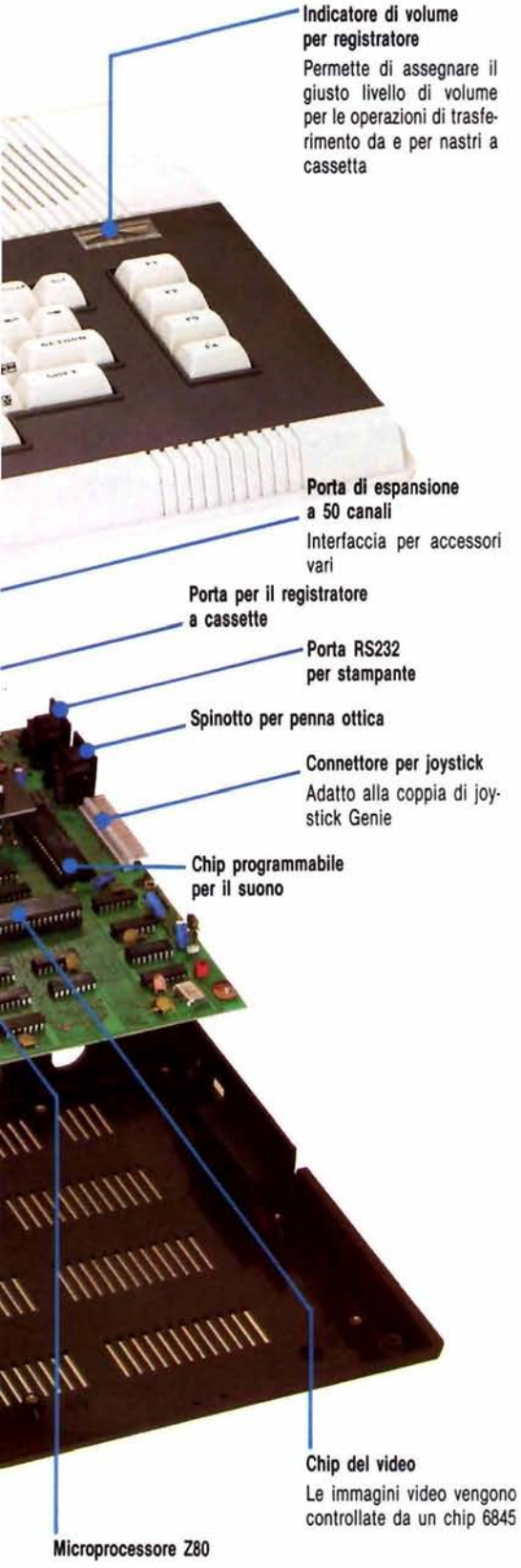
Genie, il timido
Il Colour Genie è un computer che non ha mai riscosso il grande successo di apparecchi quali lo Spectrum o il Commodore 64, pur essendo loro "coetaneo". Nonostante ciò, si è creato un piccolo ma accanito gruppo di estimatori. La macchina possiede 32 Kbyte di memoria e joystick di tipo particolare, con tastierine numeriche incorporate. Questi ultimi vengono venduti a coppie, alloggiate in un unico comodo contenitore



L'indicatore di volume del Genie
La regolazione del volume del registratore a cassette causa, in genere, non poche difficoltà: il Colour Genie è dotato di uno strumento di controllo, simile a quello di alcuni registratori, che facilita notevolmente le operazioni di trasferimento

pagina grafica ma, al termine del programma, il computer torna automaticamente al modo testuale. Nel BASIC esistono anche alcuni comandi per "ripulire" la pagina grafica e cambiare i colori di primo piano (FCOLOR) e di sfondo (FILL). Questo sistema "a due pagine" è meno pratico di quello a singola pagina, adottato sulla maggior parte degli apparecchi moderni, ma permette di colorare ciascun pixel individualmente (mentre sullo Spectrum, ad esempio, la colorazione concerne celle di 8 × 8 pixel, nonostante possieda una più alta risoluzione). La maggior parte del software di gioco tipo arcade utilizza il modo testuale, per le sue doti di velocità, assieme ai caratteri definiti dall'utente, per ottenere immagini ad alta risoluzione.
Le immagini risultano chiare e stabili, ma la forma dei caratteri rende disagevole la lettura. Il Genie dispone di otto colori (bianco, rosso, verde, giallo, azzurro, viola, blu e arancione), tutti utilizzabili contemporaneamente nel modo testuale. Nella grafica ad alta risoluzione si è invece limitati ai soli 4 colori (rosso, blu, verde e nero) ma, tramite il comando BGRD, è possibile assegnare il colore rosa allo sfondo della pagina grafica.
L'apparecchio è dotato di numerose interfacce: una porta seriale RS232 per stampante o modem; una porta d'espansione a 50 canali alla quale possono essere collegate le unità a dischetti, un segnale composito per monitor; un'uscita audio; uno spinotto per penna ottica ed una porta per joystick. Le periferiche disponibili comprendono un'interfaccia per stampante Centronics, vari tipi di joystick, una cartuccia per il sistema Prestel (da usare congiuntamente ad un modem) e diverse unità a dischetti. I joystick prodotti dalla linea Genie possiedono tastierine numeriche incorporate, ma non sono facili da usare: occorre esercitare una discreta pressione per ottenere una risposta e la cloche, se lasciata andare, non torna automaticamente alla posizione centrale. Le unità a dischetti per il Genie vengono prodotte da un'altra fabbrica britannica, che ha optato per un sistema operativo chiamato QDOS, simile al TRS-DOS della Tandy.
Fatto raro, se non unico, il Genie è dotato di un indicatore di volume, grazie al quale si eliminano molte delle difficoltà connesse alla





registrazione su cassette: basta regolare il volume del registratore in modo che l'ago dello strumento si trovi in posizione centrale. Inoltre, è possibile interporre fra il computer ed il registratore uno "stabilizzatore di trasmissione dei dati". Questo strumento filtra il segnale e favorisce le operazioni di trasferimento.

Assieme al Colour Genie vengono consegnati due manuali: una guida per principianti ed un manuale d'uso. Entrambi sono scritti in modo semplice, ma appaiono piuttosto incompleti e mancano di un indice. Per i più esperti è disponibile anche un manuale tecnico.

Il "Martian Raider"



IAN McKINNELL

Scelta del software

Il software disponibile per il Colour Genie è piuttosto limitato, ma la sua qualità è più che accettabile. Gran parte del software è costituito da giochi, che spesso derivano da versioni create per apparecchi di maggior successo

COLOUR GENIE

DIMENSIONI

mm. 90 x 280 x 340

CPU

Z80, a 2,2 MHz

MEMORIA

32 Kbyte di RAM, 16 Kbyte di ROM

VIDEO

Fino a 25 righe di 40 colonne, con 8 colori (modo testuale) a 160 x 102 pixel, con 4 colori (modo grafico)

INTERFACCE

2 porte per joystick, una RS232, una per penna ottica, una per espansioni

LINGUAGGI DISPONIBILI

BASIC

TASTIERA

Tipo macchina da scrivere, con 62 tasti di cui 4 per funzioni

DOCUMENTAZIONE

Corredato di un manuale d'uso e di uno per il BASIC ben redatti, ma un po' troppo stringati e mancanti di indice. Dispone di un manuale tecnico

PRO

Il Colour Genie è un apprezzabile home computer: dotato di un BASIC Microsoft, ha una discreta grafica e buoni effetti audio. Questi ultimi vengono riprodotti sull'apparecchio TV

CONTRO

Il progetto del Genie è un po' "antiquato", rispetto ad altri apparecchi della stessa categoria. Il display è gestito su due "pagine" e, sul mercato, c'è scarsità di software

I joystick del Colour Genie

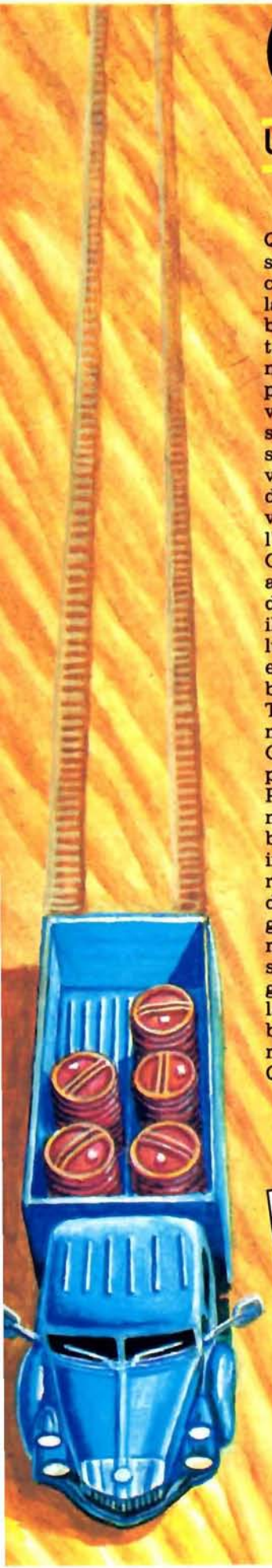
I joystick del Genie sono strumenti interessanti, ma costosi e difficili da usare. La cloche non è molto sensibile e non ritorna automaticamente in posizione centrale, se lasciata andare. La tastierina numerica di cui sono dotati è originale e può tornare utile in qualche gioco



CHRIS STEVENS

Carovana nel deserto

Un breve gioco di logica, col quale esercitarsi nella programmazione



Questo gioco è ambientato in un deserto che si estende per 1.000 Km. Ogni 100 Km. circa, c'è una stazione di deposito dove è possibile lasciare più taniche di carburante. Dalla base principale si possono prelevare quante taniche si vogliono, ciascuna capace di contenere il carburante necessario all'autocarro per spostarsi da un deposito al successivo. Il viaggio attraverso il deserto presenta una sola grossa difficoltà: l'autocarro può trasportare un massimo di otto taniche alla volta. Quindi, per compiere la traversata, si dovranno creare depositi di carburante in vari punti della strada, facendo la spola tra l'uno e l'altro.

Ovviamente, il primo obiettivo del gioco è assicurarsi di non rimanere senza benzina durante il viaggio: la base è molto lontana ed il deserto non è certo il posto ideale per una lunga passeggiata. Inoltre, la traversata deve essere compiuta seguendo l'itinerario più breve con un consumo minimo di carburante. Tuttavia, è possibile modificare il programma per rendere la soluzione più complicata. Che cosa accadrà, ad esempio, se l'autocarro può trasportare solo 4 o 6 taniche alla volta? Per sperimentare queste variazioni, basta modificare il valore di M alla linea 60. Probabilmente scopriremo che la tattica adottata in precedenza è ancora valida, ma che occorre anche cambiare tanto la distanza tra due depositi successivi, quanto il numero di viaggi per rifornirli. È possibile ideare un algoritmo che permetta di attraversare il deserto sentendosi sicuri in ogni momento del viaggio? Un simile algoritmo potrebbe costituire la base di un programma "risolutore di problemi", adatto anche a situazioni reali (ottimizzazione dei percorsi).

Questo piccolo rompicapo evidenzia una tecnica di inestimabile valore per la soluzione di particolari problemi in fase di programmazione. Si devono innanzitutto eseguire delle prove, basandosi sulle informazioni a noi note, e poi studiare un certo numero di casi già risolti con altri criteri. Infine, se tutto è andato per il meglio, dedurre un modello di comportamento. Da questo si potrà sviluppare un algoritmo e quindi iniziare la stesura di un programma adatto a risolvere il problema. Desiderando migliorare il gioco, lo si potrà corredare di un'adeguata grafica o anche di nuove difficoltà: ad esempio, la necessità di trasportare acqua e cibo, oltre al carburante.

Basic & Basic

Questo programma è scritto in Microsoft BASIC e si adatta perciò, senza modifiche, alla maggior parte dei computer con schermo 40 x 25. I possessori di Spectrum devono inserire la parola chiave LET prima di ogni istruzione di assegnazione.

CHR$(26): viene sostituita da CLS su Spectrum, Oric-1, Atmos, Dragon e BBC; da CHR$(147) su Commodore 64 e Vic-20

MID$(STR$(A(1),2)): viene sostituita da STR$(A(1)) sullo Spectrum e su tutti gli apparecchi nei quali il comando PRINT LEN(STR$(2)) dà come risultato 1.

THEN 1260 & THEN 1300: sullo Spectrum queste istruzioni vanno trasformate in THEN GOTO 1260 & THEN GOTO 1300

```
10 REM ***  Carovana nel deserto  ***
30 DIM A(10)
40 A(1)=80: REM *** Numero di taniche iniziali ***
50 S=1: REM ***        Posizione dell'autocarro  ***
60 M=8: REM *** Numero max. di taniche sull'autocarro ***
70 N=0: REM *** Numero totale di taniche usate ***
100 REM ***          Mossa successiva          ***
110 PRINT CHR$(26):REM *** Cancella lo schermo ***
120 PRINT "Fase:  1  2  3  4  5  6  7  8  9  10"
130 PRINT "Taniche:  ";
140 FOR I=1 TO 10
145 A$=MID$(STR$(A(I)),2)
147 IF LEN(A$)<2 THEN LET A$=" "+A$:GOTO 147
150 PRINT A$;" ";: NEXT I: PRINT
160 X=9+(S-1)*3: PRINT TAB(X);"^";: PRINT TAB(X-1);T
    PRINT TAB(X-2);"Autocarro";: PRINT "Ce l'hai fatta!!!":GOTO 1400
175 IF S=10 THEN PRINT: PRINT "Spiacente
180 IF T=0 AND A(S)=0 THEN PRINT:GOTO 1400
    ... puoi cominciare a camminare":GOTO 1400
190 PRINT: PRINT "Le tue scelte sono:": PRINT
200 IF T>1 THEN PRINT "S = Scarica le taniche"
210 IF A(S)>0 THEN PRINT "C = Carica le taniche"
220 IF T>0 THEN PRINT "A = Avanti di una tappa"
230 IF T>0 AND S>1 THEN PRINT "I = Indietro di una tappa"
240 PRINT: PRINT "Quale";: INPUT A$
250 IF T>1 AND (A$="S" OR A$="s") THEN 1260
260 IF A(S)>0 AND (A$="C" OR A$="c") THEN 1300
270 IF T>0 AND (A$="A" OR A$="a") THEN S=S+1:T=T-1:N=N+1:
    GOTO 110
280 IF T>0 AND S>1 AND (A$="I" OR A$="i") THEN S=S-1:T=T-1: N=N+1:
    GOTO 110
290 GOTO 110
1250 REM ***          Scarico delle taniche          ***
1260 PRINT: INPUT "QUANTE "; A
1270 IF A>T OR A<>INT(A) OR A<0 THEN PRINT:
     PRINT "Per favore, riprova":GOTO 1260
1280 A(S)=A(S)+A: T=T-A:GOTO 110
1290 REM ***          Carico delle taniche          ***
1300 PRINT: INPUT "Quante ";A
1310 IF A>A(S) OR A<>INT(A) OR A<0 THEN PRINT:
     PRINT "Per favore, riprova":GOTO 1300
1320 IF A+T>M THEN PRINT: PRINT "Spiacente... c'e'
     spazio per sole ";M;" taniche":GOTO 1300
1330 T=T+A: A(S)=A(S)-A
1380 GOTO 110
1390 REM ***          Fine del gioco          ***
1400 PRINT: PRINT "Hai usato ";N;" taniche di carburante. ";
1410 PRINT "ne hai trasportate"; T
1420 A=0: FOR I=2 TO 9: A=A+A(I): NEXT I
1430 PRINT "con te e lasciate ";A;" nel deserto."
1440 PRINT: PRINT "Digitare RUN per giocare ancora"
1450 END
```

ADRIAN MORGAN

Il Manic Miner

Un gioco di grandissimo successo, che merita conoscere e giocare

Il *Manic Miner*, nelle versioni per lo Spectrum a 48K e per il Commodore 64, è un gioco estremamente semplice, basato su un precedente di gran successo, chiamato *Kong*. In questo, lo scopo è liberare una principessa, prigioniera del grosso gorilla Kong. Per far ciò occorre salire scale e arrampicarsi su alberi, evitando tutti gli ostacoli. Nel *Manic Miner*, il giocatore interpreta il ruolo di Willy il Minatore, un cercatore d'oro che ha scoperto un "pozzo" abbandonato nel quale una civiltà ormai scomparsa ha sotterrato oro ed altre ricchezze. Sfortunatamente, gli antichi visitatori della miniera non hanno mai disattivato i Robot Scavatori, che ora rendono estremamente pericoloso il recupero del tesoro.

La miniera è composta da 20 caverne ed in ciascuna di esse ci sono 4 chiavi: ogni volta, Willy deve cercare di impadronirsi delle chiavi per poter aprire la porta di accesso alla caverna successiva.

Ogni caverna ha un certo numero di camminamenti sovrapposti (alcuni piuttosto consunti e instabili) sui quali Willy deve saltare per raggiungere le chiavi. Le caverne sono tappezzate di frasi come: "Tana di Eugene" (una pungente citazione rivolta a Eugene Evans, programmatore della rivale software house "Image"?), "Attacco di Telefoni Mutanti" (un'altra ironica frase, questa volta riferita a Jeff Minter, il programmatore ossessionato dall'idea dei lama mutanti?) e così via. Le gallerie sono infestate da presenze aliene, il cui tocco provoca morte istantanea e persino le piante sono letali.

Il giocatore aiuta Willy ad evitare tutti questi pericoli usando tre semplici comandi: "Destra", "Sinistra" e "Salta". Questa è una delle caratteristiche più interessanti del gioco: infatti la semplicità dei comandi di controllo permette di imparare a giocare in pochi secondi. Inoltre, si può scegliere l'abbinamento tra comandi e tasti che si ritiene più comodo.

Willy ha tre "vite" e in ciascuna possiede un limitato rifornimento d'aria. Quando "muore" per la terza volta, si riparte dalla Caverna 1. Ciò può essere un po' irritante, cosicché alcuni giocatori hanno modificato il programma in modo da poter iniziare da una caverna di propria scelta.

La versione Commodore è quasi identica a quella dello Spectrum, ma non riesce a trarre significativi vantaggi dal più versatile generatore audio e dalla maggiore risoluzione grafica. Le dimensioni dell'area di gioco sul 64, del tutto identiche a quelle dello Spectrum, appaiono piuttosto limitate rispetto all'area video disponibile.

Entrambe le versioni offrono senza dubbio qualche ora di buon divertimento. Il ritmo è apprezzabile ed i problemi incontrati nel gioco sono stati preparati con grande cura. Mattew Smith ha già prodotto un seguito alla storia, il *Jet Set Willy*, che sta ottenendo un successo ancor maggiore di *Manic Miner*.

Manic Miner: Per lo Spectrum 48K, e per il Commodore 64

Editori: Software Projects, (GB)

Autore: Mattew Smith

Versione con joystick: entrambe

Formato: su nastro a cassetta



Manic Miner sullo Spectrum



Manic Miner sul Commodore 64

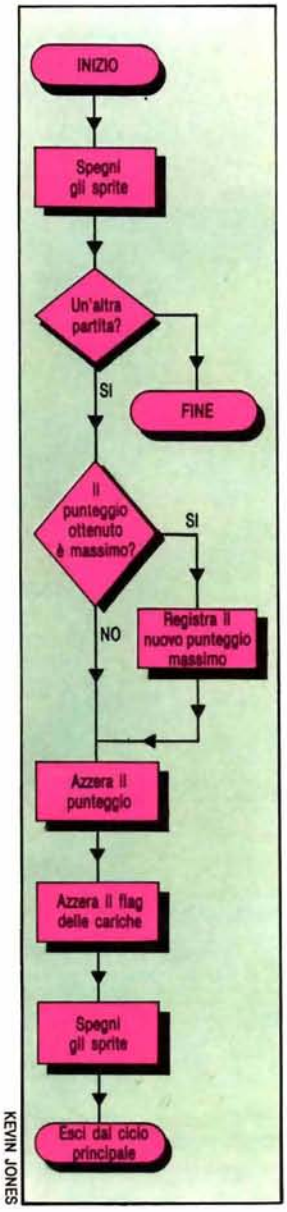
Laggiù in miniera!

Le entità soprannaturali e minacciose, che popolano il mondo sotterraneo di Manic Miner, hanno contribuito a rendere il gioco quasi un oggetto di culto. I vecchi appassionati si vantano anche di aver scoperto, nelle loro peregrinazioni sotterranee, esseri invisibili ai comuni giocatori!

IAN McKINNELL

Via ai sottomarini!

I ritocchi finali ed il listato completo della "Caccia al sottomarino"



Nella penultima parte di questa serie abbiamo visto con quanta facilità sia possibile individuare le collisioni tra gli sprite usando l'apposito "registro di collisione", il V+30. Quando si verifica una collisione, la subroutine che inizia alla linea 5000 esegue le tre seguenti operazioni: innanzi tutto visualizza un'esplosione nel punto dello schermo nel quale si è verificata la collisione; quindi somma al punteggio del giocatore il valore del sottomarino colpito, che viene calcolato in base alla sua velocità (DX) e profondità (Y3); infine azzera le coordinate per il passaggio del sottomarino successivo. Esaminiamo ora più dettagliatamente la subroutine "Bersaglio Colpito", che inizia alla linea 5000.

La linea 5010 azzera il registro di collisione V+30. La Commodore asserisce che tale registro si azzera automaticamente dopo la collisione di due sprite, ma in realtà il registro non sempre viene azzerato con sufficiente rapidità, causando a volte un'immotivata apparizione dello sprite dell'esplosione. Per eliminare questo inconveniente, conviene che il programma azzeri il registro dopo ogni collisione. Dopo quest'operazione, si potrà nuovamente posizionare ed accendere lo sprite dell'esplosione.

La linea 5030 assegna alla coordinata X dell'esplosione un valore, incrementato di 10 pixel verso destra, rispetto a quello della carica di profondità. Questo leggero spostamento fa apparire l'esplosione in una posizione più appropriata. Poiché il valore di X2 deriva dalla coordinata X (ossia X0) della nave, il suo limite superiore è 245: quindi il valore massimo della coordinata X dell'esplosione non può superare 255. La coordinata Y dell'esplosione viene ricavata direttamente da quella del sottomarino.

Lo sprite 1 è associato a quello dell'esplosione: la linea 5040 pone a uno il bit 1 del registro V+21, accendendo quindi lo sprite 1 senza modificare gli altri bit del registro. È interessante notare che lo sprite dell'esplosione apparirà sopra o davanti agli sprite del sottomarino e della carica di profondità. Questo effetto di sovrapposizione è detto "priorità" degli sprite e si basa sul principio che gli sprite con valore minore appaiono anteriormente, rispetto allo sfondo, di quelli con valore maggiore. Ecco perché l'esplosione è stata definita come sprite 1, mentre le cariche di profondità ed il sottomarino, rispettivamente, come 2 e 3.

Il colore dell'esplosione viene controllato dalla locazione V+40 del chip VIC. Si può ottenere un interessante effetto cromatico cambiando rapidamente il colore dell'esplosione: un ciclo FOR...NEXT permette di depositare in rapida successione (con una POKE) i codici di colore compresi tra 0 e 15; un secondo ciclo FOR...NEXT fa ripetere per 20 volte quest'operazione (linee 5060-5100). Terminata l'esplosione, tutti e tre gli sprite (esplosione, carica di profondità e sottomarino) devono scomparire dallo schermo: la linea 5130 spegne gli sprite 1, 2 e 3.

Come detto in precedenza, il punteggio del giocatore viene aggiornato dalla subroutine che inizia alla linea 5500. Il flag DS viene posto ad 1 quando il valore del sottomarino deve essere sommato al punteggio (o sottratto, se il sottomarino raggiunge incolume il margine destro dell'area di gioco). Infine, prima che un secondo sottomarino possa attraversare il video, è necessario accendere il relativo sprite ed azzerarne le coordinate usando la subroutine alla linea 2500. Inoltre, occorre azzerare il flag che segnala il lancio di una carica di profondità per permettere al giocatore un nuovo lancio.

Dopo circa tre minuti di gioco, il programma lascia il ciclo principale e salta alla linea 400. Quando abbiamo trattato l'uso del temporizzatore interno del Commodore 64 (vedere a pag. 714), la linea 400 era una semplice istruzione END. La routine "Fine del Gioco" (dalla linea 400 alla 660) permette adesso di avviare un nuovo gioco e di registrare i punteggi più alti (come mostrato nel diagramma di flusso a sinistra). Gran parte del programma risulta di immediata comprensione: basta ricordare che CHR$(19) posiziona il cursore sull'angolo in alto a sinistra del video e che CHR$(144) visualizza in nero i successivi caratteri sulla linea dello schermo.

A questo punto, per rendere più incisivo ed avvincente il gioco, potremmo dare più ritmo all'azione, sfruttando anche i 4 sprite ancora liberi.

Tabella delle variabili usate nel programma

V	Accensione dei registri del chip VIC
FL	Flag delle cariche di profondità, posto a uno quando viene lanciata una carica
SC	Punteggio corrente del giocatore
HS	Punteggio massimo
TI$	Temporizzatore del Commodore 64
X0	Coordinata X della nave
X2,Y2	Coordinate X e Y della carica di profondità
X3,Y3	Coordinate X e Y del sottomarino
H3,L3	I byte alto e basso della coordinata X del sottomarino
DX	Numero di pixel di cui aumenta la coordinata X del sottomarino
DS	Segnala se il punteggio deve essere incrementato (DS = 1) o diminuito (DS = −1)

CACCIA AL SOTTOMARINO: LISTATO DEFINITIVO

```
10 REM ****************************************
30 REM *  CACCIA AL SOTTOMARINO PER IL C64     *
70 REM ****************************************
90 POKE 55,0: POKE 56,48: CLR: REM LIMITE SUP. MEMORIA
100 V=53248: FL=0: SC=0
110 GOSUB 1000: REM MESSA A PUNTO DEL VIDEO
120 GOSUB 2000: REM CREAZIONE DEGLI SPRITE
130 GOSUB 2500: REM DEFINIZIONE COORDINATE SOTTOMARINO
140 TI$="000000"
200 REM ***    CICLO PRINCIPALE      ***
210 REM ***    VISIONE CONTATEMPO    ***
220 PRINT CHR$(19);: PRINT TAB (14) CHR$ (5);"TEMPO  "
    MID$ (TI$,3,2) ":" RIGHT $ (TI$,2)
225 IF VAL(TI$))259 THEN 400: REM FINE DEL GIOCO
230 GET A$
240 IF A$="I" THEN IO=IO-1.5: IF IO<24 THEN IO=24
250 IF A$="X" THEN IO=IO+1.5: IF IO)245 THEN IO=245
260 IF A$="M" AND FL=0 THEN GOSUB 3000: REM CREAZIONE
    DELLE CARICHE DI PROFONDITA'
270 REM ***   MOVIMENTO DELLA NAVE   ***
290 POKE V,IO
300 REM *** MOVIMENTO DEL SOTTOMARINO ***
310 I3=I3+DI
320 REM  *** SOTTOMARINO HA RAGGIUNTO IL MARGINE? ***
330 IF I3)360 THEN DS=-1: GOSUB 5500: GOSUB 2500
340 H3=INT (I3/256): L3=I3-256*H3
350 POKE V+6,L3
360 IF H3=1 THEN POKE V+16, PEEK (V+16) OR 8: GOTO 380
370 POKE V+16, PEEK (V+16) AND 247
380 IF FL=1 THEN GOSUB 4000: REM DISCESA DELLA CARICA DI PROFONDITA'
390 GOTO 200: REM RILANCIO DEL CICLO PRINCIPALE
400 REM ***    CONDIZIONI DI FINE GIOCO    ***
410 REM ***    SPEGNIMENTO DEGLI SPRITE    ***
420 POKE V+21,0
430 REM *** AZZERAMENTO COORDINATE SOTTOMARINO & NAVE ***
440 IO=260: GOSUB 2500
450 INPUT "UN'ALTRA PARTITA  (SI/NO)"; AN$
460 IF AN$<>"S" THEN  END
480 REM  ***  CANCELLAZIONE DEL MESSAGGIO  ***
490 PRINT CHR$(19): REM CURSORE ALL'ANGOLO IN ALTO A SIN.
500 FOR I=1 TO 120
510 PRINT " ";
520 NEXT I
540 REM *** SELEZIONA IL PUNTEGGIO MASSIMO ***
550 IF SC)HS THEN HS=SC
560 PRINT CHR$ (19); CHR$ (144); "PUNTEGGIO 000": SC=0
570 PRINT CHR$ (19);
580 PRINT TAB (26); CHR$ (144); "PUNTEGGIO MASSIMO"; HS
600 REM *** AZZERAMENTO DEL TEMPO E DEL FLAG ***
610 TI$="000000":FL=0
630 REM *** ACCENSIONE SPRITE SOTTOMARINO E NAVE ***
640 POKE V+21,9
660 GOTO 200: REM RIAVVIO DEL CICLO PRINCIPALE
1000 REM ***    MESSA A PUNTO DEL VIDEO    ***
1010 PRINT CHR$ (147): CANCELLAZIONE DEL VIDEO
1030 REM ***    COLORAZIONE DEL MARE       ***
1040 POKE 53281,14: POKE 53280,6
1050 FOR I=1246 TO 1943
1060 POKE I,160: POKE I+54272,6
1070 NEXT I
1090 REM ***        FONDALE MARINO         ***
1100 FOR I=1944 TO 2023
1110 POKE I,102: POKE I+54272,9
1120 NEXT I
1130 POKE 650,128: REM RIBATTERE I TASTI
1150 REM ***         PUNTEGGIO             ***
1160 PRINT CHR$ (19);  CHR$ (144);  " PUNTEGGIO  000";
     SPC (16); "PUNTEGGIO MASSIMO 000"
1170 RETURN
2000 REM ***     CREAZIONE DEGLI SPRITE    ***
2020 REM ***     LETTURA DATI DELLA NAVE   ***
2030 FOR I=12288 TO 12350
2040 READ A: POKE I,A: NEXT I
2060 REM ***     LETTURA DATI SOTTOMARINO  ***
2070 FOR I=12352 TO 12414
2080 READ A: POKE I,A: NEXT I
2100 REM ***  LETTURA DATI DELLE CARICHE   ***
2110 FOR I=12416 TO 12478
2120 READ A: POKE I,A: NEXT I
2140 REM ***  LETTURA DATI DELL'ESPLOSIONE ***
2150 FOR I=12480 TO 12542
2160 READ A: POKE I,A: NEXT I
2180 REM *** MESSA A PUNTO DEI PUNTATORI   ***
2190 POKE 2040,192: POKE 2041,193: POKE 2042,194
2200 POKE 2043,195
2220 REM ***     DEFINIZIONE DEI COLORI    ***
2230 POKE V+39,0: POKE V+40,1: POKE V+41,0
2240 POKE V+42,0
2260 REM *** ASSEGNAZIONE COORDINATE INIZIALI ***
2270 POKE V+1,80: IO=160: REM COORDINATE DELLA NAVE
2280 POKE V+29,15: POKE V+23,2
2300 REM ***     ACCENSIONE SPRITE 0 & 3   ***
2310 POKE V+21,9
2320 RETURN
2500 REM *** AZZERAMENTO COORDINATE SOTTOMARINO ***
2510 Y3=110+INT (RND(TI)*105)
2520 POKE V+7,Y3: POKE V+6,0
2530 I3=0: DI=RND (TI)*3+1
2540 POKE V+16,0
2550 RETURN
3000 REM *** MESSA A PUNTO DELLE CARICHE   ***
3020 REM ***     ACCENSIONE DEL FLAG       ***
3030 FL=1
3050 REM *** DEFINIZIONE COORDINATE DELLE CARICHE ***
3060 Y2=95: I2=IO
3070 POKE V+4,I2: POKE V+5,Y2
3090 REM ***     ACCENSIONE SPRITE 2       ***
3100 POKE V+21, PEEK (V+21) OR 4
3110 RETURN
4000 REM *** DISCESA CARICHE DI PROFONDITA' ***
4020 REM ***     DECREMENTO COORDINATA Y    ***
4030 Y2=Y2+2
4050 REM *** VERIFICA DI CONTATTO COL FONDALE ***
4060 IF Y2)Y3+25 OR Y2)216 THEN POKE V+21, PEEK (V+21) AND 251: FL=0
4070 POKE V+5,Y2
4090 REM *** VERIFICA PER SOTTOMARINO COLPITO ***
4100 IF PEEK (V+30)=12 THEN GOSUB 5000: REM ROUTINE BERSAGLIO COLPITO
4110 RETURN
5000 REM ***    ROUTINE BERSAGLIO COLPITO   ***
5010 POKE V+30,0: REM AZZERAMENTO DEL REG. DI COLLISIONE
5020 REM ***  ACCENSIONE SPRITE ESPLOSIONE  ***
5030 POKE V+2,I2+10: POKE V+3,Y3
5040 POKE V+21, PEEK (V+21) OR 2
5060 REM ***  VARIAZIONE COLORI ESPLOSIONE  ***
5070 FOR I=1 TO 20
5080 FOR J=1 TO 15
5090 POKE V+40,J
5100 NEXT Y, NEXT I
5120 REM ***   SPEGNIMENTO SPRITE 1, 2 & 3  ***
5130 POKE V+21, PEEK (V+21) AND 241
5150 REM ***    AGGIORNAMENTO PUNTEGGIO     ***
5160 DS=1: GOSUB 5500
5180 REM *** AZZERAM. COORDINATE SOTTOMARINO E FLAG ***
5190 FL=0: GOSUB 2500
5210 REM *** NUOVA ACCENSIONE SPRITE DEL SOTTOMARINO ***
5220 POKE V+21, PEEK (V+21) OR 8
5230 RETURN
5500 REM ***    AGGIORNAMENTO PUNTEGGIO     ***
5510 SC=SC+INT(Y3+DI*30)*DS
5520 IF SC<0 THEN SC=0
5530 PRINT CHR$(19); CHR$(144); " PUNTEGGIO";SC; CHR$(157);"    "
5540 RETURN
6000 REM **     DATI DELLA NAVE     **
6010 DATA 0,0,0,0,0,0,0,0,0
6020 DATA 0,128,0,0,192,0,0,192,0
6030 DATA 0,192,0,1,224,0,1,244,0
6040 DATA 13,244,0,3,248,128,3,253,8
6050 DATA 15,254,16,31,255,48,255,255,255
6060 DATA 127,255,254,63,255,254,31,255,252
6070 DATA 0,0,0,0,0,0,0,0,0
6100 REM **     DATI DELL'ESPLOSIONE     **
6110 DATA 0,0,0,0,0,0,0,16,0,0,8,0,4,16
6120 DATA 0,3,2,64,1,56,128,12,255,144
6130 DATA 1,238,40,5,151,0,11,121,0,1
6140 DATA 183,0,25,214,96,0,236,48,6,24
6150 DATA 152,3,98,0,8,51,0,0,96,128,0
6160 DATA 64,0,0,0,0,0,0,0
6170 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6200 REM ** DATI DELLE CARICHE DI PROFONDITA' **
6210 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6220 DATA 0,0,0,32,0,0,32,0,0,32,0,0,32,0
6230 DATA 0,0,0,0,0,0,0
6240 DATA 2,0,0,2,0,0,2,0,0,2,0,0
6250 DATA 0,0,0,0,0,0,0,0
6260 DATA 0,0,0,0,0,0,0,0
6300 REM **         DATI DEL SOTTOMARINO     **
6310 DATA 0,0,0,0,0,0,0,0,0,0,0,0
6320 DATA 0,8,0,0,12,0,0,12,0
6330 DATA 0,12,0,0,28,0,0,60,0
6340 DATA 0,126,0,199,255,255
6350 DATA 239,255,255,127,255,255
6360 DATA 255,255,254,199,255,254
6370 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Ecco il listato completo del programma "Caccia al Sottomarino" ed una tabella delle variabili in esso usate. Per favorirne la comprensione, il listato contiene molte istruzioni REM, che possono essere tralasciate durante la trascrizione del programma. Attenzione, però, a non eliminare le linee REM richiamate da altre linee del programma. Ad esempio, l'istruzione REM alla linea 400 viene richiamata dall'istruzione GOTO nella linea 225. La sua eliminazione provocherà la comparsa del messaggio "UNDEF'D STATEMENT ERROR AT LINE 225" (= "la linea richiamata nella 225 non è definita") ed un precoce arresto del programma. Il modo migliore per evitare quest'eventualità è tralasciare solo quelle REM che si trovano nella parte finale di una linea e sono precedute dai "due punti" (:)

Le divisioni

Vediamo come eseguire anche quest'operazione nei programmi in codice macchina

Partendo dal metodo manuale, abbiamo definito l'algoritmo per la moltiplicazione binaria (vedere a pag. 778). Lo stesso criterio vale per la divisione di numeri binari. Osserviamo questa divisione binaria:

```
      00001110 r00   quoziente
1011)10011010        dividendo
    -1011            sottrazione del divisore
     10000
    -1011            sottrazione del divisore
      1011
    -1011            sottrazione del divisore
        00           nessun resto
```

Questo metodo è basato sulla ripetuta sottrazione del divisore dai bit superiori del dividendo. A seconda del risultato di questa sottrazione, viene trasferito uno 0 o un 1 nel quoziente. Il resto deriva dal risultato dell'ultima sottrazione.
I vari modi, per implementare questo algoritmo nel codice macchina, non sono così evidenti come quelli per la moltiplicazione. Tuttavia, come in precedenza, la versione Z80 sfrutta la potenza e la flessibilità dei suoi registri a 16 bit, mentre il 6502 elabora solo 8 bit alla volta. Il divisore si trova all'indirizzo con etichetta DIVSR, il dividendo in DVDND, il quoziente in QUOZ ed il resto in RESTO. I programmi in linguaggio Assembly Z80 e 6502 vengono mostrati nella tabella qui accanto.
Si noti che, in entrambe le versioni, quando la sottrazione del divisore dal dividendo parziale dà un risultato negativo, il dividendo deve essere riportato allo stato originale sommandogli nuovamente il divisore. La versione 6502 è interessante per il suo modo di gestire il registro di stato del processore: il flag di riporto viene "ruotato" nel quoziente, ma il suo contenuto va conservato per indicare il risultato della sottrazione. Ecco perché il registro PSR viene "appoggiato" sullo stack prima della rotazione e recuperato in seguito, in modo da ripristinare il valore del flag di riporto.
Abbiamo così esaminato le quattro operazioni aritmetiche: tutto ciò va ovviamente considerato un utile esercizio di programmazione per una miglior conoscenza delle elaborazioni in linguaggio macchina: non è necessario vagliare tutte le varie combinazioni dell'aritmetica a byte singolo e multiplo, dato che i programmatori hanno scritto per anni queste routine su testi e riviste. Quando, nelle successive parti del corso, occorrerà modificare le routine già sviluppate, esse verranno fornite o presentate sotto forma di esercizi.

DIVISIONE 16-BIT PER 8-BIT

Z80			6502		
START	LD	A,(DIVSR)	START	LDA	#$00
	LD	D,A		STA	QUOT
	LD	E,$00		STA	RMNDR
	LD	HL,(DVDND)		LDX	#$08
	LD	B,$08		LDA	DVDHI
LOOP0	AND	A		SEC	
	SBC	HL,DE		SBC	DIVSR
	INC	HL	LOOP0	PHP	
	JP	P,POSRES		ROL	QUOT
NEGRES	ADD	HL,DE		ASL	DVDLO
	DEC	HL		ROL	A
POSRES	ADD	HL,HL		PLP	
	DJNZ	LOOP0		BBC	CONT1
	LD	(QUOT),HL		SBC	DIVSR
	RET			JMP	CONT2
DIVSR	DB	$F9	CONT1	ADC	DIVSR
DVDND	DW	$FDE8	CONT2	DEX	
QUOT	DB	$00		BNE	LOOP0
RMNDR	DB	$00		BCS	EXIT
				ADC	DIVSR
				CLC	
			EXIT	ROL	QUOT
				STA	RMNDR
				RTS	
			DIVSR	DB	$F9
			DVDLO	DB	$E8
			DVDHI	DB	$FD
			QUOT	DB	$00
			RMNDR	DB	$00

Output su video

Fino ad ora abbiamo usato la memoria RAM e la CPU come un sistema di calcolo ed abbiamo lasciato i risultati ottenuti in un'area non precisata della RAM, per poterli in seguito esaminare con un programma di controllo. Questo modo di procedere è ovviamente insufficiente, ma del resto, fino a questo momento, non avevamo abbastanza elementi per progettare la visualizzazione in codice macchina.
Quasi tutti i micro possiedono una mappa di memoria per il video, ossia un'area di RAM destinata a contenere un'immagine del video. La trama dello schermo è costituita da punti,

o pixel, che possono essere accesi o spenti. È quindi possibile esprimere i pixel con i numeri binari 1 (acceso) e 0 (spento) e si può interpretare l'intero schermo come una "conversione in punti luminosi" dei bit che compongono i byte della RAM video. Sebbene il Micro BBC, lo Spectrum ed il Commodore 64 adottino questa tecnica di rilevamento (detta "mapping"), nessuno di essi la esegue in modo semplice. Per meglio capirne il funzionamento, dividiamo ciascuna riga del video in gruppi di otto pixel (un byte) e numeriamoli da sinistra a destra e dall'alto in basso. Per vari motivi questa non è la tecnica adottata sui tre apparecchi considerati. Esaminiamo ora ciascun caso separatamente.

Il video dello Spectrum è sempre in alta risoluzione e possiede una particolare area di memoria per la definizione del video. Questo procedimento è piuttosto complicato, in quanto lo schermo viene suddiviso orizzontalmente in tre blocchi di 8 righe e ciascuna riga viene a sua volta suddivisa in 8 righe pixel.

L'indirizzamento dei byte che compongono le righe pixel ha un andamento sequenziale all'interno di ciascuna riga, ma non tra una riga e l'altra. Il Micro BBC ed il Commodore 64 non seguono il medesimo schema, ma sono egualmente complessi. Per il momento, limiteremo il nostro studio all'invio sullo schermo dei caratteri ASCII.

Questa è una procedura di uso comune e la ROM dei micro contiene apposite routine in codice macchina per la sua esecuzione. Dopo una dettagliata descrizione del loro funzionamento, potremo chiamare queste routine dall'interno dei nostri programmi in codice macchina: dobbiamo però conoscere l'indirizzo di chiamata, i registri di trasmissione e tutte le altre operazioni preliminari necessarie.

Lo Spectrum non richiede alcun intervento preliminare ed il registro di trasmissione è l'accumulatore, il quale deve contenere il codice ASCII del carattere che si vuol visualizzare. L'istruzione RST $10 permette di far comparire sul video, alla corrente posizione del cursore, il carattere il cui codice ASCII si trova nell'accumulatore. Questa procedura è comune anche agli altri due micro, ma l'opcode RST (ReSTart = riparti) appartiene ai comandi Z80. Si tratta di un'istruzione di salto a singolo byte, sulla pagina zero, che richiede uno dei seguenti operandi: $00, $08, $10, $18 ecc. fino a $38. Ciascuna di queste locazioni punta all'indirizzo iniziale di una routine della ROM. Cosicché tali routine, dedicate alla gestione di input e output, vengono richiamate per mezzo dell'istruzione RST e non tramite indirizzo. Questa procedura viene usata sia per motivi di velocità (l'uso di RST è molto più rapido di CALL, anche se solo la CPU può accorgersi della differenza) sia per favorire la trasferibilità del programma. Se ogni programmatore Z80 sà che RST $10 chiama la routine PRINT su qualunque apparecchio Z80, non sarà necessario che egli conosca i diversi indirizzi di locazione della routine PRINT all'interno dei vari sistemi software.

Il Micro BBC adotta una procedura simile: il comando JSR $FFEE, abbinato ad un codice ASCII nell'accumulatore, invierà il carattere alla cella del video occupata dal cursore: questa è la routine OSWRCH, che viene descritta in modo esauriente nel Manuale del computer.

Sul Commodore 64 il procedimento è lo stesso: JSR $FFD2 è l'istruzione di stampa mentre CHKOUT è la routine che viene documentata nel manuale.

Abbiamo dunque chiarito gli aspetti generali dell'uso delle routine ROM ed i principi su cui si basano i registri di comunicazione. La comunicazione tra un programma di chiamata ed una subroutine può avvenire in entrambi i sensi: una routine di input, ad esempio, serve per prelevare un carattere da una periferica ed inviarlo alla CPU, passando attraverso l'accumulatore. Anche quando non viene "passata" un'informazione del tipo precedente, la subroutine può inviare, tramite uno dei registri, un codice d'errore. Questo tipo di protocollo è documentato in molti manuali tecnici di consultazione oggi disponibili.

Più avanti nel corso ci occuperemo dell'input proveniente dalla tastiera o da altre periferiche, e della programmazione in codice macchina di grafica ad alta risoluzione. Concludiamo questa fase del corso con un riepilogo dei vari aspetti del linguaggio Assembly e della programmazione in codice macchina.

Riepilogo

Abbiamo iniziato il corso esaminando in modo generico il codice macchina e cercando di dissipare alcuni radicati luoghi comuni sulla sua complessità e sulle difficoltà che si incontrano nell'usarlo. Abbiamo visto come una stessa sequenza di byte in RAM possa venire interpretata in momenti successivi come una stringa di caratteri ASCII, come una linea di programma BASIC, come una stringa di indirizzi a due byte e, infine, come una serie di istruzioni in codice macchina. Pochi minuti passati usando un programma di monitor per codice macchina bastano a convincere che una qualsiasi sequenza di byte può essere disassemblata ottenendo tre distinte serie di istruzioni, diverse tra loro ma tutte valide, a seconda che si inizi il disassemblaggio dal primo, dal secondo o dal terzo byte della sequenza originale. Nessuna struttura intrinseca al codice impedisce quest'operazione e la CPU stessa non è in grado di scoprire se sta eseguendo la sequenza originale o una sua versione "corrotta" o accidentalmente trasferita in memoria.

Abbiamo poi discusso l'organizzazione della memoria e le convenzioni d'indirizzamento. Per chiarirci le idee su questo argomento, abbiamo cominciato a studiare l'aritmetica binaria, aprendo nuovi orizzonti alle nostre conoscenze sulla CPU: nei processori a 8 bit, salvo per particolari operazioni, il nostro limite massimo era stato il byte (vale a dire la

gamma di numeri decimali compresi tra 0 e 255). La conoscenza dell'aritmetica binaria ci ha permesso di cogliere interamente i limiti del sistema decimale in rapporto al mondo del codice macchina. Approfondendo il concetto di "pagina di memoria", abbiamo visto che il formato delle pagine logiche deve essere una funzione della base numerica e che in un sistema binario il formato della pagina deve essere una potenza di due. Due alla ottava dà 256, il "numero magico" in un microprocessore a 8 bit.
Ma anche l'aritmetica binaria si è ben presto dimostrata poco pratica e troppo soggetta ad errori, per essere usata come sistema di numerazione. Siamo quindi passati a quella esadecimale (in base numerica 16), imparando come sia possibile rappresentare un byte a otto bit con solo due cifre esadecimali, da $00 a $FF, dove la prima cifra rappresenta lo stato dei quattro bit più bassi e la seconda lo stato dei quattro più alti.
Abbiamo analizzato minuziosamente la procedura di immagazzinamento dei programmi BASIC nell'area ad essi riservata. Descrivendo il procedimento relativo ai "token" come un'altra forma del codice macchina e abbiamo offerto un utile strumento di approfondimento sul sistema operativo. La trattazione dei marcatori di fine linea chiariva in che modo l'interprete BASIC riesca ad individuare la fine delle varie porzioni di codice, mentre l'indirizzamento di collegamento del Commodore ha introdotto le due convenzioni d'indirizzo lo-hi ed il concetto di indirizzamento indiretto.
A questo punto abbiamo affrontato più specificatamente il linguaggio Assembly. Partendo dalle operazioni elementari della CPU, dirette dagli op-code a otto bit che formano le istruzioni di programma, siamo rapidamente passati ai codici mnemonici del linguaggio Assembly. In questa fase abbiamo compreso che la programmazione, in codice macchina, in linguaggio Assembly o in BASIC, ha sempre la sua importanza, ma che la questione fondamentale è risolvere i problemi logici prima di dedicarsi alla stesura di un programma. Questo argomento ha costituito il nucleo centrale del corso.
Il corso ha poi affrontato il caricamento ed il funzionamento dei programmi in codice macchina, con particolare riferimento ai computer progettati per lavorare principalmente con programmi BASIC. Abbiamo esaminato le variabili di sistema ed i puntatori del sistema operativo su Micro BBC, Spectrum e Commodore 64, imparando a "rubare" spazio al BASIC.
Dopo aver dato un'occhiata all'architettura di piccoli sistemi di elaborazione e alle CPU Z80 e 6502, abbiamo cominciato a scrivere programmi in Assembly per la manipolazione della memoria e dell'accumulatore.
In questa fase si è parlato di pseudoistruzioni (o pseudo-op) Assembly, che ci hanno in parte allontanato dal codice macchina, dalle procedure di assemblaggio manuale e dalla difficoltà della programmazione di basso livello.
Il corso si è poi occupato del registro di stato del processore (PSR), usato come indicatore dei risultati delle operazioni effettuate dalla CPU. Questa sua fondamentale funzione, insieme a quella del flag di riporto nelle operazioni aritmetiche, sono state immediatamente chiarite in un'introduzione all'aritmetica binaria, utilizzando l'istruzione "somma con riporto".
Abbiamo poi sinteticamente esaminato i vari modi d'indirizzamento: l'indirizzamento ad indice ha richiesto una particolare attenzione per la sua importanza nel controllo di cicli iterativi, liste e tabelle. Queste strutture ci hanno fatto chiaramente comprendere la necessità di una serie di istruzioni che permettessero di dirigere il flusso di controllo all'interno di un programma; per questo motivo ci siamo interessati delle istruzioni di salto condizionale mentre ancora stavamo esplorando le potenzialità dell'indirizzamento indiretto e a indice. Le istruzioni di salto condizionale, l'aritmetica di base e le strutture a matrice costituiscono lo scheletro di un qualsiasi linguaggio di programmazione.
Abbiamo poi descritto la chiamata di subroutine del linguaggio Assembly e la procedura di ritorno, usando poi questi procedimenti per introdurre l'ultimo argomento inesplorato del sistema operativo: lo stack. Ne abbiamo approfondito il meccanismo di funzionamento e studiata l'utilizzazione per aggiungere alcune nuove operazioni al repertorio della programmazione in codice macchina.
Un'analisi più approfondita dei registri della CPU e delle loro interazioni ci ha fornito nuove possibilità di manipolazione della memoria e del microprocessore.
Infine, ci siamo dedicati all'aritmetica binaria. Abbiamo esaminato nei dettagli gli originali meccanismi delle sottrazione e del "complemento a due" e la complessità della moltiplicazione e della divisione.
In futuro cercheremo di scoprire le potenzialità pratiche della programmazione in codice macchina e gli specifici compiti che potremmo assegnare in primo luogo ai processori di cui ci siamo inizialmente interessati (Z80 e 6502), quindi ad altri processori come la CPU 6809 usata dal Dragon 32 e 64.

Risposte agli esercizi a pag. 779

1) La soluzione che permette la risposta più rapida è certamente una routine scritta appositamente per moltiplicandi a 16 bit, sulla falsariga della routine a 8 bit vista nella lezione precedente. D'altra parte, se suddividiamo la moltiplicazione a 16 bit in due moltiplicazioni a 8 bit (moltiplicando il moltiplicatore prima per il byte basso, poi per il byte alto), allora possiamo chiamare due volte la routine a 8 bit già esistente, tener conto del riporto del byte basso e immagazzinare i risultati nei byte del prodotto.

2) Una routine di moltiplicazione, che usi ripetute addizioni, si compone essenzialmente di un ciclo iterativo, il cui contatore è il moltiplicatore: tutte le volte che il ciclo viene eseguito, il moltiplicando viene sommato al prodotto.

ROR — ROTAZIONE A DESTRA — 6502

registro ‖ 6A (1 byte)

Il contenuto del byte nell'accumulatore viene ruotato di un bit, passando attraverso il bit di riporto (Carry)

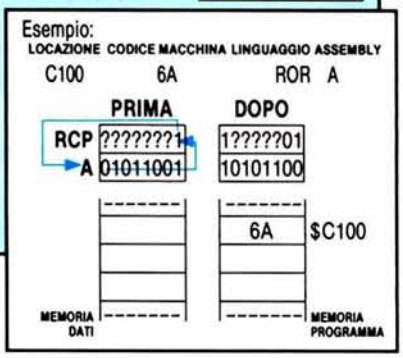
EFFETTI SU RCP

	S	V		B	D	I	Z	C
MSB	X						X	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
C100	6A	ROR A



RR — ROTAZIONE A DESTRA — Z80

registro ‖ CB 1F (2 byte)

Il contenuto del byte nell'accumulatore viene ruotato a destra di un bit, attraverso il bit di Carry

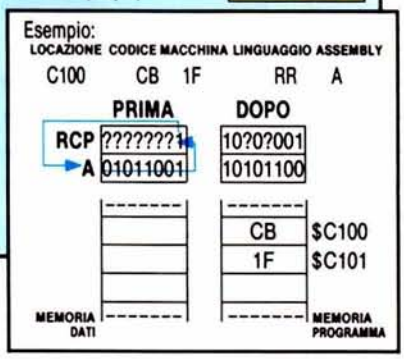
EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB	X	X		0		X	0	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
C100	CB 1F	RR A



SBC — SOTTRAZIONE CON PRESTITO — 6502

assoluto ‖ ED (3 byte)

Il contenuto della locazione di memoria è sottratto dall'accumulatore. Il bit di Carry indica la condizione di "prestito"

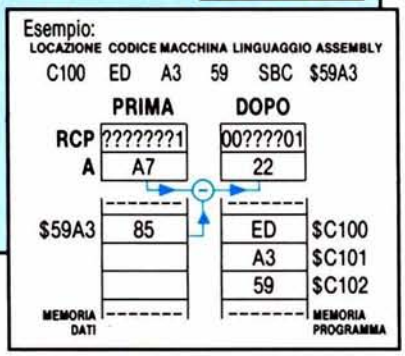
EFFETTI SUL RCP

	S	V		B	D	I	Z	C
MSB	X	X					X	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA			LINGUAGGIO ASSEMBLY	
C100	ED	A3	59	SBC	$59A3



SBC — SOTTRAZIONE CON PRESTITO — Z80

immediato ‖ DE (2 byte)

Dal contenuto dell'accumulatore viene sottratto il valore del byte che segue l'istruzione

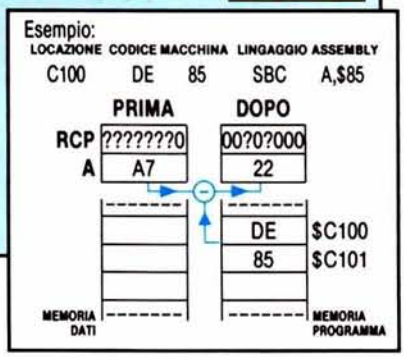
EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB	X	X		X		X	X	

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGUAGGIO ASSEMBLY	
C100	DE	85	SBC	A,$85



ASL — SCORRIMENTO ARITM. A SINISTRA — 6502

registro ‖ 0A (1 byte)

Il contenuto del byte viene fatto scorrere a sinistra, attraverso il Carry. Da destra viene inserito un valore 0 nel bit meno significativo

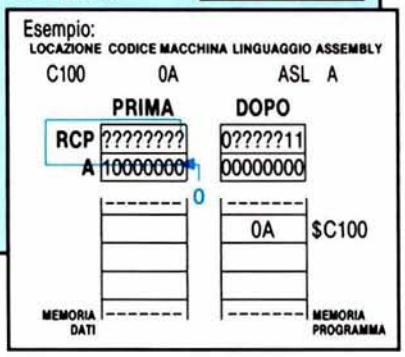
EFFETTI SU RCP

	S	V		B	D	I	Z	C
MSB	X						X	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA	LINGUAGGIO ASSEMBLY
C100	0A	ASL A



SLA — SCORRIMENTO ARITM. A SINISTRA — Z80

registro ‖ CB 23 (2 byte)

Il contenuto del registro E viene fatto scorrere a sinistra di un bit, attraverso il Carry. A destra viene immesso uno zero nel bit meno significativo

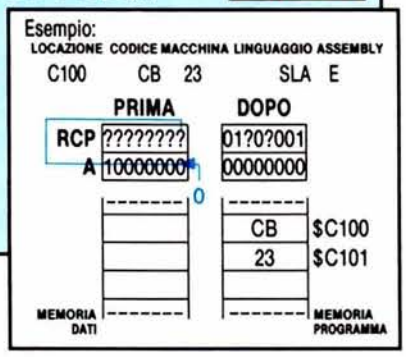
EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB	X	X		0		X	0	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGUAGGIO ASSEMBLY	
C100	CB	23	SLA	E



CMP — CONFRONTO CON L'ACCUMULATORE — 6502

assoluto ‖ CD (3 byte)

Il contenuto della locazione di memoria è confrontato a quello dall'accumulatore.

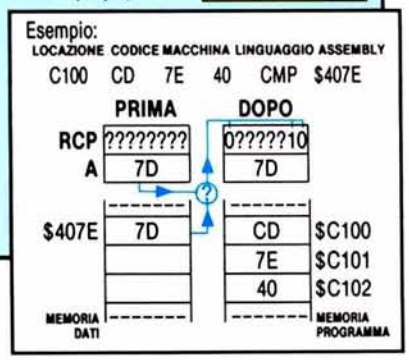
EFFETTI SUL RCP

	S	V		B	D	I	Z	C
MSB	X						X	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA			LINGUAGGIO ASSEMBLY	
C100	CD	7E	40	CMP	$407E



CP — CONFRONTO CON L'ACCUMULATORE — Z80

immediato ‖ FE (2 byte)

Il contenuto nell'accumulatore viene confrontato con il valore del byte che segue l'istruzione

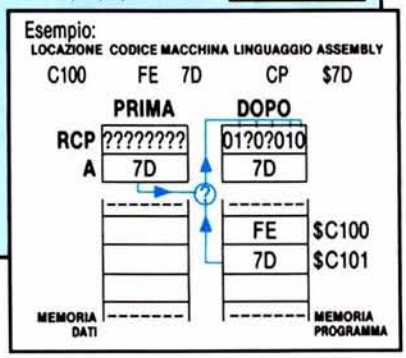
EFFETTI SUL RCP

	S	Z		H		V	N	C
MSB	X	X		X		X	1	X

LSB

Esempio:

LOCAZIONE	CODICE MACCHINA		LINGUAGGIO ASSEMBLY	
C100	FE	7D	CP	$7D



La Dragon

Il successo del Dragon 32 e 64, non ha evitato una precaria situazione economica

La Dragon Data è nata nel 1981, come società controllata dalla Mettoy, una grande fabbrica di giocattoli. Era intenzione della Mettoy trarre profitti dal boom degli home computer che proprio in quegli anni stava prendendo piede in Gran Bretagna. Grazie all'aiuto finanziario della Welsh Development Agency, fu impiantato uno stabilimento a Swansea, nel Galles, ed il Dragon 32 fece la sua prima comparsa nell'agosto del 1982.
La società preferì il 6809 della Motorola allo Z80 o al 6502 (i microprocessori abitualmente impiegati dalla maggior parte dei fabbricanti di home computer). L'hardware del Dragon seguiva attentamente l'impostazione consigliata dalla Motorola e, per questo motivo, la Data Dragon fu accusata di aver copiato il progetto base dal computer Color della Tandy, un altro apparecchio che adottava il formato della Motorola. Un effetto collaterale di questa somiglianza era che molto del software scritto per il "CoCo" (il nomignolo del "Color Computer") poteva essere usato senza problemi sul Dragon 32.
I punti di forza del Dragon erano il BASIC Microsoft (il dialetto BASIC più diffuso) e la tastiera tipo macchina da scrivere. Quando il Dragon 32 fu lanciato sul mercato, l'unica tastiera, a parità di prezzo, che poteva far concorrenza alla sua, era quella del Vic-20. La strategia di mercato della Dragon Data giocò un ruolo fondamentale nel successo dell'apparecchio; nei mesi precedenti al Natale del 1982, c'erano poche scorte tanto di Spectrum ZX quanto di Micro BBC (il Commodore 64 non era ancora entrato in gioco). Il Dragon 32 era invece disponibile in grandi quantità e, agli inizi dell'83, la Dragon aveva fatturato la vendita di ben 32.000 computer. Questo risultato era in parte dovuto ai legami con la Mettoy: alcune grosse catene di negozi, assidui clienti della fabbrica di giocattoli, furono ben felici di acquistare il nuovo computer.
Tuttavia, nell'estate del 1983, la Dragon Data subì un grave dissesto finanziario. Essendosi sviluppata proprio nel periodo in cui la Mettoy andava incontro al fallimento, si nutrivano seri dubbi sulla sua sopravvivenza. Tuttavia, la Dragon fu recuperata da un consorzio di società, guidato dalla Prutec, la sezione investimenti in "alte tecnologie" della potente compagnia assicurativa Prudential. Nel recupero furono impegnati circa 5 miliardi ed a capo della Dragon venne nominato Brian Moore, già amministratore della GEC. Questi cambiamenti permisero alla Data Dragon di superare i problemi economici in cui si dibatteva, di investire in un nuovo impianto di produzione situato a Port Talbot e di continuare la progettazione del Dragon 64 e di un'unità a dischetti.
Il Dragon 64 è dotato di 64K di RAM, di una tastiera perfezionata e di un'interfaccia seriale RS232. L'unità a dischetti utilizza minifloppy da 5 pollici e 1/4, mediante il sistema operativo Dragon DOS, adatto sia al modello 32 che al 64.
Ma, nel giugno del 1984, una nuova ombra ha oscurato tutti i progetti della Dragon: la Prutec e la Welsh Development Agency si sono rifiutate di rifinanziare le perdite e la ditta è stata così costretta a dichiarare fallimento.
I progetti per l'84, al momento della chiusura, prevedevano lo sviluppo di tre nuovi apparecchi e di altri prodotti, collegati alla nuova produzione. Uno dei micro in progetto avrebbe dovuto adottare l'MSX standard, da poco sviluppato (vedere a pag. 621).

La nascita di un Dragon

La Data Dragon si differenzia dalla maggior parte delle altre ditte del settore degli home computer perché costruisce in proprio le macchine: la Sinclair e la Acorn, viceversa, subappaltano la produzione dei loro apparecchi. Recentemente, la Dragon ha impiantato una nuova fabbrica a Port Talbot





Un "mago" del mercato

Richard Wadman, il direttore commerciale della Dragon Data, aveva già progettato la campagna di vendita dell'intera nuova gamma di computer Dragon quando la ditta è improvvisamente fallita

Una rete di computer

Lo scambio di informazioni tra computer non è riservato ai grandi e costosi sistemi



PAUL CHAVE

La Interface 1
I network non richiedono migliaia di cavi di connessione o apparecchiature costose. Uno dei sistemi a rete meno costosi è costituito da più Spectrum, collegati tra loro tramite l'Inteface 1. Ciò consente a ciascun apparecchio (detto "nodo" del network) di utilizzare unità Microdrive e stampanti in apparente simultaneità con gli altri

I "network" o "reti" sono particolari sistemi di intercomunicazione fra computer, grazie ai quali vengono messe in comune informazioni e apparecchiature. Tuttavia, ciascun computer possiede il proprio sistema operativo oppure i propri "protocolli" (procedure, regole di formattazione e così via) per comunicare con il mondo esterno. Per evitare problemi di compatibilità, le singole stazioni del network (dette anche "nodi") devono essere costituite da computer simili: tutti Spectrum oppure tutti Apple, per esempio.
Supponiamo che cinque persone desiderino collegare i propri computer ad un'unica stampante. Che cosa accadrebbe se due o più stazioni tentassero di utilizzare contemporaneamente la stampante oppure se, ad esempio, il nodo 3 desiderasse stampare un testo e, al tempo stesso, continuare a lavorare mentre la stampante è in funzione? Per risolvere questo genere di problemi occorre inserire un sesto computer nella rete (indicato col termine "print server"), dedicato esclusivamente alla gestione e allo smistamento delle operazioni di stampa. Il print server memorizza i testi da stampare, provenienti dai vari nodi, secondo un preciso ordine di priorità (generalmente, quello di accesso). Non appena termina l'invio del testo al "server", il relativo nodo può tranquillamente eseguire altri lavori.
In un network, la presenza di un apparecchio che svolga simili funzioni di "servitore" è essenziale, in quanto rappresenta l'unico canale di comunicazione tra i vari nodi. Se, oltre alla stampante, la rete prevede anche l'accesso ad un'unica memoria di massa, allora è necessario anche un "file server", che tenga sotto controllo il flusso delle informazioni tra i vari nodi della rete.
Il passo successivo è la creazione dei collegamenti elettrici tra i vari computer del network e ciò, in genere, viene realizzato mediante un impianto di speciali cavi coassiali. Esiste più di uno schema di collegamento tra le varie unità di una rete, compreso quello "ad anello" e "a stella", ma il principio rimane sempre lo stesso. Ci limiteremo ad esaminare, dunque, una configurazione di carattere generale. Il collegamento di ciascun nodo richiede una speciale interfaccia di rete, che può essere una semplice porta RS232 oppure una speciale scheda a circuiti stampati. Inoltre, la stazione servente deve essere dotata di una sufficiente quantità di memoria, per gestire senza inconvenienti il flusso di lavoro previsto e per controllare l'intera rete.
In generale, il corretto funzionamento di un

computer dipende molto dalla buona qualità del software adoperato, e ciò vale ancor più nei network. Innanzi tutto ci deve essere un "livello" software di gerarchia superiore al sistema operativo della macchina, che permetta il collegamento tra ciascun nodo ed il network stesso. Questo livello, chiamato "software di rete", mette la stazione servente in condizione di svolgere i compiti di file server e di print server, e di controllare il flusso delle informazioni all'interno della rete. Inoltre, per l'efficienza del network, il software di rete deve comunicare al computer di ciascun nodo tre informazioni molto importanti: il fatto di essere collegato all'intera rete, quello di esser connesso ad un server ed il numero delle stazioni che fanno parte del network. Infine, il software di rete assegna un "protocollo di intercomunicazione" ai computer. Ovviamente, prima che la rete sia in grado di funzionare, il software di gestione deve essere caricato ed avviato.

A questo punto ciascun nodo deve possedere uno o più programmi applicativi, che gli consentano di comunicare con la rete, tramite il software di sistema. Da quanto detto, è evidente che esistono programmi applicativi scritti appositamente per sfruttare le prestazioni di una rete. Supponiamo, ad esempio, che il nodo 1 del nostro network stia adoperando un word processor e che si appresti ad inviare un testo alla stampante, indipendentemente dagli altri nodi: in questo caso la sola modifica da apportare al programma di word processing riguarda l'introduzione dei protocolli di trasmissione adatti alla rete. La situazione si complica, invece, se ai nodi 2 e 4 occorre accedere ad una stessa serie di dati e, contemporaneamente, conoscere i risultati ottenuti dagli altri nodi. In simili casi, tanto l'hardware, quanto il software applicativo (sia esso un word processor, una lavagna elettronica, un database o anche un gioco) devono essere in grado di eseguire diverse operazioni simultaneamente. In altre parole, la CPU deve poter gestire più lavori nello stesso momento ed essere in grado di controllare allo stesso tempo le comunicazioni provenienti da almeno altre due CPU.

L'industria guida nel campo dei network privati di piccole dimensioni sembra essere la Sinclair Research: l'unità aggiuntiva "Interface 1", per lo Spectrum, già molto diffusa perché consente di adoperare i Microdrive, è dotata di un'interfaccia di rete. Il successo commerciale dell'Interface 1 potrebbe preludere ad un notevole aumento nella produzione di software in grado di sfruttare le interfacce di rete incorporate.

Anche il QL, sempre della Sinclair, è fornito di un'identica interfaccia di rete e dovrebbe quindi essere compatibile con la versione Spectrum. Tuttavia, nonostante la modica spesa di una rete di questo tipo, le possibili applicazioni per gli utenti di home computer appaiono piuttosto limitate, se si escludono i giochi.

Infatti, ci sono ancora numerosi problemi da risolvere, prima che i network per computer divengano una realtà quotidiana. Innanzi tutto, ci devono essere almeno due micro collegati tra loro. Secondariamente i due apparecchi devono essere piuttosto vicini, possibilmente nello stesso palazzo, per permettere un collegamento via cavo. Infine, l'installazione della rete ha senso solo se ci sono reali e frequenti necessità di scambio di informazioni tra i computer o se i vari utenti sono disposti a spartirsi spese e uso di attrezzature costose come stampanti o unità a dischetti.

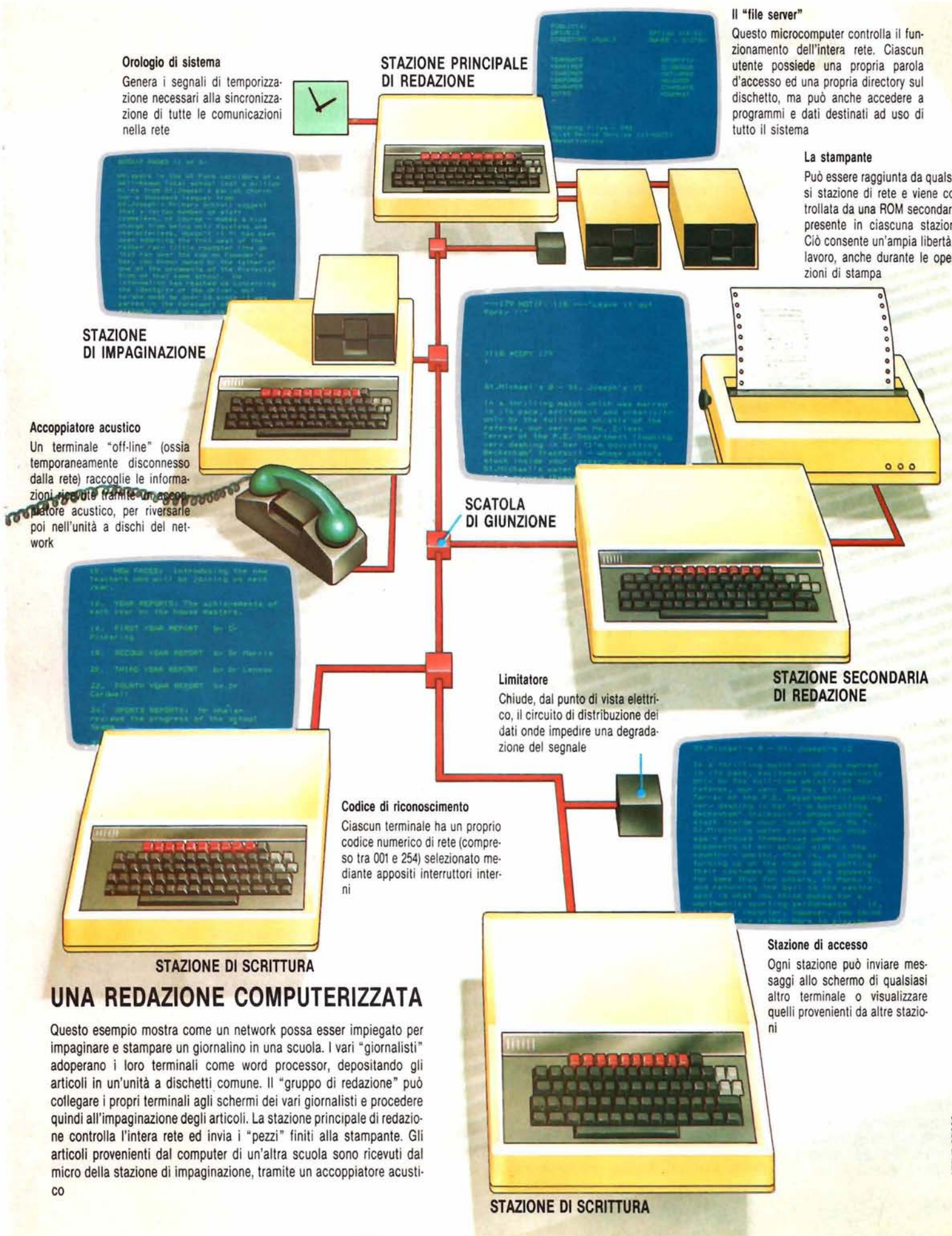
Tutto sommato, almeno per il momento, gli unici usi pratici di network per micro riguardano le piccole attività commerciali e le scuole. Il QL Sinclair rappresenta forse il primo passo verso la realizzazione e la gestione di piccoli network e, tra non molto, i computer collegati ad una piccola rete privata diverranno comuni quanto gli odierni micro ad uso domestico.



TONY SLEEP

Scambi di esperienze

Nel 1984, una scuola inglese è stata tra le prime, in Europa, a dotarsi di un "computer network", composto da 16 Micro BBC con monitor a colori, di una stampante, di una doppia unità a dischetti e di un Econet (il sistema a rete della Acorn per il Micro BBC). La spesa è stata sensibilmente inferiore a quella per l'acquisto di 16 computer indipendenti, ciascuno dotato di una propria stampante e unità a dischetti. La gestione della rete è talmente rapida, che ciascuna stazione di lavoro sembra essere l'unica fruitrice dell'unità a dischetti, anche quando 30 allievi lavorano simultaneamente. Le richieste d'accesso alla stampante vengono registrate in una speciale "lista d'attesa". L'Econet permette all'insegnante di seguire il lavoro di ciascun allievo, nel corso delle varie esperienze pratiche: la comunicazione interterminale si è dimostrata un valido strumento didattico per l'insegnamento di ogni materia



Il "file server"

Questo microcomputer controlla il funzionamento dell'intera rete. Ciascun utente possiede una propria parola d'accesso ed una propria directory sul dischetto, ma può anche accedere a programmi e dati destinati ad uso di tutto il sistema

Orologio di sistema

Genera i segnali di temporizzazione necessari alla sincronizzazione di tutte le comunicazioni nella rete

STAZIONE PRINCIPALE DI REDAZIONE

La stampante

Può essere raggiunta da qualsiasi stazione di rete e viene controllata da una ROM secondaria, presente in ciascuna stazione. Ciò consente un'ampia libertà di lavoro, anche durante le operazioni di stampa

STAZIONE DI IMPAGINAZIONE

Accoppiatore acustico

Un terminale "off-line" (ossia temporaneamente disconnesso dalla rete) raccoglie le informazioni ricevute tramite un accoppiatore acustico, per riversarle poi nell'unità a dischi del network

SCATOLA DI GIUNZIONE

STAZIONE SECONDARIA DI REDAZIONE

Limitatore

Chiude, dal punto di vista elettrico, il circuito di distribuzione dei dati onde impedire una degradazione del segnale

Codice di riconoscimento

Ciascun terminale ha un proprio codice numerico di rete (compreso tra 001 e 254) selezionato mediante appositi interruttori interni

STAZIONE DI SCRITTURA

Stazione di accesso

Ogni stazione può inviare messaggi allo schermo di qualsiasi altro terminale o visualizzare quelli provenienti da altre stazioni

STAZIONE DI SCRITTURA

UNA REDAZIONE COMPUTERIZZATA

Questo esempio mostra come un network possa esser impiegato per impaginare e stampare un giornalino in una scuola. I vari "giornalisti" adoperano i loro terminali come word processor, depositando gli articoli in un'unità a dischetti comune. Il "gruppo di redazione" può collegare i propri terminali agli schermi dei vari giornalisti e procedere quindi all'impaginazione degli articoli. La stazione principale di redazione controlla l'intera rete ed invia i "pezzi" finiti alla stampante. Gli articoli provenienti dal computer di un'altra scuola sono ricevuti dal micro della stazione di impaginazione, tramite un accoppiatore acustico

STEVE CROSS

Varietà di stampa

Anche le più economiche stampanti ad aghi offrono molte interessanti prestazioni

ELITE

```
ABCDEFGHIJKL
MNOPQRSTUVWX
YZabcdefghij
klmnopqrstuv
wxyz01234567
89 !"#$%&'()
*+,-./:;<=>?
@[\]^-`{|}~
```

PICA

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789 !"#$%&'
()*+,-./:;<=>?
@[\]^-`{|}~
```

PICA CORSIVO NERETTO

```
ABCDEFGHIJKLMN
OPQRSTUVWXYZab
cdefghijklmnop
qrstuvwxyz0123
456789 !"#$%&'
()*+,-./:;<=>?
@[\]^-`{|}~
```

ELITE ESPANSO

```
ABCDEFG
HIJKLMN
OPQRSTU
VWXYZab
cdefghi
jklmnop
qrstuvw
xyz0123
456789
! "#$%&'
() *+,-.
/:;<=>?
@[\]^-`
{|}~
```

PICA COMPRESSO

```
ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 !"#$%&
'()*+,-./:;<=>?@[\]^-`{
|}~
```

Varietà di caratteri

Alcune stampanti a matrice d'aghi dispongono di un'ampia gamma di caratteri tipografici (Pica, Elite, Italici) e di tipi di stampa (compressa, espansa, nerettata). Tutti gli esempi qui mostrati sono stati ottenuti su un'Epson FX-80

Le più recenti stampanti a matrice d'aghi sono capaci di produrre lavori più elaborati della semplice stampa di un listato di programma. Una rapida scorsa al manuale ci permette di scoprire un gran numero di "effetti speciali", facilmente ottenibili su carta. Anche le stampanti ad aghi più economiche consentono, oggi, di modificare le dimensioni dei caratteri stampati. Normalmente, in una riga di stampa ci sono 80 caratteri, ma tale valore può essere aumentato (selezionando il funzionamento "stampa compressa" che impiega caratteri più piccoli) oppure diminuito (selezionando la "stampa espansa"). Anche la spaziatura tra linee (la cosiddetta "interlinea") può essere variata, ottenendo stampati con righe più o meno fitte.
Un apparecchio dotato di svariate forme di stampa è la Epson FX-80, che offre una grande elasticità nel formato di stampa. Possiede, ad esempio, un "modo evidenziato", in cui i caratteri sono riprodotti "in grassetto" ed un "modo alternativo", in cui la forma dei caratteri è leggermente inclinata rispetto a quella normale. Queste sono soltanto due delle tante opzioni standard. Forse quella più interessante, tra le altre, è la possibilità di modificare la forma di qualsiasi carattere immagazzinato nella sua memoria. Ciò permette di stampare caratteri alfabetici stranieri o simboli scientifici. Prima di approfondire l'uso di questi effetti speciali, vediamo come viene prodotto un semplice listato di programma.
Le procedure di comunicazione, tra un computer ed una stampante, variano da macchina a macchina. Il Dragon, ad esempio, usa la variazione LLIST del comando LIST per ottenere la stampa di un listato di programma. Altri micro richiedono l'apertura di "canali" o "flussi" per accedere alla stampante. Quale sia il metodo da seguire è spiegato nel manuale del computer.
I primi tentativi di stampa, dopo aver collegato la stampante al computer, sono spesso soggetti ad inconvenienti di vario genere. Uno dei più comuni si ha quando la carta non avanza, ottenendo un'unica indecifrabile linea nera, oppure quando la spaziatura tra le righe è eccessiva. In entrambi i casi il problema deriva da un errato uso dei due caratteri di "interlinea" (line feed) e di "ritorno carrello" (carriage return). Infatti, dopo l'invio alla stampante di una linea di testo, il computer trasmette un carattere di ritorno carrello che, riportando la testina di stampa sul margine sinistro del foglio, prepara l'apparecchio per la stampa della linea successiva. Alcuni computer inviano anche un carattere di interlinea, che provoca lo scorrimento verso l'alto della carta. Altri, invece, si aspettano che tale operazione avvenga automaticamente. Inoltre, la situazione è complicata dalla presenza, sulla stampante, di piccoli interruttori interni per la selezione del tipo di interlinea. In questo caso sarà necessario consultare il manuale della stampante e posizionare gli interruttori nel modo più appropriato.
Volendo, una stampante può essere usata come dispositivo alternativo di visualizzazione, ossia per stampare direttamente i caratteri battuti sulla tastiera, senza che essi compaiano sullo schermo del computer. Anche in questo caso la procedura varia da computer a computer: generalmente in BASIC il comando è LPRINT (come sullo Spectrum o sull'Oric, che usano questo formato). Sul Commodore 64, l'istruzione OPEN 1,4, seguita da PRINT#1,"SALVE", permette la stampa della parola "SALVE". Il micro Dragon usa il comando PRINT#−2, "SALVE" mentre il Micro BBC impiega i comandi VDU2 seguito da PRINT "SALVE" e VDU3. I seguenti esempi di programma prevedono l'uso del comando LPRINT: andranno quindi modificati a seconda del micro usato.

Etichette per indirizzi

```
10 LPRINT "SIG. PAOLO BERTI"
20 LPRINT "VIA ROMA 12"
30 LPRINT "20132"
40 LPRINT "MILANO"
50 FOR I=1 TO 7
60 LPRINT
70 NEXT I
100 GOTO 10
```

Questo è un semplice programma per la stampa di etichette per indirizzi (le etichette vengono vendute in rotoli, coi margini perforati per facilitarne il trascinamento). Il programma, di applicazione generica in quanto non ricorre ad alcun particolare codice di controllo, stampa più volte lo stesso nome ed indirizzo, ma basta qualche piccola modifica per permettere l'inserimento da tastiera (o la lettura da un archivio) di nomi e di indirizzi diversi. Il ciclo FOR...NEXT (linee 50-70) serve per posizionare la testina di stampa in modo corretto su ciascuna etichetta: a seconda della stampante e del tipo di etichette usate, questo valore è soggetto a modifica.
Per operazioni di stampa più complesse, come la compilazione di una fattura o l'intestazione di una lettera, conviene usare alcuni degli effetti speciali precedentemente menzionati, per ottenere i quali occorre inviare alla stampante opportuni codici di controllo.

Il set di caratteri ASCII (vedere a pag. 557), oltre a possedere un codice per ogni carattere della tastiera, contiene un gruppo di codici "di controllo", invisibili sullo schermo, spesso usati proprio per abilitare gli effetti speciali sulla stampante. In genere, a tal fine, vengono riservati 4 codici (17, 18, 19 e 20), ma in una stampante quale la Epson FX-80, che possiede ben 70 opzioni di stampa, quattro codici non bastano. Per questa ragione molti effetti vengono abilitati inviando alla stampante opportune "sequenze di Escape", composte da due o più caratteri preceduti da un carattere ESC (codice ASCII 27). Ad esempio, per abilitare la spaziatura proporzionale, sull'Epson, occorre la sequenza ESC-p, cioè il carattere Escape seguito dalla "p" minuscola. In BASIC, ciò si ottiene con:

```
LPRINT CHR$(27);"p"
```

L'unico modo per inserire il codice ESC nell'istruzione di un programma è ricorrere alla funzione CHR$, in quanto la pressione del tasto ESC (o ESCAPE), generalmente, serve per altre operazioni. Sul Micro BBC, addirittura, viene usato il comando VDU.

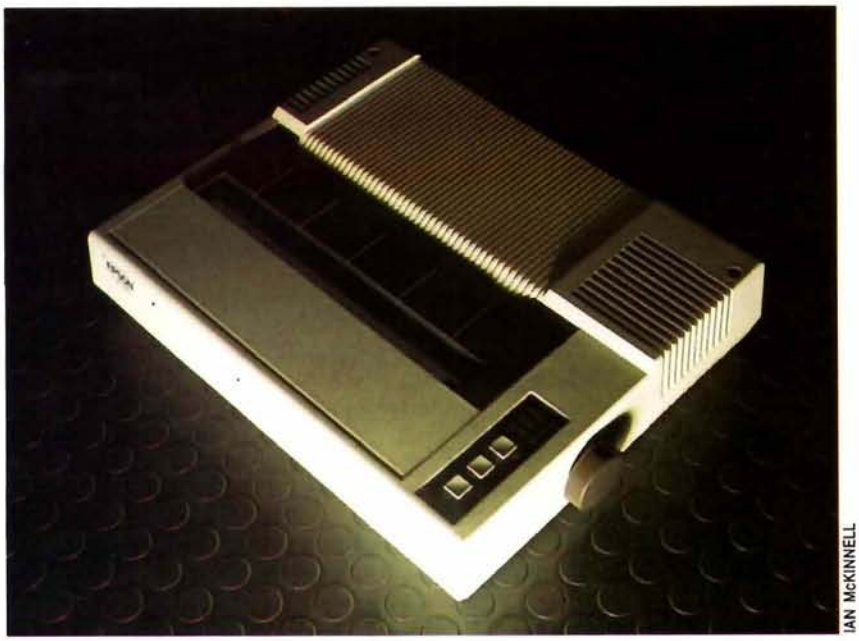
```
VDU2
VDU1,27,1,112
VDU3
```

Il comando VDU2 abilita l'uso della stampante VDU1 significa: "invia il seguente carattere alla sola stampante" (PRINT invierebbe il carattere ESC anche allo schermo, producendo risultati indesiderati). VDU3, infine, spegne (disabilita) la stampante.
Queste sequenze di comando sono tipiche dell'Epson FX-80: usandole con stampanti di diverso tipo possono non produrre alcun effetto, causare risposte inattese oppure provocare un arresto imprevisto (cioè la stampante si rifiuta di rispondere al computer).

Stampa di una fattura

Questo secondo listato mostra come usare alcune delle opzioni speciali per stampare l'intestazione di una fattura, emessa, nel caso dell'esempio, da una piccola autorimessa: anche questa volta sono stati usati i codici per l'Epson FX-80. La gamma di stampanti Epson è talmente popolare che molti fabbricanti hanno deciso di produrre modelli "Epson-compatibili". Se la stampante usata non appartiene a questa categoria, occorre consultare il manuale e sostituire opportunamente i codici di controllo qui riportati:

```
 999 REM INTESTAZIONE DI UNA FATTURA
1000 LPRINT CHR$(12)
1010 LPRINT CHR$(14); TAB(12); "AUTOFFICINA MONDIAL S.N.C."
1020 LPRINT CHR$(13); CHR$(13);
1030 LPRINT CHR$(27); "E";
1040 LPRINT TAB(36);
1050 LPRINT CHR$(27); "-"; CHR$(1);
1060 LPRINT "FATTURA"
1070 LPRINT CHR$(27); "-"; CHR$(0);
1080 LPRINT CHR$(27); "F";
1090 LPRINT CHR$(13); CHR$(13); CHR$(13)
1100 REM ESTREMI DELLA FATTURA
```



IAN McKINNELL

La Epson FX-80

Questa stampante è molto apprezzata dai possessori di micro domestici e gestionali, poiché costituisce un buon compromesso tra qualità e prezzo. Ha una testina a nove aghi ed una velocità massima di stampa di 160 cps. Molti programmi applicativi (ad esempio alcuni word processor ed alcuni programmi di grafica) sono predisposti per l'uso delle stampanti Epson (o quelle compatibili con esse)

La linea 1000 invia alla stampante il codice 12 di "pagina nuova" (form feed), che fa avanzare la carta fino all'inizio del foglio seguente. Il codice successivo, 14, è un carattere di controllo speciale, detto "shift out" (SO), che sull'Epson abilita il modo espanso: qui è usato per stampare l'intestazione della fattura. La funzione TAB serve per centrare l'intestazione. CHR$(13) è il carattere di ritorno carrello, che produce l'avanzamento di una linea: quelli usati nelle linee 1020 e 1090 creano una separazione tra l'intestazione ed il corpo della fattura. La sequenza ESC-E (linea 1030) abilita il modo evidenziato che ordina alla macchina di stampare in "neretto", ripassando più volte tutti i successivi caratteri del testo. La linea 1050 attiva l'opzione "sottolineatura", disattivata poi alla linea 1070, cosicché la parola "FATTURA" risulti sottolineata. ESC-F disabilita il modo evidenziato. Il risultato di tale programma è:

```
AUTOFFICINA MONDIAL S.N.C.

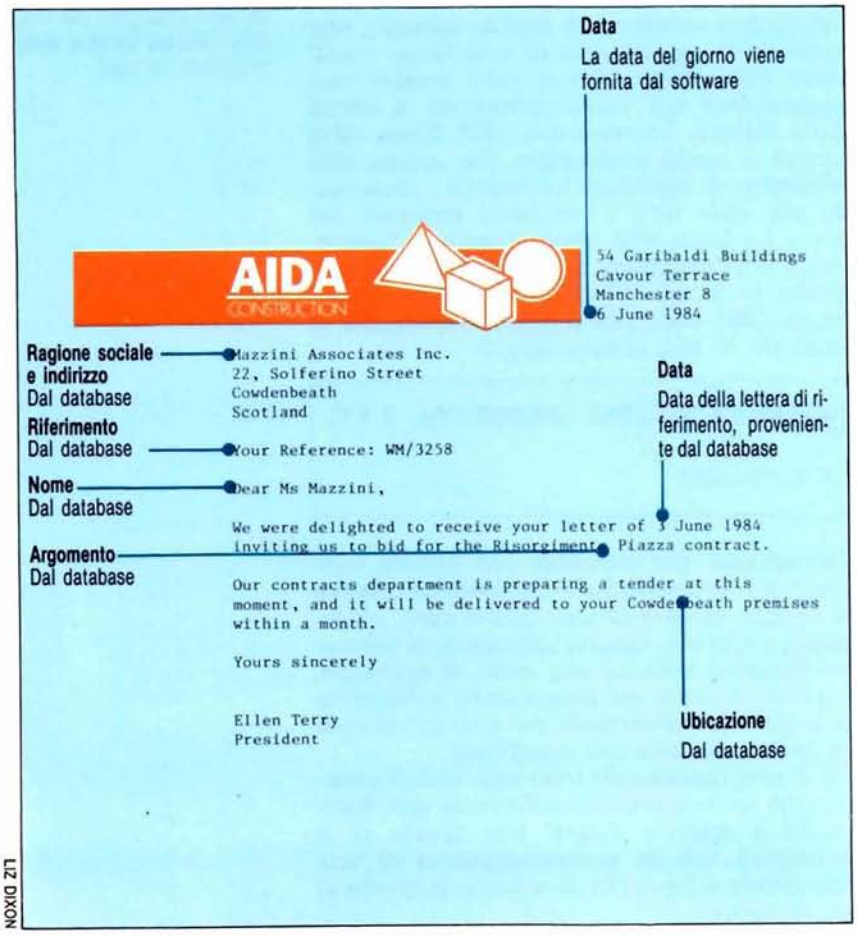
FATTURA
```

Ovviamente, per stampare una fattura completa occorrono anche le indicazioni relative al cliente, al modello dell'autovettura, all'importo, e così via. Queste informazioni verranno richieste tramite una serie di domande, inserite all'inizio del programma, e depositate in altrettante variabili, per poterle stampare nella posizione più opportuna.
I due programmi visti sono solo banali esempi delle molte possibilità offerte da una stampante a matrice d'aghi: ben presto ci si accorgerà che la programmazione di una stampante è forse più divertente di quella di un computer!

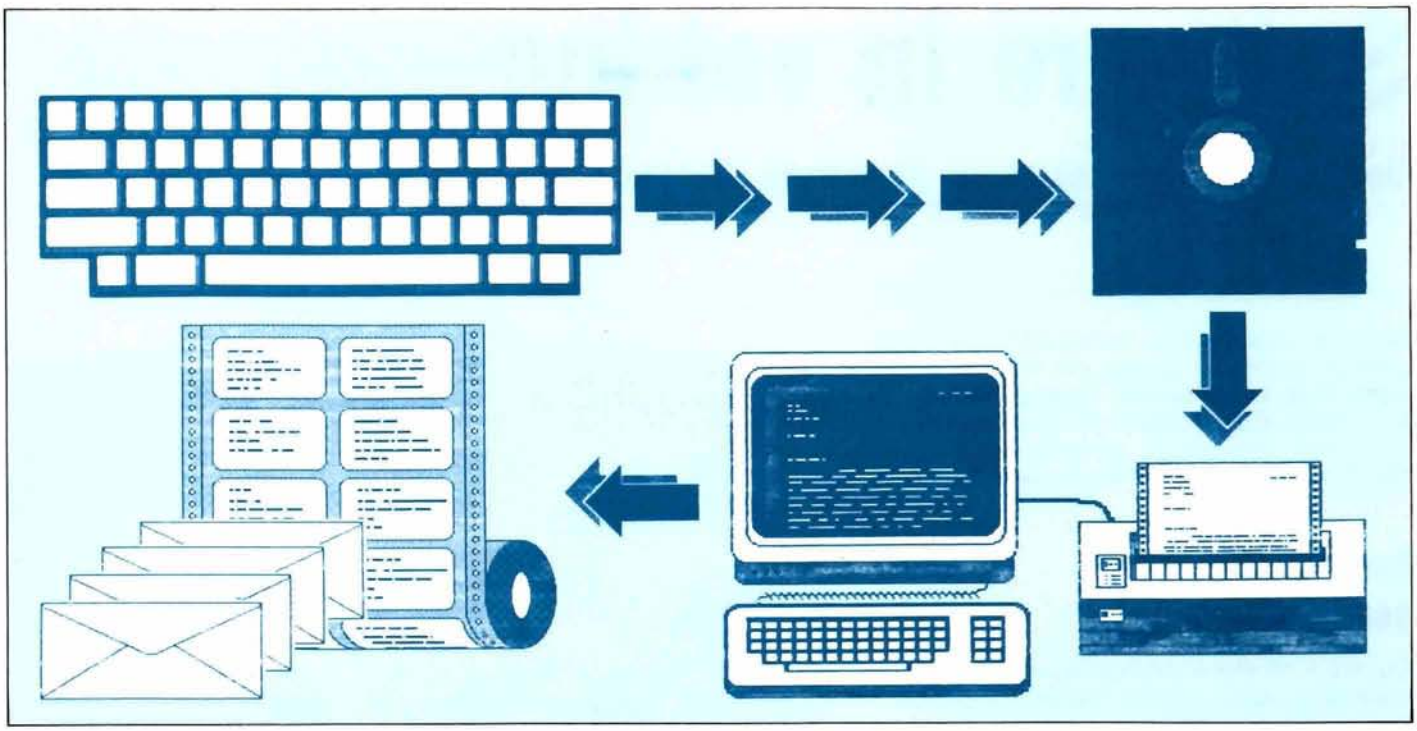
Corrispondenza automatizzata

Un buon word processor e un programma per la gestione di indirizzari facilitano la preparazione di lettere e circolari

Un sistema di posta "computerizzata" offre molto più di una semplice stampa di etichette per indirizzi. In effetti, se questo è l'unico obiettivo, si può anche fare a meno di un word processor: basta usare un comune database, che permette di creare e gestire con estrema facilità un semplice indirizzario.
Inizialmente, occorrerà registrare tutti i nominativi e gli indirizzi nei record del database, ma una volta portata a termine questa operazione, potremo utilizzare infinite volte l'archivio per stampare tutte le etichette necessarie.
Inoltre, siccome i programmi per la gestione di database possiedono, di solito, alcune sofisticate possibilità di selezione, sarà anche possibile estrarre i nominativi da stampare sulle etichette in base ad uno specifico criterio di nostra scelta, ad esempio: tutti gli indirizzi che contengono "ROMA" nel campo riservato alla città. Una simile procedura velocizza le operazioni di stampa delle etichette per le buste, ma ciò rappresenta soltanto metà del lavoro: se l'obiettivo finale consiste nell'invio di una "circolare" ad un particolare gruppo di clienti, rimane ancora da stampare su ogni lettera, sia il testo, sia l'indirizzo completo.
Del resto, anche preparando una lettera "tipo", viene stampato il nome e l'indirizzo di ciascun cliente, ma il corpo della lettera rimane tuttavia anonimo ed impersonale. Ha poco senso, infatti, lasciare spazi vuoti nel testo, da riempire successivamente con i nomi dei clienti: la loro lunghezza varia da caso a caso e sarebbe necessario lasciare uno spazio sufficiente per contenere il nome più lungo della serie.
Il risultato sarebbe una lettera "personalizzata a metà", con un aspetto né ordinato, né professionale. Inoltre, sarebbe necessario stampare tante lettere quanti sono i nomi dei clienti della lista.
La soluzione ideale a questo problema è organizzare il sistema in modo che l'utente possa redigere una lettera "tipo", servendosi di tutte le facilitazioni offerte da un word processor, lasciando poi al computer il compito di estrarre gli estremi di ogni cliente da un database, inserendoli opportunamente in ciascuna lettera.
Questo risultato può essere ottenuto in vari modi, uno dei quali è stato adottato nei due pacchetti software, prodotti dalla Acorn per il Micro BBC: il Memoplan e il Fileplan, rispettivamente un word processor ed un database. Entrambi fanno parte di una gamma di programmi offerta dalla Acorn agli acquirenti del secondo processore Z80 per il micro BBC Modello B.
Per ottenere lettere veramente personalizzate, i due pacchetti devono essere usati congiuntamente: i nomi e gli indirizzi vengono registrati col programma Fileplan, mentre il testo della lettera viene redatto usando il Memoplan. Ciascun campo del file di nomi e indirizzi è numerato, cosicché basta che l'utente digiti, nei punti della lettera standard desiderati, il numero del campo il cui contenuto deve essere inserito nella lettera. Il computer legge sequenzialmente la lista degli indirizzi, estraendo da essa quelli richiesti e copiandone il contenuto nel corpo della lettera.
Il word processor, poi, provvede automaticamente ad impaginare il testo che precede e che segue il dato inserito: il risultato finale è una lettera personalizzata, priva di inutili e



LIZ DIXON



IAN McKINNELL

Posta personalizzata
Col word processor si redige una lettera "standard", con spazi vuoti per la data, il nome e l'indirizzo. In seguito, il database vi inserisce i dati ricavati dall'archivio. Infine, vengono stampate sia le lettere che le etichette con gli indirizzi

fastidiose spaziature. Per facilitare la ricerca nel database, il Memoplan permette di specificare, assieme al numero di codice, anche una nota di richiamo. Ad esempio: 2(COGNOME) indica che il secondo campo contiene i cognomi dei vari clienti.
Alcuni pacchetti applicativi professionali, quali il Mailmerge della MicroPro (uno dei programmi di posta computerizzata di maggior successo) ed il Mailing List Manager della Peachtree, adatti a computer gestionali come l'IBM PC ed il Sirius, possiedono caratteristiche di notevole interesse e utilità. Non solo questi programmi permettono di creare un gran numero di archivi per indirizzi, ma anche di selezionare ed estrarre, in base a più criteri, particolari gruppi di nomi da un elenco generale.
La ricerca e la selezione vengono realizzate mediante sofisticate tecniche di codifica ed indicizzazione, tipiche nella gestione dei database. Inoltre, sui campi selezionati, vengono operate delle verifiche logiche, per vedere se essi corrispondono al criterio di selezione specificato dall'utente.
Questi pacchetti possiedono anche dettagliate facilitazioni di formattazione, che permettono di dimensionare gli indirizzi in modo da posizionarli convenientemente sulle etichette adesive. Nel sistema della Peachtree, ad esempio, l'opzione LABEL FORMAT visualizza sullo schermo una griglia formata dai campi che compongono ciascun record della lista: ciò permette all'utente di avere una rappresentazione visiva dell'etichetta di indirizzi che sta creando.
Esiste anche la possibilità di comunicare alla stampante quante sono le etichette affiancate orizzontalmente su ogni foglio del supporto, in modo da stampare, ad esempio, tre etichette alla volta.

Generalmente, i vantaggi offerti dai programmi di posta computerizzata si apprezzano quando i nominativi sono molto numerosi: in caso contrario può essere addirittura più rapido il classico sistema manuale, specialmente se il programma non prevede l'impiego di dischetti, ma di nastri.
Un buon pacchetto software per home computer è quello progettato per il Micro BBC dalla GCC, una software house di Cambridge (GB). La GCC ha prodotto per qualche tempo un database su ROM, chiamato "Starbase", dotato di alcune funzioni elementari per la gestione di indirizzari. La seconda versione di Starbase è stata arricchita con tutta una serie di facilitazioni, da usare congiuntamente a "Wordwise" un word processor, anch'esso su ROM. Assieme allo Starbase, è fornito un dischetto contenente alcuni programmi di utilità. Uno di questi consente l'invio di comandi alla stampante, durante la stampa delle etichette, in modo da selezionare opportunamente (se la stampante lo prevede) le dimensioni dei caratteri di stampa. È anche possibile eseguire semplici operazioni aritmetiche sui contenuti dei campi di un archivio (ciò torna molto utile per stampare, ad esempio, gli "estratti conto" dei soli clienti "in dare").
I possessori del Commodore 64 hanno un'ampia scelta: c'è, ad esempio, il Visawrite che, al contrario dello Starbase, lavora tanto su cassetta che su dischetto e può essere abbinato a qualsiasi database che usi file sequenziali. Usato da solo, comunque, può contenere fino a 500 nomi ed indirizzi.
Il Visawrite non possiede opzioni di ricerca selettiva, ma permette agli utenti di scorrere la lista degli indirizzi e di contrassegnare quelli che dovranno, in seguito, essere stampati sulle lettere.

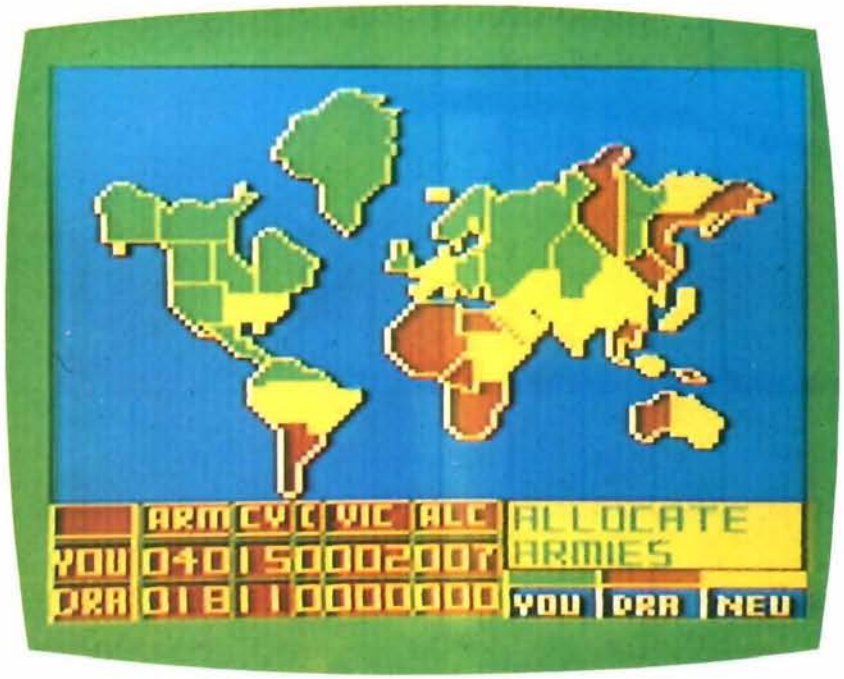
Software in vetrina

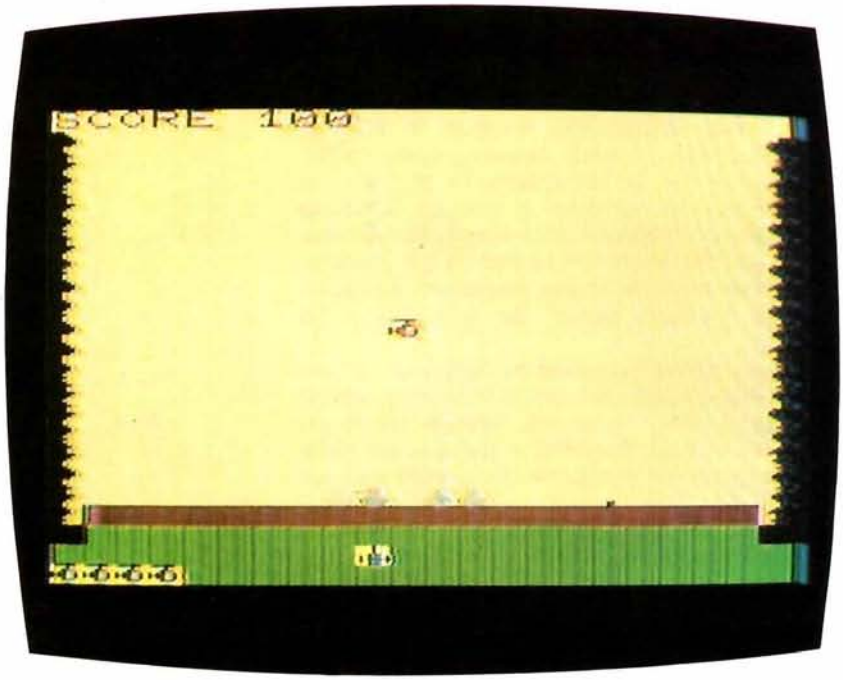
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Stavolta esaminiamo brevemente due programmi piuttosto diversi tra loro: uno che appartiene al genere dei "war games" e l'altro, invece, al più classico dei giochi "arcade". Entrambi, comunque, garantiscono ore ed ore di divertente intrattenimento.

Empire, per il Dragon

Questo è un gioco strategico, la cui impostazione assomiglia al noto "Risiko". Per prima cosa, il giocatore da una parte, ed il computer dall'altra, devono stabilire la posizione di 26 battaglioni nelle diverse parti del mondo. La scelta del computer avviene sempre dopo quella del giocatore e ciò, almeno in teoria, dovrebbe dare un piccolo margine di vantaggio a quest'ultimo.
Le varie manovre di guerra si seguono piuttosto bene sullo schermo ed il gioco mette a dura prova le doti tattiche e strategiche del giocatore. Il programma è prodotto dalla Shards Software.





Chopper, per il Vic-20

In questo gioco il giocatore pilota un elicottero, dal quale sgancia bombe nel tentativo di far saltare i carri nemici, durante il loro passaggio sulla strada sottostante. Un cannone contraereo, situato in una "fossa" protetta dalle bombe, tenta di abbattere l'elicottero, ma il pericolo maggiore, in realtà, è costituito dai missili lanciati dai carri nemici. Le prime fasi del gioco sono abbastanza semplici, ma con l'andare del tempo l'azione diviene sempre più ardua, stimolando notevolmente le reazioni del giocatore. Il programma è prodotto dalla Sumlock.

TV contro monitor

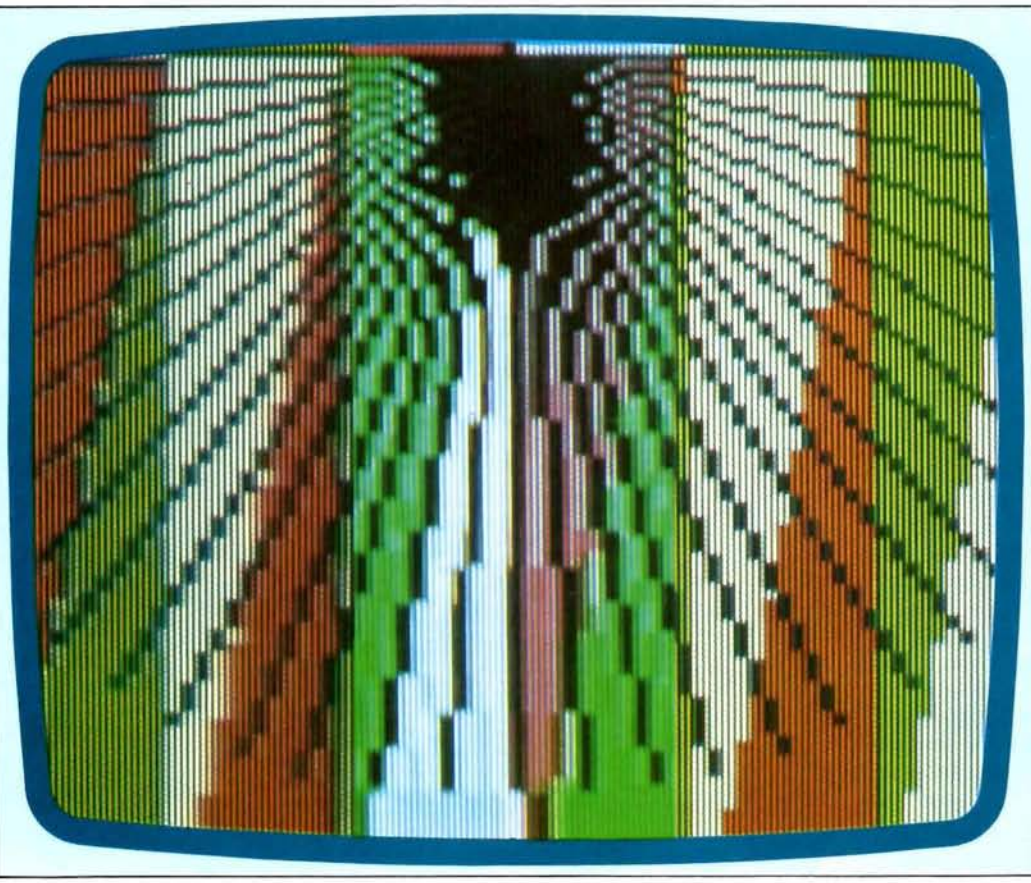
Televisore, monitor o una combinazione dei due?

Molti possessori di home computer collegano il proprio apparecchio ad un normale televisore ottenendo, quasi sempre, immagini instabili ed una colorazione piuttosto indistinta. Quando si acquista un monitor, però, la prima cosa che colpisce è la perfetta definizione delle immagini: i colori appaiono netti e decisi, l'intero schermo risulta più stabile ed è del tutto assente quel fastidioso effetto di "formicolio" tipico delle immagini ottenute sui comuni apparecchi televisivi. Purtroppo, i monitor di tipo classico hanno un costo piuttosto elevato e, per giunta, non servono per ricevere i programmi televisivi.

Tuttavia, vengono oggi prodotti apparecchi, detti "monitor/televisori", che assommano in sé il meglio di entrambe le macchine. Possono funzionare come comuni televisori e, grazie ad una presa addizionale, si trasformano in monitor quando vengono collegati ad un microcomputer.

Molti utenti possiedono già, pur senza saperlo, apparecchi di questo tipo perché molti dei più recenti apparecchi televisivi sono dotati di prese per videoregistratore adatte anche ad un computer.

Il maggior problema, nell'uso di apparecchi TV come video per un micro, deriva dal macchinoso sistema per il trasferimento dei segnali. I programmi televisivi vengono trasmessi sotto forma di onde radio che, captate dall'antenna, vengono convertite in immagini. Gli home computer imitano questo processo ed inviano i segnali video al televisore tramite un modulatore di radio frequenza (normalmente incorporato nel computer), collegato alla presa d'antenna del televisore. In sostanza, il modulatore converte gli impulsi digitali in un segnale tipo "onda radio" e l'apparecchio televisivo ricevente ritrasforma nuovamente il segnale per produrre le immagini. Più un circuito elettrico è complesso, più sono le occasioni in cui i segnali possono essere danneggiati da impulsi estranei al segnale originario: coi monitor i risultati sono migliori, poiché il segnale passa direttamente da un apparecchio all'altro, senza l'impiego di un modulatore.



IAN McKINNELL

TRE LIVELLI

I principali tipi di segnali video emessi dai computer sono tre. Tutti gli home computer producono segnali adatti ad un televisore, ma spesso la loro qualità è scadente. Molti micro, più costosi, generano anche segnali per i monitor e le immagini risultano migliori. I monitor/TV assommano le qualità visive dei monitor alla possibilità di ricevere trasmissioni televisive. Questi apparecchi si differenziano anche per il tipo di segnale elaborato. Le tre sezioni di schermo in figura sono state prodotte dallo stesso monitor/TV usando tre diversi tipi di segnale: televisivo (detto anche RF, o "Radio Frequenza"), video composito e RGB. Il segnale televisivo (al centro) dà i risultati più scadenti; quello video composito (a sinistra) è leggermente migliore. Infine il segnale RGB (a destra) dà immagini di qualità superiore

Non esiste un solo tipo di monitor: esso va scelto tenendo conto del computer al quale andrà collegato. I tipi di segnale per monitor più comuni sono due: l'RGB (Red-Green-Blue, ossia Rosso-Verde-Blu) e il "composito". I risultati migliori si ottengono col primo sistema, ma entrambi superano ampiamente le prestazioni dei normali televisori.

Anche gli apparecchi monitor/televisore sono di due tipi: nel primo si tratta di comuni ricevitori televisivi, modificati per "leggere" un segnale monitor; nel secondo gli apparecchi sono appositamente progettati per essere abbinati ad un videoregistratore. I modelli del secondo tipo sono da preferire, in quanto lavorano con segnali video "compositi" e possiedono una presa etichettata come "video" o "audio-visual". La figura della pagina seguente presenta i vari tipi di connettori usati per collegare un computer ad una presa per segnali video compositi. Per sintonizzare il televisore sul segnale del micro, si procede come nella ricerca di un normale canale televisivo.

I normali monitor, in genere, non prevedono circuiti audio, mentre i monitor/televisori sono dotati di amplificatori ed altoparlanti decisamente utili, poiché molti computer, soprattutto i modelli Atari, Commodore e Dragon, hanno ottimi generatori audio incorporati, ma la riproduzione del suono avviene tramite i circuiti dell'apparecchio televisivo. Se il computer prevede soltanto un'uscita RGB, la scelta è limitata. I più noti monitor/televisori con ingressi RGB sono tre: il sistema Sony Profeel, i televisori con connettori Peri-TV (ottima la gamma Normende) ed i modelli ITT. Il Sony Profeel accetta tanto segnali video compositi quanto RGB, ma non possiede connettori standard. Gli apparecchi ITT, invece, hanno un connettore RGB perfettamente compatibile con quello dell'Oric e dell'Atmos e possono essere collegati anche ad altri computer con uscita RGB.

Il Normende è il più usato dai possessori di home computer, in quanto è fornito di una presa Peri-TV, oramai divenuta uno standard internazionale, che è in grado di accettare segnali RGB e video compositi.

Anche altri apparecchi possono venire dotati di un connettore a innesto Peri-TV (chiamato anche "Scart") ma, in alcuni casi, il passaggio da televisore a monitor e viceversa si ottiene infilando o sfilando la presa: un sistema leggermente più scomodo rispetto al semplice cambio di sintonia. Il sistema Peri-TV è compatibile con entrambi i tipi di segnale, quindi l'acquisto di un nuovo computer non comporta anche quello di un nuovo monitor/televisore.

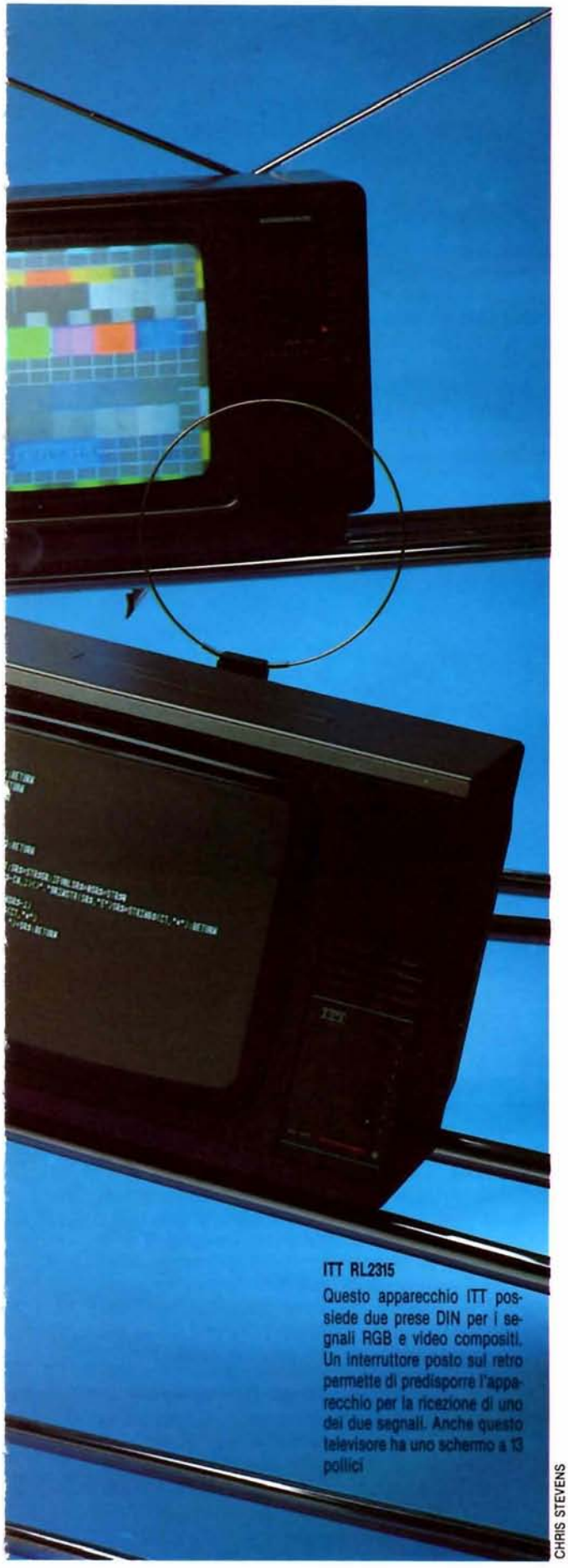
L'attuale tendenza, nel settore elettronico, è quella di fornire apparecchi "multiuso" ed il vecchio televisore non rappresenta un'eccezione: oltre alla ricezione dei programmi televisivi e come terminale per i sistemi Videotel e Televideo, oggi serve da monitor per computer. Domani?



Normende 1534
È uno dei principali televisori della gamma Normende. Tutti gli apparecchi di questa marca sono dotati di connettori a innesto Peri-TV e possono ricevere segnali RGB e video compositi. La Normende produce 7 modelli con diverse dimensioni di schermo, alcuni dei quali sono dotati di telecomando. Il televisore in figura è a 33 pollici, viene fabbricato a Singapore ed è predisposto per il telecomando

Ferguson TX con RGB
Questo monitor/TV, fabbricato in Gran Bretagna, riceve entrambi i tipi di segnale attraverso due prese DIN. I pulsanti sulla destra dello schermo a 13 pollici (33 cm.), selezionano tre diversi modi video che permettono di usare l'apparecchio come un normale televisore oppure come monitor collegato ad un computer

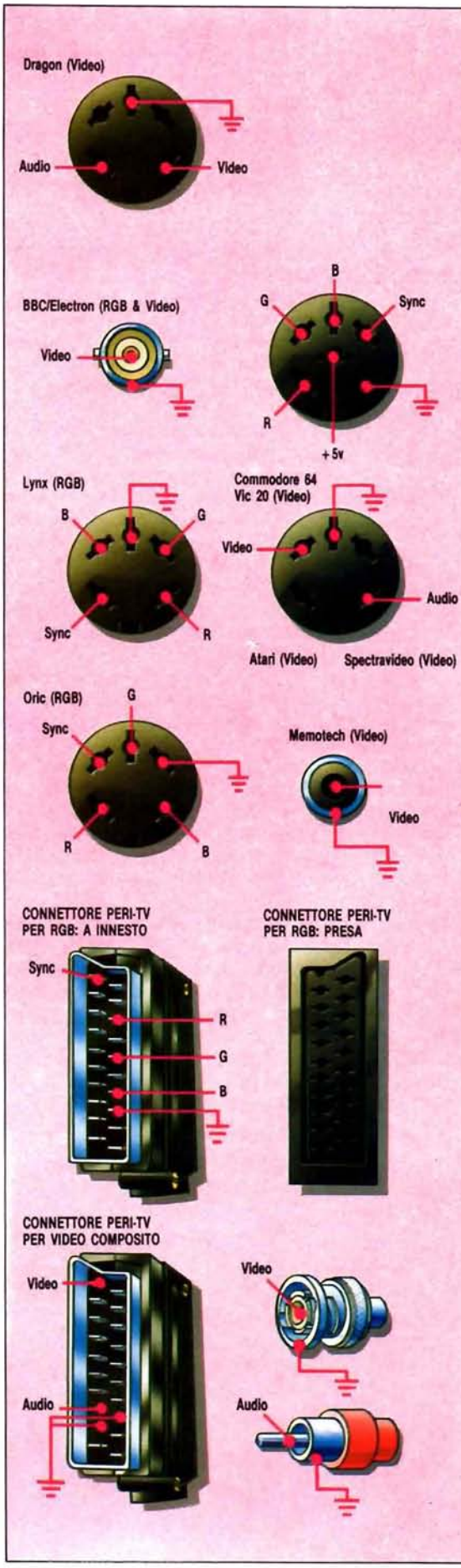
Fidelity CM14
Viene fabbricato in Gran Bretagna ed è disponibile in due modelli: uno di essi serve esclusivamente da monitor e possiede un connettore Peri-TV per segnali RGB e video compositi. Ha uno schermo a 13 pollici



ITT RL2315

Questo apparecchio ITT possiede due prese DIN per i segnali RGB e video compositi. Un interruttore posto sul retro permette di predisporre l'apparecchio per la ricezione di uno dei due segnali. Anche questo televisore ha uno schermo a 13 pollici

CHRIS STEVENS



KEVIN JONES

SPINOTTI DI COLLEGAMENTO

Ci sono due diversi tipi di segnali monitor: video compositi e quelli RGB. I segnali RGB sono composti da quattro diversi impulsi: per il rosso, il verde ed il blu, più un segnale di sincronizzazione. Gli ingressi e le uscite dei monitor RGB sono costituiti da prese a più piedini (di solito del tipo DIN). Il segnale di video composito è invece unico, formato dalla combinazione dei segnali di tutti i colori più quello di sincronizzazione. In genere i connettori adoperati per questo segnale sono a innesto (tipo baionetta) "phono" e NBC, tuttavia alcuni computer inviano anche il segnale audio su questi connettori.

Le varie figure mostrano i collegamenti necessari ai vari tipi di home computer. Se il monitor/TV presenta una delle configurazioni mostrate, basta prepararsi un cavetto seguendo le indicazioni qui riportate.

Il collegamento con monitor/TV dotati di prese Peri-TV prevede, in genere, che la selezione tra monitor e TV avvenga per mezzo dello stesso spinotto e, a tal scopo, è necessario un segnale a 5 volt, proveniente dal computer.

Lo Spectrum e lo ZX81 della Sinclair non vengono qui menzionati, in quanto sprovvisti di interfaccia per monitor, anche se si possono modificare affinché producano un segnale video composito.

Il Commodore Vic-20, l'Atari, lo Spectravideo e le prime versioni del Commodore 64 usano tutti lo stesso segnale di input per il video composito. Le più recenti versioni del Commodore 64, tuttavia, adottano un connettore ad innesto DIN a 8 piedini: il piedino due va collegato a massa, il tre porta il segnale di output del suono ed il quattro quello per il video

Un mare di onde

Come produrre affascinanti immagini tridimensionali sul computer

In passato, lo studio della trigonometria (seni, coseni, tangenti, ecc.), per non parlare dei grafici tridimensionali, ha terrorizzato più di uno studente. Ma oggi, grazie ai computer, questi argomenti possono diventare addirittura divertenti, anche per chi ha poca dimestichezza con i principi matematici di base.

Le possibilità grafiche dei micro fanno di questi apparecchi uno strumento ideale per la visualizzazione e la comprensione di grafici e di funzioni, così poco chiare quando vengono espresse come equazioni algebriche. Inoltre, la loro rappresentazione grafica sullo schermo produce immagini di notevole bellezza: anche coloro che non amano le materie scientifiche, saranno tentati di sviluppare le proprie equazioni, dopo aver visto quali risultati si possono ottenere.

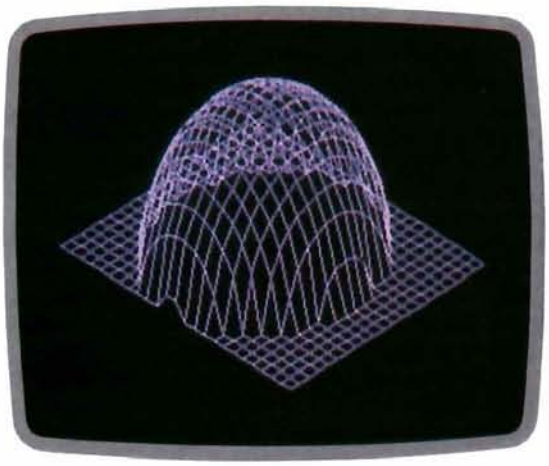
Le immagini di queste pagine, tutte create usando i listati riportati più avanti, rappresentano grafici tridimensionali. Un grafico 3-D è costituito da più grafici o curve bidimensionali, leggermente diverse tra loro, visualizzate contemporaneamente. Poiché un computer può produrre solo immagini a due dimensioni, il risultato che si ottiene non è tridimensionale nel vero senso della parola, ciononostante offre una certa illusione di profondità.

I listati di programma qui mostrati calcolano i valori di un'equazione a due variabili, X e Z. Il risultato, Y, viene ripetutamente calcolato per valori diversi di X e Z. Ciascun valore di Y è usato per tracciare un punto sullo schermo, con valori di Y corrispondenti ai punti sull'asse delle ordinate (verticale): il valore massimo di Y, quindi, corrisponde al punto più vicino al margine superiore dello schermo. I punti ottenuti vengono poi congiunti in successione con brevi linee rette in modo da formare una curva. Le curve in una direzione rappresentano i grafici di X e Y, con Z costante, mentre le curve che le intersecano sono i grafici di Y e Z, con X costante (in tal caso queste ultime vengono tracciate su un piano con assi Y e Z posto a 90 gradi rispetto al piano X-Y di una normale curva bidimensionale). Le immagini che si ottengono sono di grande aiuto nello studio di funzioni complesse.

Può essere divertente (ma anche piuttosto difficile) cercare di risalire all'equazione che ha prodotto una particolare curva. Per modificare l'immagine visualizzata, occorre cambiare la funzione contenuta nella linea 170 del programma BASIC: alcune equazioni possono essere piuttosto complicate ed il loro sviluppo richiederà più di una linea di programma. In questi casi si potranno usare i numeri di linea compresi tra 151 e 179.

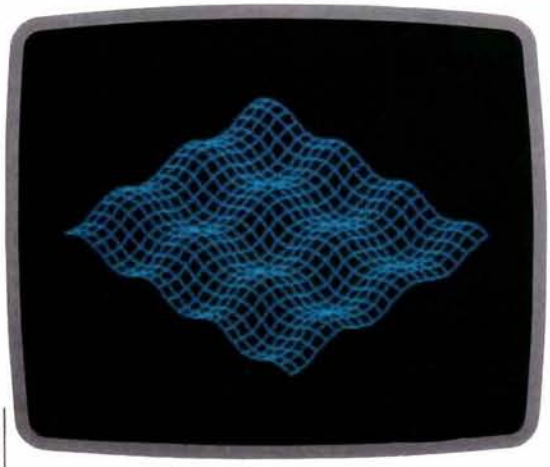
Oltre a scegliere una funzione che produca una curva gradevole, è bene assicurarsi che i valori ottenuti non generino immagini più grandi della superficie dello schermo. Per ottenere grafici di dimensioni contenute, può esser necessario dividere per un numero convenientemente grande i valori ricavati dalla funzione.

Il programma presentato può essere adattato a diversi home computer. Per favorire le varie modifiche, il programma è suddiviso in sezioni. La prima parte serve per preparare lo schermo: ciò significa che un'equazione che funziona su un particolare apparecchio dovrebbe funzionare anche su altri. La seconda parte del programma serve ad immagazzinare le coordinate dei punti che in seguito



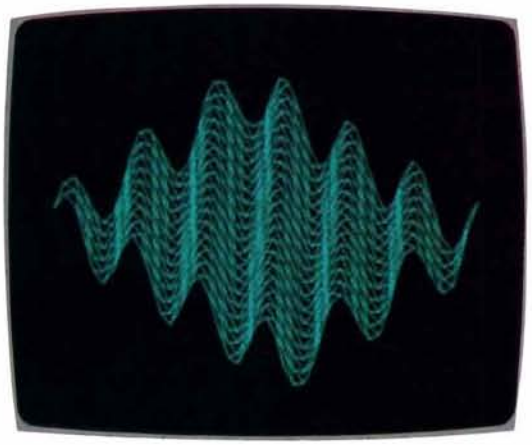
```
165 C=60-X*X-Z*Z
170 Y=SQR(C*(SGN(C)+1))/45
```

Immagini 3-D
Adottando nel programma le linee riportate sotto le illustrazioni, si otterranno queste figure 3-D



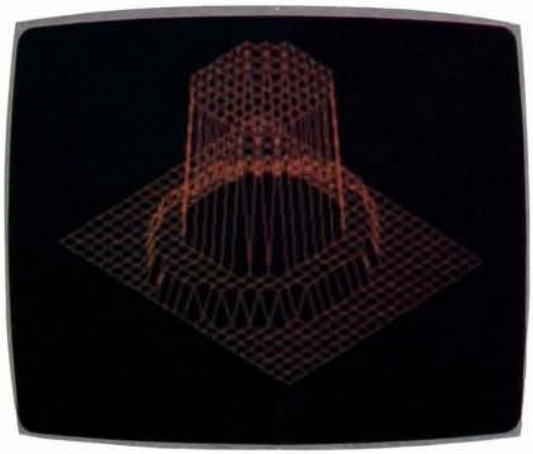
LIZ HEANEY

```
170 Y=(SIN(X)+COS(Z))/60
```



170 Y = SIN(X + Z)/12

comporranno il grafico. Questi risultati vengono depositati in una matrice ed il loro calcolo richiede un certo tempo. La procedura di calcolo dipende dalla funzione scelta e può durare anche alcuni minuti: durante questo intervallo di tempo si ha l'impressione che il computer non faccia niente. Ecco perché conviene eseguire il calcolo della funzione, memorizzandone i risultati, piuttosto che farla disegnare via via: l'immagine appare per intero, con un effetto molto più incisivo. Le illustrazioni mostrano le immagini che si ottengono usando le funzioni indicate sotto ciascuna figura, ma, dopo averle provate, è certamente più divertente creare nuovi grafici inserendo nel programma funzioni del tutto diverse. Durante tale fase si presti attenzione alle dimensioni del grafico risultante, affinché siano adeguate allo schermo, e si usino operazioni matematiche espresse nel formato consentito dal BASIC. I due errori più comuni sono la divisione per 0 e la radice quadrata di un numero negativo.
Per evitare il primo errore, è sufficiente sommare un piccolo numero (ad esempio: 0,00001) ad ogni variabile che potrebbe eventualmente assumere valore 0. Nel secondo caso basta usare la funzione ABS per rendere i numeri positivi, prima di estrarne la radice quadrata. L'impiego delle funzioni SIN, COS, LOG, ecc.



165 C = X*X + Z*Z + 0.00001
170 Y = SGN(INT(23/C))/3 + SGN(INT(55/5))/15

produce grafici 3/D di un certo interesse, ma si possono anche sperimentare altre funzioni proprie del computer come INT, SGN e ABS.
Il programma può essere migliorato notevolmente, per esempio, con un aggiustamento automatico della "scala" del disegno, oppure cercando di tracciare punti in una terza direzione, per ottenere curve sul piano di X e Z con Y costante. Ma, anche lasciando il programma così com'è, il divertimento è assicurato ed i risultati non finiranno mai di sorprendere.

```
10 REM **TRACCIAMENTO DI GRAFICI 3-D**
20 :
30 REM **PREPARAZIONE DELLO SCHERMO**
40 ORIZZ=1280: VERT=1024: SU=-1
50 INTX=25: INTZ=15
60 LARG=INT (ORIZZ/INTX/2)
70 PROF=INT (VERT/INTZ/3)
80 MODE4:CLS:PRINTTAB(12) "FASE DI CALCOLO"
90 :
100 REM **CALCOLA I PUNTI DEL GRAFICO**
110 INIZIO = 20
120 DIM G(LARG,PROF)
130 FOR A=-PROF/2 TO PROF/2
140 FOR B=-LARG/2 TO LARG/2
150 X=A*20/LARG: Z=B*20/PROF
160 REM **INSERIRE QUI LA FUNZIONE
    PRESCELTA**
170 I=(SIN(X)+COS(Z))/60
180 G(B+LARG/2,A+PROF/2)=Y*SU*VERT
190 NEXT B: NEXT A: CLS
200 :
210 REM **TRACCIA LE CURVE DEL GRAFICO;
    PIANO X-Y**
220 FOR Z=1 TO PROF
230 XBASE=INTX*Z
240 ZBASE=VERT/2+Z*INTZ+INIZIO*SU
250 XOLD=XBASE+INTX
260 ZOLD=ZBASE-INTZ-G(1,Z)
270 FOR X=1 TO LARG
280 XNEW=XBASE+X*INTX
290 ZNEW=ZBASE-X*INTZ-G(X,Z)
300 PLOT 4, XOLD, ZOLD: PLOT 5,XNEW, ZNEW
310 XOLD=XNEW: ZOLD=ZNEW
320 NEXT X: NEXT Z
330:
340 REM **TRACCIA LE CURVE DEL GRAFICO;
    PIANO Z-Y**
350 FOR X=1 TO LARG
360 XBASE=INTX*X+PROF*INTX
370 ZBASE=VERT/2-X*INTZ+PROF*INTZ+INIZIO*SU
380 ZOLD=ZBASE-INTZ-G(X,PROF-1)
390 XOLD=XBASE-INTX
400 FOR Z=0 TO PROF-1
410 XNEW=XBASE-Z*INTX
420 ZNEW=ZBASE-Z*INTZ-G(X,PROF-Z)
430 PLOT 4, XOLD, ZOLD: PLOT 5, XNEW, ZNEW
440 XOLD=XNEW: ZOLD=ZNEW
450 NEXTZ: NEXT X
460 :
470 REM **FERMA L'IMMAGINE SULLO SCHERMO**
480 GOTO 470
```

Questo programma è scritto nel BASIC del Micro BBC.

Basic & Basic

Spectrum
Inserire LET nelle istruzioni di assegnazione e le seguenti linee:
40 LET ORIZZ = 256: LET VERT = 176: LET SU = –1
50 LET INTX = 5: LET INTZ = 3
80 CLS
290 PLOT XOLD, ZOLD: DRAW XNEW–XOLD, ZNEW–ZOLD
410 PLOT XOLD, ZOLD: DRAW XNEW–XOLD, ZNEW–ZOLD

Oric-1/Atmos
Inserire le seguenti linee:
40 ORIZZ = 239: VERT = 199: SU = 1
50 INTX = 5: INTZ = 3
80 HIRES
300 CURSET XOLD, ZOLD,1: DRAW XNEW–XOLD, ZNEW–ZOLD,1
430 CURSET XOLD, ZOLD,1: DRAW XNEW–XOLD, ZNEW–ZOLD,1

I trucchi del mestiere

Utili consigli e suggerimenti per i programmatori autodidatti

Una buona tecnica di programmazione si acquisisce soltanto con una continua sperimentazione e con una notevole dose di esperienza. Anche i programmatori dilettanti, che spesso affrontano i problemi con enorme entusiasmo e concentrazione, possono gradualmente acquistare una buona tecnica, basata su astute scorciatoie e buone regole pratiche, che permettono di ottenere i risultati desiderati. Dopo un certo tempo si raggiungono la chiarezza e la sinteticità tipiche del programmatore esperto. L'apprendimento di questa materia può trarre notevoli vantaggi anche dalla conoscenza degli errori commessi da altri programmatori, che abbiano già percorso lo stesso cammino. Questa sezione, dedicata alle tecniche di programmazione, inizia con una panoramica su alcune considerazioni estremamente utili per i programmatori alle prime armi.

La programmazione non è che un procedimento per risolvere i problemi più disparati, basato soprattutto sul ragionamento logico e sull'uso di carta e penna: solo la fase conclusiva comporta la stesura del programma in un particolare linguaggio. Le varie fasi del procedimento sono ben note: ad una chiara e completa analisi del problema si fanno seguire ripetuti approfondimenti, che permettono di delimitarne in modo preciso e dettagliato i confini. La prima descrizione conterrà o implicherà sempre la soluzione del nucleo centrale del problema, che verrà poi sviluppata sempre più dettagliatamente fino a tramutarla in un concreto strumento di lavoro. Il risultato di tutto il procedimento (la stesura definitiva del programma) dipende in massima parte dalla chiarezza e dalla precisione delle fasi precedenti. Quando, invece, la fase di codifica si sovrappone a quella di analisi, si ottengono sempre scarsi risultati e programmi men che mediocri.

Le formule adottate nei programmi, chiamate anche algoritmi, non sono che una conversione in formule più o meno matematiche del processo logico sviluppato dal programma. L'efficienza di un programma dipende molto da quella dell'algoritmo adoperato, che viene giudicato in base alla sua "completezza" ed "esattezza". Queste due qualità riguardano la capacità teorica e pratica del programma a fronteggiare il maggior numero di situazioni (sia quelle prevedibili che quelle inaspettate) grazie alla coerenza della sua logica interna. In realtà, è molto più facile constatare l'assenza di queste doti, che non dimostrarne la presenza: ogni programma va sottoposto a continue verifiche logiche, possibilmente fin dalle prime fasi dello sviluppo.

Le soluzioni, oltre ad essere complete e corrette, devono soprattutto dimostrarsi "attendibili", anche in situazioni del tutto inattese. Occorre che lo stesso programma sia in grado di individuare le potenziali condizioni d'errore, di interrompere la procedura in corso (senza danneggiare le informazioni immesse) e di informare l'operatore, tramite adeguati messaggi, sull'azione da intraprendere per eliminare la fonte d'errore. Come accennato in precedenza, è più facile rilevare l'inattendibilità di un programma che non il suo contrario, e solo l'esperienza può assistere in questa operazione.

La creazione di programmi sicuri e funzionali è un obiettivo che spesso si scontra con un'altra importante esigenza: l'"economicità". Arriva sempre un momento in cui occorre decidere se continuare a lavorare ad un programma per renderlo "a prova di bomba" o metterlo da parte per dedicarsi ad un nuovo progetto. Il programmatore può forse disporre di tempo illimitato, ma la velocità d'elaborazione e la memoria del computer hanno limiti precisi: può infatti accadere che l'esecuzione delle routine di controllo e dei test d'errore, che corredano l'algoritmo principale, consumino più tempo di quello impiegato per la soluzione stessa del problema centrale.

Collaudo e "debugging"

La soluzione teorica dei problemi analitici e logici è molto importante, ma lo scopo principale di un programma è l'esecuzione di un dato compito. È quindi necessario effettuare un "collaudo" per individuare gli errori logici e sintattici. Purtroppo, questa fase della programmazione viene spesso sottovalutata. Durante il collaudo, anche dei programmi più semplici, è bene sperimentare le più svariate combinazioni di condizioni "limite" e soprattutto verificare con attenzione quei passaggi che sembrano più vulnerabili e quindi più soggetti ad errore. Il procedimento di verifica è forse quello più critico dell'intero sviluppo e richiede notevoli investimenti di tempo e di denaro. A tal proposito, i professionisti partono dal concetto che non esistono programmi perfetti, ma solo collaudi mal eseguiti.

Una volta esaurita la fase di collaudo, è necessario passare a quella detta di "debugging" o fase di correzione. Come per la precedente, anche per questa procedura non esistono indicazioni e regole precise, soprattutto perché gli errori dipendono dalle cause più disparate e, oltretutto, dagli stessi errori, introdotti dal programmatore, che essa viene chiamata a correggere. Gli errori del pro-

gramma vanno affrontati ed eliminati come un qualsiasi altro errore, ma spesso essi suscitano nel programmatore un senso d'impotenza e sgomento, con conseguenze disastrose anche per altre parti del programma stesso.
Dopo aver eseguito tutte le operazioni descritte, molti programmatori ritengono che, se il programma ha funzionato in fase di collaudo, funzionerà sempre, che non avrà più bisogno di modifiche e che il codice in cui è stato scritto sarà di una chiarezza esemplare. In realtà i programmi hanno bisogno di una costante "manutenzione" che, in particolare, significa corredare il tutto con adeguati manuali d'uso. Ogni programma dovrebbe possedere una documentazione interna (linee REM) ad uso del programmatore ed una esterna, sottoforma di manuali per l'utente, anche se quest'ultimo è lo stesso programmatore.
I principi di cui abbiamo parlato stanno alla base di un corretto approccio alla programmazione di qualsiasi computer e sono fondamentali anche per i programmatori di microcomputer, molto più di qualsiasi manuale sulla programmazione. Solo un programmatore dotato di una mente logica e di discreta esperienza potrà creare buoni programmi. Questa serie di lezioni si propone di favorire lo sviluppo di entrambe queste doti.



IAN McKINNELL

Grafici a barre

I colori e la profondità delle barre che formano il grafico vengono definiti nel programma cambiando i valori delle variabili di controllo

```
399 REM ************************************************
400 REM *            GRAFICI A  BARRE 3-D              *
401 REM ************************************************
500 GOSUB 1500:         REM INIZIALIZZAZIONE
720 YY=22: YY=2:        REM CORDINATE DI ORIGINE
760 GOSUB 3200:         REM TRACCIA GLI ASSI
800 FOR E=LT-1 TO 0 STEP -1
820 OC=IX+E*DB: OL=YY-E*DB
840 GOSUB 2200:         REM INPUT DATI
900 FOR D=1 TO NN
920 HT=DT(D)
940 IO=OC+(D-1)*(BB+LT*DB): YO=OL-HT-DB
960 GOSUB 4000:         REM VISUALIZZA LE BARRE
980 NEXT D
1000 NEXT E
1100 XP=10: YP=23: GOSUB 3500
1120 PRINT "ISTOGRAMMA TRIDIMENSIONALE"
1200 A$=INKEY$: IF A$="" THEN GOTO 1200
1400 END
1499 REM ************************************************
1500 REM *           S/R DI INIZIALIZZAZIONE            *
1501 REM ************************************************
1520 CL$=CHR$(147):     REM CANCELLA LO SCHERMO
1540 PRINT CL$
1560 PO$=CHR$(19):      REM CURSORE IN POSIZIONE "HOME"
1580 RT$=CHR$(13):      REM (RETURN)
1600 BB=2: DB=1:        REM DIMENSIONI BARRE
1620 SW=40: SD=25:      REM DIMENSIONI SCHERMO
1640 HB=SD-DB:          REM MASSIMA ALTEZZA BARRE
1660 DIM B$(HB+DB)
1680 FOR K=1 TO SD: PO$=PO$+RT$: NEXT K
1800 DIM DT(SW)
1900 GOSUB 2400:        REM CREAZIONE BARRE
2100 LT=4:              REM COEFFIC. DI PROFONDITA'
2190 RETURN
2199 REM ************************************************
2200 REM *         S/R DI INPUT PER I DATI              *
2201 REM ************************************************
2220 READ RN
2240 FOR Z=1 TO NN: READ DT(Z): NEXT Z
2310 DATA 6,12,10,4,7,8,10
2320 DATA 5,7,8,8,6,7
2330 DATA 6,7,4,8,5,3,9
2340 DATA 5,11,6,4,11,6
2390 RETURN
2399 REM ************************************************
2400 REM * S/R PER LA COSTRUZIONE DI UN'INTERA BARRA     *
2401 REM ************************************************
2500 TC$=CHR$(158):     REM LATI=GIALLO
2520 FC$=CHR$(31):      REM FRONTALE=BLU
2540 RV$=CHR$(18):      REM "REVERSE" ATTIVO
2560 NL$=CHR$(146):     REM "REVERSE" NON ATTIVO
2580 CR$=CHR$(29):      REM CURSORE A DESTRA
2600 CH$=CHR$(32):      REM SPAZIATURA DI UN CARATTERE
2620 C1$=CHR$(169):     REM CARATTERE "◩"
2640 C2$=RV$+C1$:       REM CARATTERE "◪"
2680 FOR K=1 TO SW
2700 SP$=SP$+CH$
2720 RC$=RC$+CR$
2740 FF$=FF$+CH$
2760 NEXT K
2800 TL$=SP$+"/"
2820 BL$=SP$+NL$+C1$
2840 FL$=LEFT$(FF$,BB)
2900 L$=TC$+C2$+LV$+RIGHT$(TL$,BB)
2920 FOR K=1 TO DB
2940 B$(K)=LEFT$(RC$,DB-K)+L$+LEFT$(SP$,K-1)
2960 NEXT K
3000 L$=FC$+RV$+FL$+TC$+RIGHT$(TL$,DB)
3020 FOR K=DB+1 TO HB
3040 B$(K)=L$
3060 NEXT K
3100 L$=FC$+RV$+FL$+TC$
3120 FOR K=1 TO DB
3140 B$(HB+K)=L$+RIGHT$(BL$,DB+2-K)
3160 NEXT K
3190 RETURN
3199 REM ************************************************
3200 REM *        S/R PER TRACCIARE GLI ASSI            *
3201 REM ************************************************
3300 PRINT TC$:         REM SELEZIONA IL COLORE
3320 FOR Y=2 TO YY-1
3340 XP=IX-1: YP=Y: GOSUB 3500: PRINT "!"
3360 XP=IX+YY-Y: GOSUB 3500: PRINT "/"
3380 NEXT Y
3400 YP=YY
3420 FOR X=IX-1 TO SW-1
3440 XP=X: GOSUB 3500: PRINT "-"
3460 NEXT X
3490 RETIRN
3499 REM ************************************************
3500 REM *    S/R PER MUOVERE IL CURSORE @ XP, YP       *
3501 REM ************************************************
3600 PRINT LEFT$(PO$,YP) TAB (XP-1);
3620 RETURN
3999 REM ************************************************
4000 REM *   S/R PER STAMPARE UNA BARRA INCOMPLETA       *
4001 REM ************************************************
4100 FOR V=1 TO HT
4120 XP=IO: YP=YO+V-1
4140 GOSUB 3500:        REM POSIZIONA IL CURSORE
4160 PRINT B$(V)
4180 NEXT V
4200 FOR V=1 TO DB
4220 XP=IO: YP=YO+HT+V-1: GOSUB 3500
4240 PRINT B$(HB+V)
4260 NEXT V
4290 RETURN
READY.
```

Struttura

Modulare e discretamente comprensibile

Documentazione

Questa routine, pur essendo il "cuore" del programma, manca totalmente di spiegazioni

Nomi delle variabili

Sufficientemente descritti, ma poco chiari

Correttezza/Completezza

Questa parte funziona con qualsiasi valore? È in grado di dare sempre un corretto risultato (anche quando, ad esempio il valore è minore di zero?)

"Trappole" per errori

Mancano procedure di verifica, quali: una somma di controllo (checksum) sui valori, la rappresentazione in scala e la gestione di eventuali errori

Documentazione

Buona: non sono usati "numeri magici", né enigmatici caratteri di controllo

BUONO O CATTIVO?

Questo programma per grafici a barre tridimensionali, presenta aspetti positivi e negativi: la documentazione interna, laddove esiste, è discreta e la struttura è modulare. Tuttavia, il listato ha dei punti oscuri: non ci sono test d'errore ed è privo di documentazione per l'utente

Basic & Basic

Questo programma è scritto in BASIC, versione Microsoft, e funziona senza modifiche su micro dotati di uno schermo da 40 × 25 caratteri. Le dimensioni dello schermo vengono selezionate nella linea 1620. I valori dei caratteri di controllo, inizializzati nelle subroutine 1500 e 2400, sono propri del Commodore 64: per gli altri apparecchi si dovrà consultare la tabella dei codici ASCII sui rispettivi manuali

Twin Kingdom Valley

Un divertente gioco d'avventura ricco di colpi di scena

Il capostipite dei giochi per computer del tipo "Adventure" è certamente il famoso *Dungeons and Dragons*. Intorno al 1960, alcuni programmatori di grossi elaboratori del tempo cominciarono a sviluppare i primi giochi per computer, sfruttando la grande capacità di memoria a disposizione, per conservare le descrizioni di un mitico mondo popolato da maghi e da mostri, da gnomi e da giganti. Gli odierni giochi per micro derivano tutti da quelle prime versioni, ma sono ambientati nei luoghi più disparati: dalle astronavi disperse nello spazio ai violenti sobborghi della Chicago degli anni '30.

L'obiettivo di un qualsiasi buon gioco di tipo "Adventure" è quello di coinvolgere totalmente il giocatore nella trama del racconto, facendolo calare nel fantastico mondo in cui si svolge l'azione. Inizialmente, questi giochi si basavano unicamente su messaggi scritti sullo schermo, ma l'uso delle nuove tecniche grafiche ad alta risoluzione ha reso le loro trame sempre più realistiche ed emozionanti. Il primo gioco grafico che ha riscosso un grande successo commerciale è stato *The Hobbit*, tratto dall'omonimo romanzo di J.R.R. Tolkien.

L'"Adventure game" grafico qui esaminato si intitola *Twin Kingdom Valley* ed è un gioco tradizionale, nel quale compaiono castelli medievali e foreste. La grafica aggiunge una certa tensione narrativa allo sviluppo dell'azione, ma non riesce a rimediare alla scarsa fantasia del programmatore. Il giocatore impersona un cavaliere errante, che si avventura in una vallata contesa da due re in guerra tra loro (il Re del Deserto ed il Re dei Boschi). Nella valle scorrono alcuni fiumi (la loro riproduzione grafica è particolarmente riuscita) che si gettano in un lago magico. Mentre vaga in questo ambiente fantastico, il cavaliere deve tentare di impossessarsi del maggior numero possibile di tesori. Quando il giocatore raggiunge il punteggio di 1024, accade un evento inatteso, che qui non sveleremo per non rovinare il gioco.

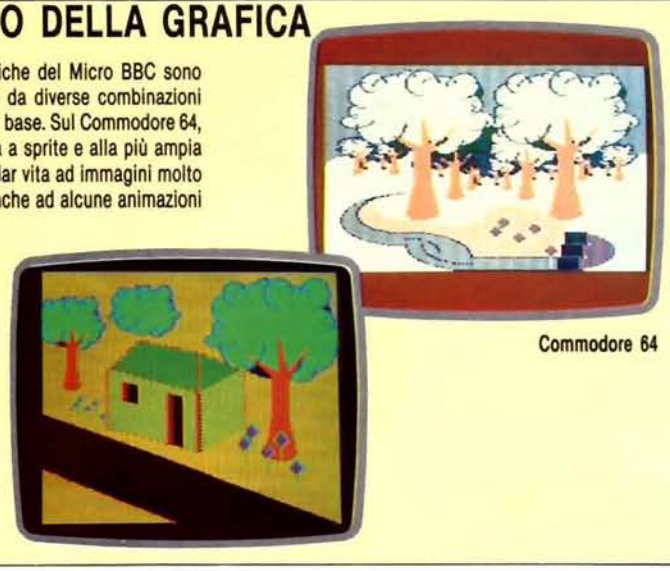
Sia il movimento che lo sviluppo dell'azione vengono guidati tramite messaggi scritti. Il programma riconosce 23 verbi che, combinati alle parole, fanno riferimento alle varie fasi di gioco. Il programma accetta comandi (in inglese, ahimè) del tipo "Colpisci la guardia con il maglio", sempre che il cavaliere abbia un maglio in mano e ci sia una guardia nelle vicinanze. La direzione del movimento viene indicata abbinando le parole "su" (up) e "giù" (down) ai vari punti cardinali. La vallata è popolata da altri personaggi ed il giocatore può usare il verbo "Chiedi" (ask) per appropriarsi delle loro ricchezze: in alcuni casi, però, la reazione che si ottiene rivolgendosi a loro è tutt'altro che amichevole!

La grafica viene impiegata per creare ben 175 scenari di gioco. La versione per il Micro BBC, in particolare, usa molta memoria per visualizzare immagini ad alta risoluzione, molte delle quali sono composte da più combinazioni delle stesse figure di base. Le foreste, ad esempio, sono composte da 10 o 12 immagini di alberi combinate in vario modo. La grafica a sprite e la maggior ampiezza di memoria del Commodore consentono, su alcuni schermi, l'uso di diverse animazioni. Per risparmiare memoria si può eliminare la componente grafica, anche se il suo consumo di memoria è piuttosto ridotto, però questa scelta renderà il gioco molto meno emozionante.

Il *Twin Kingdom Valley* è un gioco piuttosto facile e forse poco originale: esistono altri giochi, dello stesso genere, che, pur privi di grafica, stimolano con maggior efficacia la fantasia e l'interesse del giocatore.

IL REGNO DELLA GRAFICA

Le immagini grafiche del Micro BBC sono spesso composte da diverse combinazioni di alcune figure di base. Sul Commodore 64, grazie alla grafica a sprite e alla più ampia memoria, si può dar vita ad immagini molto più complete e anche ad alcune animazioni



Commodore 64

Micro BBC

IAN McKINNELL

Twin Kingdom Valley: Versione per il Micro BBC
Versione per il Commodore 64
Editori: Bug-Byte Software, (GB)
Autore: Trevor Hall
Joystick: Non necessari
Formato: Su cassetta

Pixel su pixel

La grafica del Commodore 64, usando il codice macchina

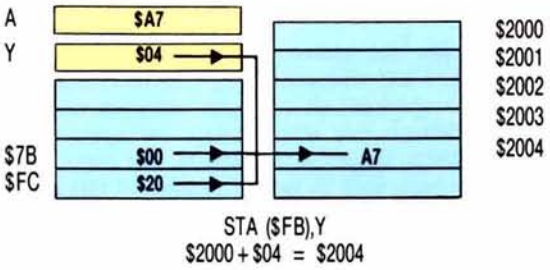
I vari passaggi e le diverse procedure necessarie per programmare la grafica ad alta risoluzione del Commodore 64 sono già state esaminate a pag. 734. Innanzi tutto occorre abilitare l'alta risoluzione sul Chip di Interfaccia Video (VIC) e cambiare il puntatore all'indirizzo di base del set di caratteri. Poi è necessario azzerare il blocco di 8 Kbyte, con locazione iniziale 8192, che contiene la mappa di memoria video, ed inizializzare la mappa di memoria del video normale. Quest'ultima operazione può risultare complicata dalla presenza di immagini a 4 colori ma, nel nostro caso, viene usato un solo colore per lo sfondo.

Il programma consente di entrare ed uscire dal modo di alta risoluzione, eseguendo tutti i calcoli necessari alla definizione di un punto sullo schermo ad alta risoluzione. La prima parte del programma, pertanto, riguarda le due importanti operazioni da effettuare prima di usare il modo di alta risoluzione, ossia: la registrazione dell'informazione di colore in ciascuna locazione del video normale e l'azzeramento della "bit map" di 8 Kbyte.

L'uso di un flag, qui chiamato HRSFLG, permette di utilizzare la stessa routine, sia per attivare che per disattivare il modo in alta risoluzione. Tuttavia, quando si abilita il modo in alta risoluzione, potremmo non voler azzerare la "bit map" (soprattutto se si vuole conservare una figura disegnata in precedenza sullo schermo). Per segnalare l'eventuale richiesta di cancellazione dello schermo ci serviamo di un secondo flag, chiamato CLRFLG. Il diagramma di flusso, qui a fianco, chiarisce l'uso dei due flag all'interno della routine in codice macchina. Vediamo adesso come accedere a blocchi di memoria di 265 byte o meno, con un lavoro relativamente semplice, utilizzando un ciclo iterativo in codice macchina. Il seguente frammento di programma assegna il numero $03, mediante una procedura di indirizzamento ad indice assoluto (vedere a pag. 676), a tutte le locazioni comprese tra gli indirizzi BASE e BASE+255 (per un totale di 256 locazioni).

```
      LDY $00
      LDA $03
NEXT  STA BASE,Y
      DEY
      BNE NEXT
```

Si noti che, con questa tecnica, il primo indirizzo a cui si ha accesso è BASE, mentre per tutti gli altri indirizzi del blocco la procedura di assegnazione procede in senso inverso, cioè da BASE+255 a BASE+1. Questo programma ci permette di accedere anche a più di un blocco di memoria di 256 byte: per far ciò dobbiamo però adottare un secondo tipo di indirizzamento, conosciuto come indiretto post-indicizzato. Questo metodo usa la pagina zero per calcolare gli indirizzi, qualunque sia la loro collocazione nella memoria. La maggior parte della pagina zero è impiegata dal sistema operativo del Commodore 64, ma esiste una piccola zona a disposizione del programmatore in codice macchina: 251 e 255 ($FB e $FC) sono appunto due dei byte di tale zona. Con questo metodo di indirizzamento, il computer presume che il byte basso dell'indirizzo sia contenuto nella specifica locazione di pagina zero e che quello alto si trovi nella successiva locazione di pagina zero. Quindi, un'istruzione del tipo STA ($FB), Y, dove $FB e $FC contengono rispettivamente $00 e $20 ed Y contiene $04, calcola l'indirizzo richiesto nel seguente modo:



STA ($FB),Y
$2000 + $04 = $2004

La forza di questo metodo risiede nel modo usato per indirizzare il byte alto dell'indirizzo BASE. Incrementando quest'ultimo di uno, si può accedere al successivo blocco di 256 byte e questa tecnica può spesso tornare utile.

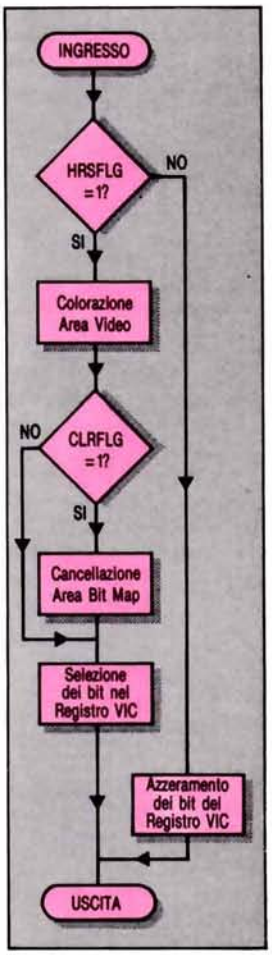
L'area video si estende da $0400 a $07E7; è quindi composta da tre blocchi di 256 byte, con un resto di $E7 byte. Il gruppo di istruzioni Assembly definito a pag. 819 come "Area del Video a Colori", si serve dell'indirizzamento indiretto post-indicizzato per assegnare a ciascun byte la propria informazione di colore. Le variabili usate sono: SCBLO e SCBHI che indicano i byte basso e alto del primo indirizzo del video normale, e SCBLK e SCREM, che rappresentano rispettivamente il numero di blocchi di 256 byte ed il resto. PRT è la locazione di pagina zero usata per memorizzare il byte basso dell'indirizzo di base.

Calcolo delle posizioni dei pixel

La seconda parte del programma riguarda il calcolo del bit, nella "bit map", corrispondente ad una data coppia di coordinate (x,y). Con

DIAGRAMMA DI FLUSSO

Questa routine per l'alta risoluzione esegue tre diverse operazioni. Compone l'informazione di colore per il video, cancella l'area bit map e pone a 1 o azzera i bit dei registri ad alta risoluzione, a seconda dello stato dei flag HRSFLG e CLRFLG



il video ad alta risoluzione, i valori dell'ascissa x sono compresi tra 0 e 319, mentre quelli dell'ordinata y vanno da 0 a 199. Questi ultimi possono essere contenuti in un unico byte mentre per i valori di x superiori a 255, saranno necessari due byte. In BASIC, date le coordinate x e y, il calcolo del bit corrispondente viene così realizzato:

```
1000 HB = XAND 248: VBYTE = INT(Y/8)
1010 RMY = YAND7: RMX = XAND7
1020 RIG = VBYTE*320 + HB
1030 BYTE = BASE + RIG + RMY
1040 POKE BYTE, PEEK (BYTE) OR (2↑(7—RMX))
```

La corrispondente routine in codice macchina esegue lo stesso tipo di passaggi. HB, RIG, BASE e BYTE impiegano tutti due byte e ciò rende il procedimento aritmetico piuttosto complicato. Il programma non richiede ulteriori chiarimenti, salvo approfondire i passaggi della divisione y/8 e della moltiplicazione VBYTE*320. La divisione per una potenza di due è di facile esecuzione:

```
45        = 00101101
45/2 = 22 = 00010110
22/2 = 11 = 00001011
11/2 =  5 = 00000101
 5/2 =  2 = 00000010
 2/2 =  1 = 00000001
 1/2 =  0 = 00000000
```

Ogni volta che viene eseguita una divisione per due, i bit che compongono il dividendo vengono spostati di una posizione a destra: una divisione per otto, quindi, comporta tre Scorrimenti Logici a Destra (LSR). Poiché a noi interessa solo la parte intera del risultato ottenuto, possiamo trascurare tutti i bit che "fuoriescono" dall'estremità destra del byte. In realtà, l'operazione LSR deposita il bit "fuoriuscito" nel Carry (riporto) e ciò può risultare utile in certe occasioni. La moltiplicazione per due segue il procedimento inverso: basta spostare i bit del moltiplicando di una posizione a sinistra (istruzione ASL). Possiamo quindi scrivere una routine che esegua la moltiplicazione VBYTE*320, scomponendo il numero 320 in 5×46 e 64 in 2×2×2×2×2×2. Prendendo adesso VBYTE, sommandolo a se stesso per 5 volte ed eseguendo sei operazioni di ALS, otterremo il risultato della moltiplicazione VBYTE×320.
Entrambe le routine in codice macchina esaminate vengono chiamate dall'interno del programma BASIC usando SYS. Tuttavia, durante l'esecuzione di un programma BASIC, l'interprete usa i registri X, Y ed A e se questi vengono utilizzati anche da una routine in codice macchina (come nel nostro caso), il loro contenuto deve essere salvato, prima di eseguire le routine, e poi recuperato prima di tornare al BASIC. Per quest'operazione viene di solito usato lo stack. I valori delle coordinate, dei colori e dei flag possono essere trasferiti al programma in codice macchina inserendoli (con POKE) nelle loro specifiche locazioni, come mostrato nel programma BASIC.

USO DELLA SUBROUTINE PLOTSUB

```
1 GOTO 200
2 POKE 53265, PEEK(53265) AND 223
3 POKE 53272, PEEK(53272) AND 240 OR 4
4 STOP
200 REM ***** C64 DIMOSTRAZIONE IN ALTA RISOLUZIONE *****
210 :
220 POKE 56,32:CLR:       REM ABBASSA LIMITE SUP. MEMORIA
250 GOSUB 3000:           REM INIZIALIZZA S/R
350 :
360 REM *****  ABILITA IL MODO IN ALTA RISOLUZIONE  *****
370 :
380 PRINT CC$:PRINT:PRINT
390 INPUT "COLORE PRIMO PIANO";FG
400 INPUT "COLORE SFONDO";BG
410 TT=FG*16+BG:          REM CALCOLO DEL COLORE
420 POKE COLORE, TT:      REM COLORE PER LA S/R IN C.M.
430 POKE HRSFLG, 1:       REM ABILITA L'ALTA RISOLUZIONE
440 POKE CLMFLG           REM CANCELLA/RIABILITA VIDEO A.R.
450 SYS BEGIN:            REM ESECUZIONE S/R IN C.M.
460 :
500 REM *****        DISEGNA UNA FIGURA        *****
515 Z=0: Y1=50: Y2=150: X1=160: SP=6
520 FOR Y=Y1 TO Y2 STEP SP
530 FOR X=Y2-Z TO Y2+Z STEPSP
540 GOSUB 1000:           REM TRACCIA UN PUNTO
550 NEXT X
555 Z=Z+SP
557 NEXT Y
560 :
565 GET Y$: IF Y$ <>"" THEN 565
570 GET A$: IF A$="" THEN 570: REM ATTESA BATTUTA TASTO
580 :
600 REM ***** CANCELLA VIDEO IN ALTA RISOLUZIONE *****
605 POKE HRSFLG,1: POKE CLMFLG, 1
610 SYS BEGIN
650 :
630 REM *****   DIMOSTRAZIONE TRACCIAMENTO LINEA *****
640 X1=0: X2=300: Y1=0: Y2=190: SP=1
670 GOSUB 1500:           REM TRACCIA UNA LINEA
690 :
695 GET J$: IF J$<>"" THEN 695
700 GET A$: IF A$="" THEN 700: REM ATTESA BATTUTA TASTO
710 :
720 REM *****       AZZERAMENTO DEL VIDEO       *****
730 :
740 POKE HRSFLG,0:        REM ABBANDONO ALTA RISOLUZIONE
750 SYS BEGIN
760 PRINT CC$: PRINT  :PRINT
770 PRINT TAB (9) "**** FINE PROGRAMMA PRINCIPALE ****"
780 END
999 :
1000 REM ***    SUBROUTINE PER DISEGNO IN A.R.    ***
1010 :
1020 XHI=INT(X/HX): XLO=X-XHI*HX
1030 POKE XBYTE, XLO: POKE XPAGE, XHI: POKE YBYTE,Y
1035 SYS PLOT:            REM ESECUZIONE S/R DISEGNO
1040 RETURN
1500 REM ***    SUBROUTINE TRACCIAMENTO LINEA    ***
1550 C9=(Y2-Y1)/(X2-X1): C8=C9*X1-Y1
1600 FOR X=X1 TO X2 STEP SP
1650 Y=X*C9-C8
1700 GOSUB 1000:          TRACCIA UN PUNTO
1750 NEXT X
1800 RETURN
3000 REM ***       INIZIALIZZA LA SUBROUTINE      ***
3020 CC$=CHR$(157):       REM AZZERA IL VIDEO
3025 HX=256
3030 HRSFLG=49408:        REM $C 100
3040 CLMFLG=49409:        REM $C 101
3050 COLORE=49410:        REM $C 102
3060 XBYTE=49411:         REM $C 103
3070 XPAGE=49412:         REM $C 104
3080 YBYTE=49413:         REM $C 105
3085 BEGIN=49422:         REM $C 10E
3090 PLOT=49539:          REM $C 183
3095 PRINT CC$: PRINT :PRINT
3100 PRINT TAB (9) "**** CARICAMENTO CODICE MACCHINA ****"
3150 PRINT TAB (9) "1 - CODICE MACCHINA SU NASTRO"
3200 PRINT TAB (9) "2 - CODICE MACCHINA IN FRASI DATA"
3250 PRINT TAB (9) "3 - CODICE MACCHINA IN MEMORIA"
3300 PRINT "BATTERE IL NUMERO DESIDERATO"
3350 FOR LP=0 TO 1 STEP 0
3400 GET OP$
3450 IF OP$<"0" AND OP$>"4" THEN LP=1
3500 NEXT LP
3600 ON VAL(OP$) GOSUB 4000, 5000, 6000
3900 RETURN
```

```
4000 REM ** S/R CARICAMENTO COD. MACCHINA SU NASTRO **
4100 PRINT "INSERIRE IL NASTRO CON LA S/R IN C. MACCHINA"
4200 IF A=0 THEN A=1: LOAD "PLOTSUB.HEX", 1, 1
4900 RETURN
5000 REM ** S/R CARICAMENTO COD. MACCHINA IN FRASI DATA **
5005 PRINT TAB (11) "**** CARICAMENTO ****"
5010 FOR I=HRSFLG TO HRSFLG+312: READ A
5020 POKE I, A: S=S+A: NEXT I
5030 READ CC: IF CC<>S THEN PRINT "CHECKSUM ERROR": END
5040 DATA 2,0,255,255,2,2,255,255,2,18
5050 DATA 255,255,2,2,72,138,72,152,72
5060 DATA 173,0,193,240,83,196,0,133,251
5070 DATA 169,4,133,252,162,3,160,0,173
5080 DATA 2,193,145,251,136,208,251,230
5090 DATA 252,202,48,8,208,244,145,251
5100 DATA 160,231,208,238,173,1,193,240
5110 DATA 24,169,0,133,251,169,32,133
5120 DATA 252,162,32,160,0,169,0,145,251
5130 DATA 136,208,251,230,252,202,208
5140 DATA 246,173,24,208,41,240,9,8,141
5150 DATA 24,208,173,17,208,9,32,141,17
5160 DATA 208,76,125,193,173,24,208,41
5170 DATA 240,9,4,141,24,208,173,17,208
5180 DATA 41,223,141,17,208,104,168,104
5190 DATA 107,104,86,72,138,72,152,72
5200 DATA 173,4,193,141,7,193,173,3,193
5210 DATA 41,248,141,6,193,173,3,193,41
5220 DATA 7,141,8,193,173,5,193,41,7,141
5230 DATA 10,193,162,3,78,5,193,202,208
5240 DATA 250,173,5,193,141,9,193,169,0
5250 DATA 141,11,193,141,12,193,162,5
5260 DATA 173,11,193,24,109,9,193,141,11
5270 DATA 193,202,208,243,142,6,14,12
5280 DATA 193,14,11,193,144,3,238,12,193
5290 DATA 202,208,242,173,11,193,24,109
5300 DATA 6,193,141,11,193,173,12,193
5310 DATA 109,7,193,141,12,193,173,11
5320 DATA 193,24,105,0,141,11,193,173,12
5330 DATA 193,105,32,141,12,193,173,11
5340 DATA 193,24,109,10,193,141,11,193
5350 DATA 173,12,193,105,0,141,12,193
5360 DATA 173,11,193,133,251,173,12,193
5370 DATA 133,252,169,1,141,13,193,56
5380 DATA 169,7,237,8,193,170,14,13,193
5390 DATA 202,208,250,160,0,177,251,13
5400 DATA 13,193,145,251,76,125,193
5410 DATA 38698: REM *CHECKSUM*
5900 RETURN
6000 REM ** S/R PER COD. MACCHINA PRESENTE IN MEMORIA **
6100 RETURN
```

Questo programma dimostrativo BASIC mostra le varie fasi riguardanti l'uso di routine di alta risoluzione in codice macchina:

1) Possedendo un programma Assembler, è possibile scrivere un programma in codice macchina, assemblarlo, salvarlo come file sorgente ed infine salvare il codice oggetto, compreso tra le locazioni $C100 e $C238, col nome "PLOTSUB.HEX". In questa fase non è possibile inviare in esecuzione la subroutine perché il video ad alta risoluzione potrebbe interferire con il funzionamento dello stesso assemblatore.
2) Per usare in un programma BASIC la routine di alta risoluzione, è necessario "abbassare" MEMTOP, ossia il limite superiore della memoria (vedere linea di programma 220), e caricare in memoria il codice dal nastro (vedere subroutine 4000).
3) In alternativa, si può salvare la subroutine 5000 come se si trattasse di un programma BASIC (chiamandola, ad esempio, "MCODLOAD"). Per usare la subroutine si dovrà abbassare MEMTOP, caricare MCODLOAD ed infine avviarne l'esecuzione. A questo punto si digiti NEW e si carichi il programma che si vuol eseguire: la routine di alta risoluzione, depositata in memoria, si può adoperare ricorrendo all'istruzione SYS.
4) L'ultimo dato della subroutine (vedere linea 5410) è una "checksum", ossia la somma di tutti i dati precedenti. Se il programma si blocca e compare il messaggio "CHECKSUM ERROR", ciò significa che si è commesso qualche errore durante la trascrizione.

```
;**************************************
;**************************************
;**                                  **
;**   DISEGNO IN ALTA RISOLUZIONE    **
;**                                  **
;**         ROUTINE PER IL           **
;**                                  **
;**          COMMODORE 64            **
;**                                  **
;**************************************
;**************************************
;
;
PTR    EQU $FB
MPBLO  EQU $00
MPBHI  EQU $20
SCBLO  EQU $00
SCBHI  EQU $04
SCBLK  EQU $03
SCREM  EQU $E7
MPBLK  EQU $20
       ORG $C100
;
HRSFLG DB $00
CLRFLG DB $00
COLORE DB $00
ILO    DB $00
IHI    DB $00
HBLO   DB $00
HBHI   DB $00
REMI   DB $00
VBYTE  DB $00
REMY   DB $00
RIGLO  DB $00
RIGHI  DB $00
BPOS   DB $00
;********** ABILITA L'ALTA RISOLUZIONE **********
;********** E CANCELLA L'AREA  BIT MAP **********
       PHA
       TXA
       PHA
       TYA
       PHA
       LDA HRSFLG
       BEQ RESET
;************* COLORA  L'AREA VIDEO **************
       LDA #SCBLO
       STA PTR
       LDA #SCBHI
       STA PTR+1
       LDX #SCBLK
       LDY #$00
       LDA COLORE
ANCOR  STA (PTR),Y
       DEY
       BNE ANCOR
       INC PTR+1
       DEX
       BMI CLTEST
       BNE ANCOR
       STA (PTR),Y
       LDY #SCREM
       BNE ANCOR
CLTEST LDA CLRFLG
       BEQ HRESON
;*********** CANCELLA L'AREA  BIT MAP ***********
       LDA #MPBLO
       STA PTR
       LDA #MPBHI
       STA PTR+1
       LDX #MPBLK
       LDY #$00
       LDA #$00
NEXT   STA (PTR),Y
       DEY
       BNE NEXT
       INC PTR+1
       DEX
       BNE NEXT
;********** SELEZIONA IL  MODO BIT MAP **********
HRESON LDA $D018
       AND #$F0
       ORA #$08
       STA $D018
       LDA $D011
       ORA #$20
       STA $D011
       JMP ESCI
;********** AZZERA IL  VIDEO NORMALE ************
RESET  LDA $D018
       AND #$F0
       ORA #$04
       STA $D018
       LDA $D011
       AND #$DF
       STA $D011
;************** ESCI DAL C.M. ***************
ESCI   PLA
       TAY
       PLA
       TAX
       PLA
       RTS
;****** CALCOLO POSIZ. IN ALTA RISOLUZIONE ******
       PHA
       TXA
       PHA
       TYA
       PHA
;********* CALCOLA IL  BYTE ORIZZONTALE *********
       LDA IHI
       STA HBHI
       LDA ILO
       AND #$F8
       STA HBLO
       LDA ILO
       AND #$07
       STA REMI
;********** CALCOLA IL  BYTE VERTICALE **********
       LDA YLO
       AND #$07
       STA REMY
       LDX #$03
SHIFT  LSR YLO
       DEX
       BNE SHIFT
       LDA YLO
       STA VBYTE
;**************** CALCOLA  LA RIGA ***************
       LDA #$00
       STA RIGLO
       STA RIGHI
       LDX #$05
CINQ   LDA RIGLO
       CLC
       ADC VBYTE
       STA RIGLO
       DEX
       BNE CINQ
       LDX #$06
MOLT   ASL RIGHI
       ASL RIGLO
       BCC NCARRY
       INC RIGHI
       DEX
       BNE MOLT
;********** SOMMA IL  BYTE ORIZZONTALE **********
       LDA RIGLO
       CLC
       ADC HBLO
       STA RIGLO
       LDA RIGHI
       ADC HBHI
       STA RIGHI
;** SOMMA L'INDIRIZZO BASE DELLA MAPPA DI A.R. **
       LDA RIGLO
       CLC
       ADC #MPBLO
       STA RIGLO
       LDA RIGHI
       ADC #MPBHI
       STA RIGHI
;************ SOMMA IL RESTO  DI YLO ************
       LDA RIGLO
       CLC
       ADC REMY
       STA RIGLO
       LDA RIGHI
       ADC #$00
       STA RIGHI
;************ COMBINA I  DUE BYTE DI ************
;********* INDIRIZZO SULLA  PAGINA ZERO *********
       LDA RIGLO
       STA PTR
       LDA RIGHI
       STA PTR+1
;******** CALCOLA LA POSIZIONE DEL PIXEL ********
       LDA #$01
       STA BPOS
       SEC
       LDA #$07
       SBC REMI
       TAX
POTEN  ASL BPOS
       DEX
       BNE POTEN
;**************** ACCENDE IL PIXEL ***************
       LDY #$00
       LDA (PTR),Y
       ORA BPOS
       STA (PTR),Y
       JMP ESCI
```

La Bug-Byte

Dopo i primi successi con i programmi per lo ZX80, questa software house si è specializzata in giochi

La Bug-Byte è stata fondata nella primavera del 1980, quando Tony Baden e Tony Milner, due studenti di chimica dell'University College di Oxford, iniziarono a scrivere programmi per il loro ZX80. Dopo qualche mese di lavoro decisero di vendere il software sviluppato, composto essenzialmente di giochi. Acquistarono una quarantina di cassette ed esposero il loro pacchetto, composto da 5 giochi, in un negozio di computer. Le prime ordinazioni arrivarono ad un ritmo di 15 alla settimana ed i due giovani investirono i profitti in pubblicità e nella produzione di nuovi programmi per ZX80. Verso la fine del 1980 la Acorn lanciò il micro Atom e la Bug-Byte decise di dedicare parte della propria produzione software a questo nuovo apparecchio.



Tony Milner



Tony Baden

All'inizio dell'81 comparve lo ZX81 e la domanda di giochi per lo ZX80 si ridusse notevolmente. Di conseguenza, Baden e Milner si dedicarono immediatamente a scrivere programmi per questo nuovo modello. Dopo essersi laureati, nel giugno dell'81, si trasferirono a Liverpool nella casa di Tony Baden e si dedicarono a tempo pieno alla loro società: in breve tempo le vendite dei loro programmi raddoppiarono.

Per meglio difendersi dalla concorrenza, la Bug-Byte affidò la propria "immagine" ad un'agenzia pubblicitaria e cominciò a servirsi di una società specializzata nella duplicazione di cassette, garantendo così ai propri clienti prodotti di alta qualità. Queste scelte, abbinate alla nuova rete di distribuzione su scala nazionale, hanno permesso alla Bug-Byte di accrescere il fatturato e di ampliare il proprio staff di programmatori, consentendo di trarre i massimi vantaggi dalla crescente diffusione di home computer, tanto da superare, più di una volta, le più agguerrite concorrenti: la Quicksilva e la Software Projects. La Bug-Byte corrisponde una "royalty" ai propri programmatori per ogni cassetta commercializzata, e se un gioco rientra tra i primi venti nella classifica delle vendite, il guadagno dell'autore (così afferma la stessa Bug-Byte) può facilmente oltrepassare i 100 milioni.

Verso la fine del 1982, la rete commerciale della software house poteva contare su più di 200 punti di vendita indipendenti, oltre ad alcune delle principali catene di negozi. Questa esplosiva crescita comportò grossi problemi ai fornitori di cassette della Bug-Byte e, nel Natale di quell'anno, anche le ditte addette alla riproduzione dei nastri non furono in grado di soddisfare le richieste del mercato. Per questi motivi la Bug-Byte decise di dar vita ad una propria catena di riproduzione, che fu chiamata Spool. Inoltre, nel giugno dell'83, la ditta ha aperto un proprio punto di vendita nel centro commerciale di Liverpool.



ROGER SANDERS

La sede centrale della Bug-Byte a Liverpool

I prodotti di maggior successo della Bug-Byte sono stati il *Twin Kingdom Valley*, un gioco grafico del tipo "Adventure" per il BBC ed il Commodore 64 (vedere a pag. 816) ed il *Manic Miner*, che gira sullo ZX80 e sul Commodore 64 (vedere a pag. 793). L'autore di quest'ultimo è Mattew Smith, che ha lasciato la Bug-Byte per fondare la Software Projects, portandosi dietro il copyright del gioco.

Oggi la Bug-Byte vende i propri prodotti in molti paesi d'Europa, in Australia, in Nuova Zelanda, in Sud Africa ed un recente accordo commerciale con la sede inglese della CBS le ha aperto le porte del mercato americano. John Phillips, il direttore commerciale della Bug-Byte, ha dichiarato che l'accordo con la CBS sarà il modello sul quale si svilupperà tutta la futura organizzazione commerciale della compagnia.

I computer... in volo

Dall'introduzione dell'Osborne 1, ci sono stati stupefacenti sviluppi nel campo dei portatili

Il significato del termine "portatile" deve oggi essere rivisto alla luce della nuova generazione di computer "da viaggio". In effetti, i micro portatili introdotti solo pochi anni fa vengono ora chiamati "trasportabili". La vera portatilità è oggi offerta da apparecchi dotati di alimentazione autonoma, di display LCD e di un'ampia memoria, il tutto delle dimensioni di un elenco telefonico.

L'Epson HX-20 è stato il primo a proporre questo nuovo tipo di portatilità, ma il suo piccolo display a cristalli liquidi, di 4 linee a 20 caratteri, appartiene già alla storia: i più recenti portatili come il Tandy 100, il NEC PC-8201A e l'Olivetti M10, a parità di prezzo, hanno un display capace di visualizzare un numero di caratteri 4 volte superiore.

Vediamo quali sono le caratteristiche principali di questi computer, quali i vantaggi e gli svantaggi rispetto ad un comune micro da tavolo. Il loro crescente successo dipende soprattutto dalla loro flessibilità: si possono usare dovunque ed in qualsiasi momento e offrono un completo sistema d'elaborazione. Molte persone consumano buona parte del loro tempo spostandosi da un ufficio all'altro, nelle hall di alberghi, aeroporti e stazioni, lontano, insomma, da unità d'elaborazione tradizionali: questi nuovi portatili da viaggio consentono di sfruttare tutti questi tempi morti. L'ultima generazione di portatili è particolarmente indicata per elaborazioni di carattere scientifico o ingegneristico, per operazioni contabili, finanziarie e di word processing: in sostanza per ogni applicazione pratica tipica di qualsiasi personal computer.

I computer da viaggio possiedono, di solito, non più di tre programmi interni: un interprete BASIC, un programma di word processing ed uno di intercomunicazione. Il Tandy 100 e l'Olivetti sono dotati anche di altri due programmi interni: uno per la gestione di una rubrica telefonica e l'altro per la pianificazione di appuntamenti.

Il software di comunicazione è di grande utilità, perché consente tutta una varietà di collegamenti (diretti o tramite linee telefoniche) tra il portatile ed altri computer, database, terminali telex e anche l'invio o la ricezione di posta elettronica. Ovviamente, per queste operazioni è necessario interfacciare l'apparecchio con un modem o un accoppiatore acustico. Chiunque si trovi in viaggio per affari, dunque, può portarsi dietro il proprio piccolo ufficio; ciò è particolarmente utile, ad esempio, per i giornalisti: un "inviato speciale" può redigere il proprio "pezzo" sul piccolo



TONY SLEEP

apparecchio e trasmetterlo, via telefono, all'elaboratore della redazione.

Sui computer portatili più sofisticati, quali lo Sharp PC-5000 e l'Epson PX-8, sono disponibili i sistemi operativi MS-DOS e CP/M, gli stessi impiegati dalla maggior parte dei personal computer: ciò consente loro di accedere ad una vastissima gamma di software commerciale.

Per l'Epson PX-8, addirittura, viene venduto un chip ROM, contenente il famoso programma di word processing "Wordstar", mentre nello Sharp si possono innestare cartucce di memorie a bolle, ciascuna di ben 128 Kbyte di memoria extra, che permettono una gestione dei dati molto più efficace, in viaggio, delle normali unità a dischetti.

Nei portatili più economici i programmi applicativi vengono di solito caricati nella RAM del computer leggendoli da una cassetta: questo processo è molto più lento rispetto al



In viaggio...

L'abitudine all'impiego di computer portatili sta diventando sempre più diffusa, soprattutto tra gli uomini d'affari. Alcuni di essi già usano la nuova generazione di portatili "da viaggio" per redigere lettere o relazioni, anche quando sono in taxi o in treno. La svolta di maggior impatto, però, sembra riguardare i rappresentanti di commercio, poiché la possibilità di portare con sé il computer quando si recano dai loro clienti, permette loro di produrre in modo immediato preventivi di spesa ed altre elaborazioni, che altrimenti richiederebbero svariati giorni di lavoro. Oltretutto, basta un telefono ed un piccolo modem (ne esistono anche di quelli funzionanti a pile) per scambiare informazioni con il computer della propria sede centrale

In volo...

Non tutte le compagnie aeree permettono ai viaggiatori di usare i loro computer portatili durante il volo. In un comunicato ufficiale, la Civil Aviation Authority (CAA) ha dichiarato che l'uso di apparecchiature e strumenti elettronici a pile può interferire col sistema di controllo di volo dell'aeroplano. Tuttavia, ogni compagnia aerea segue i propri regolamenti. Ad esempio: la tedesca Lufthansa e l'australiana Quantas vietano categoricamente l'uso, in volo, di qualsiasi apparecchio elettrico; la notissima Pan Am statunitense, vieta radio e registratori a nastro, ma permette l'uso di giochi elettronici e di computer; la Japan Airlines consente l'uso di qualsiasi apparecchio, mentre la British Airways proibisce l'uso dei computer

caricamento da cartucce di memorie a bolle o da unità a dischetti. A corredo del NEC PC-8201A viene data una cassetta che contiene numerosi programmi applicativi: un programma di calcolo, un formattatore di testi, un programma di "portafoglio titoli" ed uno di valutazione degli interessi. Il programma di calcolo trasforma l'apparecchio in una potente calcolatrice, capace di ricordare fino a 99 registrazioni. Il formattatore serve a preparare la stampa dei file creati mediante il word processor: oltre a poter specificare i margini da usare, si ha la facoltà di suddividere il testo in pagine e di assegnare a ciascuna un numero progressivo. Il programma di portafoglio titoli è di grande utilità per gli operatori di borsa: è in grado di gestire un portafoglio con un massimo di 50 operazioni di borsa, calcolando le perdite e i profitti.

Come ogni altro computer, i micro portatili possono venire collegati a stampanti, registratori a cassette e modem. Comunque, a parte gli ovvi limiti dovuti alle dimensioni ed al peso, un portatile può dirsi veramente tale soltanto se è alimentato a pile, se possiede un proprio display e vi sono adeguati programmi di word processing e di intercomunicazione, memorizzati in ROM.

Apparecchi quali l'Apple IIc e l'Apricot, invece, reclamizzati talvolta come portatili, non possono essere usati in viaggio, in quanto non possiedono un'alimentazione autonoma, devono essere collegati ad un monitor esterno ed i loro programmi vengono caricati in RAM da un disco. Pur essendo piccoli e leggeri, sono molto più vicini ad un micro da tavolo che non ad un vero portatile.

Oltre alle batterie principali, i portatili sono dotati di piccole pile "di tampone", al nickel-cadmio, che permettono di non perdere alcun dato quando l'apparecchio viene spento.

Alcuni modelli sono anche dotati di un connettore per un lettore ottico di codici a barre, che può essere usato per una rapidissima immissione di dati riguardanti, ad esempio, gli articoli di un magazzino: basta far scorrere il lettore sul codice a barre di un particolare prodotto e l'informazione viene immediatamente immagazzinata ed elaborata dal computer, permettendo di tenere facilmente sotto controllo una situazione di magazzino.

Tra le macchine di cui abbiamo parlato, solo il Tandy 100, il NEC PC-8201 e l'Olivetti M10 hanno lettori di codici a barre (il Casio FP-200 ne è sprovvisto). La somiglianza di questi tre portatili non si ferma qui. In effetti tutti e tre vengono prodotti dalla stessa industria giapponese. Tuttavia, ognuno possiede alcune caratteristiche distintive: l'Olivetti ha un display inclinabile, il NEC è dotato di una minore quantità di software interno e inoltre, mentre la memoria del Tandy e dell'Olivetti non può essere espansa oltre i 32 Kbyte, quella del NEC può arrivare fino a 64 Kbyte: le sue cartucce di memoria di 32K sono



Epson HX-20
Ha uno schermo piuttosto piccolo, ma possiede un registratore a microcassette ed una piccola stampante incorporati. Dispone anche di un limitato word processor

Casio FP-200
È il più economico dei portatili "da viaggio", ma invece di un word processor incorporato, possiede una specie di lavagna elettronica

intercambiabili e conservano le informazioni anche quando vengono disinnestate dal computer. Tutte le espansioni di memoria dei portatili da viaggio sono molto costose e ciò dipende dallo speciale tipo di chip utilizzato, che deve essere a bassissimo consumo.
È bene chiarire, comunque, che i computer portatili non sono stati progettati con l'intento di sostituire i personal computer da tavolo, ma solo per svolgere funzioni ad essi complementari. I loro piccoli visualizzatori LCD, ad esempio, li rendono inadeguati per prolungati lavori di battitura, non consentendo un'agevole lettura dei testi, e la loro "risposta" è molto più lenta di quella degli schermi a raggi catodici. Diversamente dalle tastiere dei micro da tavolo, ampie ed inclinate, quelle dei portatili più economici sono piatte e molto meno confortevoli.
I computer portatili da viaggio stanno ottenendo un notevole successo e, sull'onda del "boom" dei micro, un numero sempre crescente di persone si sta rendendo conto dei grandi vantaggi offerti da questi piccoli gioielli: tra non molto tempo l'uso dei portatili sarà comune come quello dei calcolatori tascabili.



Epson PX-8
È uno dei più sofisticati, poiché è in grado di accedere al software gestionale del tipo CP/M (l'ottimo word processor Wordstar è disponibile su ROM)

Tandy TRS-80 Modello 100 / NEC PC-8201A / Olivetti M10
Sono tre versioni modificate di uno stesso micro. Hanno in comune un eccellente word processor incorporato, una buona serie di interfacce ed un modem a pile prodotto dalla Olivetti

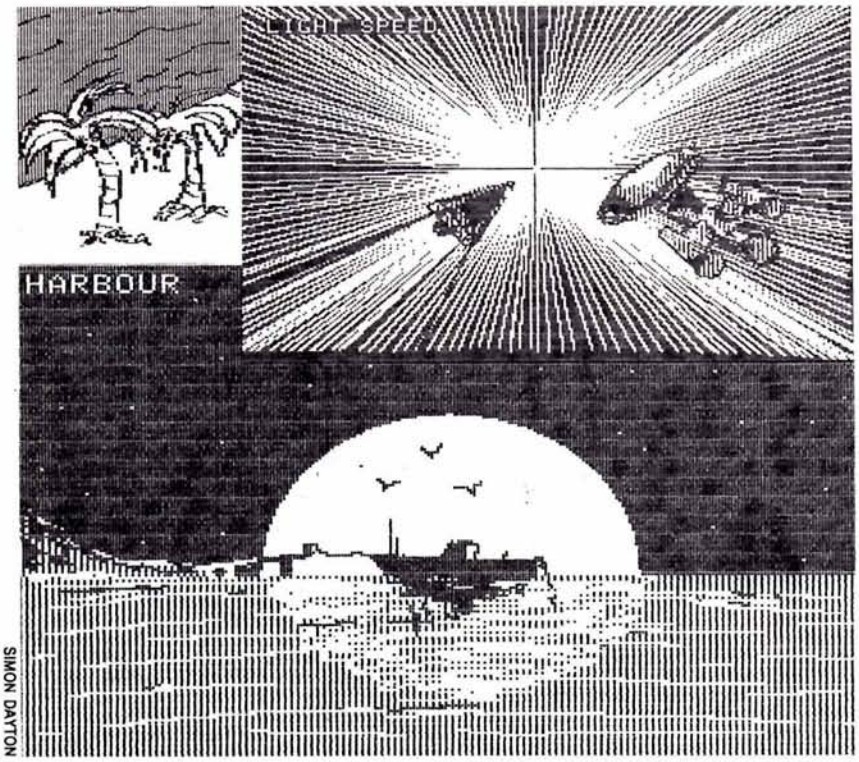
MARCUS WILSON-SMITH

Modello	Mem. Standard	Mem. Massima	Schermo	Peso
Casio FP-200	8k	32K	8 × 20	1,4 Kg
Epson HX-20	16k	32K	4 × 20	1,8 Kg
Epson PX-8	64k	64K + 120K*	8 × 80	2,3 Kg
NEC PC-8201A	16k	64K + 32K*	8 × 40	1,8 Kg
Olivetti M10	8k	32K	8 × 40	1,8 Kg
Tandy TRS-80 Modello 100	8k	32K	8 × 40	1,8 Kg

* Il NEC può usare cartucce RAM da 32K e l'Epson PX-8 "dischi" RAM da 120K.

Prima impressione

Prosegue l'esame sulle capacità grafiche delle stampanti ad aghi



Copia dello schermo
Queste immagini sono state tracciate sullo schermo con una lavagna grafica e, successivamente, trasferite su una stampante Epson FX-80. La loro qualità evidenzia le potenzialità grafiche di questa stampante a matrice d'aghi

Molti home computer possiedono un funzionamento grafico ad alta risoluzione, che permette la creazione di immagini mediante caratteri grafici delle stesse dimensioni dei normali caratteri di testo. In genere, a questa serie di simboli grafici è assegnata una numerazione di codice superiore a 127, poiché i codici da 0 a 127 sono riservati ai caratteri ASCII. Ad esempio, con l'istruzione PRINT CHR$(90) si visualizza un carattere ASCII (in questo caso Z), mentre PRINT CHR$(128) fa comparire sullo schermo un carattere grafico (un rettangolo nero, nel caso del micro Dragon). Per stampare la lettera Z, si userà l'istruzione LPRINT CHR$(90) e, scrivendo LPRINT CHR$(128), sarebbe logico aspettarsi la riproduzione su stampante di un rettangolo nero: purtroppo non è così. Infatti, i caratteri associati ai codici superiori a 127, nei computer di marche diverse, possono non essere gli stessi e, del resto, i fabbricanti di stampanti non possono produrre una stampante specifica per ogni computer sul mercato. Di conseguenza, i codici da 127 a 255 ripetono la sequenza dei caratteri ASCII standard, oppure una serie di simboli grafici non associati ad una precisa marca di computer.
La gamma di stampanti Epson non è corredata di caratteri grafici, ma permette all'utente di modificare la definizione di un qualsiasi carattere ASCII, per ottenere simboli grafici di propria invenzione: basta inviare alla stampante opportune sequenze di codici, chiamate "sequenze di Escape" (vedere alle pagg. 804-805).
La grafica ad alta risoluzione, nei computer, è ottenuta mediante piccoli punti luminosi o pixel. Analogamente, la stampa in alta risoluzione si basa su piccoli punti d'inchiostro. La testina di stampa di una stampante a matrice è costituita da un certo numero di aghi, disposti verticalmente, che si spostano sulla carta durante la stampa. Generalmente, i caratteri sono composti da una serie di punti disposti su una griglia (le dimensioni più comuni sono di 8 × 8 punti), ma su certe stampanti è possibile controllare individualmente ciascun ago della testina di stampa.
La prima operazione da compiere è abilitare sulla stampante il modo grafico, inviando una sequenza di Escape. Per l'Epson FX-80, ad esempio, occorre usare la seguente istruzione:

```
LPRINT CHR$(27);"K";CHR$(N1);(N2);
```

La lettera "K" indica il modo grafico mentre (N1) ed (N2) definiscono la lunghezza della linea grafica, cioè il numero di punti che comporranno ciascuna linea della pagina.
Quando è abilitato il modo grafico standard, la FX-80 può stampare fino a 480 punti per linea (altri modelli consentono risoluzioni comprese tra 576 e 1920 punti per linea). Per definire la lunghezza della linea sono necessari due numeri, poiché il valore massimo di ciascuno di essi è 255: il secondo numero (N2) viene automaticamente moltiplicato per 256 e sommato al primo (N1). Quindi, per stampare sull'Epson FX-80 una linea di 480 punti, si dovranno usare i numeri 1 e 224 (480=256×1+224). Il comando sarà dunque:

```
LPRINT CHR$(27);"K";CHR$(224);CHR$(1);
```

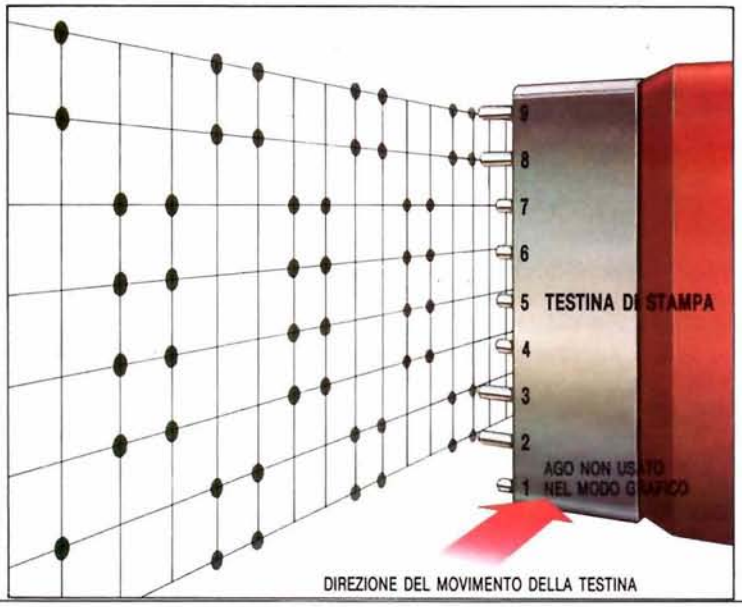
Dopo aver programmato la lunghezza della linea, si dovranno trasmettere le informazioni grafiche. Sebbene la testina di stampa della Epson FX-80 sia dotata di 9 aghi, con la maggior parte dei modi grafici è possibile usarne solo 8 che, a partire da quello più in basso, vengono così numerati: 1,2,4,8,16,32,64 e 128. L'informazione per tutti e 8 gli aghi può quindi essere espressa da un unico numero, compreso tra 0 e 255, che viene inviato alla stampante mediante il comando LPRINT CHR$(X), dove x indica il numero. Perciò se vogliamo azionare l'ago più in basso, dovremo trasmettere alla stampante CHR$(1) mentre per quello

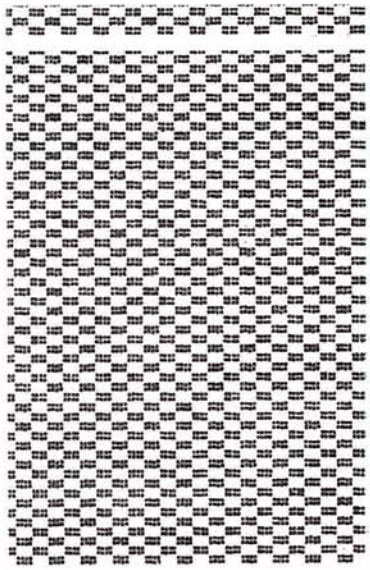
Grafica a punti

La configurazione in figura viene ottenuta inviando alternativamente i numeri decimali 195 e 60 alla stampante, dopo aver abilitato il modo grafico. La tabella mostra come gli aghi interpretino questi numeri in notazione binaria. Un appropriato controllo del trascinamento della carta permette di stampare le linee grafiche in modo sfalsato.

NUMERO DELL'AGO	VALORE DELL'AGO		
9	128	•	o
8	64	•	o
7	32	o	•
6	16	o	•
5	8	o	•
4	4	o	•
3	2	•	o
2	1	•	o
		195	60

• = AGHI IN AZIONE
o = AGHI NON IN AZIONE





STEVE CROSS

più in alto invieremo CHR$(128). Per stampare una combinazione di punti basterà sommare tra loro i numeri degli aghi interessati e questo procedimento si ripete per ciascuna delle 480 posizioni di stampa nella linea. Nell'esempio riprodotto a destra, nel riquadro, vengono usate due configurazioni di punti: CHR$(195) e CHR$(60). Per stampare le prime 4 colonne della linea occorre il comando:

```
LPRINT CHR$(195);CHR$(195);CHR$(60);CHR$(60);
```

Dopo quattro colonne le configurazioni si ripetono per cui la stampa può essere controllata tramite un ciclo FOR...NEXT. È importante notare che l'istruzione CHR$(60), usata nell'esempio, non provoca la stampa del carattere ASCII con codice 60, ma si riferisce alla combinazione grafica degli aghi sulla testina di stampa. La stampante interpreta correttamente il comando, in quanto, precedentemente, è stato selezionato il modo grafico con la sequenza CHR$(27);"K".

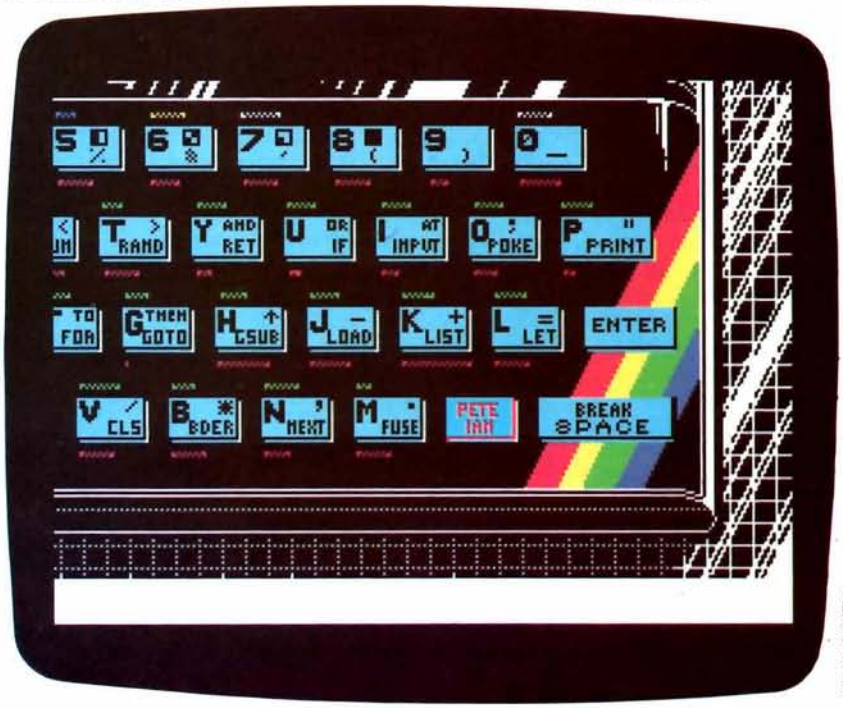
Questo metodo di stampa, chiamato anche stampa "bit image", è valido per la Epson FX-80: altri apparecchi adottano metodi simili, ma non esattamente identici. Comunque, si tratta di una procedura piuttosto laboriosa, usata soprattutto per stampare ripetute configurazioni di punti. Il sistema detto "screen dump" si dimostra molto più comodo: si tratta di un programma che trasferisce su carta l'immagine visualizzata sullo schermo del computer.

Il programma esegue una scansione completa dello schermo, verificando lo stato "on/off" di ciascun pixel. Se un pixel è acceso, il programma ordina ad un ago della testina di stampare un punto sulla carta nella posizione corrispondente. La scansione viene abilitata dalla funzione POINT(x,y) o da comandi simili presenti sulla maggior parte dei micro: se il pixel è acceso, il risultato di questa funzione è 1, altrimenti è 0. Ovviamente, le diverse risoluzioni grafiche dei vari microcomputer comportano l'introduzione di alcune modifiche.

A questo punto potremmo chiederci come un programma di screen dump possa riprodurre, su carta, immagini a colori: la soluzione più comune consiste nell'usare differenti configurazioni di punti per ciascun colore. Un pixel del video di colore nero potrebbe essere stampato usando 4 punti disposti in quadrato; un pixel rosso potrebbe essere rappresentato da 2 punti, mentre uno bianco da un'area priva di punti. La funzione POINT(x,y) produrrà numeri diversi, e quindi diverse configurazioni, a seconda del colore del pixel. Il "dump" dello schermo, mediante programmi scritti in BASIC, è piuttosto lento (fino a 5 minuti per stampare una piccola immagine). I risultati migliori si ottengono con programmi in codice macchina.

Schizzi di colore

Quest'immagine cromatica della tastiera di uno Spectrum è stata ottenuta con una stampante a colori a getto d'inchiostro. Il suo impiego è simile a quello di una stampante ad aghi, in cui , a questi ultimi, sono sostituiti dei getti d'inchiostro



IAN McKINNELL

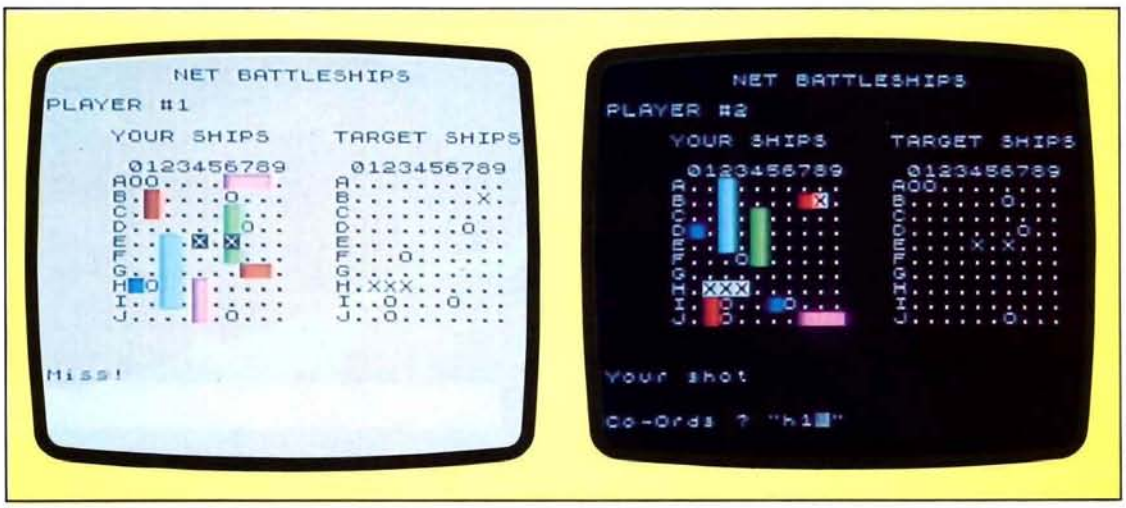
Spectrum in rete

Un gioco per mostrare il funzionamento di una rete ZX Sinclair

La *Battaglia Navale* è un classico gioco da "carta e penna": ciascun giocatore traccia due griglie, che tiene nascoste all'avversario per tutto il gioco: su una disegna le proprie navi, sull'altra segna i colpi sparati alle navi dell'avversario.
Il programma alla pagina seguente deve girare contemporaneamente su due Spectrum Sinclair, dotati dell'Interface 1 e collegati tra loro in "rete". Ciascun giocatore si siede di fronte al proprio schermo ed i due computer si scambiano messaggi riguardanti i colpi sparati ed i risultati ottenuti.
Stabilire l'identità dei due avversari è già di per sé una piccola battaglia: e per poter comunicare tra loro, a ciascuno Spectrum va assegnato un proprio "numero di stazione" all'interno della rete. I due Spectrum iniziano con programmi identici ma, in qualche modo, devono essere indicati i differenti numeri di stazione. Questa operazione viene gestita automaticamente dalla routine alla linea 2000. Quando il programma va in esecuzione, entrambe le macchine dichiarano di essere la stazione 1 sul canale di trasmissione "pubblico". L'apparecchio che invia per primo questa dichiarazione diviene la stazione 1, l'altro la stazione 2. Questo sistema si adatta perfettamente al nostro gioco. Il programma prevede che il giocatore abbinato alla stazione 1 sia il primo a "muovere". Tuttavia, se i due programmi venissero avviati contemporaneamente, il messaggio "Sono la stazione 1" verrebbe inviato ad entrambi i computer ed in tal caso il sistema ZX Net si blocca, finché i due giocatori non premono il tasto BREAK.
Quando la fase d'identificazione è conclusa, i due programmi comunicano facilmente tra loro, attraverso la rete: mentre un giocatore sta decidendo l'obiettivo del proprio colpo, l'altro attende la sua scelta. I due Spectrum si comportano allo stesso modo: uno calcola i risultati del colpo ed invia un messaggio di risposta, l'altro attende la comunicazione e, in base ad essa, aggiorna l'immagine sul video. Il gioco si svolge senza intoppi a patto che i due programmi siano sempre in comunicazione tra loro: l'uno in trasmissione e l'altro in ricezione, alternativamente. Non esistono problemi di temporizzazione: se un giocatore attende prima di inviare il proprio messaggio o si ferma a pensare dopo averne ricevuto uno dall'avversario, lo ZX Net aspetta finché entrambe le stazioni non si dichiarano "pronte" e solo allora trasmette la nuova informazione. Non ci si allarmi, quindi, se un giocatore impiega molto tempo per decidere un colpo o se il programma ritarda ad aggiornare lo schermo.
Un altro punto importante è che i dati trasmessi devono essere il più possibile sintetici. Poiché entrambi i programmi già conoscono il significato dell'informazione, è possibile comunicare mediante brevi codici. Nella *Battaglia Navale*, il programma usa una stringa di due caratteri per rimandare il risultato di un colpo. Il primo carattere è un numero di codice:

1 Colpo a vuoto
2 Colpita una nave
3 Colpita e affondata una nave
4 Colpita e affondata una nave: vinto il gioco

Il secondo indica la classe della nave colpita (o 0 per colpo mancato). Il programma ricevente decodifica l'informazione e visualizza un messaggio appropriato. Questo sistema dà buoni risultati, ma con tempi piuttosto lunghi.



LIZ HEANEY

Disposizione della flotta
La flotta di ciascun giocatore comprende navi di varie dimensioni che possono essere disposte in qualsiasi ordine. Lo schermo mostra la posizione e lo stato delle navi di un giocatore, i colpi sparati dal nemico e, sulla seconda griglia, la posizione e l'effetto dei propri colpi sulle navi del nemico: le "X" indicano i colpi giunti a segno, le "O" quelli andati a vuoto

SVILUPPO DEL GIOCO

Il programma presenta una procedura di avvio piuttosto complicata, dovendo girare separatamente su due Spectrum. Ciascun computer deve caricare il proprio programma, ma si perderà meno tempo caricando un solo programma da una cassetta (o da un Microdrive), trasmettendolo poi all'altro apparecchio tramite la stessa rete. Per effettuare la trasmissione, occorre digitare LOAD*"n";0 e quindi SAVE*"n";0 sul computer che ha il programma in memoria. A questo punto i giocatori stabiliscono a chi tocca il primo colpo: al giocatore che per primo invierà il programma in esecuzione, verrà assegnato il numero di stazione 1 e potrà sparare il primo colpo.

Avviato il programma, i giocatori devono disporre le proprie navi e ciò viene fatto indicando sulla griglia 10×10 la posizione di un'estremità di ogni nave e comunicando in quale direzione si trova lo scafo della nave stessa (in alto, in basso, a destra o a sinistra). Questa operazione è più semplice di quel che sembra. Le flotte sono composte da: 2 Motosiluranti (1 quadretto), 2 Incrociatori (2 quadretti), 2 Corazzate (3 quadretti), 1 Cacciatorpediniere (4 quadretti) e 1 Portaerei (5 quadretti).

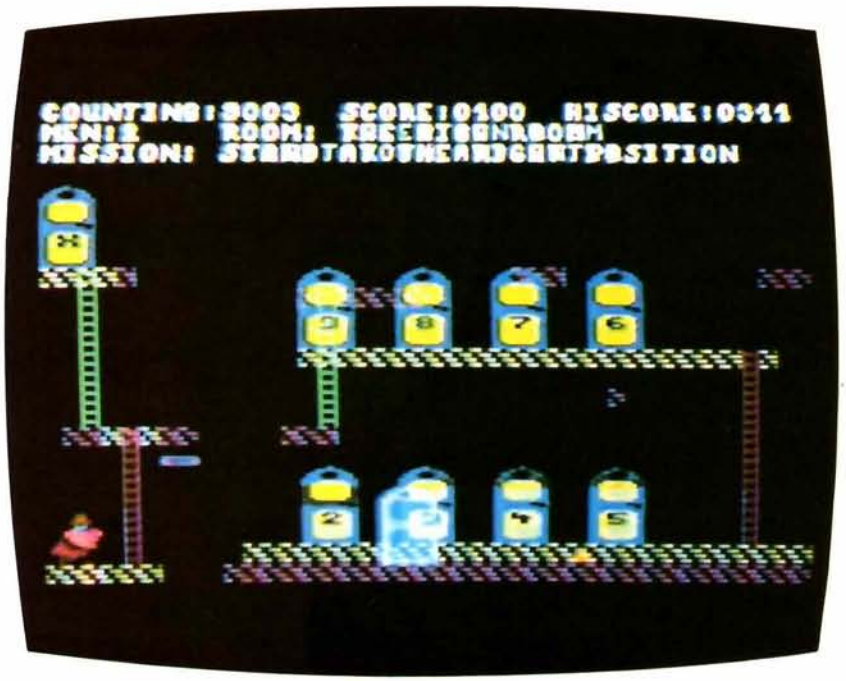
I giocatori sparano a turno un colpo sulla griglia dell'avversario ed il programma valuta i risultati di ciascun colpo: vince chi per primo affonda tutte le navi avversarie. Entrambi i giocatori devono battere RUN per giocare di nuovo ricordando che, chi lo fa per primo, spara per primo!

```
 10 REM ***    Battaglia Navale su Rete    ***
 11 REM
 12 REM ***  per 2 Spectrum con Interface 1 ***
 13 REM ***  Versione 1.6
 15 REM ***  Fase d'avvio
 30 GO SUB 2000:          REM Definizione primo colpo
 40 LET s$="                                ":REM 34 spazi
 50 DIM s(8):             REM Tipo di nave
 60 DIM n$(8,12):         REM Nome della nave
 70 DIM a(10,10):         REM Misure griglia
 80 FOR i=1 TO 8: READ s(i), n$(i): NEXT i
 90 DATA 1,"Motosilurante",1,"Motosilurante",2,"Incrociatore"
    ,2,"Incrociatore"
100 DATA 3,"Corazzata",3,"Corazzata",4,"Incrociatore",5,"Portaerei"
110 LET sc=8
120 GO SUB 3000:          REM Preparazione dello schermo
130 GO SUB 4000:          REM Disposizione delle navi
140 REM *** Ora salta in base al numero della ***
150 REM *** stazione (la n. 1 spara per prima) ***
160 IF sta=2 THEN GO TO 400
200 REM *** Spara un colpo!
210 LET a$="E' il tuo turno": GO SUB 6000
220 GO SUB 7000: IF e=1 THEN GO TO 210
230 OPEN #4;"n";lui
240 PRINT #4;a$
250 CLOSE #4
260 REM Aspetta & ritrasmetti i risultati
270 OPEN #4;"n";lui
280 INPUT #4;a$
290 CLOSE #4
300 LET r=VAL (a$(1 TO 1)): x=VAL (a$(2 TO ))
310 IF r=1 THEN LET a$="A vuoto!": PRINT AT 6+q,18+p;"O";: GO SUB
    6000
320 IF r>1 THEN LET a$="Colpito!!": PRINT AT 6+q,18+p;"X";: GO SUB
    6000
330 IF r>2 THEN LET a$="Hai affondato una nave nemica "+n$(x): GO
    SUB 6000
340 IF r=4 THEN LET a$="Complimenti ... hai vinto!!!": GO SUB 6000:
    STOP
400 REM *** Fuoco nemico
410 LET a$="Tiro del nemico": GO SUB 6000
420 OPEN #4;"n"; lui
430 INPUT #4;a$
440 CLOSE #4
450 LET p=VAL (a$(2 TO ))+1: LET q=CODE (a$)-64
460 LET a$="Il colpo nemico e' "+a$: GO SUB 6000
470 LET x=a(p,q): LET a(p,q)=0
480 IF x=0 THEN LET r=1: GO TO 530
490 LET r=2
500 LET s(x)=s(x)-1
510 IF s(x)=0 THEN LET r=3: LET sc=sc-1
520 IF sc=0 THEN LET r=4
530 LET a$=STR$ (r)+STR$ (x)
540 OPEN #4;"n"; lui
550 PRINT #4;a$
560 CLOSE #4
570 IF r=1 THEN LET a$="E' un colpo a vuoto": PRINT AT 6+q,4+p;"O";
580 IF r>1 THEN LET a$=n$(x)+" colpito": PRINT AT 6+q,4+p;"X";
585 IF r>2 THEN LET a$=a$+" e affodato"
587 GO SUB 6000
590 IF r=4 THEN LET a$="Peccato ... hai perso": GO SUB 6000: STOP
600 GO TO 210
2000 REM *** Assegna i numeri di stazione e il prim colpo
2005 CLOSE #4
2010 OPEN #4;"n";0
2020 PRINT #4;"1"
2030 CLOSE #4
2040 OPEN #4;"n";0
2045 INPUT #4;a$
2050 CLOSE #4
2060 IF a$="1" THEN OPEN #4;"n";0: PAUSE 5: PRINT #4;"2": LET staz=1
2070 IF a$="2" THEN LET staz=2
2080 CLOSE #4
2090 FORMAT "n";staz: LET lui=3-staz: RETURN
3000 REM *** Prepara lo schermo
3010 LET col=0: IF staz=2 THEN LET col=7
3020 PRINT;: BORDER 7-col: PAPER 7-col: INK col: CLS
3030 PRINT TAB 8;"DISPONI LA FLOTTA"
3040 PRINT : PRINT "GIOCATORI #";staz
3050 PRINT : PRINT "      TUE NAVI          SAGOMA NAVI"
3060 PRINT : PRINT "     0123456789        0123456789"
3070 FOR i=1 TO 10
3080 PRINT "     ";CHR$ (i+64);".....   ";CHR$ (i+64);
     ".....  "
3090 NEXT i
3100 RETURN
4000 REM *** Disposizione delle navi
4010 LET a$="Disporre le proprie navi": GO SUB 6000
4020 FOR s=1 TO sc
4030 LET a$=STR$ s+". "+n$(s)+"lunghezza "+STR$ (s(s)): GO SUB 6000
4050 GO SUB 7000: IF e=1 THEN GO TO 4030
4055 IF s(s)=1 THEN LET xd=0: LET yd=0: GO TO 4130
4070 INPUT "A, B, S o D ?";a$: LET xd=3: LET yd=3
4080 IF a$="A" OR a$="a" THEN LET xd=0: LET yd=-1
4090 IF a$="B" OR a$="b" THEN LET xd=0: LET yd=1
4100 IF a$="S" OR a$="s" THEN LET xd=-1: LET yd=0
4110 IF a$="D" OR a$="d" THEN LET xd=1: LET yd=0
4120 IF xd=3 AND yd=3 THEN GO TO 4070
4130 LET l=s(s): LET x=p: LET y=q
4140 IF x<1 OR x>10 OR y<1 OR y>10 THEN LET a$=" Allontana la nave
     dal margine": GO SUB 6000: GO TO 4030
4150 IF a(x,y)<>0 THEN LET a$="Riposiziona la nave": GO SUB 6000:
     GO TO 4030
4160 LET x=x+xd: LET y=y+yd
4170 LET l=l-1: IF l>0 THEN GO TO 4140
4180 LET l=s(s): LET x=p: LET y=q
4190 LET a(x,y)=s: INK s(s): PRINT AT 6+y,4+x;" ";: INK col
4200 LET x=x+xd: LET y=y+yd
4210 LET l=l-1: IF l>0 THEN GO TO 4190
4220 NEXT s
4230 LET a$="Preparati all'azione!!!": GO SUB 6000
4240 RETURN
6000 REM *** Stampa a$
6010 PRINT AT 20,0;s$;AT 20,0;a$: PAUSE 100: RETURN
7000 REM ***Convalida le coordinate
7010 LET e=0
7015 INPUT "Coordinate ?";a$
7020 IF LEN a$<>2 THEN LET e=1: GO TO 7100
7030 FOR i=1 TO 2
7040 LET c=CODE (a$(i TO i)): IF c>=97 AND c<=122 THEN LET a$
     (i TO i)=CHR$ (c-32)
7050 NEXT i
7060 LET q=CODE (a$(1 TO 1)): LET p=CODE (a$(2 TO 2))
7070 IF q<65 OR q>74 THEN LET x=q: LET q=p: LET p=x
7080 IF q<65 OR q>74 THEN LET e=1
7090 IF p<48 OR p>57  THEN LET e=1
7100 IF e=1 THEN LET a$="Reinserire le coordinate": GO SUB 6000:
     RETURN
7110 LET q=q-64: LET p=p-47
7120 RETURN
```

Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Abbiamo visto più volte come la fantasia, nel produrre giochi sempre nuovi, sia un fattore essenziale. Ebbene, anche nei due programmi qui presentati gli autori si sono certamente dati da fare per realizzare qualcosa di originale e divertente.



House of Usher per il Commodore 64

Agli appassionati dei romanzi dell'orrore il titolo di questo gioco ricorderà il racconto "La casa degli Usher" di E. A. Poe, ma la similitudine finisce qui, poiché, in realtà, *House of Usher* assomiglia più al *Minic Miner*, già esaminato in questa sezione.
L'azione, brillante e rapida, ha per scopo l'attraversamento di nove "stanze", infestate da ogni sorta di ostacoli, per raggiungere due locali misteriosi (chiamati x e y). Cosa si trovi in essi non possiamo svelarlo, per non rovinare la sorpresa. Il programma è prodotto dalla Anirog (GB).

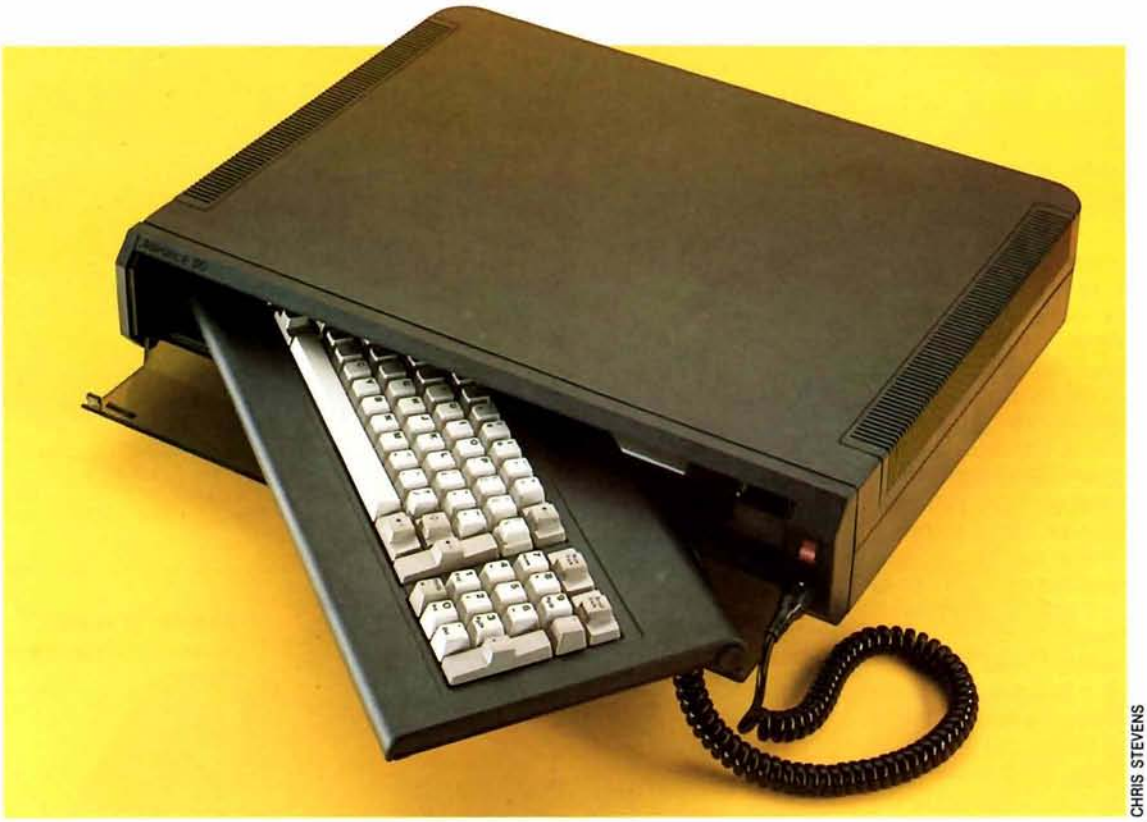


Musgy per lo Spectrum

Questo è forse uno dei giochi per microcomputer più originali finora prodotti. L'ambientazione è collocata negli anni del proibizionismo ed il giocatore assume il ruolo del "Padrino" di una banda di gangster, che deve amministrare i fondi per l'acquisto di armi, per la corruzione di funzionari e via dicendo. Lo svolgimento è veramente insolito e le azioni, accompagnate da una splendida animazione grafica, appaiono sullo schermo sotto forma di "fumetti", come si può notare dall'illustrazione. Il gioco è stato creato dalla "Melbourne House" (GB).

L'Advance 86a

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto



CHRIS STEVENS

L'Advance 86a
L'Advance 86 viene venduto in due versioni: l'Advance 86a, qui in figura, proposto come home computer e l'Advance 86b che è un micro gestionale compatibile con il Personal Computer IBM

Esistono due versioni di questo microcomputer: l'Advance 86a, con una RAM da 128 Kbyte, basato sull'impiego di nastri a cassetta e l'Advance 86b, simile al precedente, ma dotato di una doppia unità a dischetti da 5 pollici e 1/4 e di un BASIC più esteso. Con un "pacchetto" d'espansione, che include le unità a dischetti ed i circuiti di controllo, la versione 86a può esser trasformata nella 86b, ottenendo così un veloce apparecchio gestionale, compatibile con il Personal della IBM, ma ad un prezzo notevolmente inferiore.
L'86b si compone di due parti: una tastiera ed un contenitore. Quest'ultimo, forse più grande di quanto occorra (le dimensioni sono: cm. 52 × 40 × 9,5), è realizzato in materiale plastico, e contiene il microprocessore e le varie schede, oltre a poter accogliere la tastiera in un apposito vano. Tutte le prese dell'apparecchio sono montate sul contenitore, nel quale si trova anche il trasformatore dell'alimentazione.
La tastiera è dotata di un cavo coassiale, che si collega ad una presa DIN a 5 piedini, sopra la quale è situato l'interruttore generale e la relativa spia d'accensione (un diodo LED). La tastiera è certamente una delle migliori, tra quelle usate sugli home computer, e non ha nulla da invidiare a quelle adoperate sugli apparecchi professionali. Essa è dotata di 86 tasti suddivisi in tre gruppi: i tasti alfanumerici principali, 10 tasti di funzione ed una tastierina numerica, utilizzabile anche per controllare i movimenti del cursore.
Ai tasti di funzione sono associati alcuni dei più frequenti comandi del BASIC: RUN, LIST, SAVE, LOAD ecc., ma questi si possono facilmente riprogrammare. Infatti, a ciascun tasto può essere abbinata una stringa, lunga fino a 15 caratteri. Sull'ultima riga dello schermo vengono visualizzati i primi sei caratteri di ciascuna stringa. Agendo sul tasto "Num lock", la tastierina numerica può essere destinata al controllo del cursore, facilitandone il posizionamento.
La custodia del microprocessore racchiude una scheda di ridotte dimensioni, che contiene il microprocessore Intel 8086 a 16 bit (compatibile, ma più veloce dell'8088 usato dal Personal IBM) e una RAM di 128 Kbyte. Sulla scheda c'è un numero sufficiente di zoccoli liberi, utilizzabili per raddoppiare la RAM. La memoria disponibile per il BASIC è limitata a 62 Kbyte, ma si dimostra più che sufficiente per la maggior parte delle applicazioni.



Tastiera "intelligente"
La tastiera del Personal IBM viene spesso considerata la migliore sul mercato: quella dell'Advance ne ricalca molti degli aspetti, ma alcuni tasti hanno una diversa posizione e, fatto insolito, esistono due tasti Return

L'Advance è dotato di numerosi connettori ed interfacce: oltre ad una presa di corrente, utile per collegarvi il televisore o il monitor, ci sono due uscite per i segnali "video composito" ed RGB (da usare con un monitor). Su una terza uscita è disponibile il segnale video adatto ad un normale televisore. Inoltre, ci sono: un'interfaccia parallela (standard Centronics) da usare per una stampante, due porte per joystick e una presa DIN a 5 poli per un registratore a cassette. Se il sistema viene potenziato, si dispone anche di una porta RS232, adatta ad una stampante seriale oppure ad un modem. Tutte le interfacce usano conduttori tipo IBM.
Nel modo testuale, lo schermo dell'Advance prevede 16 colori (normali o lampeggianti) su 25 righe di 40 caratteri oppure, a scelta, su 25 di 80 caratteri (il formato usato dal PC della IBM), nel qual caso, però, la lettura può essere difficoltosa se non viene impiegato un monitor. Le scritte abbinate ai tasti di funzione, visualizzate sull'ultima riga dello schermo, possono essere cancellate, permettendo così un uso completo del video. Il modo grafico a media risoluzione (200 × 320 pixel o 25 righe di 40 caratteri) prevede soltanto quattro colori, mentre in alta risoluzione il display grafico, in bianco e nero, ha una risoluzione di 200 × 640 pixel o di 25 righe per 80 caratteri. Nella RAM, usando il formato 25 × 40, si possono immagazzinare e richiamare ben 7 "paginate" video (nel modo a 80 colonne, tale numero scende a 4): questa possibilità è molto utile per memorizzare menu, brani di spiegazione, ecc. di frequente consultazione. L'uso e la gestione dei colori presenta alcune restrizioni: nel modo testuale, per le immagini di "primo piano" e per il contorno, è disponibile l'intera gamma di colori, ma per lo sfondo si può scegliere soltanto uno fra gli 8 colori. Nel modo a media risoluzione, la situazione si fa ancora più complessa: i colori vengono suddivisi in due gruppi o "tavolozze" e se ne possono selezionare 4, scegliendoli però solo da una delle due tavolozze.
L'Advance 86a possiede solo i seguenti comandi grafici: PSET (seleziona il colore di un singolo pixel dello schermo) e LINE (permette di tracciare con rapidità linee e quadrati). Purtroppo altri utili comandi, quali CIRCLE (per tracciare cerchi), PAINT (per colorare l'interno delle figure chiuse), DRAW (per definire e tracciare qualsiasi sagoma), GET e PUT (che permettono di trasferire aree del video grafico all'interno di matrici e di riportarle sullo schermo con dimensioni e colori diversi), sono presenti solo nella versione BASIC dell'86b, su dischetto.
Il set di caratteri dell'Advance comprende, oltre alla normale gamma di caratteri ASCII, simboli matematici e una grafica a "blocchi", comprensiva dei "semi" delle carte da gioco, di note musicali, di lettere dell'alfabeto greco e così via. Il sistema prevede anche la creazione di UDG, ossia caratteri definiti dall'utente.
Nel complesso, il BASIC dell'Advance è un dialetto più che soddisfacente. Sebbene manchino alcune funzioni "strutturate", presenti in altri home computer, esso si dimostra veloce e facile da usare. Possiede istruzioni di numerazione e rinumerazione automatica delle linee di programma ed i comandi PRINT USING (utilissima per presentare ordinatamente le informazioni sullo schermo) e SWAP (che permette di scambiare i valori di due variabili). Le possibilità di generare suoni sono discrete, anche se non eccezionali: anche in questo caso, la serie completa di comandi è contenuta soltanto nella versione BASIC su dischetto.
L'uso delle cassette è semplice ed i programmi in BASIC e in codice macchina vengono caricati con un semplice comando LOAD. Aggiungendo la lettera "R", alla fine di questo comando, i programmi vengono automaticamente avviati, subito dopo il caricamento.
L'Advance è dotato di un eccellente "screen editor". Selezionando il funzionamento di controllo del cursore, sulla tastierina numerica, l'operatore può facilmente spostare il cursore in qualsiasi punto dello schermo, onde effettuare correzioni ed inserzioni nel testo.
In definitiva, l'Advance non ha da temere dal confronto con altri home computer.

Interfaccia per stampante parallela (Centronics)

Uscita RGB
Permette di collegarsi a un monitor a colori di alta qualità



Uscita video composito
Per collegamenti con monitor monocromatici o a colori

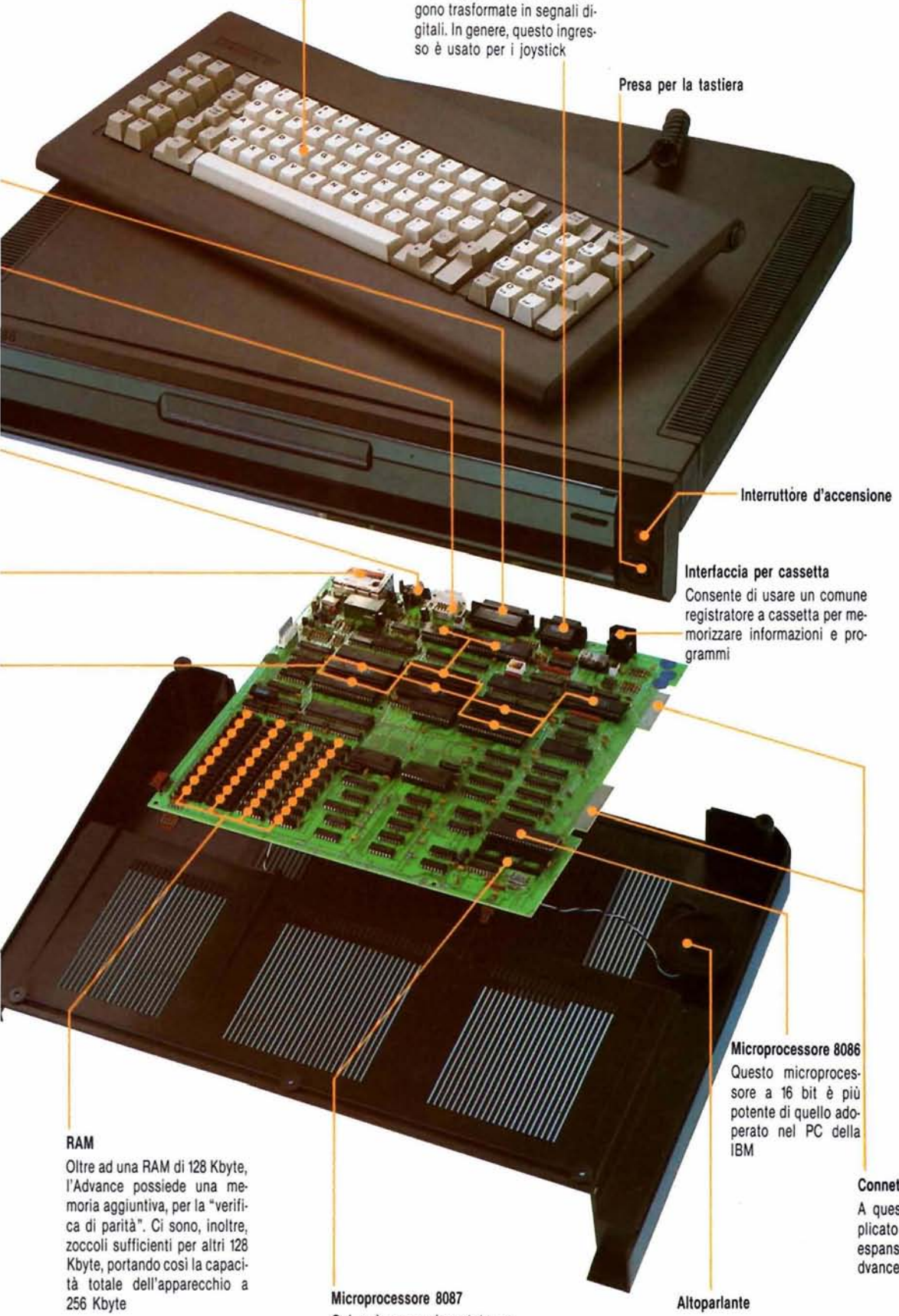
Modulatore TV ed uscita
Permette di usare un normale televisore

Chip ULA
L'adozione di questi 9 chip ULA appositamente progettati, permette di tenere basso il costo del computer



ADVANCE O IBM?

L'Advance viene venduto come home computer, ma può essere portato ai livelli di un computer da ufficio. Grazie al "pacchetto" d'espansione, il sistema diviene totalmente compatibile con il Personal della IBM. Una prova di tale compatibilità è stata ottenuta adoperando sull'Advance il programma "Simulatore di volo" scritto dalla Microsoft appositamente per il PC dell'IBM. Il modulo di espansione viene collocato sopra l'unità centrale dell'86a



Tastiera tipo IBM

Joystick e Convertitore A/D
Le tensioni analogiche applicate a questo connettore vengono trasformate in segnali digitali. In genere, questo ingresso è usato per i joystick

Presa per la tastiera

Interruttore d'accensione

Interfaccia per cassetta
Consente di usare un comune registratore a cassetta per memorizzare informazioni e programmi

RAM
Oltre ad una RAM di 128 Kbyte, l'Advance possiede una memoria aggiuntiva, per la "verifica di parità". Ci sono, inoltre, zoccoli sufficienti per altri 128 Kbyte, portando così la capacità totale dell'apparecchio a 256 Kbyte

Microprocessore 8087
Qui può essere innestato un microprocessore ad alta velocità per funzioni matematiche

Altoparlante

Microprocessore 8086
Questo microprocessore a 16 bit è più potente di quello adoperato nel PC della IBM

Connettori di estensione
A questi connettori viene applicato un apposito modulo di espansione, che trasforma l'Advance 86a in 86b

ADVANCE 86a

DIMENSIONI

mm. 95 x 400 x 520

CPU

8086 Intel a 16 bit, usata a 4,77 MHz

MEMORIA

128 Kbyte di RAM (espandibile a 256K), 64 Kbyte di ROM

VIDEO

25 righe di 40 caratteri, oppure 25 righe di 80 caratteri nel modo testuale, con 16 colori. 200 x 320 pixel (a 4 colori) oppure 200 x 640 pixel (bianco e nero) in modo grafico

INTERFACCE

Uscite RGB e video composito per monitor, connettore per 2 joystick, interfaccia parallela Centronics, porta per registratore a cassetta, porta seriale RS232 (soltanto nella versione 86b)

LINGUAGGI FORNITI

BASIC su ROM (nella versione 86a) o su dischetto (versione 86b)

TASTIERA

Tipo macchina da scrivere, con 84 tasti, incluso una tastierina numerica e 10 di tasti di funzione

DOCUMENTAZIONE

I manuali (uno operativo ed uno relativo al BASIC) sono adeguati, sebbene manchino diversi dettagli

PRO

La tastiera è una delle migliori in commercio, specialmente nel settore degli home computer e anche lo "screen editor" è di ottima qualità. Il BASIC su dischetto è molto soddisfacente e la memoria disponibile è molto più ampia di quella presente su altri home computer. La compatibilità con il Personal Computer della IBM è un grosso vantaggio

CONTRO

Il BASIC su ROM è privo di alcuni comandi, che sarebbero utili per la generazione di effetti grafici ed audio. Il numero di colori disponibile in alcuni dei modi grafici è limitato

Atterraggio lunare

Con poche linee di BASIC realizziamo una versione del più classico gioco "arcade"

Forza d'attrazione

Qui di seguito vengono dati i valori approssimativi (in metri al secondo quadrato) della forza d'attrazione del Sole e di alcuni pianeti del sistema solare. Questi valori possono essere usati per la variabile g del programma, (vedere la linea 20). Sarebbe ovviamente da pazzi o suicidi cercare di atterrare veramente su corpi celesti come il Sole o Giove, ma col nostro programma possiamo anche provarci, per sperimentare il gioco

Sole (Sun) = −274 m/s²
Mercurio (Mercury) = −3,7 m/s²
Venere (Venus) = −8,8 m/s²
Terra (Earth) = −9,8 m/s²
Luna (Moon) = −1,6 m/s²
Marte (Mars) = −3,8 m/s²
Giove (Jupiter) = −25 m/s²
Saturno (Saturn) = −10,4 m/s²
Urano (Uranus) = −10,4 m/s²
Plutone (Pluto) = −4 m/s²

Nel gioco *Atterraggio Lunare*, il giocatore pilota un piccolo veicolo spaziale durante la delicata fase di atterraggio sulla superficie della luna (o di un pianeta). Il computer di bordo è temporaneamente "fuori uso", quindi il giocatore deve provvedere manualmente all'atterraggio, frenando la discesa con brevi accensioni dei retrorazzi. Il modulo dovrà toccare la superficie a velocità ridotta e questa manovra richiederà un preciso bilanciamento tra la velocità di caduta ed il consumo delle limitate riserve di carburante. Se lo si programma in modo che l'accensione dei retrorazzi, per 2 secondi, porti ad una riduzione della velocità di discesa di 10 Km/h, l'atterraggio non presenterà alcuna difficoltà. L'obiettivo del programma è quello di creare una simulazione che riproduca il più fedelmente possibile, anche se in modo semplificato, le reali condizioni d'atterraggio di un modulo lunare.

Esaminiamo in modo specifico i problemi riguardanti il modulo:

- Il modulo lunare è soggetto alla forza di gravità del pianeta. Per atterrare, quindi, esso dovrà esercitare una forza opposta, accendendo i propri retrorazzi.
- Il modulo d'atterraggio ha una data massa (o peso) composta dal peso del modulo e da quello del carburante. Maggiore è il peso complessivo, minore è l'effetto frenante dei retrorazzi. Via via che il carburante viene consumato, però, il peso del modulo diminuisce!

Per ricreare condizioni vicine alla realtà, occorre impiegare nel programma una serie di equazioni più o meno complesse, che coinvolgono parametri quali l'accelerazione, la massa, la velocità e così via. Nel nostro caso,



prenderemo in esame un programma che usa equazioni abbastanza semplici.
Il dato principale che dobbiamo conoscere è l'altezza del modulo rispetto al suolo. Ovviamente, essa cambia continuamente, sia per effetto della gravità sia per l'accensione dei retrorazzi che tendono ad allontanare la navetta dalla superficie lunare. Per ricavare la posizione "istantanea", suddividiamo il "tempo" in una serie di fasi o periodi.
Per ciascun periodo, possiamo calcolare gli spostamenti fatti dal modulo ed i suoi cambiamenti di massa e di velocità. La lunghezza delle fasi viene stabilita dal programmatore: più brevi saranno, maggiore sarà la precisione della simulazione.
Una volta introdotto il concetto di periodo, possiamo facilmente scrivere le equazioni necessarie. La velocità viene misurata in chilometri orari ed ecco l'equazione per calcolare la distanza:

Distanza = Tempo × Velocità

Pertanto, durante ciascun periodo è possibile calcolare i movimenti del modulo verso l'alto o verso il basso semplicemente moltiplicando la sua velocità per la lunghezza del periodo (inteso come unità). Successivamente, possiamo controllare la velocità, opponendo all'accelerazione prodotta dal campo gravitazionale del pianeta, brevi e ripetute accensioni dei retrorazzi.
L'accelerazione dovuta alla gravità è sempre costante (la variabile g nel programma) e dipende dalla massa del pianeta.
L'illustrazione alla pagina precedente fornisce i dati gravitazionali dei pianeti del Sistema Solare, ma il programma potrà usare qualsiasi valore o prevedere una generazione casuale del valore di g per rendere più difficile il gioco.
La simulazione dei retrorazzi è molto più complicata. Nella versione che stiamo descrivendo, il giocatore può consumare da una a nove unità di carburante per periodo ed il programma calcola l'accelerazione prodotta, sempre tenendo conto della massa della navetta. La formula dipende dalla potenza dei retrorazzi e dal tipo di carburante usato. I valori scelti nel nostro esempio rendono l'atterraggio estremamente difficile, ma possiamo modificarli per capire come ciascuno di essi influenzi l'andamento globale del gioco. Addirittura, la simulazione può essere modificata in modo che l'azione si svolga in "tempo reale": questa frase, oggi piuttosto abusata, generalmente significa che il programma (gioco, simulazione, ecc.) gira continuamente, senza fermarsi per attendere i comandi dall'utente. In sostanza, si tratta di impiegare dei comandi INKEY$ (o GET), anziché il consueto INPUT.
Ovviamente, se questa è la strada scelta, il gioco sarà più emozionante, poiché il giocatore non avrà tempo di ponderare, per un periodo lungo a piacimento, l'azione più conveniente da intraprendere in base alla quantità di carburante ancora disponibile. Nel nostro programma c'è un ciclo iterativo che viene eseguito una volta per ogni periodo o fase. Uguagliando il periodo al tempo impiegato dal programma per eseguire un singolo passaggio nel ciclo, otterremo una simulazione la cui durata si avvicina effettivamente al tempo impiegato in un atterraggio "vero". Un tale obiettivo, tuttavia, va bene se si tratta di una simulazione ma, nel caso di un gioco, i tempi risultano troppo lunghi.
Il programma di base può essere migliorato in molti modi: in primo luogo si potrebbe introdurre una rappresentazione grafica della discesa, oppure si potrebbe visualizzare una serie di quadranti (altimetro, indicatore di carburante, ecc.). In aggiunta, potremmo prevedere uno scorrimento del sottofondo (il cielo stellato) per rendere più realistiche le immagini. Nel "vuoto" la navetta non sarà soggetta a turbolenze dell'aria, ma una volta all'interno dell'atmosfera potremmo introdurre anche questa componente, per vivacizzare il gioco. Le versioni più sofisticate di questo gioco prevedono diverse basi di atterraggio, magari situate in fondo a gallerie o crateri, e la discesa in queste aree richiede una grande perizia.
Oggi, forse, l'*Atterraggio Lunare* può sembrare un gioco di tipo "arcade" superato, ma per molte persone ha rappresentato il primo approccio ad una simulazione per computer. Inoltre, lo sviluppo di un programma di un atterraggio lunare può essere adoperato come base per la creazione di altri nuovi progetti.

ALLUNAGGIO!

```
10 REM ****     Gioco Navetta Lunare     *****
20 LET g=-1.6
30 LET t=1
40 LET f=1000
50 LET v=0
60 LET h=2000
70 LET m=2000+f
80 LET g=g*t
100 REM Aggiorna lo schermo
110 PRINT AT 0,0
120 PRINT "          Navetta Lunare"
130 PRINT : PRINT "Altezza.....";INT h;"  "
140 PRINT : PRINT "Velocita'.....";INT v;"  "
150 PRINT : PRINT "Carburante....";f; "  "
160 PRINT
165 IF h<0 THEN GOTO 400
170 IF f<=0 THEN LET f=0: PRINT "***CARBURANTE TERMINATO     ": GOTO 190
180 PRINT "Pozenza dei razzi (da 0 a 9):"
190 LET b=0: IF f>0 THEN LET a$=INKEY$: IF a$<>"" THEN LET b=VAL a$
200 IF b>f THEN LET b=0
210 LET h=h+v*t
220 LET v=v+g
230 LET v=v+(b*3000)/m
240 LET f=f-b: LET m=m-b
250 FOR i=1 TO 50: NEXT i
300 GOTO 110
400 REM Contatto con la crosta lunare
410 IF v>-10 THEN PRINT "***Atterraggio Perfetto... Ottimo!": GOTO 500
420 IF v>-20 THEN PRINT "*** SBAMM!... la navetta e'distrutta,
    ma l'equipaggio e' sopravvissuto!": GOTO 500
430 PRINT "CRASH!...Navetta distrutta...Nessun superstite"
440 PRINT : PRINT "Hai creato un nuovo cratere lunare,
    largo "; INT (-v*2.1);" Km."
500 PRINT : PRINT "Giochi ancora (S/N) ? ";
510 LET a$=INKEY$: IF a$="" THEN GOTO 510
520 IF a$="s" OR a$="S" THEN RUN
530 IF a$<>"n" AND a$<>"N" THEN GOTO 510
540 CLS: STOP
```

Basic & Basic

Quella a fianco è la versione adatta allo Spectrum: sugli altri apparecchi le istruzioni LET possono essere omesse. Sul Micro BBC, sostituire la linea 110 con la seguente:

110 PRINT TAB(0,0)

Sostituire, nelle linee 190 e 510, INKEY$ con INKEY$(0), mentre nella linea 190 cambiare VAL a$ con VAL(a$). Nelle linee 130 e 140 sostituire INT h e INT v con INT(h) e INT(v).
Sul Commodore 64 e sul Vic-20, sostituire la linea 110 con la seguente:

110 PRINT CHR$(19)

Sostituire nelle linee 190 e 510 LET a$=INKEY$ con GET a$. Sostituire nella linea 190 VAL a$ con VAL(a$). Sostituire INT h e INT v nelle linee 130 e 140 con INT(h) e INT(v).

Sull'Oric/Atmos, sostituire la linea 110 con la seguente

110 PRINT @0,0

Sostituire nelle linee 190 e 510 INKEY$ con KEY$. Sostituire nella linea 190 VAL a$ con VAL(a$). Sostituire INT h e INT v nelle linee 130 e 140 con INT(h) e INT(v)

Ci vuole stile...

Non basta qualche semplice REM per documentare in modo adeguato un programma

Consideriamo la prima versione (Listato 1) del programma a pagina seguente: ad un primo esame la sua funzione è piuttosto oscura. Il programma prevede l'immissione di due numeri, li moltiplica con due altri numeri, somma tra loro i risultati e stampa la risposta: lo scopo delle operazioni non è affatto evidente. Osserviamo adesso la seconda versione del programma (listato 2): tutto appare più comprensibile, benché privo di un titolo, di linee di commento (le REM del BASIC), e di una documentazione esterna.
Esaminiamo ora, in modo più dettagliato, le differenze tra queste due versioni. Innanzi tutto, i numeri dall'oscuro significato del primo listato sono stati sostituiti con nomi (ANNO e MESE). I numeri il cui valore non cambia durante l'esecuzione del programma, prendono il nome di "costanti". Alcuni linguaggi, come il PASCAL, hanno una particolare notazione per le costanti (nel Listato 2, le due costanti vengono definite separatamente dalle variabili), mentre altri linguaggi, come il BASIC, ne sono privi (le linee 10 e 20 del programma BASIC usano delle variabili per definire le costanti). Assegnare dei nomi alle costanti è utile solo se queste ultime vengono usate con frequenza: in caso contrario è più comodo usare linee di commento.
Una seconda differenza è che alle variabili sono stati assegnati nomi chiari ed estesi per evitare che si possano confondere, per esempio, "A" ed "e" sono stati sostituiti con "ANNO" e "insecondi", scelti perché lunghi meno di 10 caratteri e perché i primi due caratteri sono diversi tra loro.
È infatti una buona abitudine assegnare alle variabili nomi che richiamino il ruolo da esse svolto nel programma. Ad esempio, potremmo assegnare il nome CICLO (invece che C o I) ad un contatore di ciclo iterativo e nomi altrettanto appropriati al primo ed all'ultimo valore del contatore che dovranno venire inseriti in una costante o in una variabile. Quindi il seguente ciclo:

```
FOR I = 1 TO 10 ... NEXT J
```

potrebbe essere scritto nel seguente modo:

```
FOR CICLO = DA UNO TO DIECI ... NEXT CICLO
```

Ovviamente, l'uso di nomi lunghi, per le variabili, implica più tempo di battitura ed occupa più memoria, ma rende i programmi più comprensibili e più veloce la fase di "debugging". Se il linguaggio di programmazione usato riconosce soltanto i primi due caratteri dei nomi, si dovranno accuratamente evitare accidentali ripetizioni. Di conseguenza, due nomi di variabili quali, ad esempio, CODART e COSTO, possono apparire diversi al programmatore, ma del tutto identici al computer.
Un'altra differenza sostanziale tra i due listati è che il secondo visualizza lunghi ed espliciti messaggi tanto per le richieste di input quanto per le descrizioni di output (le linee PRINT in BASIC e "write" in PASCAL). Ciò presenta due vantaggi: rende il programma più leggibile anche quando i nomi delle variabili sono semplici lettere e, cosa ancor più importante, lo rende comprensibile anche a coloro che lo esaminano per la prima volta.

Struttura del programma

Chi programma in PASCAL sa bene quanto sia importante poter redigere con chiarezza un programma sullo schermo. Semplici accorgimenti, come l'introduzione di capoversi e di linee bianche o l'uso misto di maiuscole e minuscole, permettono di trasformare un'impenetrabile sequenza di simboli in un listato chiaro e decifrabile. L'aspetto "estetico" di un programma assume un'importanza primaria, se si usano strutture del tipo FOR... NEXT, WHILE... WEND, REPEAT... UNTIL, ecc., soprattutto quando esse sono "nidificate".
Purtroppo, soltanto una piccola parte degli interpreti BASIC consente accorgimenti di questo tipo e, peggio ancora, in alcune versioni di BASIC le eventuali spaziature vengono automaticamente eliminate. Al contrario, quando si adoperano linguaggi compilati, la redazione del programma "sorgente" viene fatta ricorrendo ad un text editor o ad un word processor, e può esser data un'impostazione molto più chiara. In certi casi, esistono specifici programmi di formattazione del testo, che provvedono in modo automatico a curare l'aspetto finale del listato.
Una versione BASIC che si distingue dalle altre è quella del Micro BBC: esiste addirittura un comando specifico (il LISTO) per inserire automaticamente opportune spaziature nel programma, in modo da consentire una rilettura molto più agevole.
In una situazione così ingarbugliata, conviene stabilire una propria convenzione, che tenga conto dei limiti imposti dal linguaggio usato. Comunque, le righe di commento restano il principale strumento di documentazione di un programma, anche se il loro impiego è regolato a seconda del linguaggio

impiegato. In BASIC si usa l'istruzione REM, che deve precedere il commento: l'interprete ignora tutto ciò che si trova dopo di essa fino al termine della linea. In altri linguaggi (PASCAL, PL/1, PROLOG, ecc.) i commenti vengono indicati dai simboli /*...*/ (a volte {...}) e tutto ciò che si trova al loro interno viene ignorato dal compilatore: questo sistema permette di scrivere commenti su più linee ma, se si dimentica di digitare il simbolo di "fine commento", tutto il resto del programma verrà considerato un commento e, di conseguenza, ignorato!

I commenti devono rappresentare sintetici ma efficaci messaggi da inserire nei punti "cardine" del programma: nella definizione di costanti, nell'inizializzazione di variabili, nella definizione di una funzione, o all'inizio di una nuova procedura (o subroutine); oppure quando il codice è molto complicato. La collocazione migliore è quella immediatamente precedente la sezione cui si riferiscono, oppure a fianco di alcune delle stesse linee che compongono la sezione, purché non interferiscano con la lettura della sua struttura logica. Attenzione, però, a non esagerare: commenti lunghi e verbosi renderanno ancor più difficile la comprensione degli algoritmi che compongono le varie parti del programma. Il Listato 3 mostra alcuni buoni esempi di documentazione interna.

La preparazione della documentazione esterna, intesa come manuali o annotazioni di corredo, comporta un lavoro difficile e noioso, ma la sua consultazione può far risparmiare tempo e fatica, soprattutto quando si deve revisionare un vecchio programma o correggerne uno appena terminato. Tuttavia, se il programma non è eccessivamente lungo e possiede una buona documentazione interna, solo raramente il programmatore dovrà ricorrere alla consultazione dei manuali. La documentazione per l'utente è tutt'altra cosa e questo argomento viene trattato a parte e con maggior approfondimento.

Alcuni linguaggi della "quinta generazione" sono in grado di produrre automaticamente una specie di documentazione, basandosi su informazioni contenute nelle fasi di sviluppo progettuale del programma come, del resto, anche la documentazione scritta a mano dal programmatore, si basa essenzialmente sulle annotazioni scritte durante il lavoro di progettazione.

Durante lo sviluppo di un nuovo programma, conviene conservare ordinatamente tutte le annotazioni fatte nel corso dell'analisi, compresi i progetti degli algoritmi e i diagrammi di flusso. È molto importante conservare anche il diagramma di flusso definitivo, ossia quello usato per scrivere il codice, ed eseguire una stampa del listato. Un'ottima regola è quella di indicare, sul listato della versione finale, il nome del programma, il numero della versione e la data dell'ultimo intervento fatto: tutto ciò facilita qualsiasi intervento successivo e l'individuazione (magari tra diverse decine di versioni del medesimo programma) dell'ultima versione definitiva.

LA DOCUMENTAZIONE INTERNA

Listato 1

BASIC

```
(a) 10 INPUT A, B
    20 C=A*31536000
    30 D=B*2592000
    40 E=C+D
    50 PRINT E
```

PASCAL

```
(b) program abcde (input,output);
    var a,b,c,d,e:integer;
    begin
    read (a,b);
    c:=a*31536000;
    d:=b*2592000;
    e:=c+d;
    writeln(e);
    end.
```

Listato 2

BASIC

```
(a) 10 ANNO=31536000
    20 MESE=2592000
    30 PRINT "Inserisci la tua eta' (con anni e mesi separati da virgola) ";
    40 INPUT NUANNI, NUMESI
    50 ASEC=NUANNI*ANNO
    60 MSEC=NUMESI*MESE
    70 ETAINSEC=ASEC+MSEC
    80 PRINT "La tua eta' in secondi e' (all'incirca)  ";ETAINSEC
```

PASCAL

```
(b) program etainsecondi (input,output);
    const
      aanno=31536000;
      amese=2592000;
    var
      nanni,nmesi,asec,msec,etainsec:integer;
    begin
      write('Inserisci la tua eta' (con anni e mesi separati da virgola) ');
      read (nanni,nmesi);
      asec:=nanni*aanno;
      msec:=nmesi*amese;
      etainsec:=asec+msec;
      writeln('La tua eta' in secondi e' (all'incirca) ',etainsec);
    end.
```

Listato 3

BASIC

```
(a) 10 REM "ETA' IN SECONDI"   Settembre 1984
    20 REM Il programma riceve l'eta' in anni e mesi (a,m)
    30 REM ed effettua una conversione approssimata (mese = 30 giorni)
    40 REM per calcolare l'eta' in secondi.
    50 REM
    60 AANNO=31536000:REM        secondi in 365 giorni
    70 AMESE=2592000:REM         secondi in 30 giorni
    80 PRINT "Inserisci la tua eta' (con anni e mesi separati da virgola) ";
    90 INPUT NANNI, NMESI
   100 REM l'eta' in secondi e' (eta' in anni * secondi in un anno) piu'
       (mesi dall'ultimo compleanno * secondi in un mese)
   110 ASEC=NANNI*AANNO
   120 MSEC=NMESI*AMESE
   130 ETAINSEC=ASEC*MSEC
   140 PRINT "La tua eta' in secondi e' (all'incirca) ";ETAINSEC
```

PASCAL

```
(b) program etainsecondi (input,output);
    /*   Settembre 1984
       legge l'eta' in anni e mesi (a,m) ed effettua una conversione
       approssimata (mese = 30 giorni) per ottenere l'eta' in secondi.    */

    const
      aanno=31536000;     /* secondi in 365 giorni */
      amese=2592000;      /* secondi in trenta giorni */
    var
      nanni,nmesi,asec,msec,etainsec:integer;
    begin
      write('Inserisci la tua eta' (con anni e mesi separati da virgola) ');
      read (nanni,nmesi);
      /* l'eta' in secondi e' (eta' in anni * secondi in un anno) piu' (mesi
         dall'ultimo compleanno * secondi in un mese)                       */
      asec:=nanni*aanno;
      msec:=nmesi*amese;
      etainsec:=asec+msec;
      writeln ('La tua eta' in secondi e' (all'incirca) ',etainsec);
    end.
```

Apocalypse

Mettiamo alla prova le nostre abilità tattiche e strategiche

Una delle più frequenti critiche, mosse nei confronti dei giochi da computer, è che questi tendono a far isolare "socialmente" l'individuo: benché la pubblicità spesso mostri intere famiglie raccolte a giocare attorno ad un home computer, in realtà la maggior parte dei giochi è, in genere, progettata per un solo giocatore.

Apocalypse, al contrario, si basa sulla rivalità tra avversari e richiama alla mente giochi da tavolo tradizionali, giocati da più persone. Addirittura, la versione per il Micro BBC prevede fino ad un massimo di 15 giocatori. È un vero gioco collettivo, e non un semplice passatempo nel quale i vari contendenti partecipano a turno ad un gioco per una sola persona.

Apocalypse, in origine un gioco da tavolo simile a *Diplomacy*, è stato tradotto in versione elettronica perché ben si adatta alle potenzialità grafiche e di elaborazione numerica dei microcomputer. La versione base presenta 4 "teatri di guerra": Europa, Caraibi, Gran Bretagna e Londra, ma è possibile acquistare versioni più estese che prevedono operazioni militari in Sud Africa, in Oriente, nell'Artico, negli USA, nel Sud-Est Asiatico. Altre ancora riproducono le condizioni di battaglie realmente avvenute durante la Seconda Guerra Mondiale, oppure offrono campi di battaglia più originali (come quelli delle campagne di guerra degli antichi romani o di Napoleone). Tutte queste estensioni sono contenute in tre cassette, da abbinare al gioco base.

Lo schermo viene suddiviso in quadrati, organizzati in matrici di 20 × 20 (Spectrum) o 40 × 20 (BBC). I quadrati blu rappresentano il mare mentre gli altri colori indicano le diverse caratteristiche geografiche: aree urbane o rurali, città, montagne o deserti. Esistono anche simboli per individuare le diverse forze in campo, ma la grafica dipende molto dal computer usato. Quella dello Spectrum è meno curata, ma più chiara: è possibile lasciare i quadrati vuoti o colorarli per evidenziare il territorio occupato da una forza attaccante. La grafica del BBC è più attraente, ma i simboli associati a ciascun giocatore tendono a coprire eccessivamente l'area di gioco.

I giocatori, a turno, dispongono le proprie forze sul campo di battaglia e, tramite il computer, possono muovere eserciti e navi, ingaggiare scontri e ritirarsi. I movimenti vengono programmati selezionando le opzioni desiderate da vari menu che non sempre, però, sono un esempio di chiarezza. L'opzione "nucleare" può essere selezionata solo nella versione base del gioco ed ha effetti devastanti (nella finzione del gioco, beninteso!), ma può essere richiesta solo negli appropriati contesti storici: ad esempio, il programma non permetterà mai il lancio di un attacco nucleare preventivo contro una legione di antichi romani.

Lo svolgimento del gioco è lungo, richiede notevoli doti di memoria e molti giocatori affermano che la versione da tavolo non ha nulla da invidiare a quella del computer.

Apocalypse: versione Micro BBC (2-15 giocatori)
versione Spectrum (2-4 giocatori)
Editori: Red Shift (GB)
Autori: Helmut Watson (BBC); Bob Tyler (Spectrum)
Progetto originale: Mike Hayes
Joystick: non necessari
Formato: Cassetta

ZONE DI GUERRA

Apocalypse è un "war game" per tutta la famiglia: la versione BBC consente la partecipazione contemporanea di un massimo di 15 giocatori mentre nella versione per lo Spectrum, qui illustrata, i giocatori non possono essere più di 4. Questi ingaggiano battaglie che possono anche sfociare in un conflitto nucleare (simulato!)



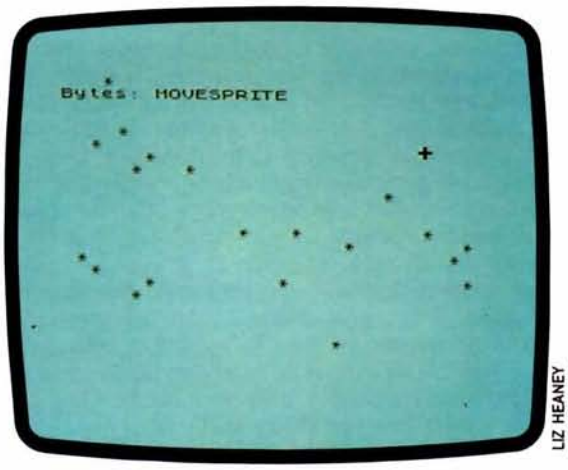


LIZ HEANEY

Gli sprite sullo Spectrum

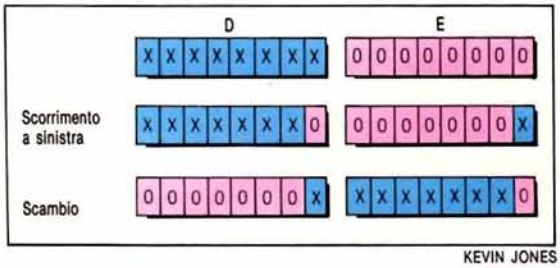
Ecco alcune utili routine per scrivere i propri giochi in codice macchina

Il BASIC dello Spectrum ha molti limiti, soprattutto per quanto riguarda la grafica a sprite, ma un buon gioco d'azione richiede l'uso di sprite che possano compiere movimenti uniformi in tutte le direzioni.
Non è facile scrivere routine grafiche in Assembly, ma il programma presentato può fornire alcuni chiarimenti visualizzando una serie di asterischi disposti a caso e permettendo di muovere sullo schermo una forma definita dall'utente (in questo caso una croce), usando i tasti per il cursore. La croce si muove di un pixel alla volta nelle quattro direzioni senza interferire con lo sfondo. Questa routine in codice macchina viene caricata e richiamata tramite un programma in BASIC.
La prima cosa da fare, quindi, è trascrivere il programma BASIC presentato a pag. 839. Il codice macchina verrà poi utilizzato usando l'apposito programma di caricamento in BASIC, che legge il codice dalle istruzioni delle frasi DATA e lo deposita in memoria con delle POKE, oppure adoperando un assembler (ad esempio il programma CHAMP). Per ciò viene riportato anche il listato "sorgente" in linguaggio Assembly. Qualunque sia la forma scelta, le linee 9000 e 9010 del programma BASIC possono essere impiegate per memorizzare su nastro il codice macchina risultante.
Per capire come funzioni il programma, esamineremo per prima la parte in BASIC. La subroutine alla linea 1000 legge la definizione dello sprite, estraendola dalle frasi DATA, depositando i valori nella RAM, dove possono essere usati dal programma in codice macchina. Le linee da 90 a 110 visualizzano lo sfondo, mentre la linea 120 definisce la posizione di partenza della croce. L'istruzione PRINT AT 10,16 fa sì che l'interprete BASIC calcoli l'indirizzo di schermo corrispondente alle coordinate specificate, immagazzinandolo poi nella variabile di sistema DFCC (indirizzi 23684 e 23685). Queste locazioni verranno poi ispezionate ed utilizzate dal programma in codice macchina. La linea 130 richiama il sottoprogramma di inizializzazione della routine in codice macchina. Le linee da 140 a 180 sono un ciclo che attende la battuta di un tasto. Il valore corrispondente viene depositato in una locazione di memoria, a disposizione della routine in codice macchina. Infine, viene richiamato il codice macchina, per spostare lo sprite di un pixel nella direzione indicata dal tasto battuto.
Il programma in Assembly comincia definendo i nomi delle locazioni di memoria usate. In TASTO viene depositato il codice del tasto premuto; in SPRPOS viene copiato l'indirizzo di memoria corrispondente alla posizione dello sprite sullo schermo; SPRTAB è una tabella nella quale il programma memorizza la definizione dello sprite ed il contenuto delle locazioni di schermo che lo sprite attraverserà. Lo sprite può essere mosso dovunque sullo schermo ed il suo spostamento non si limiterà a salti di un'intera cella di carattere. Ciò significa che gli 8 bit relativi che formano ciascuna riga dello sprite possono anche occupare due byte della memoria video. Di conseguenza, nella tabella servono due byte per memorizzare gli 8 bit, a seconda della combinazione da riportare sullo schermo. La locazione di memoria BITPOS viene usata per memorizzare la posizione degli 8 bit, all'interno dei due byte. La sezione di inizializzazione del programma legge da DFCC l'indirizzo iniziale di schermo dello sprite, quindi salta alla sezione chiamata SAVSCR, che deposita tale indirizzo in SPRPOS, carica il valore 1 nel registro D e chiama la subroutine SOTTO. Quando D è posto a 1, SOTTO copia il contenuto dell'area video sulla quale verrà visualizzato lo sprite per poter ricreare lo sfondo dopo che lo sprite si sarà spostato. Il programma quindi chiama la subroutine PRSPRT per visualizzare lo sprite sullo schermo.
La sezione del programma che controlla il movimento dello sprite inizia dall'etichetta MOVSPR. La procedura comincia caricando 0 nel registro D e chiamando la subroutine SOTTO. Con D posto a 0, SOTTO copia dalla tabella sul video i contenuti dello schermo salvati in precedenza, cancellando lo sprite dal video. Quindi il programma legge le coordinate dello



LIZ HEANEY

Sprite in movimento
Il programma dimostrativo in BASIC muove lo sprite (la cui forma, a crocetta, è definita nelle istruzioni DATA) su uno sfondo stellato, in risposta alla pressione dei tasti per il movimento del cursore



KEVIN JONES

Sposta e scambia
Per muovere lo sprite di un solo pixel alla volta, occorre eseguire uno scorrimento del valore contenuto nel doppio registro DE. Nella figura, i bit relativi allo sprite sono indicati con delle X. Qualora lo scorrimento provocasse la fuoriuscita di un bit a sinistra (facendolo automaticamente "entrare" a destra, nel registro E) i contenuti dei due registri vanno scambiati tra loro

Indirizzamento dello schermo

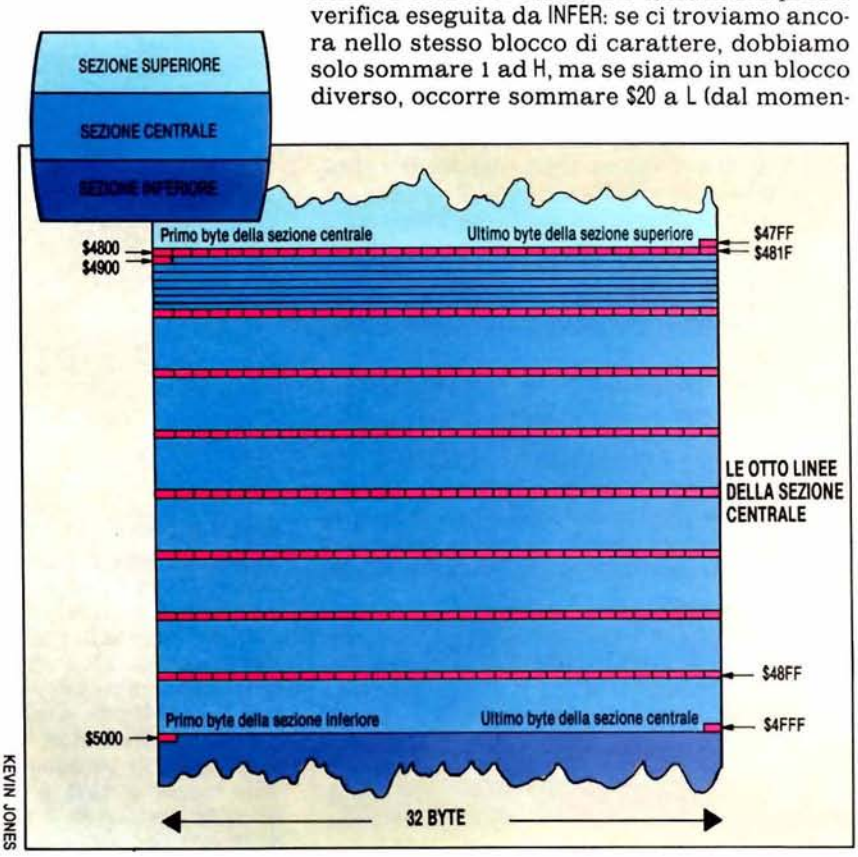
Per una migliore gestione della memoria, lo schermo a 24 linee dello Spectrum viene suddiviso in tre sezioni o "fasce" di 8 linee ciascuna: l'illustrazione in basso contiene informazioni sull'indirizzamento della fascia centrale. Ogni linea dello schermo viene suddivisa in 8 righe di alta risoluzione di 32 byte, dove ciascun bit dei byte corrisponde ad un pixel. I byte che compongono le righe sono numerati in modo sequenziale quindi, i byte posti nelle 8 righe delle prime 8 linee della sezione superiore, hanno indirizzi consecutivi. L'indirizzo successivo è quello del primo byte nella seconda riga della prima linea di schermo. Tutti i byte situati nelle 8 seconde righe di alta risoluzione della sezione di schermo vengono indirizzati in modo sequenziale a partire da questo indirizzo e così via. L'indirizzo del primo byte nella riga superiore della prima linea di schermo di una sezione è uno più l'indirizzo dell'ultimo byte nell'ottava riga dell'ottava linea della sezione precedente. Il seguente programma per lo Spectrum a 48K chiarisce questa complessa organizzazione inserendo a rotazione una linea in ciascun byte dello schermo di alta risoluzione:

```
 50 LET INIZSCRMO = 16384
 60 LET FINESCRMO = 22527
100 FOR B=INIZSCRMO TO FI-
    NESCRMO
200 POKE B,255
300 NEXT B
```

sprite ed il valore del tasto, lo verifica e chiama la routine per guidare il movimento nella giusta direzione. Infine, salta a SAVSCR per salvare i contenuti dello schermo e visualizza lo sprite.

Per capire le routine SUPER e INFER (usate nel controllo dei movimenti verticali dello sprite), occorre prima comprendere come lo Spectrum abbini gli indirizzi di memoria alle locazioni di schermo. La procedura è spiegata per esteso nel manuale dello Spectrum; si immagini lo schermo suddiviso in tre "fasce" o sezioni: osservando gli indirizzi in esadecimale, si noterà che, per i 256 caratteri in ciascuna sezione dello schermo, il byte "basso" dell'indirizzo di ciascuno degli 8 byte, relativi ad un singolo carattere, è uguale per tutti i caratteri del blocco, mentre il byte "alto" dell'indirizzo aumenta di uno, quando ci si sposta, sullo schermo, di un pixel verso il basso. Per questo motivo, le 8 righe di pixel che compongono ciascun carattere hanno indirizzi compresi tra $4000 e $47FF (per i caratteri della fascia superiore) tra $4800 e $4FFF (per quelli della fascia centrale) e tra $5000 e $57FF (per i caratteri della fascia inferiore). Il simbolo '$' significa che i numeri sono in notazione esadecimale: alcuni assembler, però, richiedono l'uso del simbolo #.

La subroutine INFER si aspetta di trovare un indirizzo di schermo nella coppia di registri HL, calcola l'indirizzo del byte che, sullo schermo, si trova immediatamente sotto questa posizione e carica il nuovo valore in HL. Se consideriamo gli indirizzi di schermo in binario, ci accorgiamo che, quando i tre bit bassi di H sono 111, la successiva riga di pixel fa parte di un altro blocco di carattere. Questa è la prima verifica eseguita da INFER: se ci troviamo ancora nello stesso blocco di carattere, dobbiamo solo sommare 1 ad H, ma se siamo in un blocco diverso, occorre sommare $20 a L (dal momento che una linea contiene $20=32 caratteri). Se il nuovo valore di L è compreso tra 0 e $1F (i tre bit alti sono 000), siamo in una diversa sezione dello schermo.



KEVIN JONES

Se siamo ancora nello stesso blocco di schermo, dobbiamo sottrarre 7 da H. Il procedimento appare più chiaro quando si lavora sul codice poiché si può osservare direttamente come vengono manipolati gli indirizzi contenuti nella tabella. La routine SUPER è simile a INFER, ma calcola l'indirizzo del pixel situato sopra la posizione di schermo.

Le subroutine SINIS e DESTR funzionano in modo analogo, ma spostano la configurazione dei bit del pixel a sinistra o a destra sulla tabella. In SINIS, ad esempio, l'accumulatore viene caricato con il puntatore alla locazione del bit, rappresentato da un numero a singolo byte compreso tra i valori 0 e 7 (che corrispondono alla numerazione dei bit in un byte). A tale valore viene quindi sottratto 1, per realizzare il movimento; ciò vale anche per gli altri numeri da 0 a 7, a meno che il valore iniziale del contatore di bit non sia 0: in questo caso l'accumulatore conterrà 255. L'istruzione AND 7 garantirà che il valore nell'accumulatore sia sempre compreso tra 0 e 7. C'è un ciclo per ciascuna delle 8 righe dello sprite. Per ogni riga, caricheremo nella coppia di registri DE i due byte della tabella contenenti la riga di pixel ed eseguiremo su DE una rotazione verso sinistra dei bit. Se quest'operazione non comporta l'uscita di un bit (di valore 1) dall'estremità sinistra di D (ed un suo rientro dalla estremità destra di E), allora possiamo tranquillamente depositare la nuova configurazione dello sprite nella tabella e procedere con la successiva riga di pixel. Al contrario, se un bit dello sprite esce da D e rientra in E, allora dobbiamo scambiare i valori di D ed E, prima di depositarli nella tabella. Infine, al termine della routine, dovremo sottrarre 1 da HL per visualizzare lo sprite una posizione più a sinistra.

La subroutine finale PRSPRT provvede a visualizzare lo sprite sullo schermo. Si compone di due cicli nidificati, uno per le 8 righe di pixel dello sprite controllate dal registro D, l'altro per i due byte sui quali si trova spartita la riga di pixel, controllata dal registro B. La sezione centrale della routine deposita sullo schermo i bit che formano lo sprite, senza disturbare quanto sia eventualmente già presente sullo schermo. L'indirizzo della locazione di schermo si trova nella coppia di registri HL e quello della tabella per lo sprite nel registro IX. La subroutine PRSPRT prende un byte dalla configurazione pixel dello sprite e lo pone in OR con quello presente sullo schermo: quest'ultima operazione serve a sovrapporre lo sprite ad un'eventuale immagine già raffigurata sullo schermo.

Il programma ha alcuni limiti: la dimensione massima dello sprite è di 8 × 8 pixel, c'è un unico sprite ad un solo colore. Nulla vieta di migliorarlo; tuttavia, anche senza modifiche, questo programma si dimostrerà un utile strumento in molti programmi BASIC o in codice macchina.

MOVIMENTI DELLO SPRITE

```
  5 REM     *Programma 1*
 10 CLEAR 45055:        REM AFFF HEX
 20 LOAD "MOVSPRITE"CODE
 30 LET TASTO=45056
 40 LET BITPOS=45059
 50 LET SPRTAB=45060
 60 LET INIT=45312:     REM B100 HEX
 70 LET MOVSPR=45317:   REM B105 HEX
 80 GO SUB 1000:        REM DEFINIZIONE SPRITE
 90 FOR I=1 TO 20
100 PRINT AT 21*RND,31*RND;"*";
110 NEXT I
120 PRINT AT 10,16;
130 RANDOMIZE USR INIT
140 LET X$=INKEY$: IF X$=" " THEN GO TO 140
150 LET X=VAL X$
160 POKE TASTO,X
170 RANDOMIZE USR MOVSPR
180 GO TO 140
1000 POKE BITPOS,0
1010 FOR I=0 TO 7
1020 READ X
1030 POKE SPRTAB+2*I,X
1040 POKE SPRTAB+1+2*I,0
1050 NEXT I
1060 RETURN
2000 DATA BIN 00011000
2010 DATA BIN 00011000
2020 DATA BIN 00011000
2030 DATA BIN 11111111
2040 DATA BIN 11111111
2050 DATA BIN 00011000
2060 DATA BIN 00011000
2070 DATA BIN 00011000
```

```
  5 REM     *Programma 2*
 10 LET A=45312
 20 FOR L=1000 TO 1420 STEP 10
 30 LET S=0
 40 FOR A=A TO A+7
 50 READ B
 60 POKE A,B
 70 LET S=S+B
 80 NEXT A
 90 READ C
100 IF C<>S THEN PRINT "ERRORE ALLA LINEA ";L: STOP
110 NEXT L
120 READ B
130 POKE A,B
140 READ B
150 POKE (A+1),B
200 PRINT "INSERIRE LA CASSETTA DEL PROGRAMMA"
250 SAVE "MOVESPRITE"CODE 45312,400
1000 DATA 42,132,92,24,44,22,0,205,561
1010 DATA 61,177,42,1,176,58,0,176,691
1020 DATA 254,5,32,5,205,165,177,24,867
1030 DATA 24,254,6,32,5,205,109,177,812
1040 DATA 25,15,254,7,32,5,205,136,678
1050 DATA 177,24,6,254,8,192,205,228,1094
1060 DATA 177,34,1,176,22,1,205,61,677
1070 DATA 177,205,35,178,201,42,1,176,1015
1080 DATA 221,33,4,176,14,8,229,61,691
1090 DATA 2,203,66,32,6,221,126,16,672
1100 DATA 119,24,4,126,221,119,16,35,644
1110 DATA 205,83,178,221,35,16,234,225,1197
1120 DATA 13,200,221,35,197,213,205,109,1193
1130 DATA 177,209,193,24,217,62,7,164,1053
1140 DATA 254,7,40,2,36,201,17,32,589
1150 DATA 0,25,62,244,165,32,4,205,717
1160 DATA 83,178,201,124,214,7,103,201,1111
1170 DATA 62,7,164,40,2,37,201,17,530
1180 DATA 32,0,167,237,82,62,224,165,969
1190 DATA 254,224,32,4,205,76,178,201,1147
1200 DATA 124,198,7,103,201,221,33,3,890
1210 DATA 176,121,126,0,61,230,7,221,1042
1220 DATA 119,0,79,221,35,6,8,221,689
1230 DATA 94,0,221,86,1,203,3,245,853
1240 DATA 203,11,241,203,18,203,19,62,960
1250 DATA 7,185,32,3,122,83,95,221,748
1260 DATA 115,0,221,114,1,221,35,221,928
1270 DATA 35,16,220,62,7,185,192,43,760
1280 DATA 205,76,178,201,221,33,3,176,1093
1290 DATA 221,126,0,60,230,7,221,119,984
1300 DATA 0,79,221,35,6,8,221,94,664
1310 DATA 0,221,86,1,203,11,245,203,970
1320 DATA 3,241,203,26,203,27,62,0,765
1330 DATA 185,32,3,122,83,95,221,115,856
1340 DATA 0,221,114,1,221,35,221,35,848
1350 DATA 16,220,62,0,185,192,35,205,915
1360 DATA 83,178,201,42,1,176,221,33,935
1370 DATA 4,176,14,8,229,6,2,221,660
1380 DATA 126,0,47,87,126,162,221,182,951
1390 DATA 0,119,35,205,76,178,221,35,869
1400 DATA 16,237,225,13,200,197,205,109,1202
1410 DATA 177,193,24,224,62,63,188,192,1123
1420 DATA 38,87,201,62,88,188,192,38,894
1430 DATA 64,201,256
```

```
TASTO  EQU  £B000
SPRPOS EQU  £B001
BITPOS EQU  £B003
SPRTAB EQU  £B004
DFCC   EQU  £5C84
       ORG  £B100
INT    LD   HL,(DFCC)
       JR   SAVSCR
MOVSPR LD   D,0
       CALL SOTTO
       LD   HL,(SPRPOS)
       LD   A,(TASTO)
       CP   5
       JR   NZ,LO
       CALL SINIS
       JR   SAVSCR
LO     CP   6
       JR   NZ,L1
       CALL INFER
       JR   SAVSCR
L1     CP   7
       JR   NZ,L2
       CALL SUPER
       JR   SAVSCR
L2     CP   8
       RET  NZ
       CALL DESTR
SAVSCR LD   (SPRPOS),HL
       LD   D,1
       CALL SOTTO
       CALL PRSPRT
       RET
SOTTO  LD   HL,(SPRPOS)
       LD   IX,SPRTAB
       LD   C,8
LINEA  LD   B,2
BYTE   BIT  0,D
       JR   NZ,SVESCR
CANCEL LD   A,(IX+£10)
       LD   (HL),A
       JR   CONT
SVESCR LD   A,(HL)
       LD   (IX+£10),A
CONT   INC  HL
       INC  IX
       DJNZ BYTE
       DEC  C
       RET  Z
       INC  IX
       DEC  HL
       DEC  HL
       PUSH BC
       PUSH DE
       CALL INFER
       POP  DE
       POP  BC
       JR   LINEA
INFER  LD   A,7
       AND  H
       CP   7
       JR   Z,BDIFCB
BSAMCB INC  H
       RET
BDIFCB LD   DE,£20
       ADD  HL,DE
       LD   A,£E0
       AND  L
       RET  Z
BSAMSB LD   A,H
       SUB  7
       LD   H,A
       RET
SUPER  LD   A,7
       AND  H
       JR   Z,ADIFCB
ASAMCB DEC  H
       RET
ADIFCB LD   DE,£20
       AND  A
       SBC  HL,DE
       LD   A,£E0
       AND  L
       CP   £E0
       RET  Z
ASAMSB LD   A,H
       ADD  A,7
       LD   H,A
       RET
SINIS  LD   IX,BITPOS
       LD   A,(IX+0)
       DEC  A
       AND  7
       LD   (IX+0),A
       LD   C,A
       INC  IX
```

```
DESTR  LD   IX,BITPOS
       LD   A,(IX+0)
       INC  A
       AND  7
       LD   (IX+0),A
       LD   C,A
       INC  IX
       LD   B,8
PAIOD  LD   E,(IX+0)
       LD   D,(IX+1)
       RRC  E
       PUSH AF
       RLC  E
       POP  AF
       RR   D
       RR   E
       LD   A,0
       CP   C
       JR   NZ,DEPODS
       LD   A,D
       LD   D,E
       LD   E,A
DEPODS LD   (IX+0),E
       LD   (IX+1),D
       INC  IX
       INC  IX
       DJNZ PAIOD
       LD   A,0
       CP   C
       RET  NZ
       INC  HL
       RET
PRSPRT LD   HL,(SPRPOS)
       LD   IX,SPRTAB
       LD   C,8
PRLINE LD   B,2
PRBYTE LD   A,(IX+0)
       CPL
       LD   D,A
       LD   A,(HL)
       AND  D
       OR   (IX+0)
       LD   (HL),A
       INC  HL
       INC  IX
       DJNZ PRBYTE
       DEC  C
       RET  Z
       DEC  HL
       DEC  HL
       PUSH BC
       CALL INFER
       POP  BC
       JR   PRLINE
       LD   B,8
PAIOS  LD   E,(IX+0)
       LD   D,(IX+1)
       RLC  E
       PUSH AF
       RRC  E
       POP  AF
       RL   D
       RL   E
       LD   A,7
       CP   C
       JR   NZ,DEPOSN
       LD   A,D
       LD   D,E
       LD   E,A
DEPOSN LD   (IX+0),E
       LD   (IX+1),D
       INC  IX
       INC  IX
       DJNZ PAIOS
       LD   A,7
       CP   C
       RET  NZ
       DEC  HL
       RET
```

Come usare questi programmi

1) Scrivere e salvare (SAVE) il "Programma 1"
2) Battere e eseguire (RUN) "Programma 2": questa operazione salva il codice macchina su nastro, meglio se subito dopo il "Programma 1"
3) Caricare (LOAD) "Programma 1" che andrà automaticamente in esecuzione

La Motorola

Notissimo sul mercato della componentistica, questo colosso americano produce i più recenti chip a 16 bit

La Motorola, fondata da Paul Galvin nel 1928 sotto il nome di Galvin Manufacturing Corporation, era specializzata nella fabbricazione di apparecchi radiofonici per uso domestico. Già nel 1930 la compagnia, diversificando la propria produzione, iniziò a fabbricare autoradio e rice-trasmittenti per la polizia sotto il marchio "Motorola". Nel 1940 la società, ora chiamata Motorola Incorporated, fu una delle prime industrie elettroniche a produrre semiconduttori.

Paul Galvin morì nel 1959 e gli successe come presidente il figlio Robert. Nella decade seguente, la Motorola subì una forte concorrenza da parte di altre industrie, soprattutto giapponesi. La recessione economica dei primi anni '70 causò enormi perdite alla compagnia che fu costretta a rivedere la propria strategia commerciale. La Motorola assunse nuovo personale, proveniente in gran parte dalla rivale Texas Instruments, e decise di abbandonare il mercato elettronico tradizionale per concentrare la produzione sull'elettronica ad alta tecnologia. Questa scelta comportò la vendita di alcune consociate (in particolare quelle che producevano televisori a colori), l'investimento di grossi capitali nella ricerca e l'acquisto di altre società già operanti in questo settore di mercato.

Questa rischiosa riconversione dette però i suoi frutti: alla fine degli anni '70, la Motorola non aveva ancora colmato il distacco rispetto alle industrie leader nel campo dei semiconduttori ma, dopo nuovi forti investimenti in quest'area di mercato, la compagnia poté dichiarare di aver superato, come vendite, la Texas Instruments, sua principale concorrente. In quell'occasione Robert Galvin dichiarò: "Le industrie che fanno concorrenza alla Motorola sono destinate alla sconfitta, perché non sono in grado di adattarsi ai nostri ritmi di mercato".

Tuttavia, anche la Motorola aveva il problema di rendere i propri prodotti disponibili al momento giusto, tant'è vero che, nella seconda metà degli anni '70, quando ancora l'industria dei microcomputer era ai suoi primi passi, il processore Motorola 6800 a 8 bit perse terreno, da una parte a causa del chip Mostek 6502, che fu scelto dalla Apple per i propri personal computer di grande successo, e dall'altra a causa dell'Intel 8085 e lo Zilog Z80 per i computer CP/M. Nel 1976, la compagnia lanciò il 6809, oggi considerato il miglior microprocessore ad 8 bit mai prodotto, ma la partita per la conquista del mercato era stata ormai giocata e questo chip fu adottato soltanto su pochi micro, quali il Dragon ed il Tandy Color.

Nonostante ciò, la Motorola ha continuato ad investire molto denaro nella ricerca ed ha ottenuto notevoli risultati nel mercato dei processori a 16 bit. Nel '79 ha progettato il microprocessore 68000, utilizzato poi dalla Apple per il Lisa ed il Macintosh e dalla Sinclair Research per il QL. Questo potente processore possiede 17 registri a 32 bit, un bus dati a 16 bit ed un bus indirizzi a 24 bit.

Oggi la Motorola possiede tre centri di ricerca: a Phoenix in Arizona, a Ginevra in Svizzera ed a East Kilbride nel Galles (GB). Quest'ultimo centro produce chip secondo la tecnica CMOS e MOS per una vasta gamma di applicazioni. La compagnia è stata suddivisa in 5 sezioni produttive: comunicazioni, semiconduttori, sistemi informativi, elettronica industriale e di controllo e ricerca teorica. Nonostante alcune perdite in qualche settore, il fatturato della Motorola è nell'ordine di diverse migliaia di miliardi e questa solidissima ditta sembra decisa a rafforzare ancor più la propria posizione di mercato.



La Sede Centrale della Motorola nell'Illinois, USA



Robert Galvin, presidente della Motorola

Tecniche di marketing

Per vendere software o hardware occorre creare un'"immagine"



SOFTWARE per concessione della PILOT SOFTWARE

LIZ HEANEY

L'apparenza può ingannare!
Il costo dei due giochi qui raffigurati è pressoché identico, ma uno è stato confezionato in modo molto fantasioso, che attira immediatamente l'occhio, suggerendo implicitamente che si tratta di un ottimo "affare"; l'altro, ad un superficiale confronto, può esser giudicato semplicemente come un gioco poco attraente, dal prezzo molto elevato

Il "marketing" è quella particolare scienza che fa da ponte tra il produttore di un certo articolo e le "tasche" del potenziale acquirente. Affinché questo immaginario ponte possa reggere ad un forte traffico, gli esperti di marketing devono riuscire ad indovinare (basandosi tuttavia su approfondite indagini di mercato) quali siano i gusti, le preferenze e le tendenze del pubblico, per poi dirigere i propri sforzi e soprattutto fortissimi investimenti, in tale direzione. Ad esempio, nel momento culminante, il lancio commerciale del Macintosh in Gran Bretagna prevedeva una spesa di oltre 2 miliardi al mese!

Le prime decisioni di marketing vengono prese addirittura quando il prodotto da vendere è ancora in fase progettuale, poiché elementi quali la funzione dell'apparecchio, a che livello impostare la produzione e quanto viene a costare ogni singolo pezzo influenzano le scelte di marketing.

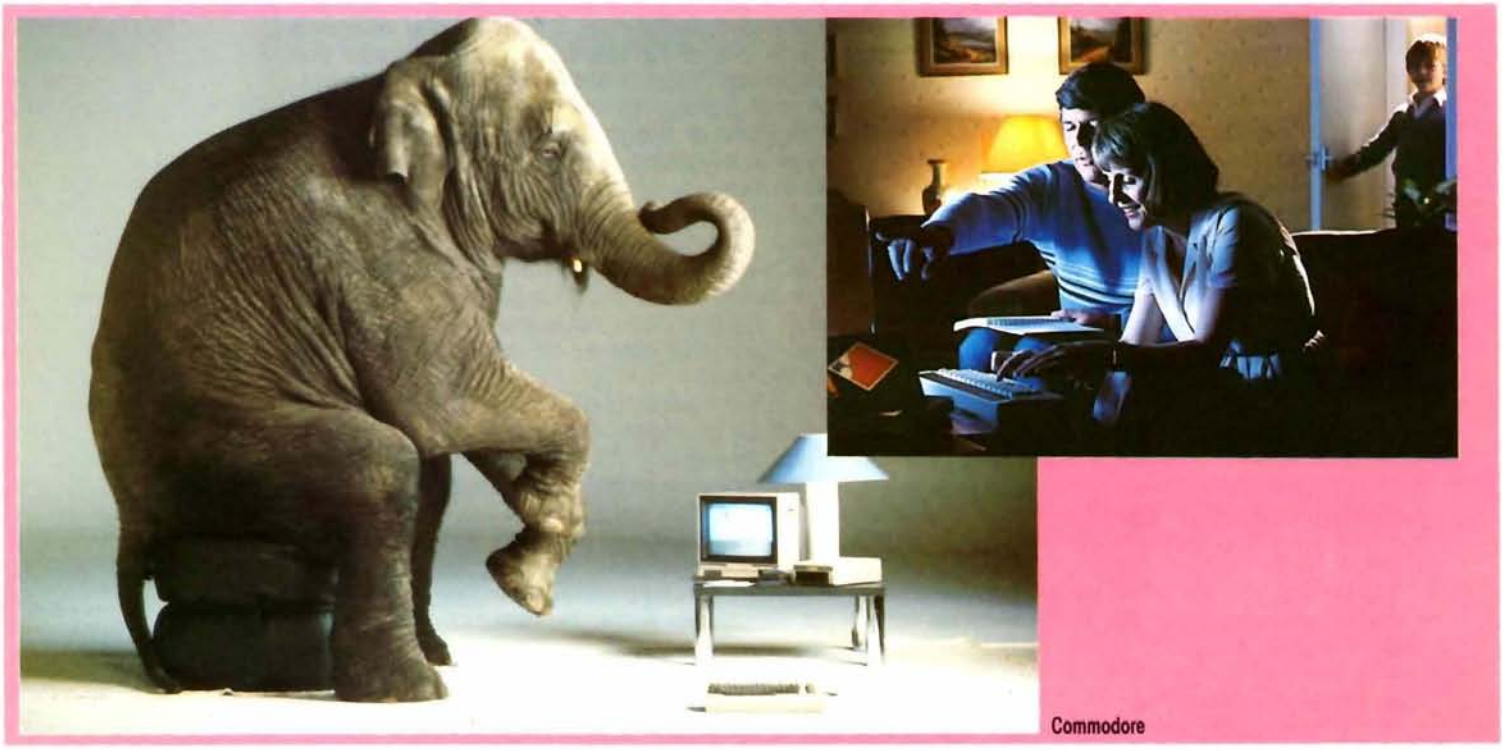
Anche nel settore software vigono le stesse considerazioni: non vale certo la pena investire ingenti somme di denaro per lo sviluppo e la commercializzazione di un programma, se poi il suo prezzo di vendita è tale da non invogliare potenziali acquirenti, causando così enormi perdite. D'altro canto, una software house tirchia in fase di sviluppo può ritrovarsi con un prodotto finito commercialmente inutile, in quanto non risponde alle esigenze di mercato, oppure mal si adatta alle aspettative dei potenziali compratori. Così facendo si rischia di compromettere la propria reputazione e la futura vendita di altri programmi. Tuttavia, esiste anche il rischio di spendere molto per lo sviluppo di un programma, ma troppo poco nel marketing. L'immediata conseguenza sarà un ottimo prodotto finale, di cui tutti ignorano l'esistenza!

Un altro elemento da tenere in considerazione è la collocazione di un nuovo prodotto rispetto a quelli già presenti sul mercato ed è qui che entra in ballo l'"immagine", sia della ditta, sia del prodotto stesso. Si pensi, ad esempio, a quanto ciò conti nella vendita di saponette o di detersivi, che, tutto sommato, si assomigliano tutti fra loro. Per i fabbricanti di computer, forse, la situazione è diversa, ma non è sempre facile descrivere in due parole i pregi di un nuovo apparecchio (in altre parole "far presa"), senza rischiare che termini troppo tecnici risultino incomprensi-

bili ai più. Anche molti prodotti software non rivelano le proprie doti, se non dopo essere stati acquistati ed usati.
È per questo motivo che i produttori di hardware e di software, come quelli dei detersivi, ricorrono alla creazione di un'"immagine del prodotto", una specie di telegrafica proiezione psicologica dell'oggetto in vendita. È molto più semplice vendere un'"idea" che non un mero oggetto e ciò può essere esemplificato da uno slogan del tipo "vendi la sete, non la birra"!
Nello sviluppo di un'immagine per un nuovo apparecchio, l'apparenza esteriore può costituire un buon punto di partenza. Ad esempio, il Micro BBC ha un aspetto sobrio e pratico, che ben si adatta all'immagine di un computer destinato alla didattica. Il design di altri home computer, al contrario, è incentrato sull'impiego di forti tinte o forme stilizzate, per sottolineare il loro principale uso: quello ricreativo. La gamma XL della Atari costituisce un altro buon esempio di design concepito in funzione del marketing: l'aspetto di questi computer è piuttosto "militaresco" (adattissimo per giocare coi "war game"), nel tentativo di recuperare la fetta di mercato perduta verso la fine del 1983. La Commodore, addirittura, ha commissionato il design da applicare alla propria produzione niente meno che a Ferdinand Porsche, cercando di dare agli apparecchi un aspetto futuristico, ma al tempo stesso accattivante.
Un ultimo esempio è lo Spectrum della Sinclair: il suo involucro, minuto e nero, sul quale compaiono numerose ed elaborate iscrizioni, offre l'immagine di una "piccola botte piena di vino buono", ossia di un prodotto economico ma validissimo.
L'immagine del prodotto, comunque, va ben oltre il suo aspetto esteriore: la pubblicità deve adeguatamente rinforzare i concetti di base ad esso legati. Ad esempio, i 64 Kbyte di memoria del Commodore, in alcuni "short" pubblicitari, sono sottolineati mostrando un elefante, intento a far uso di un Commodore. Nel Micro BBC, il marchio usato (una civetta), fa leva sulla saggezza comunemente associata a questo animale.
Ovviamente, anche lo stesso nome del prodotto ha estrema importanza: i primi home computer hanno dovuto abbattere l'immagine, particolarmente diffusa dai film degli anni '60, in cui il computer è l'orwelliano "Grande Fratello", disumano e spesso sfuggente al controllo dell'uomo. Ecco dunque un fiorire di nomi di comune uso "domestico", per mascherare l'alta tecnologia dei nuovi apparecchi: PET (cucciolo), Apple (mela) e via dicendo.
La campagna pubblicitaria per il Macintosh, negli USA ed in Gran Bretagna, insiste proprio nell'abbattimento di questa erronea immagine: nel filmato si vede una donna che fracassa l'enorme schermo di un "Grande Fratello", a sottolineare la libertà offerta dal nuovo prodotto della Apple. Lo slogan di accompagnamento suonava all'incirca così: «Grazie al Macintosh, il 1984 non sarà come il 1984!». Detto per inciso, la Mackintosh è una nota varietà di mele, pertanto il nome è molto appropriato, ma chi lo ha proposto inizialmente, si è dimenticato una "k" e l'errore, almeno apparentemente involontario, è poi rimasto.
L'uso di nomi familiari, però, non deve al tempo stesso far credere al pubblico che l'apparecchio sia un semplice giocattolo. Questo genere di ostacolo lo ha dovuto affrontare il PET: inizialmente, ma con scarso successo, la sigla venne fatta passare come acronimo di Personal Electronic Transactor,

L'importanza di un'immagine

Azzeccare l'"immagine" commerciale per un nuovo prodotto è tutt'altro che facile, per i risvolti "a doppio taglio" che spesso possono sfuggire ad un primo esame.
Ecco tre esempi di immagini pubblicitarie. La prima è della Commodore: all'immagine del computer è stata abbinata quella di un elefante: ciò suggerisce la potenza della sua memoria, ma può anche insinuare il dubbio che l'apparecchio sia lento.
La seconda riguarda il Micro BBC, del quale viene mostrato l'impiego didattico o "in famiglia": se da un lato ciò attribuisce al computer un alone di semplicità e di praticità, dall'altro l'impostazione può sembrare scontata e stereotipata.
La terza concerne l'Apple, ed è quella con maggiori sottintesi di alta tecnologia: il cupo scenario "orwelliano" e la conseguente liberazione da esso, grazie al facile impiego del Macintosh, è di grande effetto, ma lo spettatore potrebbe risentirsi, per essere indirettamente paragonato ai personaggi privi di volontà descritti da Orwell



Commodore

dal significato molto più scientifico, poi il nome dell'apparecchio venne addirittura cambiato in Commodore Business Machine, da cui CBM.
In un secondo momento, essendosi il pubblico abituato alla presenza di questi piccoli computer, la tendenza al "tecnologico" ed al "futuribile" è tornata nuovamente in auge. Di conseguenza sono apparsi computer i cui nomi non sono che sigle più o meno prive di qualsiasi significato concreto: ZX81, Z380, MTX500 e così via.
Nell'immagine di un prodotto software conta molto anche la confezione o "packaging", mentre nel caso dell'hardware essa ha uno scopo puramente protettivo. Fondamentalmente, esistono due strategie nella vendita di software: ridurre al minimo le spese per la confezione (la semplice cassetta, magari con una fascetta riccamente colorata) puntando sul concetto di "acquisto a buon mercato", oppure dedicarvi molta cura, inscatolando il tutto in un involucro di grande effetto, nel quale inserire anche altri piccoli accessori più o meno utili al programma. Ad esempio, assieme al gioco *Hobbit* viene dato in omaggio l'omonimo libro di Tolkien. Questa seconda strategia punta a valorizzare al massimo il prodotto e, di conseguenza, consente anche di tenerne alto il prezzo.
L'immagine, poi, viene proiettata e diffusa dalla pubblicità. In senso lato, il rapporto esistente tra marketing e pubblicità è identico a quello intercorrente tra strategia e tattica. Le decisioni relative a "quando" reclamizzare un certo prodotto e su "quanto" investire nel budget pubblicitario concernono il marketing. Nel tipico caso di un home computer, il produttore riterrà opportuno lanciarlo sul mercato poco prima di Natale e riversare gran parte del budget pubblicitario in tale periodo dell'anno, per appoggiare il lancio. Naturalmente, per quanto buona possa essere la campagna promozionale di un prodotto, se l'immagine è tutt'altro che efficace, esso è destinato a fallire e ci sono numerosi casi di catastrofiche conseguenze, anche nel settore dei computer.
Anche quella che viene definita la parte "logistica" del marketing è altrettanto importante: spesso, anzi, il punto debole di una campagna promozionale è proprio una pessima scelta del canale di vendita.
Ad esempio, vendere un nuovo prodotto per posta, tramite ordine diretto (come sperimentato dalla Sinclair in Gran Bretagna), può essere un metodo economico, ma sorge poi il problema di offrire un'adeguata assistenza e viene a mancare un aspetto fondamentale della vendita: il contatto personale venditore/acquirente, per non parlare poi degli inevitabili ritardi postali, che certo non giovano.
La vendita tramite accordi con catene di negozi specializzati può rassicurare i probabili acquirenti sulla buona qualità del prodotto, ma in tal caso gran parte del margine di profitto verrà fagocitata dalle spese per garantire un volume di produzione sufficientemente alto e dalla percentuale giustamente pretesa dall'organizzazione commerciale cui ci si rivolge. In alternativa, ci si può affidare ad un ristretto numero di rivenditori ben selezionati, in grado di illustrare singolarmente i pregi del proprio software.
L'accanita lotta per la sopravvivenza, nel commercio, se da un lato porta ad un migliore standard produttivo, dall'altro rende la realizzazione di un nuovo prodotto (per quanto valido possa essere) soltanto il primo passo di un lungo cammino: conquistare e, soprattutto, mantenere la propria fetta di mercato è la parte più difficile!



Margherite per stampare

Conviene acquistare una stampante a margherita: ecco le sue principali doti

A prima vista, l'acquisto di una stampante "a margherita", da parte di un possessore di home computer può sembrare un fatto insolito: questo tipo di stampante non è molto utile per la stampa di lunghi listati, è lento, ed ha un prezzo superiore a quello di una normale stampante ad aghi. Nonostante ciò, per certe applicazioni rappresenta una scelta validissima, considerata l'ottima qualità di stampa ottenibile, di gran lunga superiore a quella delle stampanti ad aghi.
In sostanza, il principio di funzionamento delle stampanti a margherita è pressoché identico a quello delle semplici macchine da scrivere: su di un supporto a forma di fiore sono disposti i vari martelletti, uno per ogni carattere. La stampa avviene provocando la percussione di uno dei martelletti, selezionato facendo ruotare la margherita, sul nastro inchiostrato. Poi, il carrello di supporto viene spostato nella posizione di stampa successiva.
Confrontando questo sistema con quello adottato nelle stampanti ad aghi, nelle quali i caratteri vengono formati e stampati durante lo stesso movimento della testina di stampa, si comprende facilmente perché vi sia una così notevole differenza di velocità. A parità di costo, una stampante a margherita ha una velocità di stampa di circa 20 caratteri per secondo (cps), mentre una a matrice d'aghi raggiunge comodamente i 160 caratteri per secondo. Esistono stampanti a margherita con velocità maggiori (fino ad 80 cps), ma il loro costo è molto elevato.
In alcune stampanti, dell'uno e dell'altro tipo, si ottiene un'alta velocità di riproduzione ricorrendo alla stampa bidirezionale, alla ricerca logica o ad entrambe queste possibilità.
Per "stampa bidirezionale" si intende la capacità di stampare una linea di testo sia durante lo scorrimento della testina da sinistra a destra, sia durante il percorso inverso. La "ricerca logica", invece, consiste nel posizionare immediatamente la testina di stampa sul primo carattere da stampare in una riga, senza perder tempo a "stampare" (si fa per dire) gli spazi all'inizio o alla fine di una riga di testo.
Nelle stampanti a matrice d'aghi la forma di ciascun carattere è definita in alcuni chip di ROM, mentre in quelle a margherita la forma è fisicamente riprodotta sui martelletti. I due metodi hanno i rispettivi pro e contro: per cambiare il "set" di caratteri nelle prime, occorre inviare opportune sequenze di codici alla stampante (vedere a pag. 804), mentre nelle seconde basta sostituire la margherita. Le margherite vengono vendute in numerosissime versioni, a seconda del "corpo" di stampa e della "densità" dei caratteri. I corpi più comuni sono: il Pica, l'Elite, il Gotico e il Corsivo. La densità è generalmente espressa in pollici e quelle più comuni sono da 10 e da 12 caratteri per pollice (characters per inch, o più brevemente cpi). Inoltre, esistono margherite in plastica o in metallo: le prime sono più economiche, ma durano meno.
Un problema comune con le stampanti a margherita è che talvolta alcuni simboli riportati sulla tastiera del computer non corrispondono a quelli effettivamente stampati. Può così accadere che, anziché il simbolo "#", si ottenga un "£" e un "1/2" al posto di "[", e via dicendo. A seconda del corpo usato, poi, può accadere che lo zero "0" si confonda con la "O" maiuscola, mentre il numero "1" si confonda con la "l" minuscola.
Le stampanti a margherita più costose possiedono 127 caratteri per margherita, ma quelle più comuni ne hanno 96 o 92.
Le peggiori conseguenze di tali inconvenienti si riscontrano nel tentativo di individuare gli errori in un listato di programma, dove la confusione tra gli "1" e le "l" certamente non aiuta! Se l'utilizzazione principale è la stampa di listati, allora le stampanti a margherita non sono le più adatte.

Effetti speciali

Anche le stampanti a margherita, opportunamente programmate, possono offrire una serie di prestazioni "extra", come abbiamo già visto per quelle a matrice d'aghi, sebbene in misura ridotta. Anche in questo caso si adoperano delle particolari "sequenze di Escape" (vedere a pag. 804). Sulle stampanti Diablo, ad esempio, il codice ESC-E abilita la sottolineatura automatica del testo, mentre ESC-R la disabilita. Usando un BASIC versione Microsoft, gli equivalenti comandi sarebbero: LPRINT CHR$(27);"E" e LPRINT CHR$(27);"R". Altri codici sono: ESC—1 per fissare un fermo di tabulazione; ESC—9 per fissare il margine sinistro ed ESC-U per far avanzare la carta di una mezza interlinea (utile per la stampa di formule matematiche). Per far risaltare una sequenza di lettere o di parole, i caratteri vengono ripassati per quattro volte, ed il codice necessario per questa funzione è ESC—0. Ovviamente, ciò rallenta enormemente il tempo necessario alla stampa.
In alcune stampanti, più sofisticate, è consen-

tito variare sia la distanza fra i vari caratteri, che quella tra una riga di scritto e l'altra, e ciò permette di usare la stampante per riprodurre immagini grafiche estratte, magari, direttamente dallo schermo: per ogni pixel "acceso" viene stampato un punto sulla carta. Dimensionando adeguatamente la distanza tra i caratteri, si riesce a riprodurre un'intera riga dello schermo su di una sola linea di stampa. Analogamente, variando la misura delle interlinee, si ottiene un corretto avvicinamento dei punti fra riga e riga. Tuttavia, questo procedimento richiede tempi molto lunghi.

Una caratteristica interessante, assente sulle stampanti ad aghi, è la centratura automatica dei titoli: l'invio del codice ESC=, sulla Diablo, provoca la stampa di quanto segue in una posizione ben centrata sul foglio. Un'altra dote insolita è la possibilità di introdurre dei fermi di tabulazione "decimali", che provoca l'incolonnamento automatico di valori numerici, in base alla posizione della virgola decimale.

Tra le altre varie opzioni di stampa, è giusto citare la "spaziatura proporzionale", grazie alla quale i caratteri non vengono stampati ad una distanza costante l'uno dall'altro. Al contrario, lo spazio occupato da alcuni caratteri, quali le "i", è inferiore a quello occupato dalle "w" o dalle "m". Di conseguenza, in un brano di testo, i caratteri non risultano incolonnati esattamente l'uno sotto l'altro ed il risultato estetico è ancor più gradevole e professionale. Per la corretta esecuzione di questo effetto è spesso necessaria una margherita speciale, ma l'effetto stesso è abilitato o disabilitato mediante i soliti codici Escape. Sulla Silver-Reed EXP 770, ad esempio, vengono usati i codici ESC-P e ESC-Q.

In un'attività commerciale, la presenza di una stampante a margherita è più giustificabile, sebbene esista un'alternativa più economica: adattare una macchina da scrivere elettronica. Spesso, infatti, quest'ultima fa già parte del macchinario d'ufficio e possiede un connettore seriale del tipo RS232. Se non fosse previsto questo genere di collegamento, esistono in commercio delle interfacce dal prezzo sempre più ragionevole che non quello per acquistare una stampante a margherita. Un vantaggio secondario, nell'adattare una macchina da scrivere elettronica, è la possibilità di usarla per il suo fine originario, quando non è adoperata dal computer.

Print sample with 10 characters per inch
Print sample with 12 characters per inch
Print sample with 15 characters per inch

A half-line feed is used for subscripts:-
H_2O

ESC-E turns on the automatic underlining and ESC-R turns it off.

Bold print makes text **stand out** from the rest of the text.

The ESC= code is used to centre text:-

A Centred Heading

These lines of text were printed without the proportional spacing feature. Note especially how the numbers 0123456789 are spaced out. Without proportional spacing, the width of each character is the same. For example, a 'W' is the same width as an 'i':-
WWWWWWWWWWW
iiiiiiiiiii

These lines of text were printed using the proportional spacing feature. Note especially how the numbers 0123456789 are spaced out. Using proportional spacing, the width of each character varies. For example, a 'W' is much wider than an 'i':-
WWWWWWWWWWW
iiiiiiiiiii
The overall effect is to make the text more attractive.



LA QUALITÀ HA UN PREZZO

In questa illustrazione sono riportati numerosi esempi delle prestazioni offerte da una stampante a margherita, oltre all'ottima qualità di stampa dei caratteri.

Nelle prime tre righe si può apprezzare la variazione di densità tra 10, 12 e 15 caratteri per pollice; seguono poi esempi di scrittura di formule matematiche, di sottolineatura, di intensificazione, di centratura del testo e, infine, dell'effetto globale sullo stesso brano di testo, stampato una volta in modo normale, la seconda con la "spaziatura proporzionale": si noti la diversa densità dei caratteri nei due esempi

La tastiera dello Spectrum

Codici assegnati ai tasti e come leggerli direttamente

Il BASIC dello Spectrum prevede due comandi per l'immissione di dati da tastiera: INPUT e INKEY$. Il primo "legge" una serie di caratteri digitati dall'operatore, trasferendoli in una variabile numerica o alfanumerica, finché non viene premuto un ENTER. Il comando INKEY$, invece, esegue una scansione della tastiera e torna al programma principale con il carattere digitato oppure, se nessun tasto è stato premuto, con il carattere spazio (ossia con una stringa "nulla"). Entrambi questi comandi sono molto utili, ma, per alcune applicazioni (quali la lettura simultanea di più tasti), torna più comodo l'impiego della funzione IN.

La tastiera dello Spectrum è collegata al microprocessore Z80 tramite porte di ingresso/uscita. Ognuna delle 65536 porte di ingresso/uscita disponibili è indirizzabile individualmente. Per la "lettura" e la "scrittura" di dati attraverso le porte di ingresso/uscita si usano IN e OUT, con un funzionamento analogo a PEEK e POKE, quando si tratta di leggere o scrivere nella memoria del computer: IN(m) legge il contenuto della porta m, mentre OUT(m,n) emette (ossia, scrive) il valore n dalla porta m.

La tastiera consiste di quattro file di 10 tasti ciascuna ed ogni fila può essere ulteriormente suddivisa in due metà (si veda l'illustrazione): ciascuna di queste sezioni è collegata ad una precisa porta del computer. Si noti che la configurazione di bit adoperati nelle porte è diversa per le due metà destra e sinistra: in una vengono adoperati i bit più a destra, nell'altra quelli più a sinistra.

Ad ogni bit della porta corrisponde un tasto della rispettiva sezione: esso vale 0 se il tasto viene premuto, oppure 1 in caso contrario. I bit contrassegnati con delle X non vengono adoperati ed il loro valore può essere sia 0 che 1. Tuttavia, l'incertezza di questi valori significa che il risultato di IN(m) è indefinito. Il problema si aggira facilmente, via software, "forzando" sempre i bit non significativi a 0, ed un comodo sistema per far ciò è definire la seguente funzione:

```
DEF FN a(p) = p - INT(p/32)*32
```

dove p è il valore letto dalla porta, ossia il byte fornitoci da IN(m). La divisione p/32 converte il numero in uno compreso tra 0 ed un valore inferiore ad 8. Con INT(p/32) vengono eliminati tutti i valori decimali dopo la virgola, ottenendo un valore intero compreso tra 0 e 7. Adesso, moltiplicando questo risultato per 32, si risale al valore dei tre bit più a sinistra di p, che, sottratto al valore originale di p, rimuove i tre bit indesiderati. Il risultato di questo procedimento, complicato solo in apparenza, è un valore compreso tra 0 e 31, nel quale influiscono soltanto i cinque bit che

I codici della tastiera

Nello Spectrum, i tasti vengono suddivisi in 8 sezioni di cinque tasti ciascuna e ogni sezione viene collegata ad una precisa porta del computer. Ad ogni tasto della sezione corrisponde uno dei bit degli otto a disposizione, pertanto i tre bit più significativi (quelli più a sinistra) non vengono mai usati. Premendo un tasto, il valore del bit corrispondente diventa 1, altrimenti esso vale 0. Si noti che l'ordine di corrispondenza, tra bit e tasti, varia da una metà all'altra della tastiera. Dunque, la pressione simultanea dei tasti Y, I ed O dà per risultato, alla porta 57342, il valore XXX01001 (le X indicano i bit indefiniti). Se però vengono premuti i tasti A ed S, il valore, alla porta 65022, è XXX11100



KEVIN JONES

a noi interessano. Di conseguenza, qualsiasi valore assumano i tre bit più a sinistra, FN a(p) ci darà sempre l'esatto valore dei cinque più a destra.
Se, ad esempio, viene premuto il tasto corrispondente alla lettera V, la porta 65278 conterrà:

X	X	X	0	1	1	1	1

cosicché FN a(IN(65278)) fornirà il valore:

0 0 0 0 1 1 1 1 = 15

qualora, assieme alla V, venga premuto anche il tasto Caps Shift, la porta conterrà:

X	X	X	0	1	1	1	0

ed FN a(IN(65278)) ci fornirà il valore 14. Quando non viene premuto alcun tasto di questa sezione, otterremo il valore 31 (00011111). Si provi questo programma:

```
10 REM Visione dei valori (mascherati) delle porte
20 REM Definizione della funzione di "mascheramento"
30 DEF FN a(p) =p-INT(p/32)*32
40 REM Visualizza le porte
50 PRINT AT 1,1;"Porta","Valore mascherato"
60 PRINT "32766", FN a(IN(32766))
70 PRINT "49150", FN a(IN(49150))
80 PRINT "57342", FN a(IN(57342))
90 PRINT "61438", FN a(IN(61438))
100 PRINT "63486", FN a(IN(63486))
110 PRINT "64510", FN a(IN(64510))
120 PRINT "65022", FN a(IN(65022))
130 PRINT "65278", FN a(IN(65278))
140 FOR i=1 TO 250:NEXT i
150 CLS
160 GO TO 50
```

Lanciato il programma, si provi a digitare qualche tasto, per vedere come variano i valori delle rispettive porte. Poi si tenti d'indovinare il valore risultante dalla pressione di uno o più tasti di ciascuna sezione. Premendo i tasti del cursore (5, 6, 7 ed 8), dalle porte 61438 e 63486, qui indicate rispettivamente con A e B, si dovrebbe ottenere:

	Porta A (61438)	Porta B (63486)
←	31	15
↓	15	31
↑	23	31
→	27	31

Premendo il tasto Caps Shift, otterremo il valore (mascherato) 30 dalla porta 65178 (che chiameremo porta C).
Si osservi il semplice programma grafico presentato più avanti: esso permette di tracciare linee orizzontali, verticali e diagonali, agendo sui tasti del cursore. Adoperando simultaneamente il Caps Shift, i detti tasti servono per spostare la "penna", da un punto all'altro, senza però tracciare linee. Le diagonali si ottengono premendo contemporaneamente due tasti del cursore.
La linea 30 del programma definisce la solita funzione FN(a), per mascherare i tre bit non significativi. La posizione della penna grafica è stabilita dai valori di x (orizzontale) e y (verticale). La linea 50 fissa la posizione iniziale della penna più o meno al centro dello schermo.
Le linee 70 e 80 leggono i valori delle porte associate ai tasti del cursore: la sezione che comprende i tasti da 0 a 6 corrisponde alla porta 61438 (qui indicata con A), quella che contiene i tasti da 5 a 1 corrisponde alla porta 63486 (porta B). La linea 90 legge la porta 65278 (ossia la C) relativa al tasto Caps Shift.
Le linee da 110 a 180 verificano ciascuna delle combinazioni di tasti consentite, correggendo, di conseguenza, i valori delle coordinate x e y della penna grafica:

Linea	Verifica
110	↑
120	↑ e →
130	→
140	↓ e →
150	↓
160	↓ e ←
170	←
180	↑ e ←

Le linee da 200 a 240 garantiscono che la penna non fuoriesca dai margini dello schermo. Infine, se il tasto Caps Shift non è premuto, la linea 250 traccia sullo schermo un punto della linea, ripetendo poi il processo daccapo.

Muoversi coi tasti

Il programma qui riportato impiega un'insolita tecnica per leggere i codici associati alla tastiera dello Spectrum, che permette di rilevare anche la contemporanea pressione di più di un tasto. Per tracciare linee orizzontali, verticali e diagonali si usano i soli tasti del cursore (anche combinati tra loro), mentre per spostare la "penna" in un nuovo punto (senza tracciare linee) si impiega anche il tasto Caps Shift, simultaneamente ai precedenti

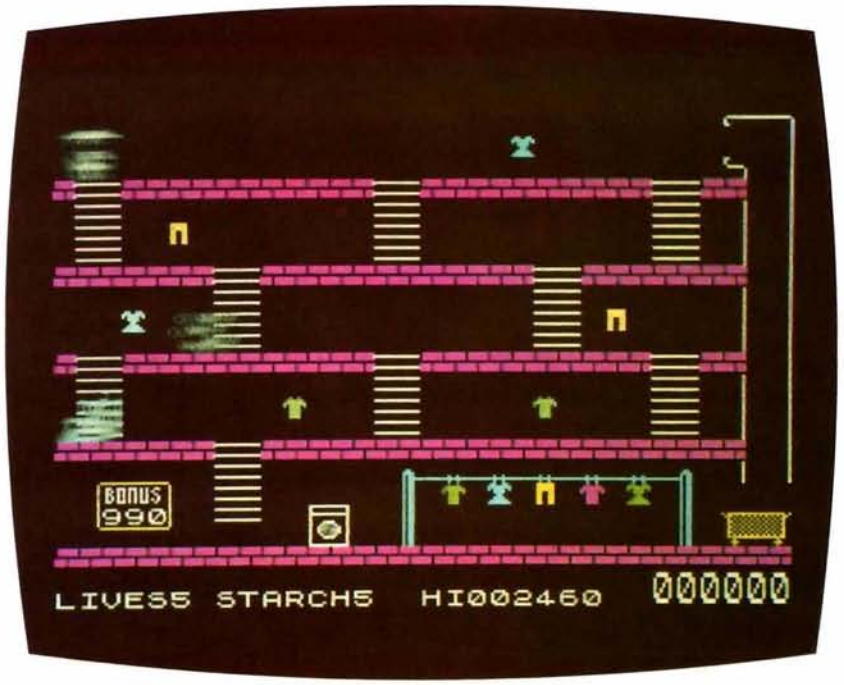
LA PRESSIONE SIMULTANEA DI PIÙ TASTI

```
 20 REM Definizione della funzione di "mascheramento"
 30 DEF FN a(p)=p-INT (p/32)*32
 40 REM Posizionamento iniziale della penna
 50 LET x=127: LET y = 35
 60 REM Lettura porte (mascherando i  3 bit indefiniti)
 70 LET portA = FN a(IN 61438)
 80 LET portB = FN a(IN 63436)
 90 LET portC = FN a(IN 65278)
100 REM Varia la posizione della penna in base alla combinazione
    di tasti premuta
110 IF portA=23 AND portB=31 THEN LET y=y+1
120 IF portA=19 AND portB=31 THEN LET x=x+1: LET y=y+1
130 IF portA=27 AND portB=31 THEN LET x=x+1
140 IF portA=11 AND portB=31 THEN LET x=x+1: LET y=y-1
150 IF portA=15 AND portB=31 THEN LET y=y-1
160 IF portA=15 AND portB=15 THEN LET x=x-1: LET y=y-1
170 IF portA=31 AND portB=15 THEN LET x=x-1
180 IF portA=23 AND portB=15 THEN LET y=y+1: LET x=x-1
190 REM Controllo fuoriuscita dallo schermo
200 IF x<0 THEN LET x=0
210 IF x>255 THEN LET x=255
220 IF y<0 THEN LET y=0
230 IF y>175 THEN LET y=175
240 REM Visualizza nuovo punto (se CAPS SHIFT non premuto)
250 IF portC=31 THEN PLOT x,y
260 GO TO 70
```

Software in vetrina

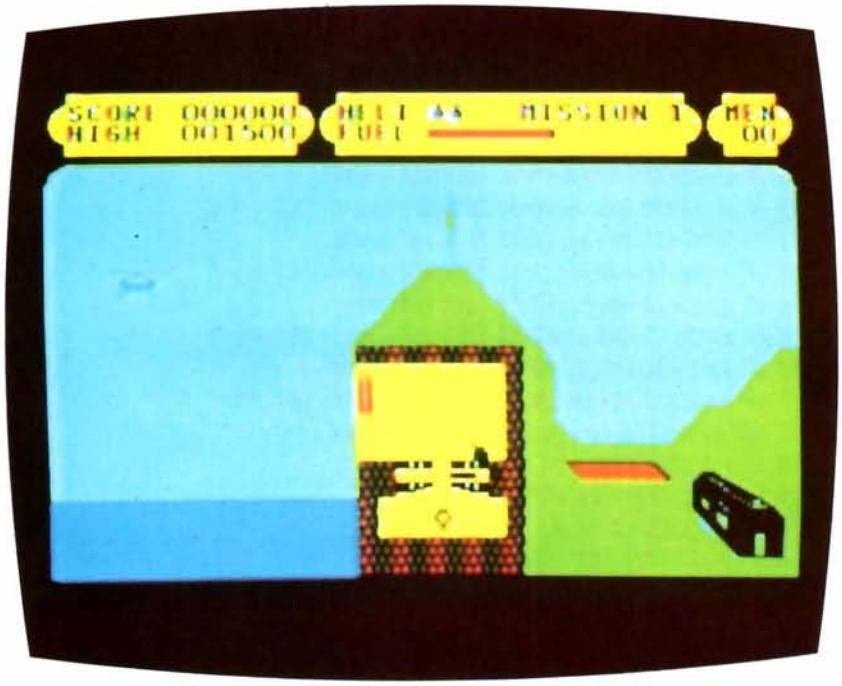
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Esaminiamo brevemente due simpatici e divertenti giochi che appartengono al genere "arcade". Il primo è un classico "acchiappino", mentre il secondo mette alla prova le nostre capacità nella guida di un elicottero.



Mr. Wong's loopy laundry per lo Spectrum

In questo gioco, il protagonista è il piccolo Mr. Wong, proprietario di una lavanderia. Lo svolgimento è abbastanza classico, e lo si può intuire già dall'illustrazione: Mr. Wong deve raccogliere il bucato, cercando di sfuggire a una serie di oggetti (una pila di panni sporchi, saponette, ecc.) che lo inseguono alacremente. A sua difesa, Mr. Wong ha un'insolita arma: uno spray all'amido, che "intecchierisce" i suoi inseguitori. Sebbene originale nell'idea, lo scenario del gioco è abbastanza consueto per gli appassionati del genere "arcade". L'azione è forse un po' lenta (adoperando il joystick si consegue facilmente la vittoria). Il programma è prodotto dalla Artic (GB).



Blue thunder per il Commodore 64

Questo è un tipico gioco d'azione, in cui il giocatore si trova implicato in una pericolosa missione (tra le cinque disponibili), alla guida di un modernissimo elicottero. La prima missione, quella più facile, consiste nel distruggere un reattore nucleare, protetto da uno scudo mobile. La zona in cui esso si trova è controllata da un'agguerrita postazione contraerea ed occorre molta abilità e tempismo nel manovrare l'elicottero. Una volta distrutto l'obiettivo, si può scegliere se tornare alla base oppure se cimentarsi nell'eroica, ma quantomai pericolosa, impresa di salvare i "sopravvissuti". Ritornato alla base, il giocatore affronta la missione successiva, di difficoltà sempre crescente. Ovviamente, i migliori risultati si ottengono adoperando il joystick. Il gioco è prodotto da Richard Wilcox (GB).

L'unità Torch per il BBC

Un efficace modo per aumentare le prestazioni del Micro BBC

Volendo, il Micro BBC può essere impiegato anche per applicazioni di tipo commerciale, purché si adottino opportuni moduli di espansione. Una delle prime ditte a progettare e a realizzare un modulo di questo tipo è stata la Torch. Il Torch Disk Pack non solo aggiunge due unità a dischetti, ma consente anche di ampliare la memoria RAM e, fatto molto notevole, di usare il sistema operativo CP/M. Il tutto ad un prezzo leggermente superiore a quello di una doppia unità a dischetti della Acorn.

Come è noto, il CP/M "gira" soltanto sui computer dotati di microprocessori Z80 e di almeno un'unità a dischi. L'espediente dell'unità Torch è quello di affiancare, appunto, un chip Z80 al 6502, usato dal BBC, utilizzando quest'ultimo soltanto per le funzioni di ingresso/uscita. In sostanza, la tastiera, lo schermo e le unità a dischi vengono controllate dal 6502, mentre lo Z80 si dedica all'esecuzione dei programmi CP/M (sfruttando al massimo la memoria, espansa a 64 Kbyte) ed al controllo globale dell'apparecchio. I dati vengono scambiati tra i due microprocessori tramite la porta di espansione del BBC, nota col nome "Tube".

Lo Z80 e la memoria aggiuntiva sono fornite su un'apposita scheda, da inserire all'interno del BBC. Per non sovraccaricare il circuito di alimentazione del BBC, viene adoperato un analogo circuito, contenuto nell'involucro che racchiude anche le unità a dischetti. Queste ultime prevedono l'impiego di dischetti a doppia faccia da 80 tracce e sono molto simili alle unità da 800 Kbyte prodotte dalla stessa Acorn.

La Torch propone anche un'alternativa, a chi già possieda le proprie unità a dischetti: il modulo ZEP 100, essenzialmente identico al precedente, ma privo dei dischetti. Inoltre, la Torch produce anche computer gestionali, ricavandoli da unità centrali BBC, incorporandovi però anche un monitor ed un modem.

Il Disk Pack è progettato in modo da formare un comodo piedistallo per il BBC, sotto il quale si incastra perfettamente, senza brutte sporgenze o rientranze. La lunghezza delle connessioni è ridotta al minimo, con evidente guadagno estetico. L'unico fatto negativo è che l'apparecchio sopreleva la tastiera di 7 centimetri buoni e l'uso di quest'ultima può risultare scomodo.

Il Disk Pack viene venduto completo di tutte le parti necessarie all'espansione del BBC, con un'importante eccezione: i chip d'interfaccia per il collegamento delle unità a dischetti. Alcuni possessori di BBC li avranno acquistati assieme al computer, ma gli altri, per poter usare il Disk Pack, dovranno mettere in conto anche questa spesa.

All'accensione, con il Disk Pack correttamente installato, il computer entra automaticamente in MCP, il sistema operativo scritto dalla stessa Torch, compatibile con il CP/M della Digital Research. In effetti, la sola differenza apprezzabile tra CP/M ed MCP è il fatto che quest'ultimo è memorizzato in ROM e quindi disponibile immediatamente, senza alcun caricamento da dischetto. Il sistema operativo MCP consente di utilizzare una vastissima quantità di programmi scritti per il CP/M.

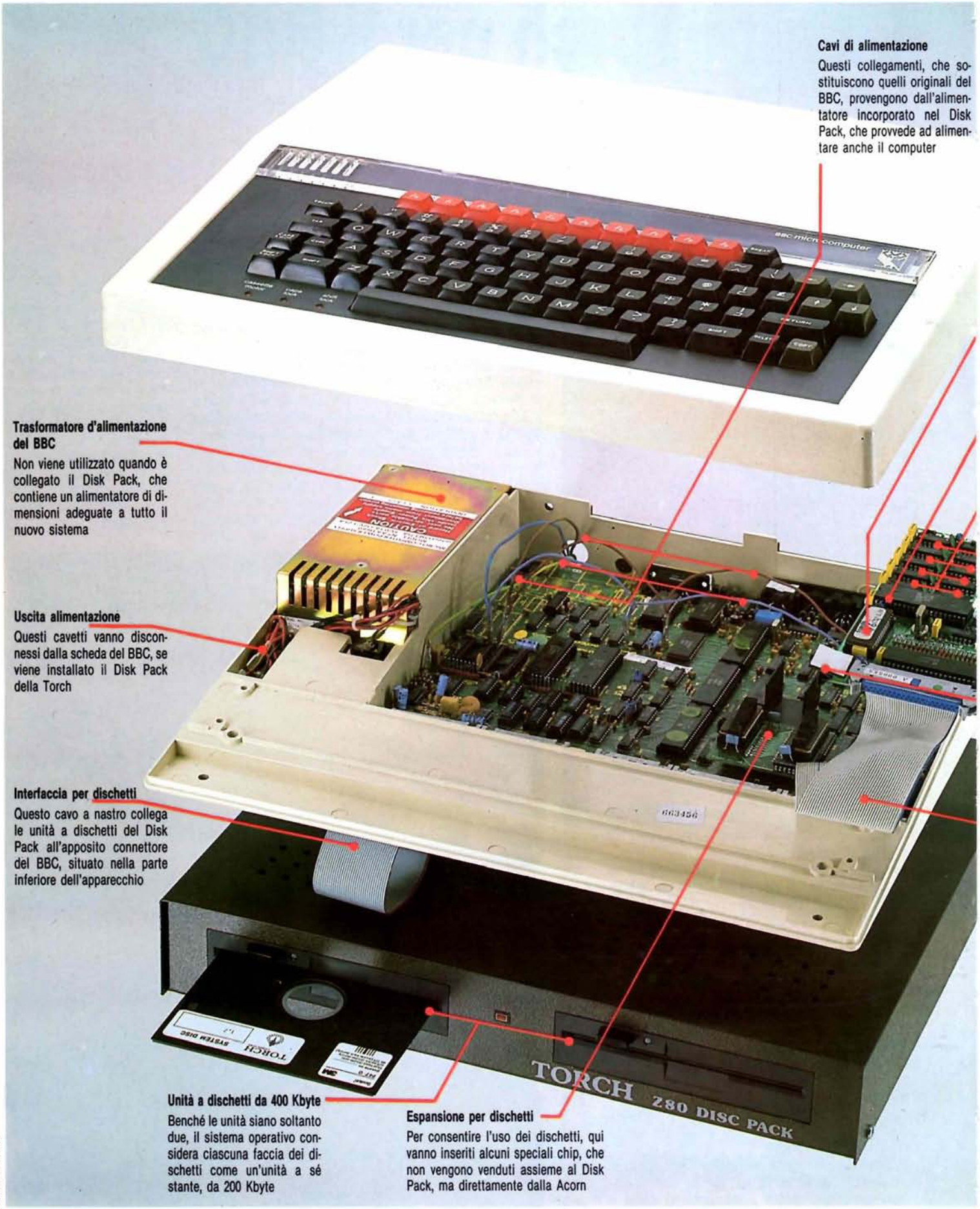
Molti dei comandi dell'MCP sono familiari

Estensione al BBC

Il modulo Disk Pack, prodotto dalla Torch, trasforma il Micro BBC in un apparecchio per applicazioni commerciali, aggiungendovi un microprocessore Z80, 64 Kbyte di memoria, un sistema operativo MCP, compatibile con il diffusissimo CP/M della Digital Research e, "dulcis in fundo", il modulo contiene due unità a dischetti da 400 Kbyte. Il tutto ad un prezzo poco superiore a quello di una doppia unità a dischetti della Acorn



CHRIS STEVENS



Cavi di alimentazione
Questi collegamenti, che sostituiscono quelli originali del BBC, provengono dall'alimentatore incorporato nel Disk Pack, che provvede ad alimentare anche il computer

Trasformatore d'alimentazione del BBC
Non viene utilizzato quando è collegato il Disk Pack, che contiene un alimentatore di dimensioni adeguate a tutto il nuovo sistema

Uscita alimentazione
Questi cavetti vanno disconnessi dalla scheda del BBC, se viene installato il Disk Pack della Torch

Interfaccia per dischetti
Questo cavo a nastro collega le unità a dischetti del Disk Pack all'apposito connettore del BBC, situato nella parte inferiore dell'apparecchio

Unità a dischetti da 400 Kbyte
Benché le unità siano soltanto due, il sistema operativo considera ciascuna faccia dei dischetti come un'unità a sé stante, da 200 Kbyte

Espansione per dischetti
Per consentire l'uso dei dischetti, qui vanno inseriti alcuni speciali chip, che non vengono venduti assieme al Disk Pack, ma direttamente dalla Acorn

CHRIS STEVENS

agli utenti del BBC: sono presenti tutti i comandi *FX e VDU, oltre alla selezione del MODE per il video. Detto per inciso, all'accensione viene automaticamente selezionato il MODE 3. Il comando *KEY serve ancora per definire i tasti di funzione, quattro dei quali, comunque, sono già assegnati ad altrettanti utili comandi.
L'MCP amplia enormemente la quantità di comandi per la gestione dei file: oltre alla possibilità di caricare ed eseguire i programmi in codice macchina semplicemente digitandone il nome, il sistema offre anche un comando per il caricamento di speciali file, coi quali conservare e riprodurre immagini video, create tramite il sistema operativo del BBC. Il testo contenuto in un file può essere stampato col semplice comando PRINT, mentre TYPE lo visualizza sullo schermo. Una funzione simile alla EXEC del BASIC BBC, per l'esecuzione di una sequenza di comandi, si ottiene con COMMAND.
Altri programmi di utilità sono forniti su di un dischetto, venduto assieme al Disk Pack: per adoperarli, basta digitarne il nome. Tra questi, si trovano: una routine per modificare la forma dei caratteri sul video, una per creare e riprodurre brani musicali, un "debugger" per codice macchina, un "disk editor" (per esaminare e modificare il contenuto di un dischetto) e, infine, un utilissimo programma che consente al BBC di leggere i dischetti sia nel formato Acorn che in quello Torch.
Benché i formati Torch ed Acorn per i dischetti siano diversi, entrambi usano la strana convenzione di trattare le due unità a dischetti come se fossero quattro.
Nonostante tutti questi miglioramenti, il sistema può ancora venir usato in modo standard: basta impartire il comando *BASIC, ed il computer si comporterà come se l'espansione Disk Pack non esistesse affatto. L'unico fatto strano è che, a questo punto, il sistema è selezionato per il funzionamento col registratore a cassette e non con i dischetti, pertanto è necessario effettuare un'ulteriore selezione, specificando che i dischetti sono effettivamente collegati. Per tornare ad usare il Disk Pack, basta digitare il comando *MCP in BASIC, oppure spegnendo e riaccendendo il computer (ovviamente se non si hanno lavori... in corso!).

ROM della CCCP
La "Cambridge Console Command Processor" è una parte del sistema operativo MCP, necessaria all'interpretazione dei comandi. Il resto dell'MCP si trova su altri chip ROM

Microprocessore Z80
Questo secondo processore controlla l'esecuzione dei programmi, sotto il sistema operativo MCP e l'intero sistema nel suo complesso, lasciando tutte le operazioni di ingresso/uscita al 6502 pre-esistente

RAM
Questo è il "banco" di memoria da 64 Kbyte, utilizzato dallo Z80. I 32 Kbyte usati dal BBC, vengono dedicati al video



Alette di fissaggio
Queste alette autoadesive fissano la scheda al contenitore del BBC

Cavo per la scheda dello Z80
Questo cavo a nastro serve a collegare la scheda col secondo processore all'unità principale, sfruttando l'interfaccia "Tube"

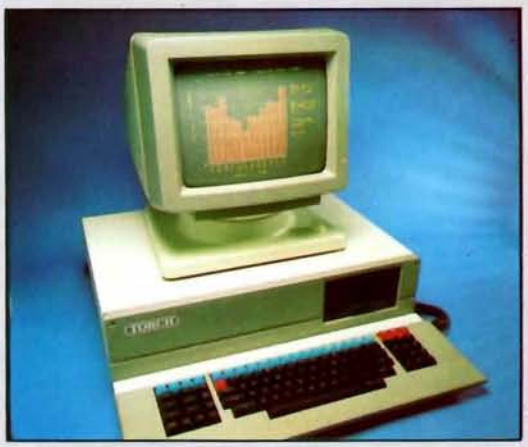
Installare il Disk Pack
Le operazioni per il montaggio di questa espansione sono relativamente semplici. Il computer deve essere un BBC Modello B, dotato di un'interfaccia per dischetti. Poiché l'alimentazione originale non è sufficiente per tutto il sistema, occorre sostituire sette cavetti (fortunatamente dotati di connettori a pressione) con i corrispondenti, provenienti dal Disk Pack.
Un cavo a nastro collega il Disk Pack all'interfaccia per dischetti del BBC. Infine, va inserita la ROM che contiene il sistema operativo MCP



CHRIS STEVENS

Software a volontà
Assieme al Disk Pack vengono venduti numerosi "pacchetti" software, tra cui un word processor (Perfect Writer), un correttore ortografico (Perfect Speller), un programma per la gestione di archivi (Perfect Filer) ed uno "spreadsheet" (Perfect Calc).
Inoltre, viene fornita anche una versione del BASIC BBC, adatta al secondo microprocessore: tutti i comandi sono identici alla versione normale in ROM, eccezion fatta per le istruzioni dell'assembler, che riguardano lo Z80 e non il 6502. Indubbiamente, tutta la programmazione risente positivamente dei 48 Kbyte liberi, qualsiasi MODE si usi per lo schermo (il BBC, normalmente, non ha più di 28 Kbyte effettivamente liberi, ma a seconda del MODE scelto, si può scendere fino a soli 9 Kbyte!)

Il chip DFS
Come già accennato, il Disk Pack della Torch non include i chip ROM necessari all'interfaccia per dischetti. Questi vengono venduti separatamente, poiché occorrono qualunque sia il sistema a dischetti usato. La sigla DFS sta per "Disk Filing Sistem", ossia "sistema di gestione file su disco". L'acquisto di questi chip è utile anche se non si prevede quello del Disk Pack.
Siccome il formato dei dischetti Acorn differisce da quello usato con il Disk Pack, la Torch fornisce un apposito programma che consente la lettura e la scrittura su dischetti di entrambi i formati



Altre offerte della Torch
La Torch produce anche elaboratori completi, alcuni dei quali dotati del sistema operativo Unix (la serie 700, di cui un esemplare è qui illustrato). Esiste anche una gamma di terminali, chiamata Serie 300, per il collegamento in rete di più computer Torch

Colori e numeri

Un gioco per esercitare la propria memoria visiva e auditiva

Alcuni anni fa, divenne molto diffuso un gioco elettronico, basato su di un vecchissimo gioco per bambini, chiamato *Simon*. Lo svolgimento è semplice: l'apparecchio, di forma circolare, è dotato di quattro grosse piastre mobili a forma di quarto di cerchio, che costituiscono i "tasti".
Ogni tasto ha un proprio colore e, se premuto, emette una nota di tono diverso da quello degli altri tasti. In questa versione elettronica, si gioca contro il computer, che costruisce una sequenza casuale di colori (o note) da ripetere con precisione sui tasti. Indovinata (o per meglio dire "ricordata") la prima breve sequenza, ad essa viene aggiunto un altro colore e la prova è ripetuta. Il giocatore può scegliere tra più livelli di difficoltà, in base alla velocità ed alla lunghezza delle sequenze di colori (o note emesse).
Il programma *Ripeti questa!*, qui presentato, è una versione computerizzata di questo vecchio gioco da bambini: i colori sono quattro e a ciascuno di essi è associata una nota musicale ed un tasto (da 1 a 4) sulla tastiera. Viene visualizzato un colore e, se si preme il tasto ad esso corrispondente, ne viene presentato un altro e così via, fino al termine del gioco.
Se si commettono più di tre errori in una sola partita o se si impiega troppo tempo a rispondere, il gioco termina con la nostra sconfitta.
Per rendere lo svolgimento un po' più difficile, la velocità di presentazione aumenta progressivamente, benché non sia necessario rispondere allo stesso ritmo. La massima lunghezza di una sequenza è di 50 colori, ma ben pochi riusciranno a superare la barriera dei 15, oltre i quali la memoria viene veramente messa a dura prova!
Vale la pena di esaminare la struttura del programma. Tutte le subroutine relative all'emissione di colori e di suoni sono raggruppate in fondo al programma:

1000 Visualizza il colore di numero a
1500 Rileva la pressione dei tasti 1-4
2000 Emette un suono di fine gioco
2500 Emette un suono per avvertire che una risposta è sbagliata
6000 Visualizza un messaggio sulla riga 20 dello schermo, seguito (eventualmente) da una pausa

Queste routine potrebbero essere assimilate alle varie componenti hardware di un giocattolo. Tenerle separate dal corpo principale del programma offre due vantaggi: in primo luogo i gruppi di istruzioni relative all'emissione di suoni e colori, che possono variare da un computer all'altro, sono raggruppate tutte assieme; in secondo luogo, ciò facilita l'impiego di queste routine in altri giochi di simile impostazione.
Anche lo stesso programma qui presentato può essere modificato in più modi per adattarlo ai propri gusti. Magari, potremo aggiungervi un contapunti ed un contatempo, assieme ad un conteggio del punteggio massimo raggiunto da ciascun giocatore.

KEVIN JONES

RIPETI QUESTA

```
 10 REM *** Ripeti questa! ****
 30 LET h=0:  LET n=0:  LET w=3
 40 DIM c(4):  DIM p(4):  DIM a(50)
 50 LET p(1)=5:  LET p(2)=8:  LET p(3)=12:
    LET p(4)=15
 60 LET c(1)=1:  LET c(2)=2:  LET c(3)=4:
    LET c(4)=6
 70 LET s$=" ":  FOR i=1  TO 5:  LET s$=s$+s$:
    NEXT i
 80 LET b$=CHR$(143) : REM un qudratino grafico
100 REM *** Pagina delle istruzioni ***
110 CLS : PRINT TAB (10);"Ripeti questa!"
120 PRINT : PRINT
130 IF n>0 THEN PRINT:PRINT "Hai vinto in
    ";n;" mosse"
140 IF n>h THEN PRINT:PRINT "... un nuovo
    record!!!":  LET h=n
150 IF h>0 THEN PRINT:PRINT "Il miglior
    risultato finora e' ";h;" mosse"
155 PRINT:PRINT "Prova a ripetere la
    sequenza       di luci e suoni generata
    dal computer"
160 PRINT "usando i tasti da 1 a 4"
170 PRINT:PRINT "Premere G per Giocare, F per
    Fine"
180 LET a$=INKEY$:  IF a$="" THEN GO TO 180
190 IF a$="f" OR a$="F" THEN CLS:STOP
200 IF a$<>"g" AND a$<>"G" THEN GO TO 180
205 REM *** Nuova partita ***
210 CLS:PRINT TAB(10);"Ripeti questa!"
220 FOR a=1 TO 4: GO SUB 1000: NEXT a
230 LET n=0:  LET m=0:  RANDOMIZE
240 REM *** Mossa successiva ***
250 LET n=n+1
270 LET a(n)=INT  (RND*4)+1
280 IF m=w THEN GO SUB 2000: LET m$="*"+STR$(w)+
   " risposte errate!":GO SUB 6000: GO TO 100
285 LET m$="*Eccola...": GO SUB 6000
290 FOR i=1  TO n
300 LET a=a(i):  GO SUB 1000
310 FOR j=1  TO 100/n:  NEXT  j
320 NEXT i
330 LET m$="Ripeti questa...": GO SUB 6000
340 LET i=1
350 GO SUB 1500
360 IF t=0 THEN GO SUB 2000: LET m$="*Troppo
    lento!": GO SUB 6000: GO TO 100
370 IF a<>a(i) THEN LET m=m+1:  GO SUB 2500:
    GO TO 280
380 LET i=i+1:  IF i<=n THEN GO TO 350
390 IF n>50 THEN LET m$="*Vinci con 50 mosse!":
    GO SUB 6000: GO TO 100
400 LET m$="*Preparati a ricominciare": GO
    SUB 6000
410 GO TO 250
1000 REM *** Illumina lo scomparto a ***
1010 INK c(a)
1015 LET p=(a-1) *8+2
1020 FOR l=10  TO 14
1030 PRINT AT l,p;
1040 IF l=12  THEN PRINT b$;b$;a;b$;b$;
1050 IF l<>12  THEN PRINT b$;b$;b$;b$;b$;
1060 NEXT l
1070 BEEP 2/(n+1),p(a)
1080 PRINT AT 10,p;"      "
1090 PRINT AT 11,p;" ";b$;b$;b$;" ";
1100 PRINT AT 12,p;" ";b$;a;b$;" ";
1110 PRINT AT 13,p;" ";b$;b$;b$;" ";
1120 PRINT AT 14,p;"      "
1130 INK 0: RETURN
1500 REM *** Immissione di un carattere ***
1510 LET  t=250
1520 LET a$=INKEY$:  IF a$=""  THEN LET t=t-1:
    IF t>0 THEN  GO TO 1520
1530 IF t=0  THEN RETURN
1540 IF a$<>"1"  AND  a$<>"2"  AND a$<>"3"
    AND a$<>"4 THEN  GO TO 1520
1550 LET a=VAL  (a$): GO SUB 1000
1560 RETURN
2000 REM *** Avviso acustico di un errore ***
2010 BEEP 3,0: RETURN
2500 REM *** Suono di avvertimento ***
2510 BEEP 1,0: RETURN
6000 REM *** Visualizza m$ ***
6010 PRINT AT 20,1;s$;AT  20,1;
6030 IF m$(1)="*"  THEN PRINT m$(2 TO):
     FOR z=1 TO 200: NEXT z
6040 IF m$(1)<>"*"  THEN PRINT m$
6050 RETURN
```

Basic & Basic

Commodore 64

Sostituire CLS con PRINT CHR$(147)
Sostituire LET A$=INKEY$ con GET A$
Sostituire RANDOMIZE con XX=RND(—TI)
Sostituire (RND*4) con (RND(1)*4)
Sostituire M$(1) con LEFT$(M$,1)
Sostituire M$(2 TO) con MID$(M$,2)
Sostituire CHR$(143) con CHR$(166)
Sostituire PRINT AT L,C; con PRINT LEFT$(DN$,L+1)TAB(C);
Ad es. la linea 6010 diventa:
6010 PRINT LEFT$(DN$,21)TAB(1);S$; LEFT$(DN$,21)TAB(1);
Sostituire DIM C(4) con DIM C$(4)
Sostituire b$;b$;a;b$;b$; con BBZBB$; in 1040
Sostituire b$;a;b$; con BZB$ in 1100
Inserire:

```
20 VL=54296:AD=54277:SR=AD+1:WF=AD—1:
N0=17:N1=N0:LF=AD—5:HF=LF+1
25 POKE AD,255:POKE SR,48:POKE VL,15
50 C$(1)=CHR$(31):C$(2)=CHR$(28):C$(3)=
   CHR$(30):C$(4)=CHR$(158)
60 P(1)=51:P(2)=34:P(3)=64:P(4)=38
90 DN$=CHR$(17):FOR K=1 TO 5:
DN$=DN$+DN$:NEXT K:
DN$=CHR$(19)+DN$
1010 PRINT C$(A);
1015 P=(A—1)*9+3:Z$=RIGHT$(STR$(A),1)
1130 PRINT CHR$(144):RETURN
2010 SD=15:SP=4:N1=33:GOSUB 7000:RETURN
2510 SD=10:SP=10:N1=33:GOSUB 7000:RETURN
7000 REM*** BEEP SD,SP
7010 POKE VL,15:POKE WF,N1
7020 POKE LF,SP:POKE HF,SP:
FOR DD=1 TO SD*50:NEXT DD
7030 POKE HF,0:POKE LF,0:N1=N0:RETURN
```

Micro BBC

Sostituire AT Y,X con TAB(X,Y)
Ad esempio, la linea 6010 diventa:
6010 PRINT TAB(1,20);S$;TAB(1,20)
Sostituire INKEY$ con INKEY$(0)
Inserire:

```
20 MODE 2
25 COLOUR 135:CLS
60 C(1)=1:C(2)=2:C(3)=4:C(4)=5
80 B$=CHR$(35)
1010 COLOUR C(A)
1015 P=(A—1)*4
1070 SOUND 1,—10,P(A),40/(N+1):FOR DE=1
TO 2000/N:NEXT
1130 COLOUR 0:RETURN
2010 SOUND 1,—15,2,40
2510 SOUND 1,—15,40,40
```

Piano d'azione

L'impostazione da dare allo sviluppo di un programma, prima di codificarlo

Il progetto di un programma è considerato dagli interessati (ossia il programmatore e l'utente) una complessa esercitazione formale sulla risoluzione di problemi (il "problem solving"). Malauguratamente, si pensa sempre che i problemi maggiori derivino da ostacoli tecnici (come presentare le informazioni sullo schermo o velocizzare un ciclo o, ancora, come far rientrare il programma nella RAM disponibile), ma in realtà molti degli inconvenienti sono di tutt'altra natura, e sono già presenti, anche se non evidenti, al primo incontro tra l'utente e l'"esperto". Tuttavia, il più delle volte, chi commissiona un programma non possiede un'esatta cognizione dei problemi che intende risolvere con esso. Dal canto suo, l'esperto pecca talvolta di presunzione e ritiene di conoscere già la soluzione a detti problemi, prima ancora che essi vengano esposti. Il risultato di incontri di questo tipo è un'impostazione iniziale sbagliata, dalla quale, il più delle volte, scaturisce un prodotto finale inutile o perlomeno non adatto alle concrete esigenze dell'utente. Nella produzione di software per home computer, spesso e volentieri il programmatore lavora per se stesso, ossia svolge anche il ruolo di "analista", oltreché quello di utente finale. Già questo fatto dovrebbe diminuire i rischi derivanti da una cattiva comunicazione programmatore/utente, tuttavia conviene attenersi a precise regole anche in questo caso, per avere le idee ben chiare fin dalla partenza.

Consideriamo un immaginario utente di computer appassionato di modellini aeronautici: egli desidera crearsi un archivio relativo ai vari materiali impiegati nei vari progetti (tipo di colla, di giuntura, ecc.), per poterne ricavare utili informazioni in successivi progetti. Ciò che l'"utente" deve ben specificare è:

- La funzione del programma. Inizialmente, questa può esser descritta anche vagamente, con un semplice "voglio poter archiviare informazioni sui miei modellini". Il programmatore dovrà però indagare sulla qualità e la struttura di queste "informazioni": «È prevista una descrizione del modellino?», «La ricerca va fatta in base al solo nome dell'aereo, o anche in base ad alcuni dettagli descrittivi?», ecc.
  Oltre a chiarire che sorta di informazioni andranno elaborate, ciò offre qualche indicazione anche sulle tecniche di programmazione da usare (memorizzazione, ricerca, indicizzazione, ecc.).

- Come verrà usato il programma. Alcuni dettagli sul tipico impiego del programma possono risaltare dalla descrizione sulla funzione, ma potrebbero non esser complete. Per esempio, le varie descrizioni dei modellini potrebbero essere più utili su carta, anziché su video, per esser meglio consultati durante il lavoro.

- Quale formato devono avere l'input e l'output. Spesso i programmatori si avvalgono di speciali reticoli per progettare la più opportuna disposizione delle informazioni sullo schermo: questa è una pratica consigliabile, specialmente se si prevede l'uso di schermi grafici, e rappresenta l'"interfaccia" tra macchina e utente: una disposizione logica delle informazioni sullo schermo è importante quanto il corretto impiego delle istruzioni nel programma.

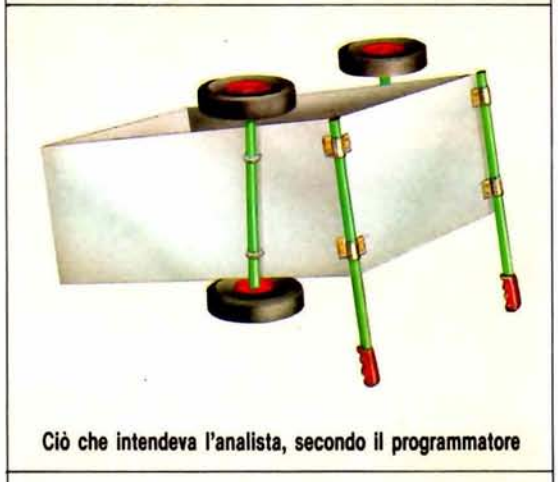
- Quale deve essere il formato dell'archivio. Può darsi che la particolare applicazione preveda l'archiviazione di oltre un centinaio di descrizioni di singoli modellini, oppure che non si preveda di superare mai una dozzina di descrizioni "standard". La quantità dei dati ha un'enorme influenza sui metodi di accesso da adottare nel programma: attendere che il programma ricerchi sequenzialmente una descrizione, tra una decina registrate, su nastro, può essere ancora accettabile, ma se le descrizioni diventano 80 o 100 questo metodo sarebbe fuori questione. In un caso simile, si potrebbe prevedere l'impiego di svariate cassette, ognuna relativa ad una particolare "famiglia" di aerei.
  Anche le dimensioni dello stesso programma possono creare dei problemi: se il programma prevede un largo impiego di messaggi, la sezione necessaria alla gestione dei file ed alla ricerca dei dati è complessa ed ingombrante o, in definitiva, il programma è troppo grande, e quindi andrà opportunamente suddiviso in più moduli.

- Cosa dovrebbe fare, ossia quali procedure e calcoli speciali deve prevedere. Nel progetto preso ad esempio, questi elementi hanno forse scarsa rilevanza, ma in altre applicazioni la loro importanza è grandissima. Può darsi che esistano 20 o più vie diverse, ma altrettanto valide, per raggiungere uno stesso scopo, ma l'utente spesso si cristallizza su di un'unica soluzione: non adottando questa, egli sarà quasi certa-

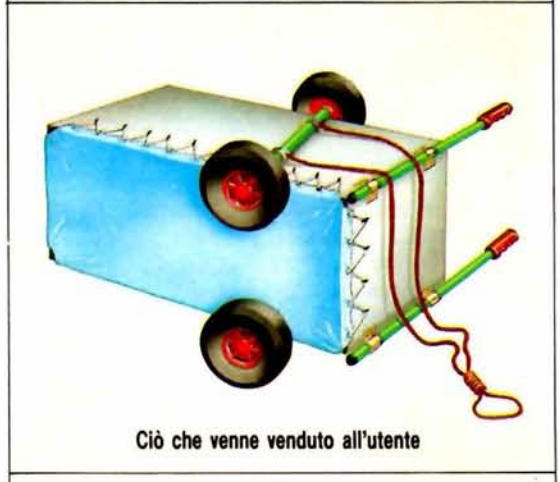
mente scontento del programma. Una soluzione diversa da quella preferita dall'utente, infatti, può forse tentare il programmatore, che la ritiene più elegante o più efficiente, ma tali apprezzabili vantaggi possono venir giudicati in modo negativo e far cadere definitivamente l'interesse dell'utente.
Per quanto riguarda eventuali procedure di calcolo, non è forse più semplice chiedere direttamente all'utente quali formule applichi per calcolare la distribuzione dei pesi sulle ali, ad esempio, piuttosto che reinventarsele da soli?
Dopo essersi annotata tutta questa serie di informazioni, può iniziare la conversione delle specifiche in un programma. È preferibile, comunque, iniziare col progetto del "dialogo" tra utente e computer, passando poi al formato ed alla gestione dei file ed infine, al metodo per controllare il tutto. Il termine "dialogo" non è stato scelto a caso, poiché si tratta, in effetti, di uno scambio di informazioni tra l'operatore ed il sistema che non include soltanto l'immissione del nome del modello d'aereo, ad esempio, ma tutta la serie di messaggi ed informazioni che possono apparire sullo schermo durante l'esecuzione del programma. Inoltre, va fissato uno "stile" nella presentazione e nella richiesta delle informazioni: si può scegliere un metodo impostato su "Menu", oppure sulla semplice immissione di comandi specifici. Terminata questa fase di progetto, si avrà una chiara idea sull'ammontare di memoria impegnato dai vari messaggi, spiegazioni, ecc. e si può passare a definire le caratteristiche dei file.
Nel caso preso ad esempio, dove le informazioni sono costituite da grossi blocchi di testo, può convenire una suddivisione dell'archivio su più nastri o, se si ritiene che ciò valga la pena, si può ricorrere a qualche sistema di compressione del testo, con relative routine di codifica e decodifica.
A questo punto, la struttura globale del programma dovrebbe già essere evidente: esisteranno routine per l'immissione, per la ricerca, per la variazione dei dati, e per la gestione dell'errore.
Se, eseguita una verifica generale sul progetto, tutto sembra a posto, si può passare alla codifica del programma. Tuttavia, ("tra il dire e il fare...") il lavoro non è finito, perché il programma andrà verificato in ogni suo modulo, soprattutto controllandone il comportamento di fronte ad errate immissioni (anche le più improbabili) da parte dell'utente. Le beghe non mancheranno di certo!



Ciò che occorreva all'utente, secondo l'analista



Ciò che intendeva l'analista, secondo il programmatore



Ciò che venne venduto all'utente



Ciò che l'utente desiderava in primo luogo

STEVE CROSS

Descrivere, definire e progettare

Uno dei maggiori problemi, nello sviluppo del software applicativo, è quello dovuto ad una imperfetta comunicazione tra i vari individui coinvolti: l'utente, l'analista ed il programmatore. Nell'illustrazione qui a fianco, si è cercato di sintetizzare quali conseguenze possano avere dei malintesi non chiariti fin dal primo momento

Atic Atac

Una felice combinazione tra il genere "adventure" e quello "arcade"

Atic Atac è un rarissimo esempio di come si possa efficacemente mescolare la rapidità d'azione di un gioco "arcade" con la complessità di un gioco d'avventura. Come ogni buon classico di quest'ultimo genere, il gioco si svolge attraverso le stanze di un castello stregato. Il giocatore è prigioniero del castello, ma può fuggire se trova la chiave d'oro, che apre uno dei portoni principali. La vita del protagonista è messa in pericolo da ogni sorta di creature infernali (ragni, streghe, diavoli e voraci mostri), per non parlare degli eterni favoriti: il Conte Dracula, Frankenstein e la Mummia. Abbondano, naturalmente, passaggi segreti e trappole micidiali, ma anche utili oggetti da raccogliere e portare con sé.

Fin qui, l'*Atic Atac* può dare l'impressione di essere un qualsiasi gioco d'avventura, ma la grafica adoperata nell'animazione è veramente eccezionale ed aggiunge al gioco quegli elementi tipici dei giochi "arcade". Di ogni stanza, o caverna, viene presentata una particolareggiata prospettiva dall'alto. Si possono vedere le porte che conducono nelle varie direzioni (nord, sud, est e ovest) ed altre decorazioni: insegne, scaffali, quadri, vecchi orologi a pendolo e via dicendo.

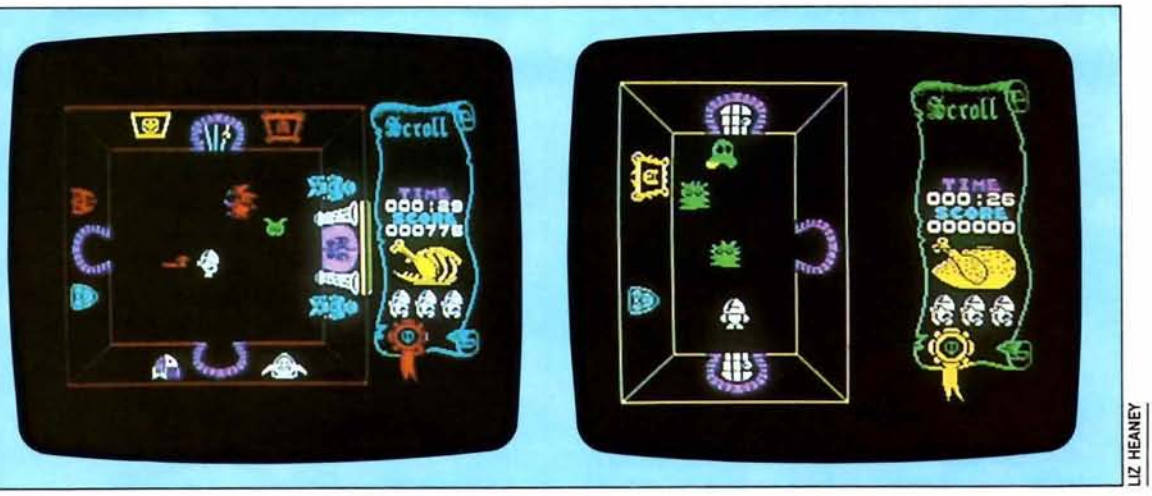
Il giocatore, che può scegliere se impersonare un Cavaliere, un Mago o un Servo, si muove da un ambiente all'altro, armato come conviene al personaggio prescelto. Di continuo appaiono mostri, preannunciati da sbuffi di fumo: ogni loro "tocco" indebolisce seriamente il protagonista, che, tuttavia, può difendersi da loro con le armi a disposizione. Il movimento di alcuni mostri è puramente casuale, ma quello di altri è decisamente più antisociale: puntano dritti dritti verso il protagonista!

Per gli spostamenti, il giocatore usa la tastiera, ma è previsto anche l'impiego di joystick. Nel primo caso, però, la cosa è resa difficoltosa dal fatto che i tasti relativi al movimento (Q, W, E, ed R) si trovano tutti sulla stessa fila. Sarebbe stato più logico abbinare agli spostamenti nord/sud, ad esempio, i tasti Q ed A, mentre a quelli est/ovest i tasti P e L.

Lo svolgimento del gioco si basa molto sull'impiego di colori: non basta una semplice chiave per aprire una porta, ci vuole quella del colore appropriato. Oltre a cercare l'uscita, esiste anche un obiettivo secondario nel gioco, ma non per questo meno importante: la ricerca di viveri. Sulla parte destra dello schermo è visualizzato un pollo: esso indica, indirettamente, la nostra vitalità, poiché se non raccogliamo via via del cibo, il pollo si trasformerà ben presto in un mucchietto d'ossi, a significare che siamo morti di fame! Inizialmente, il gioco può lasciare vagamente indifferenti, ma imparando a combattere i mostri e ad aggirarsi nel castello, si apprezzerà sempre più l'aspetto "avventuroso" del gioco, e si troverà che esso avvince sempre più, fino a farci trascorrere svariate ore di divertimento.

LA GRAFICA

Dalle riproduzioni presentate qui a fianco, ci si rende conto dell'accuratezza usata nella grafica di questo divertente programma. I colori giocano un ruolo molto importante, perché spesso la salvezza o il disastro completo dipendono unicamente dal possesso di un oggetto del colore giusto. Lo scenario è tipico dei giochi d'avventura, e il ritmo è vivace ed incalzante



LIZ HEANEY

Atic Atac: versione per Spectrum 48K
Editori: Ashby Computer e Graphics Ltd. (GB)
Autori: Ultimate, Play the Game
Accessori: è previsto l'uso di joystick
Formato: cassetta

Sprite sul BBC

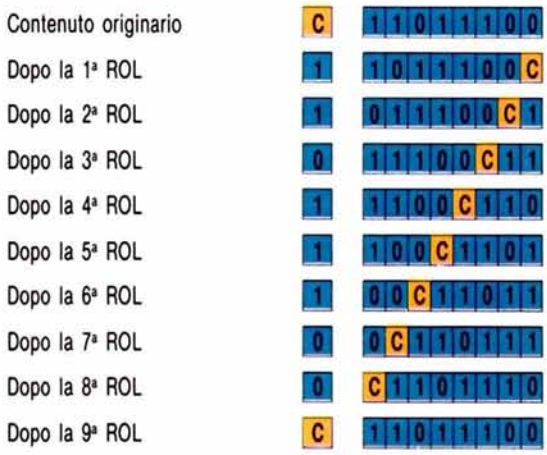
La creazione e l'uso degli sprite sul Micro BBC

Gli sprite possono avere diverse sagome e forme, ma tutti devono esser prima definiti su di una griglia. Non vi sono restrizioni particolari sulle dimensioni della griglia, benché convenga sempre stabilirne la larghezza in un numero pari di byte (ad esempio: 8, 16, 24, ecc.) e l'altezza in modo che lo spazio di lavoro risulti quadrato o leggermente rettangolare. La routine qui presentata prevede una griglia larga 24 pixel ed alta 21. Pertanto, saranno necessari 63 byte per contenere le informazioni sulla sagoma dello sprite: ad ogni pixel da "illuminare" corrisponderà un bit posto ad 1, se invece un pixel va "spento" il bit corrispondente varrà 0. Una volta definiti i valori di tutti i byte, questi vanno depositati, in forma decimale o esadecimale (hex) in una zona di memoria opportunamente protetta. Qui a fianco è riportata la definizione dello sprite usato come esempio, con tutti i relativi valori.

I 63 numeri, che definiscono la forma dello sprite, possono venir immessi in frasi DATA e successivamente letti e depositati in memoria da un breve programma in BASIC. La zona di memoria RAM, ove depositare questi valori, può esser scelta a piacere, purché non interferisca con altre parti del programma e venga adeguatamente "protetta" da sovrascritture. Il posto più ovvio da scegliere, comunque, è in cima all'area riservata al BASIC. Siccome l'indirizzo finale di tale zona può essere modificato tramite la variabile MIMEM, possiamo "accorciare" l'area disponibile al BASIC, riservandoci così una piccola fetta di memoria "protetta": le linee 220 e 230 provvedono proprio a questo, fissando, inoltre, l'indirizzo iniziale dei dati relativi allo sprite (SPRDAT) subito sopra il nuovo valore di HIMEM. Le linee da 1740 a 1770 leggono e depositano i valori nella zona RAM così preparata.

La routine in codice macchina

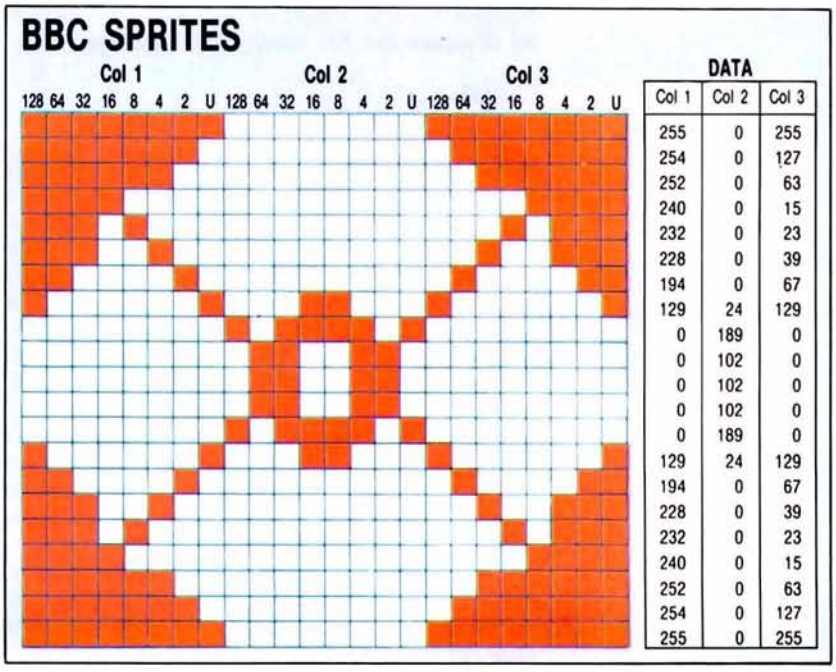
La funzione principale del modulo in codice macchina è quella di analizzare i dati che definiscono lo sprite, e di espletare le operazioni necessarie alla sua visualizzazione. Per far ciò, occorre che la routine esamini, un bit alla volta, tutti i 63 byte della definizione, per decidere se illuminare o meno il corrispondente pixel. Il modo più semplice per eseguire tale controllo, è ricorrere ad una delle istruzioni di Rotazione. La linea 1100 del codice sorgente usa l'istruzione ROL (rotazione a sinistra) che provoca uno scorrimento a sinistra, di un bit, del contenuto di un byte. Nel flag di Carry (riporto) finisce il bit più significativo del byte (quello più a sinistra) ed il precedente contenuto del flag di Carry viene depositato come bit meno significativo nel byte. Con un ripetuto impiego dell'istruzione ROL si può esaminare ciascun bit dell'intero byte, se l'operazione viene ripetuta, senza modificare il contenuto del Carry, e al nono passaggio si riottiene il byte originario.



Da questo esame, si può notare come il contenuto originario del flag di Carry (C) non ci interessi particolarmente, in quanto viene gradualmente ruotato attraverso il byte per

Definizione dello sprite

Lo sprite misura 24 x 21 pixel, cosicché occorrono 63 byte per la sua definizione (un bit per pixel). Il colore dello sprite è determinato dalla variabile logcol, mentre le dimensioni dello sprite sono fissate da due fattori di scala: XSCALE e YSCALE. Per queste ultime due variabili si dovrebbero sempre usare dei numeri pari



DATA

Col 1	Col 2	Col 3
255	0	255
254	0	127
252	0	63
240	0	15
232	0	23
228	0	39
194	0	67
129	24	129
0	189	0
0	102	0
0	102	0
0	102	0
0	189	0
129	24	129
194	0	67
228	0	39
232	0	23
240	0	15
252	0	63
254	0	127
255	0	255

LIZ DIXON

tornare, alla fine, nuovamente nel flag di Carry. Si noti, inoltre, come la ROL sia usata, nella linea 1100, con indirizzamento indicizzato assoluto, cosicché ciascuno dei 63 byte, a turno, viene esaminato bit per bit.
Una volta stabilito se un particolare bit ha valore 0 o 1, va operata l'accensione o meno del pixel corrispondente sullo schermo. Sul BBC esistono due maniere per far ciò. La prima consiste nell'eseguire direttamente una POKE del valore nell'area video. Tuttavia, un simile metodo è insicuro, sia perché l'ampiezza dell'area video dipende dal MODE grafico usato, sia perché la relazione matematica che lega ciascun bit ai pixel dello schermo è piuttosto complessa. Nel MODE 2, ad esempio, ciascun byte di memoria controlla soltanto due pixel. Inoltre, poiché i colori disponibili sono 16, occorrono 4 bit per definire il colore di ogni pixel. Nel MODE 0, che invece dispone di due soli colori, basta un solo bit per la definizione di ciascun pixel. Pertanto, se progettassimo una routine adatta al MODE 2, non la potremmo poi applicare al MODE 0.
L'alternativa consiste nell'adoperare il sistema operativo del BBC per compiere le necessarie conversioni e manipolazioni, rendendo così la routine indipendente dal particolare MODE usato. Il funzionamento della routine assomiglia a quello del comando VDU del BASIC. Se, ad esempio, desideriamo "accendere" un singolo punto sullo schermo, si userebbe il seguente comando:

```
VDU25,300;700;
```

In questo caso, le coordinate x e y corrispondono, rispettivamente, ai valori 300 e 700. L'equivalente in codice macchina del comando VDU consiste nel ripetuto uso di una chiamata al sistema operativo, la OSWRCH, previo deposito di opportuni valori nell'accumulatore. Poiché quest'ultimo può contenere soltanto un byte alla volta, le coordinate x e y devono essere suddivise nel formato LO/HI ed il comando VDU risultante sarebbe:

```
VDU25,68,44,1,188,2
```

Per eseguire tale comando in codice macchina, occorre "chiamare" OSWRCH ben sei volte. Il vettore contenente l'indirizzo di OSWRCH si trova nella locazione &FFEE, quindi alla routine si accede con JSR &FFEE. Qualsiasi comando VDU può essere riprodotto tramite il ripetuto impiego di OSWRCH, come si può osservare nel listato della routine. Si noti che la OSWRCH, pur non alterando i valori dell'accumulatore e dei registri X e Y, modifica il flag del Carry. Ciò significa che, quando il contenuto del Carry è significativo, come lo è nella nostra routine di ROL, esso va memorizzato prima di usare la chiamata a OSWRCH, e recuperato subito dopo. Il modo più semplice per far questo consiste nel depositare il registro di condizione del processore (status register) sullo stack, con un'istruzione PHP, per recuperarlo poi con una PLP.

Esaminiamo adesso la struttura del programma in codice macchina. La routine che si occupa dell'analisi dello sprite e della sua visualizzazione, è la SPRPLT, che inizia alla linea 890 del listato: la prima operazione eseguita è quella di fissare il metodo impiegato nella visualizzazione, ossia tramite un OR esclusivo. Questo sistema corrisponde al comando GCOL del BASIC: i punti tracciati mediante questo sistema vengono cancellati semplicemente ed automaticamente, ad ogni nuovo tracciamento, non interferendo, al tempo stesso, con le immagini già presenti sullo schermo. All'inizio del codice macchina, viene eseguito un posizionamento sull'angolo superiore sinistro dello sprite, dopodiché ogni riga di esso viene analizzata, esaminando tre byte alla volta, mediante le rotazioni prima descritte.
Ogni spostamento relativo, con o senza tracciamento di punti, viene fatto in base ad un fattore di scala orizzontale, XSCALE, determinato dal valore del bit contenuto nel flag di Carry. Terminato il lavoro su una riga di tre byte, la routine compie uno spostamento assoluto nella posizione individuata dalla medesima coordinata x dell'angolo superiore sinistro dello sprite, ma ad un'altezza y, determinata dal fattore di scala verticale YSCALE. Questo processo è ripetuto per tutti i 63 byte che formano lo sprite.
La routine SPRPLT è usata due volte: prima per cancellare lo sprite "precedente", poi per tracciare quello nuovo. Fatto ciò, le coordinate del nuovo sprite vengono depositate in OLDX e OLDY, ad uso della successiva chiamata alla routine.

COME USARE LA ROUTINE IN CODICE MACCHINA IN BASIC

La routine può esser facilmente adoperata in BASIC, anche se il programmatore non desidera addentrarsi nel significato delle varie istruzioni. I passi necessari sono:

1. Disegnare e definire lo sprite desiderato, depositandone i valori risultanti in una zona di memoria, come mostrato nel programma
2. Fissare il MODE grafico voluto
3. Fissare i valori di XSCALE e YSCALE, come illustrato nella linea 1870
4. Determinare il colore "logico" per lo sprite, come mostrato nella linea 1890
5. Stabilire le coordinate x e y della posizione nella quale far apparire lo sprite ed usare la procedura esposta alle linee da 2010 a 2060 per convertire le coordinate assolute nel formato LO/HI
6. Eseguire la routine, tramite una CALL SPRITE

La routine in codice macchina qui presentata, può essere facilmente incorporata in altri programmi BASIC. La visualizzazione del listato può essere soppressa, modificando la linea 260 in:

```
FOR opt% = 0TO2STEP2
```

In alternativa, la routine può essere conservata su nastro (una volta assemblata, ossia dopo l'esecuzione di un RUN), ricorrendo al comando *SAVE, nel quale andranno indicati gli indirizzi iniziale e finale del codice, visualizzati in fase di assemblaggio

SPRITE SUL BBC

```
10 REM ** Sprite sul BBC **
20 :
30 REM ** Preparazione variabili in pagina zero **
40 TYPE   =&70
50 OLDXLO =&71:?OLDXLO=0
60 OLDXHI =&72:?OLDXHI=0
70 OLDYLO =&73:?OLDYLO=0
80 OLDYHI =&74:?OLDYHI=0
90 NEWXLO =&75
100 NEWXHI =&76
110 NEWYLO =&77
120 NEWYHI =&78
130 XSCALE =&79
140 YSCALE =&7A
150 logcol =&7B
160 ROW    =&7C
170 YTMPLO =&7D
180 YTMPHI =&7E
190 XTMPLO =&7F
200 XTMPHI =&80
210 :
220 HIMEM=HIMEM-150
230 SPRDAT =HIMEM+1
240 OSWRCH =&FFEE
250 DIM MCX &01FF
260 FOR op tX =0TO 3 STEP 3
270   PX =MCX
280   [
290   OPT op tX
300    \ ** SPOST. VECCHIA POSIZIONE X,Y **
310    \
320    .SPRITE  LDA #25
330             JSR OSWRCH
340             LDA #48
350             JSR OSWRCH
360             LDA OLDXLO
370             STA XTMPLO
380             JSR OSWRCH
390             LDA OLDXHI
400             STA XTMPHI
410             JSR OSWRCH
420             LDA OLDYLO
430             STA YTMPLO
440             JSR OSWRCH
450             LDA OLDYHI
460             STA YTMPHI
470             JSR OSWRCH
480    \
490    \ ** CANCELLA VECCHIO SPRITE **
500    \
510             JSR SPRPLOT
520    \
530    \ ** SPOST. NUOVA POSIZIONE X,Y **
540    .NEWMOV   LDA #25
550             JSR OSWRCH
560             LDA #48
570             JSR OSWRCH
580             LDA NEWXLO
590             STA XTMPLO
600             JSR OSWRCH
610             LDA NEWXHI
620             STA XTMPHI
630             JSR OSWRCH
640             LDA NEWYLO
650             STA YTMPLO
660             JSR OSWRCH
670             LDA NEWYHI
680             STA YTMPHI
690             JSR OSWRCH
700    \
710    \ ** TRACCIA NUOVO SPRITE **
720    \
730             JSR SPRPLOT
740    \
750    \ ** MEMORIZZA X,Y COME 'VECCHIA POSIZIONE' **
760             LDA NEWXLO
770             STA OLDXLO
780             LDA NEWXHI
790             STA OLDXHI
800             LDA NEWYLO
810             STA OLDYLO
820             LDA NEWYHI
830             STA OLDYHI
840    \
850    \ ** RITORNO AL BASIC **
860    \
870           RTS
880    \
890    \ ** SUBROUTINE DI TRACCIAMENTO DELLO SPRITE **
900    \
910    \ ** ESEGUE UN OR ESCLUSIVO **
920    .SPRPLOT LDA #18
930             JSR OSWRCH
940             LDA #3
950             JSR OSWRCH
960             LDA logcol
970             JSR OSWRCH
980    \
990    \
1000   \ ** INIZIALIZZA I CONTEGGI          **
1010   \ ** X CONTA I BYTE, Y CONTA I BIT   **
1020   \ ** ROW CONTA I GRUPPI DI TRE BYTE **
1030            LDX #&00
1040   .NEWROW  LDA #&00
1050            STA ROW
1060   \
1070   .BYTE    LDY #&09
1080   .BIT     LDA #&5
1090            STA TYPE
1100            ROL SPRDAT,X
1110            PHP          \ SALVA IL CARRY SULLO STACK
1120            BCS DOPLOT
1130            LDA #64
1140            STA TYPE
1150   \ ** COMANDO VDU DI TRACCIAMENTO **
1160   .DOPLOT  LDA #25
1170            JSR OSWRCH
1180            LDA TYPE
1190            JSR OSWRCH
1200            LDA XSCALE
1210            JSR OSWRCH
1220            LDA #&00
1230            JSR OSWRCH
1240            JSR OSWRCH
1250            JSR OSWRCH
1260   \ ** FINE DEL COMANDO DI TRACCIAMENTO **
1270            PLP          \ RECUPERA IL FLAG DI CARRY
1280            DEY
1290            BNE BIT
1300   \ ** SE IL BYTE E' FINIT **
1310            INX
1320            CPX #63
1330            BEQ FINISH
1340   \ ** VERIFICA SE RIGA FINITA **
1350            INC ROW
1360            LDA ROW
1370            CMP #3
1380            BNE BYTE
1390   \ ** SE FINE DI RIGA, SOTTRAE YSCALE DA Y **
1400   \
1410            LDA YTMPLO
1420            SEC
1430            SBC YSCALE
1440            STA YTMPLO
1450            BCS NOSUB
1460            DEC YTMPHI
1470   \ ** SPOSTAMENTO ASSOLUTO A INIZIO NUOVA RIGA **
1480   \
1490   .NOSUB   LDA #25
1500            JSR OSWRCH
1510            LDA #48
1520            JSR OSWRCH
1530            LDA XTMPLO
1540            JSR OSWRCH
1550            LDA XTMPHI
1560            JSR OSWRCH
1570            LDA YTMPLO
1580            JSR OSWRCH
1590            LDA YTMPHI
1600            JSR OSWRCH
1610   \ *
1620   \ ** RIGA SUCCESSIVA **
1630            JMP NEWROW
1640   \
1650   \ ** FINE DELLA SUBROUTINE **
1660   .FINISH RTS
1670   \
1680 ]
1690 NEXT
1700 :
1710 REM **** QUI INIZIA IL PROGRAMMA IN BASIC ****
1720 :
1730 REM ** LETTURA DATI DELLO SPRITE **
1740 FOR indir = SPRDAT TO SPRDAT + &2
1750   READ data:?indir = data
1760 NEXT indir
1770 :
1780 REM ** PREP. PARAMETRI PER C.M. **
1790 :
1800 MODE 1
1810 GCOL0,129
1820 CLS
1840 ?XSCALE  =4:?YSCALE  =4
1870 ?logcol=1
1880 :
1890 X=700:Y=800
1900 PROCCOORDS (X,Y)
1910 CALL SPRITE
1920 :
1930 REM ** ATTESA PRESSIONE TASTI CURSORE **
1940 FOR S=0 TO 1 STEP 0
1950 PROCKEYS
1960 PROCCOORDS (X,Y)
1970 CALL SPRITE
1980 NEXT S
1990 END
2000 :
2010 DEF PROCCOORDS (X,Y)
2020 XH=X DIV 256:XL=X MOD 256
2030 YH=Y DIV 256:YL=Y MOD 256
2040 ?NEWXLO=XL:?NEWXHI=XH
2050 ?NEWYLO=YL:?NEWYHI=YH
2060 ENDPROC
2070 REM ** SCANSIONE TASTIERA **
2080 DEF PROCKEYS
2090 LOCAL LT,ZZ:LT=2
2100 FOR ZZ=0 TO 1 STEP 0
2110 IF INKEY(-58) THEN Y=Y+30:ZZ=LT
2120 IF INKEY(-42) THEN Y=Y-30:ZZ=LT
2130 IF INKEY(-26) THEN X=X-30:ZZ=LT
2140 IF INKEY(-122) THEN X=X+30:ZZ=LT
2150 NEXT ZZ
2160 ENDPROC
2170 REM ** DATI CHE DEFINISCONO LO SPRITE **
2180 DATA 255,0,255,254,0,127,252,0,63
2190 DATA 240,0,15,232,0,23,228,0,39
2200 DATA 194,0,67,129,24,129,0,189,0
2210 DATA 0,102,0
2220 DATA 0,102,0
2230 DATA 0,102,0
2240 DATA 0,189,0,129,24,129,194,0,67
2250 DATA 228,0,39,232,0,23,240,0,15
2260 DATA 252,0,63,254,0,127,255,0,255
```

Muoviti, sprite!

Il movimento dello sprite, la cui forma è definita dai valori contenuti nelle frasi DATA, si ottiene agendo sui tasti del cursore. Gli spostamenti sono relativamente lenti (ciò deriva dall'uso della routine OSWRCH, conservata in ROM), ma in compenso la routine è adatta a qualsiasi MODE grafico selezionato.

Quando si impartisce un RUN, sullo schermo scorre velocemente il listato in Assembly del codice macchina, seguito immediatamente dalla comparsa dello sprite

LA LLAMA SOFT

Jeff Minter è l'animatore di questa software house di enorme successo

Molte delle software house dedicate alla produzione di giochi per computer iniziano la propria attività basandosi sul lavoro di uno o due programmatori "casalinghi". Talvolta, però, il caos inevitabilmente suscitato dai primi successi comporta l'assunzione di altri programmatori e, nella foga di produrre nuovo software, lo "stile" e le caratteristiche iniziali, che avevano determinato il successo, vengono dimenticate, portando così la produzione allo stesso livello di quella di tante altre software house. Comunque, ciò non è accaduto alla Llamasoft: i due famosissimi giochi *Mutant Camels* e *Sheep in Space* sono talmente caratteristici, da renderne inconfondibile la fonte.

Il creatore del tipico stile della Llamasoft è Jeff Minter, un giovane che, a soli 22 anni, è diventato uno dei "divi" nel mondo del software. È la sua ossessione per particolari animali (cammelli, lama, ecc.) ad aver inciso profondamente sull'immagine della Llamasoft. Nella pubblicità dei propri prodotti il termine "awesome" (terrificante, mostruoso) è onnipresente.

La fama e la notorietà di Minter sono tali, che, nel gioco *Manic Miner* (prodotto da una software house concorrente), il giocatore deve affrontare un "Attacco dei telefoni mutanti", con evidente ironia sulle ossessioni di Minter.

Jeff Minter iniziò a scrivere giochi per computer nel 1981, subito dopo aver lasciato l'università (studiava matematica e fisica): nonostante gli ottimi risultati nelle materie informatiche, egli non superò l'esame di matematica!

La sua prima operazione commerciale fu la stesura di una versione di *Defender* per il Vic-20, e la creazione, nel 1982, della Llamasoft per commercializzare questo prodotto. Seguirono ben presto altri due giochi, *Traxx* e *Gridrunner*, che riscossero lo stesso successo: *Gridrunner*, in particolare, ebbe grandissima diffusione anche negli Stati Uniti.

Da allora, Jeff Minter ha continuato a scrivere programmi per il Vic-20, per il Commodore 64 e per gli Atari, rifiutandosi di dedicare i propri sforzi allo Spectrum, che egli considera un "orribile apparecchio". Di conseguenza, le versioni per lo Spectrum dei giochi prodotti dalla Llamasoft sono state preparate, su licenza, da altre software house, in particolare dalla Quicksilva e dalla Salamander Software.

Talvolta, la Llamasoft assume programmatori "ad hoc" per specifiche conversioni di programmi, ma il principale lavoro di programmazione resta sempre nelle abili mani di Minter, che ne controlla ogni fase dello sviluppo. Tutti i giochi sono frutto della fantasia di Minter, quindi egli non ritiene opportuno affidarsi ad altri programmatori.

La Llamasoft è un'impresa a conduzione familiare: il padre di Jeff, Patrick, aiuta nella programmazione, mentre la madre, Hazel, cura gli aspetti contabili ed amministrativi. Gli elementi "esterni" sono soltanto due: un contabile ed un "artista".

Minter ha prodotto più di una ventina di giochi per la Llamasoft, tutti più o meno conditi dall'ossessiva presenza di cammelli, a partire dall'ormai famosissimo *Attack of Mutant Camels*, seguito da *Revenge of Mutant Camels* (la vendetta dei cammelli mutanti).

La Llamasoft ha stipulato un contratto con la CBS, che cura la distribuzione dei giochi in Gran Bretagna (eccezione fatta per le versioni prodotte dalla Quicksilva e dalla Salamander Software), mentre negli Stati Uniti è la Human Engineering Software che si occupa di ciò.

I prodotti della Llamasoft sono richiestissimi, tanto che di ogni nuovo gioco ne vengono subito prodotte 10.000 copie: successive duplicazioni (sempre sul medesimo ordine di grandezza), dipendono poi dalle vendite.

Minter non prevede di modificare l'impostazione e lo stile della Llamasoft in futuro, e forse questa tenacia rappresenterà la chiave di un successo duraturo.

TONY LODGE



per concessione della "YOUR 64 AND VIC-20"

La vendetta dei cammelli mutanti

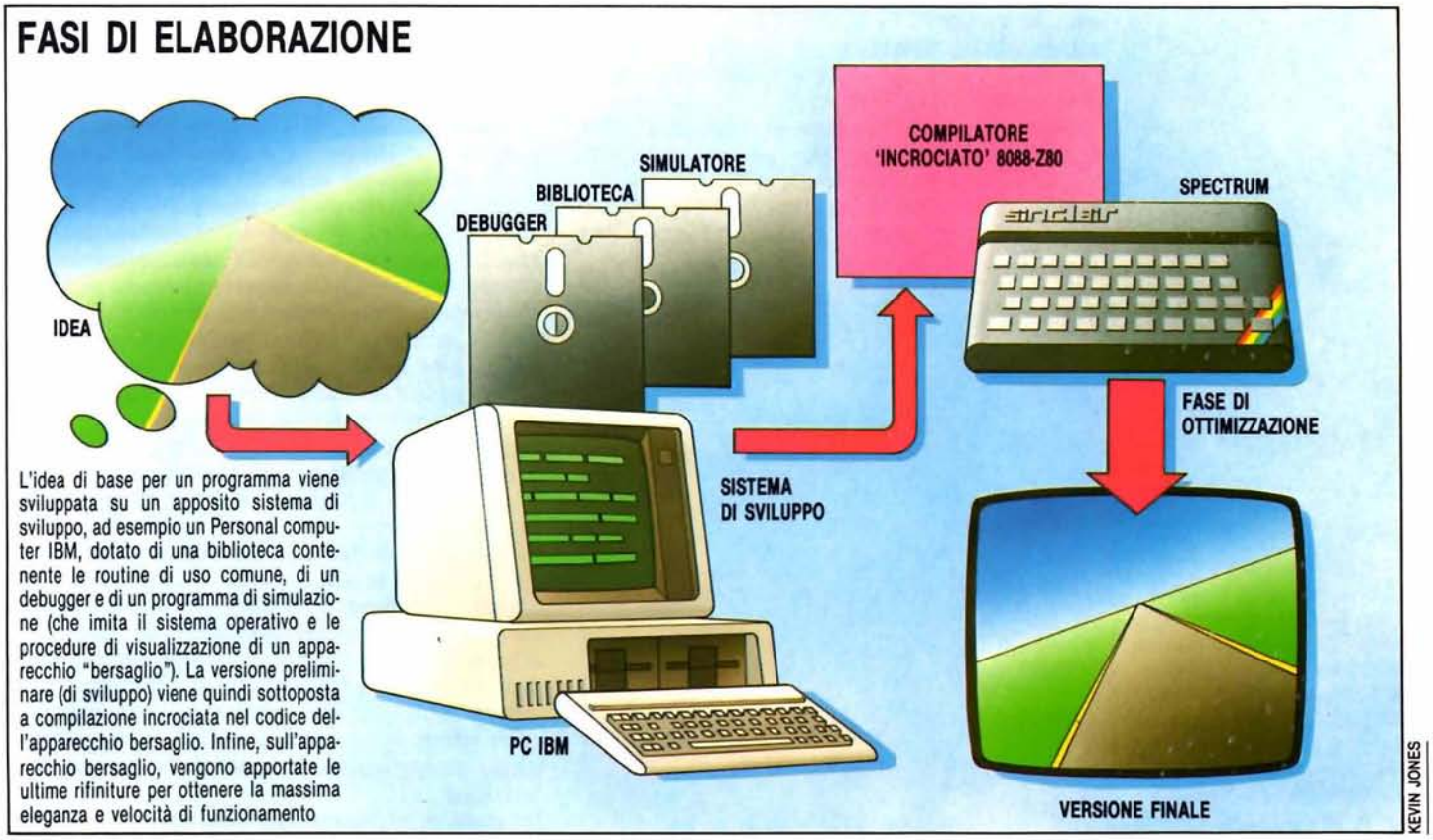
L'ossessivo stile surrealistico della Llamasoft è graficamente illustrato da questa scena, tratta dal gioco *Revenge of the Mutant Camels*, un ideale seguito del notissimo *Attack of Mutant Camels*

Programmare per vendere

Scrivere un buon software non basta per ottenere un successo commerciale

L'uso di sofisticate apparecchiature non significa necessariamente creare programmi di ottimo livello. Ad esempio, alcuni semplici appassionati di programmazione hanno accumulato una piccola fortuna, vendendo i propri programmi scritti su di uno Spectrum. Tuttavia, lo sviluppo delle grandi software house, verificatosi negli ultimi anni, rende sempre più angusti gli spazi della programmazione, per così dire, "domestica": i potenti computer e le sofisticate tecniche di programmazione di cui dispongono le software house permettono la realizzazione di programmi di gran lunga migliori di quelli fatti in casa.
Uno dei punti di forza del software commerciale per home computer è la velocità d'elaborazione: ciò implica l'impiego del codice macchina, almeno per alcune parti del programma. Purtroppo, la programmazione in codice macchina è piuttosto complessa e costringe i programmatori a servirsi di software complementare per la stesura dei propri programmi: come minimo occorre un programma assembler, che trasformi il codice sorgente in codice oggetto (ossia eseguibile dalla macchina) e questa operazione si dimostra piuttosto impegnativa se il programma da assemblare è di grosse dimensioni. Ad esempio, per scrivere un programma adatto allo Spectrum, molti programmatori usano un assembler che "gira" sullo stesso apparecchio, ma anche questo metodo di lavoro presenta alcuni limiti.
Innanzi tutto, la qualità dei programmi assembler prodotti per gli home computer è piuttosto bassa; inoltre, questi pacchetti occupano una discreta quantità di memoria ed impongono, quindi, forti limitazioni alle dimensioni del programma. Infine, molti micro sono del tutto inadeguati a lunghe sessioni di lavoro: possiedono tastiere e schermi scadenti e, in molti casi, sono privi delle utili unità a dischetti.
Per queste ragioni, nella fase di programmazione, molte software house non usano il micro a cui il programma è destinato (chiamato "apparecchio bersaglio"), ma si servono di elaboratori gestionali dotati di apposito software (noti come "sistemi di sviluppo"). Su tali apparecchi i programmatori spesso usa-

FASI DI ELABORAZIONE

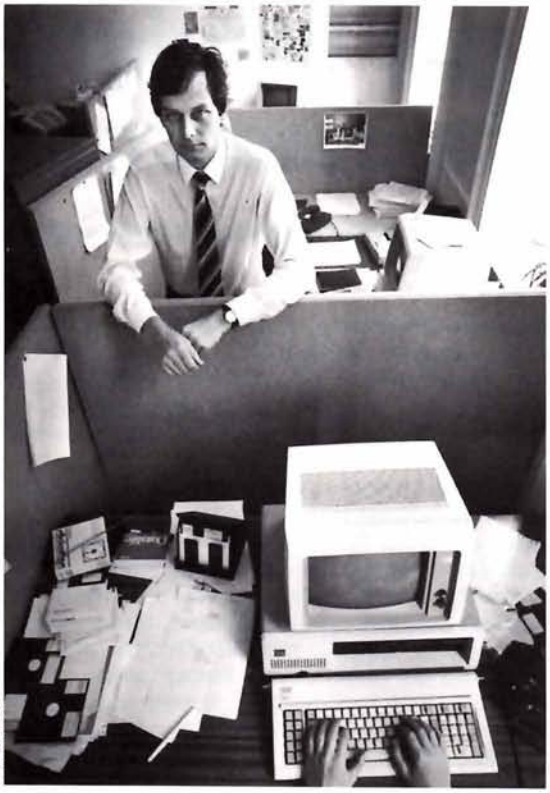


L'idea di base per un programma viene sviluppata su un apposito sistema di sviluppo, ad esempio un Personal computer IBM, dotato di una biblioteca contenente le routine di uso comune, di un debugger e di un programma di simulazione (che imita il sistema operativo e le procedure di visualizzazione di un apparecchio "bersaglio"). La versione preliminare (di sviluppo) viene quindi sottoposta a compilazione incrociata nel codice dell'apparecchio bersaglio. Infine, sull'apparecchio bersaglio, vengono apportate le ultime rifiniture per ottenere la massima eleganza e velocità di funzionamento

KEVIN JONES

La "Intelligent Software"

La IS è una software house specializzata in giochi di scacchi e strategici. All'inizio, per lo sviluppo del proprio software, si serviva di computer IBM ed Apple, dotati di apposite interfacce. Gli originali sistemi di sviluppo e le tecniche standardizzate di segmentazione hanno permesso alla IS di creare programmi per molti apparecchi e per tantissimi computer per scacchi



TONY SLEEP

no compilatori per linguaggi quali il PASCAL ed il C, oppure "cross assembler" che permettono, ad esempio, di lavorare su un micro basato sul processore 8086, generando però programmi utilizzabili su apparecchi con processori Z80. Questi compilatori "incrociati" sono linguaggi ad alto livello (come il BASIC) facili da usare, ma i programmi risultanti sono scritti in codice macchina. Infine, questi ultimi vengono esaminati e spesso ottimizzati da programmatori esperti in codice macchina.

Ovviamente, un sistema di sviluppo offre grandi vantaggi rispetto ad un semplice home computer: un assembler su disco o che si serve di una RAM espansa per memorizzare tabelle di grandi dimensioni, lavora con maggior efficienza rispetto ad un assembler registrato su nastro e che opera ai limiti consentiti da un comune micro. È inoltre possibile includere speciali routine per la correzione degli errori (debugging) nella versione di sviluppo del codice, sempreché quest'ultima non diventi troppo grande per la memoria a disposizione. Infine, gli apparecchi gestionali offrono il vantaggio di possedere tastiere e display di buona qualità e, magari, più unità a dischetti.

Una software house che ha fatto proprie queste tecniche di programmazione è la Intelligent' Software (IS), fondata nel 1981 da David Levy, un esperto di scacchi, e da Robert Madge, presidente della ANT Microware. La IS si è specializzata nella produzione di giochi di strategia per i micro di maggior successo e si occupa anche della parte software dei computer per scacchi. Come gran parte dei giochi di strategia, gli scacchi ed il bridge non presentano immagini a rapido movimento; è tuttavia necessario che le loro procedure di elaborazione possiedano la velocità d'esecuzione tipica dei programmi scritti con un assembler.

La IS, oltre a servirsi delle stesse macchine "bersaglio" per lo sviluppo dei programmi, usa apparecchi IBM PC ed Apple dotati di interfacce appositamente progettate per il trasferimento del codice da un computer all'altro. Questa ditta si occupa anche di progetti di conversione, come l'adattamento di un gioco di scacchi a computer di diverso tipo. Per questo motivo, i programmi sviluppati dai suoi programmatori devono risultare facilmente segmentabili. Una tecnica di segmentazione, che facilita in modo decisivo le operazioni di trasferimento di un programma da un processore ad un altro, è la suddivisione del programma in "codice di gioco" e "codice di input/output". Sull'apparecchio ricevente, il codice di I/O avrà differenti indirizzi di memoria e forse anche la strategia di gioco presenterà delle diversità (ad esempio, l'interrogazione ciclica sostituita da interruzioni o altro). Poiché il superamento dei condizionamenti imposti dall'hardware richiede un complesso intervento, non si dovrà eccedere nell'uso del codice di I/O. Al contrario, il codice di gioco potrà essere molto sviluppato perché, non essendo strettamente dipendente dall'hardware, la sua conversione non presenterà grossi problemi.

La IS ha ridotto al minimo le "regole interne" di programmazione a cui si devono attenere i programmatori nello scrivere un programma; inoltre un aspetto su cui insiste molto è la presenza, nelle routine del codice sorgente, di chiare linee di commento. Alcune software house si avvalgono di programmatori esterni che lavorano a casa propria: ciò, se da un lato esalta la programmazione individuale, dall'altro comporta uno spreco di tempo e di lavoro, poiché molto spesso accade che ciascun programmatore progetti e sviluppi per conto proprio alcune routine con la stessa funzione.

La casa di software Psion si serve di compu-



BOB BROMIDE

Visions

La maggior parte della programmazione, per il gioco *Visions*, è stata fatta da programmatori esterni, che lavoravano a casa propria. Una volta definita la logica di base e stabiliti gli scenari del gioco, le routine sono state scritte in linguaggio Assembly (Z80 oppure 6502), servendosi di assembler quali il DevPak della HiSoft, per lo Spectrum

ter molto più grandi dell'IBM PC ed è una tra le poche che producono programmi di gioco per home computer, sviluppando gran parte del lavoro direttamente su minicomputer. La Psion iniziò la propria attività, sviluppando software per lo ZX81, direttamente su questo apparecchio. Per produrre la cassetta *Horizons*, per lo Spectrum, si è servita però di un TRS-80, basato sullo stesso processore Z80, collegato ad uno Spectrum tramite una speciale interfaccia.

Nel 1982, la Psion ha deciso di abbandonare questa dispendiosa strategia produttiva ed ha acquistato una coppia di grossi computer Vax 750 con sistema operativo VMS sviluppato dalla DEC. Questi apparecchi forniscono capacità d'elaborazione superiori a quelle precedenti e sono sufficientemente flessibili per adattarsi a qualsiasi necessità di programmazione.

Questa scelta presenta due vantaggi: l'alta qualità del software, appositamente progettato dalla DEC, e le intrinseche potenzialità offerte dalla combinazione tra sistema operativo e hardware. Il sistema ha spazio sufficiente per accogliere una serie di programmi sussidiari (come compilatori, librerie delle subroutine più usate, programmi per l'individuazione degli errori) che, raccolti su di un solo apparecchio, possono essere utilizzati contemporaneamente da ben venti programmatori. Inoltre, i due apparecchi possono, all'occorrenza, scambiarsi il software immagazzinato.

I nuovi apparecchi Vax hanno permesso, dunque, un uso comune dei programmi sussidiari tanto ai programmatori impegnati in uno stesso progetto, quanto a quelli coinvolti in progetti diversificati: questo vantaggioso sistema di lavoro viene detto multiprogrammazione in "time sharing", ossia a spartizione di tempo.

Inoltre, i due Vax 750 possono gestire anche l'amministrazione della Psion, senza interferire col lavoro di programmazione.

Comunque, non si pensi che il semplice acquisto di simili computer significhi portarsi alla pari con la Psion: le operazioni di trasferimento dal vecchio al nuovo sistema, solo in parte realizzate dalla DEC, fornitrice dell'hardware, hanno richiesto alla Psion un notevole impegno di uomini e tempo.

La Psion usa il C, un linguaggio di livello intermedio, in grado di produrre codici oggetto discretamente compatti e veloci per i chip a 16 bit come l'8086 (benché ciò non sia altrettanto vero per i chip a 8 bit). Quindi, per la programmazione di macchine "bersaglio" quali lo Spectrum, la Psion ha ritenuto opportuno sviluppare un proprio compilatore 'ad hoc', del quale è molto gelosa: l'unico fatto certo è l'uso del C per scriverlo. Questo compilatore, che in mancanza di un nome specifico è stato battezzato dalla Psion "il nostro linguaggio di lavoro", ha molti punti di contatto col C, può essere usato su diversi processori e crea codici di estrema compattezza.

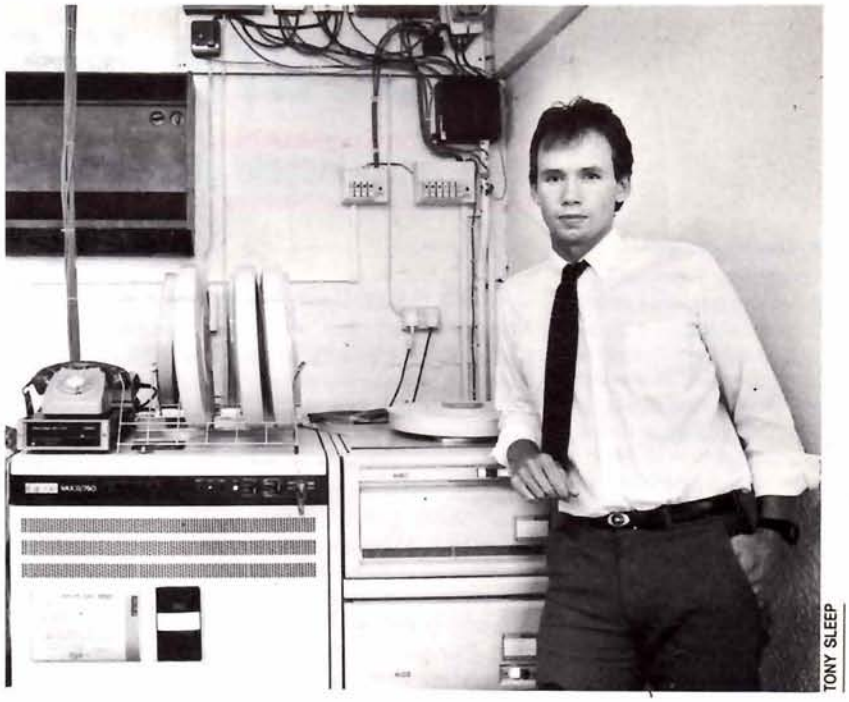
In genere, lo sviluppo in proprio di programmi sussidiari, come quello appena descritto, e di procedure di aggiornamento del sistema occupa circa il 30 per cento dell'intera attività di programmazione; la Psion tuttavia vi ha dedicato molto più tempo, poiché ritiene che il possesso di un proprio compilatore permetta ad una software house di eseguire esperimenti di frammentazione, adattabilità e correzione, del tutto impossibili col comune software in commercio.

Lo speciale software sviluppato dalla Psion comprende programmi che consentono di simulare perfettamente i più comuni processori, come lo Z80 ed il 6502: essi permettono ai grossi computer Vax di imitare il funzionamento del Commodore 64 o dello Spectrum. Le simulazioni sono progettate per funzionare ad una velocità molto minore rispetto a quella delle macchine bersaglio: ciò per dar modo al programmatore di esaminare, in qualsiasi fase del programma, il contenuto dei registri del microprocessore.

Questa tecnica si dimostra assai utile per l'individuazione degli errori e permette al programmatore di risparmiare moltissime ore di lavoro.

Negli ultimi tempi, molta dell'attività della Psion è stata dedicata all'implementazione dei quattro programmi gestionali standard, che corredano il QL della Sinclair. La famiglia dei chip Motorola 68000, uno dei quali viene impiegato sul QL, è stata progettata appositamente per linguaggi di alto livello: in effetti, la compilazione di programmi scritti in C è talmente efficiente su questi chip, che l'uso dell'assembler appare superfluo.

Se tutti gli home computer seguissero l'esempio del QL, il linguaggio C potrebbe sostituire del tutto l'assembler e le software house si libererebbero per sempre dell'interminabile lavoro di conversione semimanuale dei programmi.



TONY SLEEP

La Psion

Dal 1982, la Psion adopera una coppia di minicomputer Vax 750 come sistema di sviluppo per il proprio software. Fino ad un massimo di venti programmatori possono lavorare contemporaneamente su di un solo apparecchio, utilizzando i compilatori 'incrociati', le biblioteche di sistema ed i programmi di correzione degli errori, permettendo la creazione e la conversione dei programmi in più linguaggi

Giocare via modem

MUD, un colossale gioco d'avventura per più giocatori

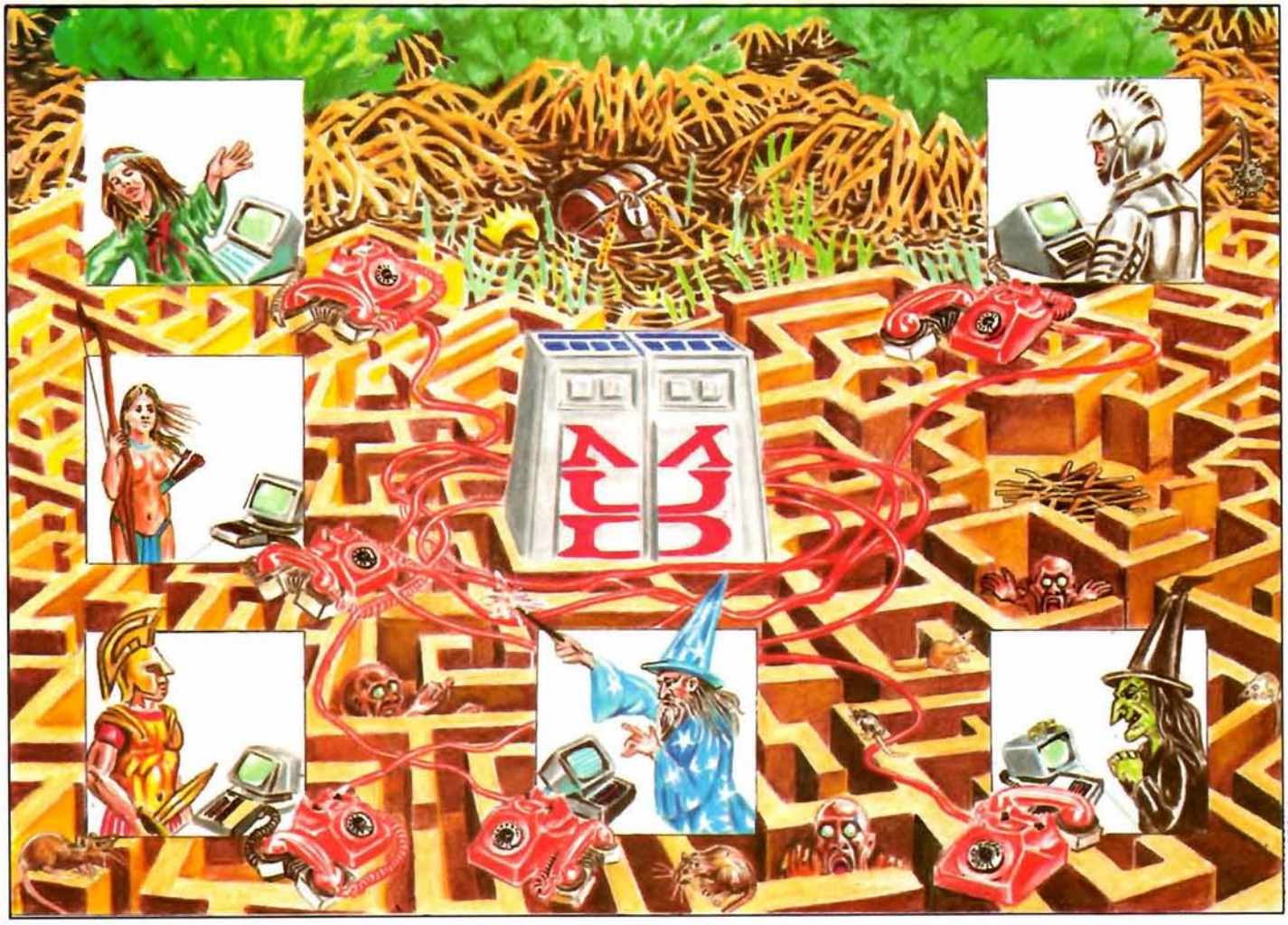
In "Lotta" contro MUD

I giochi stile "Adventure", che girano su grosse unità centrali, permettono l'accesso a più giocatori (il Multi-User Dungeon ad un massimo di 43 Novizi, Guerrieri, Eroi ecc.) che possono allearsi o combattersi a vicenda, per impadronirsi di un tesoro e trasformarsi in potentissimi Maghi o Streghe. Inoltre, questi giochi offrono ai partecipanti ambientazioni molto particolareggiate: il MUD propone la Terra, caverne di varie dimensioni, una foresta, un'isola dei draghi, il mare e la palude. Tutte queste zone, popolate da crudeli folletti e da zombie, possono nascondere un tesoro. Sei linee telefoniche permettono ai giocatori di collegare i propri micro all'unità centrale

Il MUD (Multi-User Dungeon) è un gioco in tempo reale, sul genere "Adventure", che permette a più giocatori di incontrarsi, scambiarsi messaggi e consigli, scontrarsi tra loro o allearsi per combattere un nemico comune. I giocatori possono uscire ed entrare nel gioco liberamente e, solo in questo secondo caso, le loro azioni interagiscono con quelle degli altri contendenti.

Il MUD viene giocato in Gran Bretagna (anche se nulla vieta, in via di principio, una partecipazione internazionale) e gira su un grande computer DEC10, della Essex University. Per partecipare al gioco basta possedere un micro con un programma che simuli le funzioni di un terminale, un telefono, un modem ed un codice di conto 'Packet Switching System' (PSS). Il programma simulatore, che può essere acquistato o creato individualmente, deve permettere al micro di comunicare con l'unità centrale attraverso la rete telefonica. Occorre, inoltre, che consenta lo "scroll" dello schermo ed una lunghezza di 80 caratteri per riga: il software buono per il Micronet, dunque, non è adatto a questo scopo perché privo di queste caratteristiche. L'uso di un micro con uno schermo in cui i caratteri per riga sono meno di 80, può causare qualche problema. Tuttavia, esistono in commercio alcuni eccellenti pacchetti: il Termi ed il Communicator, ad esempio, su chip ROM per il Micro BBC.

Il modem deve essere in grado di comunicare con il PSS della British Telecom ed avere una velocità di trasmissione di 300/300, 1200/75 o 1200/1200 baud: un accoppiatore acustico per il Micronet dà risultati discreti. Il PSS è un servizio di rete, che permette di mettersi in contatto con sistemi di elaborazione centrali, anche molto distanti, al costo (beati sudditi di



ADRIAN MORGAN

Sua Maestà!) di una semplice telefonata urbana. A questa spesa va però aggiunta quella della trasmissione di dati sulla rete PSS. La procedura di accesso all'unità centrale è semplice. Dopo aver avviato il programma di simulazione del terminale, il giocatore compone il numero telefonico della centralina PSS a cui fa capo, collega il modem e batte sulla tastiera il proprio nome e cognome. Quindi indica l'"indirizzo" PSS del DEC10 della Essex University. Uno speciale "codice di conto" consente il libero accesso al computer tutti i giorni, dalla mezzanotte alle 7 di mattina e, durante i fine settimana, dalle 22 alle 7, quando cioè l'unità centrale è meno impegnata. Infine, battendo questo codice di conto seguito dal comando RUN MUD, il giocatore entra nel gioco.
Il programma viene costantemente aggiornato e la prima cosa che appare sullo schermo del micro è la data dell'ultima versione. Il giocatore, a questo punto, digita il nome del personaggio attribuitogli, detto "Persona", ed informa il computer della sua presenza. Se è la prima volta che partecipa al gioco, il giocatore riceve una nuova Persona e viene classificato come "Novizio". I livelli di esperienza sono:

LIVELLO	PUNTI	MASCHILE	FEMMINILE
1	0	NOVIZIO	NOVIZIA
2	400	GUERRIERO	GUERRIERA
3	800	EROE	EROINA
4	1600	CAMPIONE	CAMPIONESSA
5	3200	SUPEREROE	SUPEREROINA
6	6400	INCANTATORE	INCANTATRICE
7	12800	STREGONE	STREGONA
8	25600	NEGROMANTE	NEGROMANTE
9	51200	LEGGENDARIO	LEGGENDARIA
10	102400	MAGO	STREGA

Se si esce dal gioco senza essere stati uccisi, il punteggio ottenuto verrà registrato dal programma ed il giocatore riprenderà il gioco dal punteggio ottenuto in precedenza. Si guadagnano punti, salendo di livello, quando ci si libera di oggetti che poi risultano inutili, si superano difficoltà inattese, si distruggono esseri ripugnanti come topi e zombie o si vince uno scontro con un altro giocatore. Se, invece, il personaggio abbinato al giocatore muore, egli viene eliminato dal gioco e deve ricominciare come Novizio, a punteggio 0. È quindi consigliabile cercare di evitare gli scontri diretti, soprattutto quando l'occasionale avversario sembra dotato di forza maggiore e di armi micidiali.
Il gioco comincia in un "angusto sentiero di campagna". Digitando "WHO", si ottiene una lista di tutti i partecipanti al gioco. Scrivendo un nome della lista seguito da una frase, ci si può rivolgere ad un particolare giocatore, ma è anche possibile inviare un messaggio a tutti i giocatori contemporaneamente.
Battendo "HELP" si otterranno alcuni consigli su come muoversi (ad esempio, N, SO, O, giù, salto ecc.) ed una succinta spiegazione su molti dei comandi disponibili, mentre digitando "COMMAND", compare un elenco di tutti i comandi. Alcune versioni del software simulatore del terminale consentono di trasferire su dischetto tutte le immagini che compaiono sul video per poterle esaminare successivamente. Questa possibilità permette di crearsi una propria mappa dello scenario di gioco e aggiornarla dopo ogni seduta. In una particolare fase del gioco i comandi disponibili possono essere i seguenti:

COMANDI (la forma abbreviata è in maiuscolo)

AutoWho, <secondi>
BRIEF
CONVERSE
EMPTY<borsa>
FOLLOW<nome>
Give<ogg.>TO<nome>
Help
HouRS
Inventory
KISS<nome>
<livello>,<messaggio>
Look<direzione>
NoPassWord
ProNouns
"<messaggio>"
RETaliate<oggetto>
SHout,<messaggio>
STeal<oggetto>From<nome>
tell<nome>,<messaggio>
VERBOSE
WHO

Back
BUG
DRop<oggetto>
EXITS
Get<oggetto>

HELP<nome>
HUG<nome>
KEEP<oggetto>
LEVEL
Look intorno
LOSE<nome>
PassWord
QuickWho
REFUSE<nome>
SAVE
SLEEP

WEIGH<oggetto>
WRITE<argomento>, <messaggio>

BERSERK
BYE
DRop ALL
Flee<direzione>
GetALL
go<direzione>
HINTS
INFO
Kill<nome>
LOG
Look<borsa>
MEDITATE
PERSONA
Quit

SCore
SPELL

UNKEEP
WHEN

MUD è un gioco basato su testi scritti, con descrizioni lunghe ed accurate dei vari ambienti. Una volta acquisita una buona conoscenza dello scenario, il comando "BRIEF" permette di saltare le parti puramente descrittive. Gli abbonati al Prestel, abituati alla lentezza ed ai limiti della grafica teletext, apprezzeranno molto i giochi stile "Adventure" dotati di grafica, ma i veri competenti di giochi "Adventure" sembrano sempre preferire le versioni basate su testi scritti, in quanto questi ultimi lasciano maggior spazio alla fantasia del giocatore. Un'ulteriore pecca dei giochi basati sulla grafica è che devono essere scritti in diverse versioni, per potersi adeguare alle caratteristiche grafiche dei vari tipi di micro. Chi gioca con il MUD è invece libero di usare una vasta gamma di apparecchi come l'Apple, lo Spectrum o il micro BBC, oppure vecchi terminali di seconda mano.
Il MUD tra breve comparirà anche sul mercato. Gli autori del programma originale, Richard Bartle e Roy Trubshaw, ne stanno scrivendo una versione per un VAX Computer che verrà commercializzata dalla Century Communication. Questa versione potrà essere giocata anche via cavo e su reti telefoniche come il Prestel e il PSS. I giocatori dovranno corrispondere una quota di collegamento quadrimestrale, oltre ad una modesta tariffa oraria.

Gli algoritmi

Cosa sono gli algoritmi e come servirsene nella programmazione

Un algoritmo non è che la traduzione, in termini logici o matematici, di un procedimento in funzione di altri procedimenti già noti o talmente fondamentali e chiari da non aver bisogno di essere precisati. In una ricetta culinaria, ad esempio, l'istruzione "preparare la salsa di pomodoro" implica un'operazione (algoritmo) già spiegata in un'altra parte del libro di ricette, mentre l'istruzione "portare l'acqua ad ebollizione" non richiede alcun chiarimento perché si dà per scontato che il "cuoco" sappia ciò che deve fare. Nell'ambito della programmazione, gli algoritmi sono costituiti o da istruzioni che si servono di algoritmi (procedure, routine, funzioni) scritti in un'altra parte del programma o da istruzioni che sono parte integrante del linguaggio (comandi come PRINT e DIM o funzioni matematiche come LOG e TAN). Qui esaminiamo la costruzione di alcuni algoritmi a partire da altri algoritmi e da operazioni "primitive", intese come comandi e funzioni del linguaggio usato. Tali primitive permettono di formulare algoritmi capaci di eseguire alcune semplici operazioni (ad esempio, muovere uno sprite oppure accettare un numero come input). Questi algoritmi di base vengono poi usati per costruirne altri di carattere più generale (aggiornare il punteggio di un gioco o controllare un sistema pilotato da menu). Infine, questi ultimi vanno a comporre algoritmi di dimensioni ancora maggiori. Quindi un programma completo, inteso come singolo algoritmo, può essere considerato un insieme di algoritmi di livello più basso concatenati tra loro. Questo concetto sta alla base della programmazione detta "strutturata" o "modulare", argomento di una fase successiva del corso.

Progettare gli algoritmi

Un algoritmo deve possedere un ingresso (input) e un'uscita (output), vale a dire che, come procedimento, esso elaborerà alcuni dati iniziali per produrre un risultato. I dati iniziali vengono forniti dall'esterno in forma di "parametri", che si mantengono costanti per una particolare operazione dell'algoritmo, ma che possono cambiare se riferiti ad operazioni differenti. La procedura di immissione dei parametri nell'algoritmo è estremamente semplice. Ad esempio, l'istruzione:

```
10 PRINT "Ciao Mondo!"
```

trasferisce il parametro "Ciao Mondo" all'algoritmo associato al comando PRINT. Altri esempi sono FNA(P), TAN(P), LEFT$(P$,5) e POKE P,5, dove P, P$ e 5 costituiscono i parametri. Analogamente, anche il risultato di un algoritmo viene "restituito" sotto forma di parametro. Se il linguaggio di programmazione usato possiede variabili locali (come il PASCAL ed il C), allora i parametri verranno probabilmente immessi con una chiamata "di procedura" del tipo:

```
procedura (parametro1, parametro2, ecc.);
```

Nel progetto di un algoritmo, occorre innanzi tutto tenere sotto controllo i contenuti dei parametri di input e output, la loro tipologia (interi, in virgola fluttuante, reali, stringhe, ecc.), le loro dimensioni ed il campo di variabilità.
Dopo aver determinato l'input e l'output di un algoritmo, è necessario riflettere sulle procedure che trasformeranno il dato in ingresso nel risultato dell'algoritmo. Sfortunatamente, non esiste alcun metodo precostituito per sviluppare tali procedure ed il programmatore deve affidarsi alle proprie capacità logiche ed alla propria fantasia.
Un metodo molto comune (e molto comodo) è quello di servirsi di algoritmi già esistenti. A livello più semplice, le funzioni incorporate in un linguaggio di programmazione forniscono molti utili algoritmi, quali la manipolazione delle stringhe, le funzioni trigonometriche, operazioni di input/output e, talvolta, il riordinamento e la manipolazione di matrici. Oltre a questi, si potranno inserire nel nuovo programma algoritmi già adoperati in altri programmi (per questo scopo è molto utile crearsi una "biblioteca" di algoritmi). Infine, una fonte inesauribile di algoritmi è la vasta gamma di riviste di informatica e di pubblicazioni specializzate. Particolarmente raccomandabile è il testo *The Art of Computer Programming, Vol. I: Fundamental Algorithms* di D.E. Knuth.
Infine, esistono algoritmi, sviluppati per attività totalmente diverse dalla programmazione, ma che possono dimostrarsi assai utili: i testi di contabilità o di ingegneria, ad esempio, contengono molte formule di calcolo il cui uso facilita la stesura di programmi gestionali, permettendo di ottenere risultati affidabili. Ciò vale anche per altre discipline come l'elettronica, la matematica ecc.
Che ci si serva o meno di un algoritmo già esistente, le istruzioni in esso contenute devono sempre attenersi a due criteri fondamentali: la "chiarezza" e l'"efficacia". Chiarezza significa che l'istruzione non deve mai essere ambigua, come spesso accade, invece, nelle prime fasi di analisi descrittiva dell'algoritmo, in cui si usano termini come "e" ed "o",

attribuendo loro un significato diverso da quello di AND e OR dell'algebra Booleana. Ad esempio, se un algoritmo deve individuare tutti i nomi di una lista che cominciano con A "e" tutti quelli che cominciano con B, scrivendo l'istruzione:

```
IF PRIMALETTERA = "A" AND PRIMALETTERA = "B" THEN
```

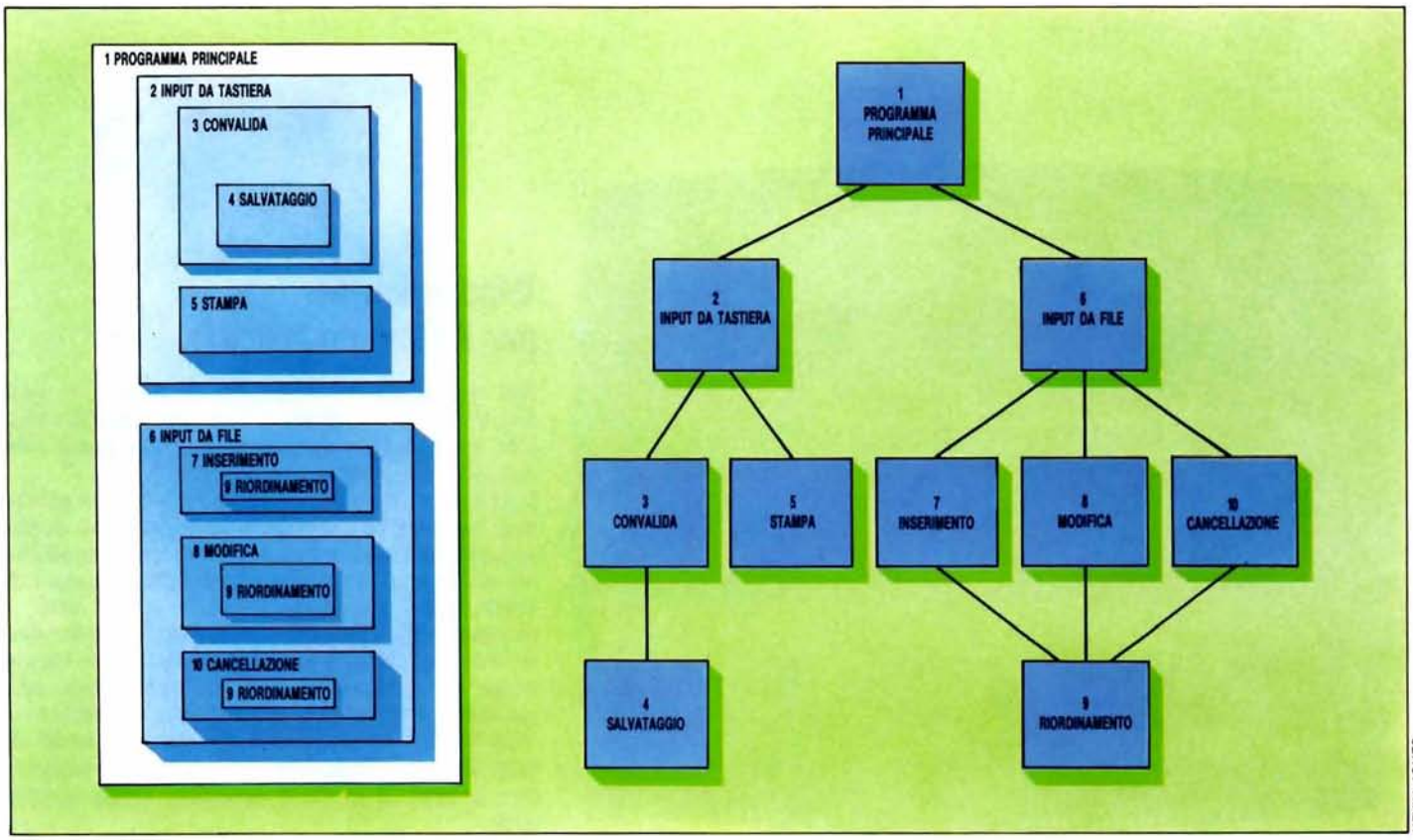
si commette un errore, poiché in questo caso occorre adoperare un OR logico!
Efficacia significa invece che il programma non deve contenere istruzioni "impossibili". Un'istruzione viene detta "efficace", quando può essere eseguita con carta e penna in un tempo finito, pertanto un algoritmo del tipo "LET X = il più alto numero primo", non è efficace (non esiste un numero primo più grande di tutti gli altri!).

Considerazioni generali

Un algoritmo, considerato nel suo insieme, deve rispettare anche altri criteri: ad esempio, deve "concludersi". L'algoritmo che segue non ha termine (anche se le sue istruzioni sono chiare ed efficaci) e, qualora inserito in un programma, darebbe vita ad un ciclo iterativo senza fine:

```
fase 1 poni K = 1
fase 2 se K>3 esci dall'algoritmo
fase 3 torna a fase 1
```

Non sempre è facile accorgersi se un algoritmo si conclude ma, in generale, gli algoritmi che contengono dei cicli, verificano una particolare condizione prima di concludersi (nel nostro esempio, se 1>3): occorre dunque accertarsi che la condizione sia effettivamente realizzabile.
"Efficienza, genericità ed eleganza" distinguono un algoritmo da un altro. L'efficienza viene giudicata in termini di tempo e di consumo di memoria. Questi due fattori sono spesso strettamente correlati: programmi veloci occupano di solito poca memoria anche se non sempre è così. Per esempio, un calcolo è molto più rapido e occupa meno memoria se si serve di numeri interi e non di decimali. La genericità indica la capacità di un algoritmo di adattarsi a situazioni differenti da quella per la quale è stato inizialmente progettato. L'obiettivo del programmatore deve essere quello di scrivere programmi il più possibile generici: se un programma prevede un frequente uso di immissioni del tipo SI/NO, conviene scrivere una routine che visualizzi il messaggio "Digitare S o N", accetti il tasto digitato, verificando se si tratta di S o N, e che riproponga il messaggio se la risposta è diversa da S o da N, altrimenti accetti la risposta fornita dall'operatore. Tuttavia, la routine sarà ancor più generica ed adatta a differenti situazioni, se potrà gestire messaggi e potenziali risposte di vario genere. Eleganza significa sviluppare algoritmi che sono, al tempo stesso, semplici e ben congegnati: in ogni caso è più importante creare algoritmi efficienti e generici, che non solamente eleganti. Un altro importante aspetto degli algoritmi è il loro flusso di controllo e dei dati e come questo possa essere rappresentato con appositi diagrammi: questo argomento verrà trattato più avanti nel corso.



Diagrammi piramidali e di flusso

Il diagramma a blocchi sovrapposti (a sinistra) evidenzia con chiarezza le nidificazioni contenute nell'algoritmo del programma, mentre il diagramma di flusso (a destra) sottolinea i collegamenti ed i livelli di elaborazione dello stesso programma. Gli algoritmi più "primitivi" sono quelli contenuti nelle nidificazioni più interne, e sono i più "bassi", in termini gerarchici

Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

È la volta di due giochi abbastanza diversi tra loro, sia nell'ambientazione che nello svolgimento. Il primo è una delle tante versioni di giochetti "spaziali", mentre il secondo è di un genere più "sportivo".

Space command per lo Spectrum

Questo gioco non è, tutto sommato, molto originale: si tratta della "solita" base spaziale, attaccata dai "soliti" belligeranti alieni e difesa strenuamente dall'astronave del giocatore. Sembra in effetti di tornare ai primi giochini tipo "arcade". Tuttavia, come nelle abitudini della Virgin (la software house che ha prodotto il gioco), la grafica è molto curata e l'azione è veloce. Per assicurarsi maggiori probabilità di successo (e di divertimento) è consigliabile l'uso di joystick.





Wimbledon 64 per il Commodore 64

Fin dai primi videogiochi, il tennis, e vari derivati, è sempre stato uno dei temi favoriti (chi non si ricorda i primi "ping pong" da applicare al televisore di casa?).
Il gioco in questione, concepito per due giocatori "armati" di joystick, è una versione molto più sofisticata di questo genere di competizione: sullo schermo sono riprodotti il campo di gioco, due scattanti tennisti e, in alto, i punteggi di ciascun giocatore. Il controllo, tramite joystick, è molto accurato e si riesce a far compiere interessanti evoluzioni alla pallina. Per esser veramente divertente, Wimbledon 64 va usato da due giocatori di pari livello, altrimenti le partite terminano ben presto. Il gioco è prodotto dalla Merlin (GB).

L'Adam

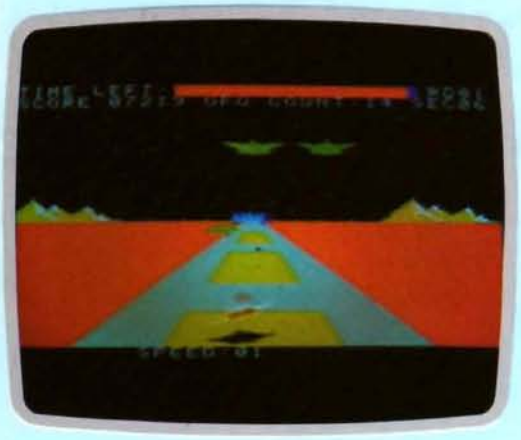
Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Il sistema Adam si basa sulla consolle per videogiochi della ColecoVision, potenziata con una serie di unità addizionali. In effetti, tutti gli Adam oggi in commercio si servono del modulo ColecoVision, sebbene la Coleco stia progettando una nuova macchina più compatta.
Il sistema Adam si compone di vari elementi: tanto il modulo ColecoVision quanto quello di memoria (che contiene un microprocessore Z80A e due lettori di nastri) vengono fissati ad una larga base di plastica. Il collegamento elettrico tra i due avviene tramite una porta d'espansione. La tastiera è collegata al sistema da un lungo cavetto "a molla" ed una delle due unità di controllo per i giochi può essere inserita sul lato destro della tastiera, per svolgere le funzioni di tastierina numerica. La stampante viene collegata al modulo di memoria e il suo cavo d'alimentazione fornisce corrente all'intero sistema.
La tastiera ha 75 tasti (i soliti tasti alfanumerici, 6 tasti di funzione, una serie di tasti di controllo per il word processor e 4 per i movimenti del cursore). È ben strutturata, ha una discreta "sensibilità" e il suo uso è confortevole.
All'accensione, l'Adam abilita automaticamente il modo "macchina da scrivere elettronica", inviando i caratteri digitati sulla tastiera contemporaneamente allo schermo e alla stampante. L'utilità pratica di questo modo è piuttosto limitata, ma con la semplice pressione di un tasto si può passare al modo "SmartWriter", un semplice ma funzionale word processor assai adatto ad un principiante.
Nel modo SmartWriter, lo schermo conserva le caratteristiche precedenti, ma sull'ultima linea compaiono i comandi associati ai tasti di funzione. Questi, abbinati ai tasti di gestione del word processor, facilitano sensibilmente l'uso dello SmartWriter. Inoltre, le opzioni contenute nel menu permettono all'utente di eseguire le principali operazioni di archiviazione, visualizzazione e stampa di un testo senza dover ricorrere al manuale operativo.
Il formato dello schermo è di 20 linee per 36 colonne ma, usando il video come una "finestra" che scorre sul testo, è possibile lavorare su righe più lunghe.
Il lettore di nastri è concettualmente simile al Microdrive della Sinclair, ma possiede una capacità di immagazzinamento superiore (circa 256 Kbyte per nastro) ed è molto più veloce di un comune registratore a cassetta.



CHRIS STEVENS

Il sistema Adam
L'Adam è un apparecchio "tutto o nulla". È infatti un sistema completo composta da: micro, lettore di nastri ad alta velocità, stampante a margherita, tastiera, 2 joystick e tre pacchetti software (due eccellenti giochi ed un word processor). Il tutto costituisce un buon home computer, benché, in definitiva, si tratti di un apparecchio per videogiochi potenziato



Scaricabarili

Buck Rogers è un gioco su nastro, adatto all'unità di lettura dell'Adam. Il programma del gioco è piuttosto lungo ed è stato necessario suddividerlo in più fasi, caricate sequenzialmente nella memoria del computer e giocate una dopo l'altra. Alla fine del gioco, l'unità di lettura riavvolge automaticamente il nastro e registra su di esso il punteggio massimo. Quindi la tabella dei punteggi contiene tutti i migliori punteggi realizzati, non solo quelli ottenuti da quando l'apparecchio è stato acceso

Il lettore si trova sotto il completo controllo del computer che può gestire lo scorrimento del nastro per la ricerca di un particolare file. Poiché i nastri (o 'data pack', come li chiama la Coleco) vengono venduti già formattati, sul lettore non si possono usare comuni audiocassette. Quando viene acceso, l'Adam carica automaticamente in memoria il programma presente sul nastro, sia che si tratti di un pacchetto di giochi, della cassetta SmartBASIC o di un programma qualsiasi scritto dall'utente. Se l'unità di lettura è vuota, il sistema seleziona automaticamente il modo "macchina da scrivere elettronica".

La stampante, del tipo "a margherita", che accetta indifferentemente fogli singoli o moduli continui, è molto lenta e relativamente rumorosa, ma la qualità della stampa è paragonabile a quella di una macchina da scrivere elettrica.

Nonostante gli 80 Kbyte di RAM, il programma SmartBASIC su cassetta lascia solo 28 Kbyte a disposizione dell'utente. Lo SmartBASIC è simile al dialetto BASIC della Apple, ha buone qualità grafiche ed offre una gamma di sedici colori. Purtroppo, il manuale originale del BASIC pecca di superficialità, di scarsa chiarezza ed usa termini fortemente gergali. Tuttavia, la stretta somiglianza con il BASIC della Apple permette agli utenti dell'Adam di servirsi della vasta biblioteca di consultazione della Apple.

Ad un primo esame, l'Adam può apparire un buon investimento: il software incorporato, per l'elaborazione di testi, dà ottimi risultati ed il sistema non solo può contare su un discreto numero di cartucce di gioco ColecoVision ma, corredato di un adattatore, può accedere alla vastissima gamma di giochi

Tastiera separata

Solo due modelli di home micro possiedono una tastiera separata, caratteristica tipica dei computer gestionali. La tastiera dell'Adam possiede tasti di funzione e alcuni tasti dedicati al word processor fornito a corredo della macchina

Manopola del joystick

Sostituisce la più comune "leva" di controllo del joystick



Alloggiamento del joystick

Qui vengono riposti i joystick quando non vengono usati. Ai lati di questa cavità ci sono le prese di collegamento per gli stessi joystick

Sistema per videogiochi ColecoVision

Questo modulo viene venduto anche singolarmente come computer dedicato per giochi ma, collegato ad altre unità, può trasformarsi in un home computer

Vassoio di supporto

Su di esso vengono fissate le due unità mostrate in figura

Presa per la stampante

Permette il collegamento con la stampante e riceve da quest'ultima la corrente d'alimentazione per tutto il sistema. La stampante, quindi, deve sempre essere collegata

Unità a nastri

Non si tratta di comuni lettori di cassette, ma di vere unità a nastri ad alta velocità, completamente controllate dal computer

CHRIS STEVENS



Pulsanti di "fuoco" del joystick

Tastierina numerica

Entrambi i joystick di corredo all'Adam sono dotati di una tastiera numerica

Astuccio del joystick

Viene innestato sul lato destro della tastiera. Su di esso viene fissato un joystick



Pulsante di reset

Cartuccia di giochi

Le cartucce di giochi vengono inserite nell'apposita fessura dell'unità di gioco. Le cartucce disponibili per l'Adam sono numerose, ma piuttosto costose



Connettore per espansioni

Si innesta sul retro del modulo d'espansione

Connettore per il cavo della tastiera

Un cavo a molla collega la tastiera all'unità centrale

Alloggiamento per una seconda unità a nastri

"Data pack" digitali

Non vanno confusi con le comuni audio-cassette. L'Adam usa particolari nastri pre-formattati, capaci di immagazzinare 256 Kbyte ciascuno



I joystick dell'Adam

Il corredo standard dell'Adam prevede 2 joystick: quello che possiede una tastierina numerica può essere innestato in uno speciale alloggio. La Coleco, inoltre, propone altri 2 sistemi "deluxe" di controllo per giochi. Il Super Action Controller è composto da una coppia di joystick con tastierina numerica e grilletti tipo "fucile", dotati di connettori-D (standard Atari). Nel prezzo di acquisto è compresa una cartuccia contenente il gioco "Baseball". Il Roller Controller è un sistema di controllo per giochi stile "Arcade". È composto da due joystick standard Coleco e da un'unità di controllo "trackball" per risposte rapide e spostamenti di precisione. Due serie di doppi pulsanti di "fuoco" consentono la partecipazione al gioco di due giocatori ed un interruttore di selezione abilita l'uso del joystick o dell'unità "trackball". Nel kit del Roller Controller è incluso lo "Slither", un gioco "Arcade" stile "Millepiedi"



prodotti dalla Activision e dall'Atari. Al contrario, i pacchetti software per usi professionali sono costosi e piuttosto limitati: comprendono un database, una lavagna elettronica, un programma di elaborazione di testi ed una versione del linguaggio LOGO. La Coleco sta progettando la produzione di un'unità a dischetti e di un'espansione di memoria di 144 Kbyte che dovrebbe permettere all'apparecchio di usare il CP/M.
Tuttavia, l'Adam presenta anche alcuni limiti. La qualità della stampa non è quella che ci si potrebbe aspettare da un modello a margherita, la stampante è lenta e rumorosa, la RAM utente è piuttosto ridotta ed il caricamento del BASIC da nastro, a lungo andare, è fastidioso. Comunque, l'Adam si dimostra una buona scelta per coloro che già possiedono il modulo ColecoVision.

ADAM

DIMENSIONI

mm. 381 x 279 x 102 (unità di espansione per la memoria)
mm. 381 x 355 x 152 (stampante)
mm. 381 x 152 x 51 (tastiera)

CPU

Z80A

MEMORIA

80 Kbyte di RAM (16 Kbyte usati per la memoria video), con possibilità di espansione fino a 144 Kbyte

VIDEO

20 righe di 36 caratteri (24 righe di 31 caratteri, in BASIC). Risoluzione grafica di 256 x 159 punti, con 16 colori

INTERFACCE

Connettore per stampante SmartWriter, per linea telefonica (Rete Adam), tre connettori per espansioni, interfaccia per espansione ColecoVision

LINGUAGGI FORNITI

SmartBASIC (su cassetta)

TASTIERA

Tipo macchina da scrivere, formato QWERTY, con 75 tasti (di cui 6 di funzione, programmabili)

DOCUMENTAZIONE

I manuali in dotazione sono tre: uno, di 64 pagine, spiega il collegamento ed il funzionamento delle varie unità, il secondo riguarda il word processor SmartWriter, il terzo concerne lo SmartBASIC. La documentazione è abbastanza facile da seguire, ma talvolta lascia qualche perplessità. Il manuale originale del BASIC è spesso considerato incomprensibile, almeno per un principiante

PRO

La gran quantità e la buona qualità delle cartucce per giochi fanno dell'Adam un apparecchio ideale per l'intrattenimento. Il word processor è molto facile da usare

CONTRO

La stampante, a margherita, è lenta e piuttosto rumorosa. I 'Data pack' sono pre-formattati (non è possibile adoperare i comuni nastri a cassetta)

Grafica BBC ed Electron

Lo sviluppo di un gioco grafico per questi due apparecchi

Il BASIC BBC offre al programmatore due importanti vantaggi rispetto al BASIC Microsoft standard: è più rapido nella elaborazione e possiede funzioni che agevolano la programmazione strutturata. Il "cuore" dello sviluppo di un programma strutturato è la creazione di piccole sezioni di programma indipendenti, da poter mettere a punto individualmente, prima di collegarle tutte tra loro. Qualsiasi programma BASIC può essere strutturato, suddividendo i vari moduli in subroutine, ma il BASIC BBC offre uno strumento in più: le cosiddette "procedure", ossia sezioni di codice, progettate per eseguire uno specifico compito all'interno del programma. Immaginiamo che una parte del programma debba provocare una data pausa dopo l'esecuzione di ogni istruzione. Nel BASIC standard si dovrà usare un "falso ciclo iterativo", ovvero un ciclo il cui unico compito è rallentare l'esecuzione del programma:

```
10 PRINT "PRIMA SEZIONE"
20 FOR I = 1 TO 100: NEXT I
30 PRINT "SECONDA SEZIONE"
40 FOR I = 1 TO 100: NEXT I
50 PRINT "TERZA SEZIONE"
60 FOR I = 1 TO 100: NEXT I
70 PRINT "QUARTA SEZIONE"
80 END
```

Tuttavia, sarebbe meglio inserire il ciclo di ritardo in una subroutine:

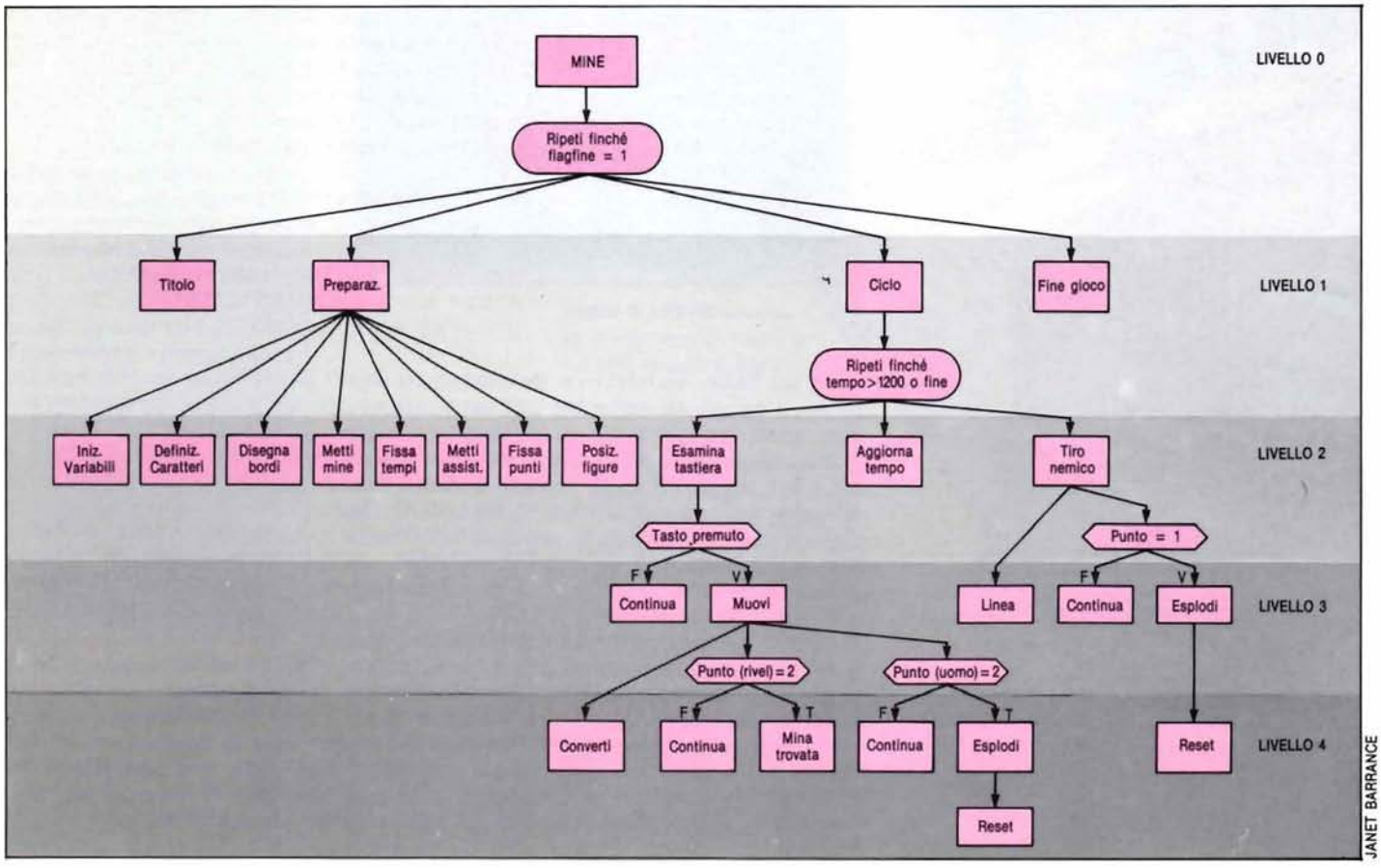
```
 10 PRINT "PRIMA SEZIONE"
 20 GOSUB 100
 30 PRINT "SECONDA SEZIONE"
 40 GOSUB 100
 50 PRINT "TERZA SEZIONE"
 60 GOSUB 100
 70 PRINT "QUARTA SEZIONE"
 80 END
100 REM ** SUBROUTINE **
110 FOR I = 1 TO 100: NEXT I
120 RETURN
```

In BASIC BBC, sostituendo la subroutine con una procedura, si ottiene:

```
10 PRINT "PRIMA SEZIONE"
20 PROCritardo
```

Diagramma di struttura

A differenza dei diagrammi di flusso, questo diagramma di struttura illustra la struttura procedurale del programma e non il flusso di controllo che lo attraversa. La "capsula" indica l'inizio del ciclo REPEAT...UNTIL, le losanghe corrispondono alle "decisioni": quando la verifica fallisce, i cicli iterativi collegati continuano. I numeri di livello evidenziano la struttura del programma: tutti i cicli e le chiamate di procedura "aprono" una nuova sezione del programma ed un livello logico inferiore



```
 30 PRINT "SECONDA SEZIONE"
 40 PROCritardo
 50 PRINT "TERZA SEZIONE"
 60 PROCritardo
 70 PRINT "QUARTA SEZIONE"
 80 END
100 REM ** DEFINISCE LA PROCEDURA **
110 DEF PROCritardo
120 FOR I = 1 TO 100: NEXT I
130 ENDPROC
```

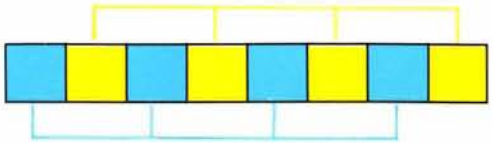
La realizzazione di una subroutine e quella di una procedura hanno molti aspetti comuni. Per esempio, entrambe vengono definite dopo l'istruzione END e possono essere richiamate più volte dal corpo principale del programma. Le procedure offrono il vantaggio di venir richiamate "per nome", anziché col numero di linea. La definizione (DEF) della procedura (PROC) può iniziare dovunque, dopo l'istruzione END.
Se il programma portato ad esempio richiede pause di diversa durata, prima di ciascuna sezione, con le procedure l'obiettivo è presto raggiunto, tramite l'impiego di "parametri" inclusi nella definizione. Poniamo che l'intervallo tra la prima e la seconda sezione debba essere di 100 cicli, quello tra la seconda e la terza di 200 cicli ed infine quello tra la terza e la quarta di 175 cicli. Nel BASIC standard si dovrà assegnare ad I l'appropriato valore prima di ogni chiamata di subroutine. Con le procedure, invece, il valore può essere fornito tra parentesi tonde, alla fine del comando di chiamata:

```
 10 PRINT "PRIMA SEZIONE"
 20 PROCritardo (100)
 30 PRINT "SECONDA SEZIONE"
 40 PROCritardo (200)
 50 PRINT "TERZA SEZIONE"
 60 PROCritardo (175)
 70 PRINT "QUARTA SEZIONE"
 80 END
100 REM ** DEFINISCE LA PROCEDURA **
110 DEF PROCritardo (N)
120 FOR I = 1 TO N: NEXT I
130 ENDPROC
```

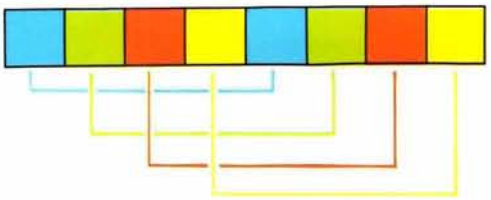
È possibile fornire alla procedura anche una sequenza di valori, purché essi siano separati da virgole, e la stessa tecnica si usa anche per assegnare valori successivi alle variabili, quando la procedura viene richiamata.
Il gioco che ci apprestiamo a sviluppare prevede un solo giocatore e si serve dei tasti di controllo del cursore per muovere un "rivelatore di mine" attraverso un campo minato. Il punteggio aumenta tutte le volte che il giocatore individua una nuova mina. Tuttavia, la ricerca è ostacolata da alcuni fattori. Innanzi tutto, l'"assistente" (l'addetto al rivelatore) esegue con precisione tutti i movimenti impartiti da tastiera e, poiché le mine possono esplodere al più lieve contatto, occorre far attenzione che l'assistente non ne calpesti una. Inoltre, l'assistente è sottoposto al fuoco del nemico e la durata massima del gioco è di soli due minuti. Nella versione finale, il giocatore dispone di 4 assistenti per ogni partita e può selezionare uno dei livelli di difficoltà compresi tra 0 (il più facile) e 9 (il più difficile).
Poiché il BASIC BBC si serve di procedure, i diagrammi di flusso sono di utilità relativa. Si potranno invece usare i "diagrammi di struttura" per illustrare come le procedure debbano essere collegate tra loro per comporre il programma definitivo. In questi diagrammi, i cicli REPEAT...UNTIL sono rappresentati da blocchi a forma di capsula; i blocchi di decisione hanno invece la forma di losanghe schiacciate. Infine, è importante sottolineare che la stesura del diagramma di struttura non si è conclusa prima dell'inizio della fase di programmazione, ma è scaturita da una serie di successive rifiniture.

Le procedure di preparazione

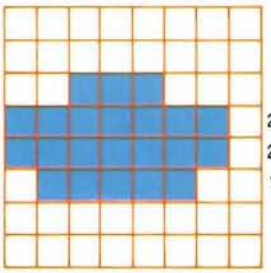
Prima di sviluppare le routine che visualizzeranno le varie figure sullo schermo è necessario scegliere il MODE grafico più adatto, tenendo conto di tre fattori principali: la risoluzione, il colore e la memoria. In generale, il consumo di memoria cresce proporzionalmente alla quantità di informazioni presenti sullo schermo. Perciò l'uso dell'alta risoluzione e di più colori implica una maggior occupazione di memoria. Ciò ha scarsa rilevanza quando i programmi sono piccoli ma, nel nostro caso, il programma ha dimensioni discrete. Inoltre, dovremo usare colori diversi per le mine, per l'assistente e per il rivelatore, affinché siano facilmente distinguibili. Il BBC Modello B possiede due MODE di media risoluzione: il MODE 2, che dispone di 16 colori, ed il MODE 5, con soli 4 colori. Esaminando come il BBC interpreti la configurazione di bit di ciascun "modo", ci accorgiamo che il MODE 5 usa meno memoria del MODE 2.
A differenza di altri micro, il BBC immagazzina in un byte i colori e l'informazione di stato dei pixel di ciascuna piccola area dello schermo. Nel MODE 2 occorrono quattro bit per rappresentare i 16 colori possibili: quindi un byte può contenere l'informazione per il colore di due soli pixel. I bit all'interno del byte sono organizzati nel seguente modo:



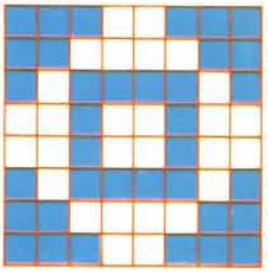
Poiché il MODE 5 ha solo 4 colori, bastano due bit per contenere l'informazione del colore. Quindi un byte potrà rappresentare quattro pixel così:



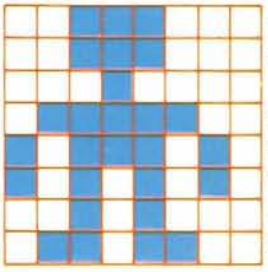
Mina



Rivelatore



Assistente



I caratteri del gioco

Le mine, il loro rivelatore e l'assistente sono tutti caratteri definiti dall'utente (UDG). L'informazione relativa ai pixel viene suddivisa in 8 byte e ciascun byte contiene la rappresentazione binaria di una riga dell'immagine

Dato che entrambi i modi hanno una risoluzione di 160 × 256 pixel, il numero di byte necessari alla memoria video del MODE 2 sarà (160×256)/2=20 Kbyte, mentre quello del MODE 5 sarà (160×256)/4=10 Kbyte. Quindi il MODE 5 lascia ben 10 Kbyte in più a disposizione del programmatore.
Sul BBC, la creazione dei caratteri definiti dall'utente (chiamati brevemente UDG) è semplice: ogni carattere è composto da 8 righe, ciascuna rappresentata da un numero decimale.
Il comando VDU23 consente al programmatore di assegnare un codice ASCII compreso tra 226 e 255 al carattere. Il secondo valore, nel comando VDU, indica il numero di codice attribuito al carattere definito dagli otto numeri che lo seguono.

```
VDU23,224,0,0,56,254,254,124,0,0
```

definisce CHR$(224) come immagine della mina, che apparirà sul video tramite l'istruzione PRINT CHR$(224). La seguente procedura definisce i tre caratteri utilizzati nel gioco.

```
2380 DEF PROCcreafig
2390 REM ** MINE **
2400 VDU23,224,0,0,56,254,254,124,0,0
2410 REM ** RIVELATORE DI MINE **
2420 VDU23,225,231,195,189,36,36,189,195,231
2430 REM ** ASSISTENTE **
2440 VDU23,226,56,56,16,124,186,170,40,108
2450 ENDPROC
```

Dimensioni dello schermo

Per visualizzare sullo schermo le tre figure così definite, si può usare il comando PRINT TAB(X,Y). Nel MODE 5, il video è composto da 32 righe di 20 caratteri: ciò significa che X è compresa tra 0 e 19 ed Y tra 0 e 31. L'area del campo minato è quella mostrata dalla figura in basso. Per disporre le mine in modo casuale, ci possiamo servire della funzione RND(N). Se N è un numero intero, RND(N) dà come risultato un numero intero tra 1 ed N. Le coordinate orizzontali di ogni mina devono essere comprese tra 2 e 17. RND(16) genera valori tra 1 e 16, perciò RND(16)+1 seleziona le coordinate all'interno del campo minato. Questa procedura colloca sul terreno un numero di mine pari al valore "ricevuto" dal programma principale:

```
2560 DEF PROCmettimine(numeromine)
2570 REM ** ASSEGNA IL COLORE 2 AL VERDE **
2580 VDU19,2,2,0,0,0
2590 FOR I=1 TO numeromine
2600 PRINT TAB(RND(16)+1,RND(25)); CHR$(224)
2610 NEXT I
2620 ENDPROC
```

Nel MODE 5, i quattro colori comunemente usati sono nero, rosso, giallo e bianco, corrispondenti ai numeri di colore 0, 1, 2 e 3. Tuttavia, il comando VDU19 permette di cambiarli con uno dei 16 colori disponibili sull'apparecchio. I numeri da 0 a 3 indicano i colori "logici" di sfondo, ma tutti e 16 i colori del BBC possiedono un numero proprio, detto numero di colore "attuale" (vedere la tabella nel manuale per l'utente), indipendente dal MODE adoperato. Se vogliamo che nel gioco le mine siano verdi (numero di colore attuale 2) e non vogliamo il giallo (normalmente indicato come colore logico 2), il comando VDU eseguirà questo cambiamento.
All'inizio del gioco il rivelatore e l'assistente compaiono sullo schermo, rispettivamente, negli angoli in basso a sinistra ed in alto a destra. Per cambiare la loro posizione iniziale si dovranno usare le variabili (xrivel,yrivel) per le coordinate del rivelatore e (xuomo,yuomo) per quelle dell'assistente.

```
2830 DEF PROCmettifigure
2840 COLOUR 1
2850 PRINT TAB(xrivel,yrivel); CHR$(225)
2860 PRINT TAB(xuomo,yuomo); CHR$(226)
2870 COLOUR 2
2880 ENDPROC
```

I comandi COLOUR all'inizio ed alla fine della procedura selezionano i colori logici dello sfondo: COLOUR 1 seleziona il colore logico 1 (rosso) per il rivelatore e l'assistente, mentre COLOUR 2 ripristina il colore verde per successive visualizzazioni. Comunque, prima di usare questa procedura, occorre inizializzare le variabili xrivel, yrivel, xuomo ed yuomo. Ciò viene fatto in un'altra procedura, assieme alla inizializzazione di altre variabili, usate altrove.

```
2320 DEF PROCinizializza
2330 xrivel=2: yrivel=25: xuomo=17: yuomo=1
2340 xinizio=120: xfine=1144
2350 zero$="000000"
2360 ENDPROC
```

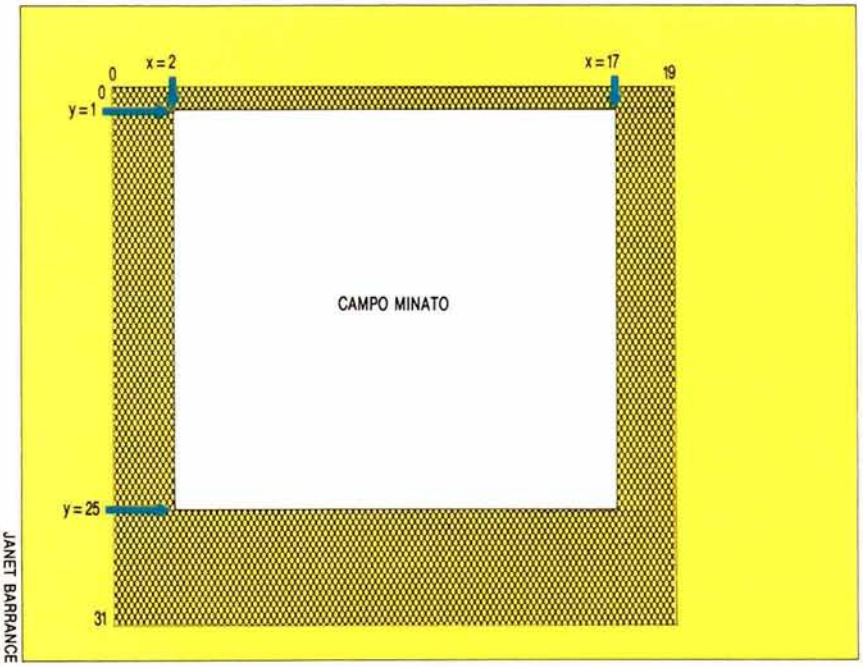
Tutte le procedure viste possono ora essere sottoposte a verifica con il seguente programma di chiamata:

```
 5 MODE 5
10 COLOUR 2
20 PROCinizializza
30 PROCcreafig
40 PROCmettimine(40)
50 PROCmettifigure
60 END
```

Più avanti verrà trattata la sincronizzazione ed il controllo dei movimenti tramite la tastiera.

Campo minato

L'area del campo minato è di 16 righe di 20 caratteri: le mine vengono disposte in modo casuale, durante la procedura di preparazione dello schermo



JANET BARRANCE

Jet Pac

Questo gioco ha segnato il debutto della "Ultimate, Play the Game"

I giochi "arcade" del tipo "discesa di alieni" sono spesso poco apprezzati per il loro sviluppo semplice e prevedibile. Tuttavia, anche la loro progettazione richiede un notevole impegno di programmazione ed il *Jet Pac* ne è un ottimo esempio.

Come spesso accade, l'immagine di copertina della cassetta promette molto più della realtà: quale capo pilota collaudatore della Compagnia di Trasporti Interstellari Acme, il giocatore deve viaggiare per la galassia costruendo astronavi su diversi pianeti; durante queste soste raccoglie tutto l'oro e le pietre preziose che può trovare. Nella costruzione del razzo gli è di grande aiuto un Hydrovac Jet Pac, capace di sollevare qualsiasi peso, che permette al giocatore di spostare con facilità i pezzi da assemblare. Il pilota collaudatore è armato di Laser Fotonico a due bocche di fuoco, da usare contro tutti gli alieni che vogliono impedirgli di impossessarsi delle ricchezze del loro pianeta.

La descrizione del gioco richiama alla memoria eroici viaggi in Tecnicolor attraverso pianeti inesplorati, ma la realtà è molto meno appassionante. I pianeti che il pilota visita sono più o meno tutti uguali, tanto da suscitare nel giocatore una fastidiosa sensazione di "dejà vu" ad ogni atterraggio.

Tuttavia, il gioco si fa appassionante non appena compaiono gli alieni: ciascun pianeta ne ospita una sola specie (del resto si "riproducono" a velocità vertiginosa!).

Le varie specie hanno le forme più disparate (da salsicce volanti a sfere saltellanti), ma esiste una caratteristica comune a tutte: un semplice contatto con essi provoca morte istantanea.

L'astronave da costruire è composta da tre elementi, sparsi sulla superficie del pianeta, che devono essere assemblati prima di poter decollare verso un altro pianeta. Per questo lavoro non c'è bisogno di particolari attrezzature: basta semplicemente sollevare gli elementi e lasciarli cadere sulla base del razzo per veder "crescere" la nuova astronave.

Comunque, il ruolo principale appartiene agli alieni: intere schiere di essi calano ininterrottamente dal cielo, muovendosi minacciosamente in direzione del giocatore. Ben presto ci si accorge che gli assalti seguono uno schema predeterminato, pur partendo da posizioni casuali. La prima ondata di attaccanti cala lentamente e ciò dà al giocatore il tempo di distruggerla col proprio laser. La seconda ondata è composta da sfere saltellanti, che rimbalzano per tutto lo schermo tra le sporgenze rocciose ed il terreno. Questa combinazione di movimenti predeterminati e casuali richiede al giocatore una notevole abilità e riflessi rapidi per non essere colpito.

I giochi di questo tipo vengono spesso danneggiati da una scarsa scelta dei tasti di controllo. Nel *Jet Pac*, invece, la scelta dei tasti è stata fatta molto accuratamente: gli spostamenti a destra ed a sinistra sono controllati da due tasti dell'ultima riga della tastiera, che si adattano bene tanto ai giocatori destri quanto a quelli mancini. Un qualsiasi tasto della seconda riga mette in azione il laser, capace anche di raffiche continue. I tasti della riga superiore accendono i vettori di decollo e quelli della riga più alta controllano il volo dell'astronave. Se in fase di volo non si preme alcun tasto, il razzo viene lentamente attratto verso la superficie del pianeta dalla forza gravitazionale.

La grafica del gioco è discreta. Il pilota è ben disegnato e l'Hydrovac Jet Pac emette realistici sbuffi di fumo quando vengono azionati i razzi di decollo o di volo. Il laser spara raggi luminosi multicolori e gli alieni, quando vengono colpiti, esplodono in dense nubi di fumo. Il gioco è praticamente identico sullo Spectrum e sul Vic-20, anche se il formato video di quest'ultimo appesantisce notevolmente la figura del pilota collaudatore. Il *Jet Pac* si dimostra un gioco piacevole e di facile uso su entrambi gli apparecchi.

Jet Pac: per lo Spectrum a 16 o 48 K; per il Vic-20 ad 8 K espanso

Editori: Ashby Computers and Graphics Ltd. (GB)

Autori: Ultimate, Play The Game

Joystick: Kempston Competition-Pro (Spectrum); la maggior parte dei joystick Commodore-compatibili (Vic-20)

Formato: Cassetta



ASSEMBLAGGIO NELLO SPAZIO

Il lavoro di assemblaggio eseguito dal pilota collaudatore del Jet Pac è reso pericoloso da continui attacchi di alieni, ma è in parte compensato dalle ricchezze che gli... piovono dal cielo!

Il Jet Pac sullo Spectrum

LIZ HEANEY

Finestre sullo Spectrum

Creare riquadri sullo schermo è utile in moltissime applicazioni



Finestra "aperta"
La parola "HELLO" viene spostata orizzontalmente e verticalmente all'interno della finestra, un pixel alla volta

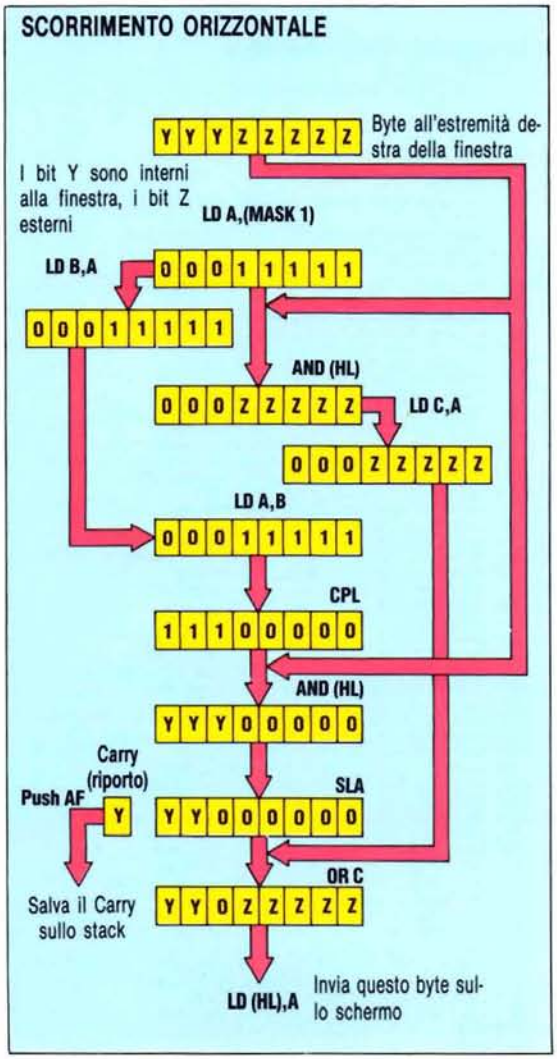
Il programma in codice macchina qui presentato permette di aprire "finestre" sullo schermo dello Spectrum e di farle scorrere verso destra, sinistra, l'alto ed il basso. Le finestre possono avere qualsiasi dimensione e disposizione sullo schermo e, inoltre, il loro contorno non deve necessariamente coincidere con i margini di un carattere di 8 × 8 pixel.
Per trattare i parametri della finestra e per memorizzare temporaneamente le informazioni, il programma in codice macchina si serve di una serie di tabelle, il cui indirizzo iniziale è $B004 (45060 in decimale); l'indirizzo della tabella per la finestra è depositato in $B000 e $B001 (45056 e 45057 in decimale). La tabella di una finestra occupa il byte, quindi l'indirizzo di un'eventuale seconda finestra sarà $B00F (57071 in decimale), quello di una terza $B01A (45082 in decimale) e così via.
Il programma dimostrativo in BASIC prevede l'impiego di una sola finestra, i cui parametri vengono depositati in memoria, con delle POKE, dalle linee 180-230, mentre la routine di inizializzazione viene chiamata alla linea 240. In modo analogo si possono aggiungere altre finestre. Per passare da una finestra all'altra, basta depositare l'indirizzo della relativa tabella nelle locazioni di memoria WT e WT+1. La direzione di "scroll" viene selezionata depositando opportuni valori (0 = a destra, 1 = a sinistra, 2 = verso l'alto, 3 = verso il basso) nella locazione di memoria TABFIN+DIR.
Il programma in linguaggio Assembly inizia definendo un certo numero di costanti. Nella ROM dello Spectrum, INDPIX è una routine che calcola l'indirizzo del byte di schermo ed il numero del bit all'interno di quel byte, relativi al pixel individuato dalle coordinate usate in PLOT. La coordinata y viene "passata" alla routine INDPIX nel registro B e la coordinata x nel registro C. La routine restituisce (o, detto in gergo, "ritorna") l'indirizzo di schermo nella coppia di registri HL e la posizione del bit nel registro A.
La routine INITW verifica che le coordinate per l'angolo inferiore destro della finestra si trovino effettivamente più in basso e a destra rispetto a quelle dell'angolo superiore sinistro e che i lati destro e sinistro non coincidano, cioè non si trovino nello stesso byte di memoria video. Quest'ultima procedura garantisce che la larghezza della finestra sia di almeno una cella di carattere: lo scroll di una finestra di minor larghezza richiederebbe l'aggiunta di un'ulteriore sezione di programma.
Eventuali errori, commessi nella fase di inizializzazione della finestra, vengono indicati da una routine ROM di "messaggio d'errore", chiamata tramite l'istruzione RST 8 (linea 2110 del listato Assembly), che riporta il sistema al modo di comando BASIC. La direttiva DEFB 25 provoca la visualizzazione del messaggio "Q Parameter error".
L'ultima sezione di INITW calcola MASKSIN e MASKDES: questi valori sono usati per lo scroll di quei byte di schermo posti "a cavallo" del perimetro della finestra. I bit di questi byte, corrispondenti ai pixel esclusi dal perimetro della finestra, vengono posti a 1, mentre gli altri sono posti a 0.
Il programma di scroll vero e proprio inizia all'etichetta SCROLL. Per prima cosa il programma controlla la direzione dello "scrolling", chiamando la routine ORIZZ (per scorrimenti verso destra o sinistra) oppure la routine VERT (per scorrimenti verso l'alto o il basso).
Lo scorrimento a destra e quello a sinistra si ottengono in modo molto simile, quindi si usa una sola routine: la selezione avviene in base al valore del bit zero del byte di direzione. Per capire come funziona ORIZZ, esaminiamo la procedura di scroll verso sinistra.
Entrambi gli scorrimenti orizzontali partono dalla prima riga di pixel della finestra e procedono verso il basso, pertanto ORIZZ inizia copiando le coordinate della prima riga in una locazione di memoria temporanea, per la linea "corrente". Nello scorrimento verso sinistra, la copia deve iniziare dal margine destro di ciascuna riga di pixel e procedere verso destra. Questa operazione viene preceduta dal trasferimento di MASKD e MASKS rispettivamente in MASK1 e MASK2, dal calcolo e dall'immagazzinamento nella coppia di registri DE dell'indirizzo del primo byte di schermo a sinistra della corrente riga di pixel e, infine, dal calcolo e deposito, nella coppia di registri HL, dell'indirizzo dell'ultimo byte di schermo a destra della corrente riga di pixel. Viene quindi chiamata la routine HLNSCR, per lo scroll della riga di pixel. Questa routine verifica anche se è stata raggiunta l'ultima riga della finestra; in tal caso, la routine passa alla riga di pixel successiva e ritorna a ORIZZ3 per eseguire un nuovo scroll.
HLNSCR inizia all'estremità di una linea in cui il byte estremo può essere "a cavallo" del margine della finestra. Successivamente, si sposta sui byte della riga interamente contenuti nella finestra e termina all'altra estremità della riga dove, nuovamente, alcuni bit possono essere interni ed altri esterni alla finestra. Lo schema di scorrimento orizzontale (vedere la figura) illustra come la routine operi sul byte più a destra, nella procedura di

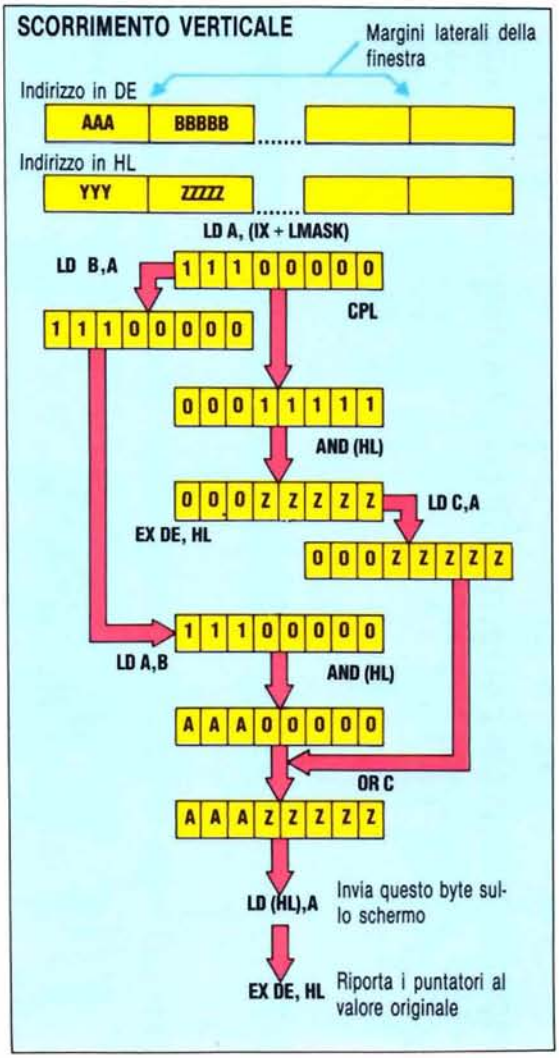
scroll verso sinistra. La sezione di HLNSCR che inizia all'etichetta PROX, esegue lo scroll dei byte interni alla finestra. Il bit trasferito nel flag di Carry (o di riporto), viene appoggiato sullo stack dall'istruzione PUSH AF, per esser poi recuperato con POP AF. Nello scroll verso sinistra, la routine impiega l'istruzione RL (HL) ed il byte sullo schermo viene spostato verso sinistra, cosicché il bit proveniente dal flag di Carry viene inserito dall'estremità sinistra del byte e quello proveniente dall'estremità destra viene trasferito nel Carry. Quindi PUSH AF appoggia il flag di Carry sullo stack per poterlo inserire, in seguito, nel byte successivo. Per individuare la fine della riga, la routine confronta semplicemente i due registri L ed E, poiché il byte alto dell'indirizzo di schermo è identico per tutti i byte di una stessa riga. Lo scroll parziale dell'ultimo byte della riga è del tutto simile a quello del primo byte.

Anche le operazioni di scorrimento verso l'alto e il basso sono riunite in una sola routine. Esaminando il funzionamento di VERT, nello scroll verso l'alto, si nota che la routine inizia trasferendo le coordinate della prima riga in una locazione di memoria temporanea. Vengono quindi calcolati gli indirizzi di schermo dei byte posti alle due estremità della riga e viene valutata la lunghezza di quest'ultima. Prima di chiamare la subroutine di scroll verticale (VLNSCR), la routine VERT deposita in DE l'indirizzo del byte all'estremità sinistra della riga ed in HL quello del byte corrispondente della riga sottostante. Quindi verifica se è stato raggiunto il margine inferiore della finestra; in tal caso, la sezione della routine che comincia all'etichetta CLREDG pone a zero tutti i pixel dell'ultima riga della finestra, cancellando tale riga dal video. In caso contrario, l'esecuzione passa alla linea successiva, saltando indietro a VERT5, ed esegue lo scroll della successiva riga di pixel.

VLNSCR opera separatamente sui byte marginali (si veda il diagramma che illustra lo scorrimento verticale), in modo analogo a HLNSCR: per spostare in alto la sezione centrale della riga (composta da tutti i byte contenuti per intero nella finestra), la routine incrementa HL e DE in modo che puntino al primo byte interno della linea corrente ed al corrispondente byte della linea che la precede. Quindi VLNSCR calcola la lunghezza della sezione centrale, inserisce il valore ottenuto in BC e usa l'istruzione LDIR per spostare la sezione centrale sulla riga superiore.

Queste routine di scroll sono relativamente lente perché il programma è costretto ad eseguire frequenti verifiche.





Direzioni di "scroll"

Lo scorrimento orizzontale presenta particolari problemi ai margini della finestra: i byte di schermo estremi devono essere scomposti per separare i bit interni alla finestra da quelli esterni ed il loro contenuto deve essere ruotato. Entrambe le procedure si servono di AND, OR e SHIFT logici ed usano lo stack per salvare il contenuto dei flag.

Lo scorrimento verticale è più semplice, grazie alla struttura della mappa di memoria video dello Spectrum (vedere a pag. 838). Anche in questo caso, i byte di schermo estremi vengono sottoposti alla procedura prima descritta. L'istruzione EX viene usata per scambiare i contenuti dei registri DE ed HL, che contengono i puntatori all'indirizzo di schermo

FINESTRE SULLO SPECTRUM

Istruzioni operative

1) Trascrivere il programma dimostrativo in BASIC
2) Usare il comando SAVE "SCROLL" LINE 5
3) Trascrivere ed eseguire (RUN) il programma in BASIC per il caricamento del codice macchina
4) Usare il comando SAVE "SCROLLMC" CODE 45312,410 salvandolo su nastro subito dopo il programma dimostrativo
5) Riavvolgere il nastro e battere LOAD "SCROLL"

Listato Assembly

```
 10 INDPIX EQU  #22AA
 20 WT     EQU  #B000
 30 MASK1  EQU  #B002
 40 MASK2  EQU  #B003
 50 TABFIN EQU  #B004
 60 XSINS  EQU  0
 70 YSUP   EQU  1
 80 XDESTR EQU  2
 90 YINF   EQU  3
100 BITS   EQU  4
110 BITD   EQU  5
120 YCORR  EQU  6
130 DIR    EQU  7
140 MASKS  EQU  8
150 MASKD  EQU  9
160 LUNGH  EQU  10
170        ORG  #B100
180 INIT   LD   HL,(WT)
190        PUSH HL
200        POP  IX
210        CALL INITW
220        RET
230 SCROLL LD   HL,(WT)
240        PUSH HL
250        POP  IX
260        BIT  1,(IX+DIR)
270        PUSH AF
280        CALL Z,ORIZZ
290        POP  AF
300        CALL NZ,VERT
310        RET
320 ORIZZ  LD   A,(IX+YSUP)
330        LD   (IX+YCORR),A
340        BIT  0,(IX+DIR)
350        JR   Z,ORIZZ1
360        LD   B,(IX+MASKS)
370        LD   A,(IX+MASKD)
380        JR   ORIZZ2
390 ORIZZ1 LD   B,(IX+MASKD)
400        LD   A,(IX+MASKS)
410 ORIZZ2 LD   (MASK2),A
420        LD   A,B
430        LD   (MASK1),A
440 ORIZZ3 LD   C,(IX+XSINS)
450        LD   B,(IX+YCORR)
460        CALL INDPIX
470        EX   DE,HL
480        LD   C,(IX+XDESTR)
490        LD   B,(IX+YCORR)
500        CALL INDPIX
510        BIT  0,(IX+DIR)
520        JR   Z,ORIZZ4
530        EX   DE,HL
540 ORIZZ4 CALL HLNSCR
550        LD   A,(IX+YINF)
560        CP   (IX+YCORR)
570        RET  Z
580        DEC  (IX+YCORR)
590        JR   ORIZZ3
600 HLNSCR LD   A, (MASK1)
610        LD   B,A
620        AND  (HL)
630        LD   C,A
640        LD   A,B
650        CPL
660        AND  (HL)
670        BIT  0,(IX+DIR)
680        JR   Z,HLN1
690        SRA  A
700        JR   HLN2
710 HLN1   SLA  A
720 HLN2   PUSH AF
730        OR   C
740        LD   (HL),A
750 PROX   BIT  0,(IX+DIR)
760        JR   Z,HLN3
770        INC  HL
780        JR   HLN4
790 HLN3   DEC  HL
800 HLN4   LD   A,L
810        CP   Z
820        JR   Z,ULTI
830        POP  AF
840        BIT  0,(IX+DIR)
850        JR   Z,HLN5
860        RR   (HL)
870        JR   HLN6
880 HLN5   RL   (HL)
890 HLN6   PUSH AF
900        JR   PROX
910 ULTI   LD   C,(HL)
920        LD   A,(MASK2)
930        AND  C
940        LD   B,A
950        POP  AF
960        BIT  0,(IX+DIR)
970        JR   Z,HLN7
980        RR   C
990        JR   HLN8
1000 HLN7  RL   C
1010 HLN8  LD   A,(MASK2)
1020       CPL
1030       AND  C
1040       OR   B
1050       LD   (HL),A
1060       RET
1070 VERT  LD   C,(IX+XSINS)
1080       BIT  0,(IX+DIR)
1090       JR   Z,VERT1
1100       LD   B,(IX+YINF)
1110       JR   VERT2
1120 VERT1 LD   B,(IX+YSUP)
1130 VERT2 LD   (IX+YCORR),B
1140       CALL INDPIX
1150       PUSH HL
1160       PUSH HL
1170       LD   C,(IX+XDESTR)
1180       LD   B,(IX+YCORR)
1190       CALL INDPIX
1200       POP  DE
1210       AND  A
1220       SBC  HL,DE
1230       LD   A,L
1240       DEC  A
1250       LD   (IX+LUNGH),A
1260 VERT3 BIT  0,(IX+DIR)
1270       JR   Z,VERT4
1280       INC  (IX+YCORR)
1290       JR   VERT5
1300 VERT4 DEC  (IX+YCORR)
1310 VERT5 LD   C,(IX+XSINS)
1320       LD   B,(IX+YCORR)
1330       CALL INDPIX
1340       POP  DE
1350       PUSH HL
1360       CALL VLNSCR
1370       LD   A,(IX+YCORR)
1380       BIT  0,(IX+DIR)
1390       JR   Z,VERT6
1400       CP   (IX+YSUP)
1410       JR   VERT7
1420 VERT6 CP   (IX+YINF)
1430 VERT7 JR   NZ,VERT3
1440 CLREDG POP  HL
1450       LD   A,(IX+MASKS)
1460       AND  (HL)
1470       LD   (HL),A
1480       LD   B,(IX+LUNGH)
1490       LD   A,0
1500 CLR1  INC  HL
1510       LD   (HL),A
1520       DJNZ CLR1
1530       INC  HL
1540       LD   A,(IX+MASKD)
1550       AND  (HL)
1560       LD   (HL),A
1570       RET
1580 VLNSCR LD  A,(IX+MASKD)
1590       CALL ENDBYT
1600       INC  HL
1610       INC  DE
1620       LD   B,0
1630       LD   C,(IX+LUNGH)
1640       LDIR
1650       LD   A,(IX+MASKD)
1660 ENDBYT LD  B,A
1670       CPL
1680       AND  (HL)
1690       LD   C,A
1700       EX   DE,HL
1710       LD   A,B
1720       AND  (HL)
1730       OR   C
1740       LD   (HL),A
1750       EX   DE,HL
1760       RET
1770 INITW LD   A,(IX+XDESTR)
1780       CP   (IX+XSINS)
1790       JR   Z,ERROR
1800       JR   C,ERROR
1810       LD   A,(IX+YSUP)
1820       CP   (IX+YINF)
1830       JR   Z,ERROR
1840       JR   C,ERROR
1850       LD   C,(IX+XSINS)
1860       LD   B,(IX+YSUP)
1870       CALL INDPIX
1880       PUSH HL
1890       LD   (IX+BITS),A
1900       LD   C,(IX+RIGHTX)
1910       LD   B,(IX+YSUP)
1920       CALL INDPIX
1930       LD   (IX+BITD),A
1940       POP  BC
1950       LD   A,C
1960       CP   L
1970       JR   Z,ERROR
1980 MSKSIN LD  B,(IX+BITS)
1990       LD   A,0
2000 L1    SCF
2010       RRA
2020       DJNZ L1
2030       LD   (IX+MASKS),A
2040 MSKDES LD  B(IX+BITD)
2050       LD   A,255
2060 L2    AND  A
2070       RRA
2080       DJNZ L2
2090       LD   (IX+MASKD),A
2100       RET
2110 ERROR RST  8
2120       DEFB 25
```

Programma dimostrativo BASIC

```
  5 CLEAR 32767
 10 LOAD ""CODE
 20 LET WT=45056
 30 LET TABELLAFIN=45060: REM B004 HEX
 40 LET XSINS=0
 50 LET YSUP=1
 60 LET XDESTR=2
 70 LET YINF=3
 75 LET DIR=7
 80 LET SCROLL=45322
 90 LET INIT=45312
100 BORDER 6
110 PAPER 4: INK 2
120 CLS
180 POKE WT,4: POKE WT+1,176
190 REM WT & WT+1 ORA CONTENGONO L'INDIRIZZO 45060
    IN FORMATO BASSO,ALTO
200 POKE TABELLAFIN+XSINS,5
210 POKE TABELLAFIN+YSUP,80
220 POKE TABELLAFIN+XDESTR,250
230 POKE TABELLAFIN+YINF,35
240 RANDOMIZE USR INIT
250 FOR Y=0 TO 175
260 PLOT 0,Y
270 DRAW 255,0
280 NEXT Y
290 POKE TABELLAFIN+DIR,2
300 FOR Y=0 TO 45
310 RANDOMIZE USR SCROLL
320 NEXT Y
400 PRINT AT 12,20;"HELLO";
410 POKE TABELLAFIN+DIR,0
420 FOR I=1 TO 130
430 RANDOMIZE USR SCROLL
440 NEXT I
470 POKE TABELLAFIN+DIR,3
480 FOR I=1 TO 35
490 RANDOMIZE USR SCROLL
500 NEXT I
510 POKE TABELLAFIN+DIR,1
520 FOR I=1 TO 130
530 RANDOMIZE USR SCROLL
540 NEXT I
550 POKE TABELLAFIN+DIR,2
560 FOR I=1 TO 35
570 RANDOMIZE USR SCROLL
580 NEXT I
590 GO TO 410
999 STOP
```

Caricamento Codice Macchina

```
100>LET a=45312
110 FOR l=1000 TO 1500 STEP 10
120 LET s=0
130 READ b
140 POKE a,b
150 LET s=s+b
160 NEXT a
170 READ b
180 IF s<>b THEN PRINT "ERRORE ALLA LINEA ";L: STOP
190 NEXT l
1000 DATA 42,0,176,229,221,255,205,69,1167
1010 DATA 178,201,42,0,176,229,221,225,1272
1020 DATA 221,203,7,78,245,204,29,177,1164
1030 DATA 241,196,184,177,201,221,126,1,1347
1040 DATA 221,119,6,221,203,7,70,40,887
1050 DATA 8,221,70,8,221,126,9,24,687
1060 DATA 6,221,70,9,221,126,8,50,711
1070 DATA 3,176,120,50,2,176,221,78,826
1080 DATA 0,221,70,6,205,170,34,235,941
1090 DATA 221,78,2,221,70,6,205,170,973
1100 DATA 34,221,203,7,70,40,1,235,811
1110 DATA 205,103,177,221,126,3,221,190,1246
1120 DATA 6,200,221,53,6,24,215,58,783
1130 DATA 2,176,71,166,79,120,47,166,827
1140 DATA 221,203,7,70,40,4,203,47,795
1150 DATA 24,2,203,39,245,177,119,221,1030
1160 DATA 203,7,70,40,3,35,24,1,383
1170 DATA 43,125,187,40,16,241,221,203,1076
1180 DATA 7,70,40,4,203,30,24,2,380
1190 DATA 203,22,245,24,226,78,58,3,859
1200 DATA 176,161,71,241,221,203,7,70,1150
1210 DATA 4,40,203,25,24,2,203,17,518
1220 DATA 58,3,176,47,161,176,119,201,941
1230 DATA 221,78,0,221,203,7,70,40,840
1240 DATA 5,221,70,3,24,3,221,70,617
1250 DATA 1,221,112,6,205,170,34,229,978
1260 DATA 229,221,78,2,221,70,6,205,1032
1270 DATA 170,34,209,167,237,82,125,61,1085
1280 DATA 221,119,10,221,203,7,70,40,891
1290 DATA 5,221,52,6,24,3,221,53,585
1300 DATA 6,221,78,0,221,70,6,205,807
1310 DATA 170,34,209,229,205,40,178,221,1286
1320 DATA 126,6,221,203,7,70,40,5,678
1330 DATA 221,190,1,24,3,221,190,3,853
1340 DATA 32,209,225,221,126,8,166,119,1106
1350 DATA 221,70,10,62,0,35,119,16,533
1360 DATA 252,35,221,126,9,166,119,201,1129
1370 DATA 221,126,8,205,58,178,35,19,850
1380 DATA 6,0,221,78,10,237,176,221,949
1390 DATA 126,9,71,47,166,79,235,120,853
1400 DATA 166,177,119,235,201,221,126,2,1247
1410 DATA 221,190,0,40,67,56,65,221,860
1420 DATA 126,1,221,190,3,40,57,56,694
1430 DATA 55,221,78,0,221,70,1,205,851
1440 DATA 170,34,229,221,119,4,221,78,1076
1450 DATA 2,221,70,1,205,170,34,221,924
1460 DATA 119,5,193,121,189,40,25,221,913
1470 DATA 70,4,62,0,55,31,16,252,490
1480 DATA 221,119,8,221,70,5,62,255,961
1490 DATA 167,31,16,252,221,119,9,201,1016
1500 DATA 207,25,0,0,0,0,0,0,232
```

Logica divertente

Reverse è un piccolo "rompicapo", col quale si apprendono alcuni importanti concetti

Il programma genera in modo casuale un elenco di numeri, che deve essere poi riordinato. La disposizione dei numeri può essere cambiata solamente invertendo l'ordine di gruppi specifici all'interno della lista. Supponiamo che il giocatore chieda al computer una serie di nove numeri e che il programma generi la seguente lista di numeri casuali:

2 8 4 7 1 5 6 9 3

Se alla successiva domanda che appare sullo schermo: "Scambio?" il giocatore risponde "5", allora l'ordine dei primi 5 numeri verrà invertito e la lista diverrà la seguente:

1 7 4 8 2 5 6 9 3

In teoria, la soluzione di questo "rompicapo" non dovrebbe richiedere né un tempo né un impegno eccessivi e ci si aspetterà di poter facilmente risolvere il problema usando un algoritmo. In pratica, la formulazione di tale algoritmo non si dimostra affatto facile! Supponiamo che la lista contenga n numeri. L'algoritmo più ovvio sarà:

- Trovare il numero più grande della lista e scambiare tutti i numeri fino alla sua posizione. Il numero più grande si trova ora all'estremità sinistra della lista.
- Scambiare tutti gli n numeri in modo che il numero più grande venga a trovarsi nella posizione voluta, cioè al margine destro della lista. Per questa operazione sono stati necessari due scambi.
- Trovare il secondo numero più grande e ripetere l'intera procedura. Per portare questo secondo numero nella posizione voluta saranno necessari n−1 scambi.
- Ripetere la procedura fino ad ottenere un riordinamento completo della lista.

Questo algoritmo risolve "sempre" il rompicapo in 2n−3 movimenti: è possibile ottenere un riordinamento completo con un minor numero di passaggi. Per dimostrare come una strategia di "verifica in avanti" possa ridurre il numero di movimenti, esaminiamo l'esempio in figura. L'algoritmo richiederebbe 7 passaggi (2×5−3), ma un giocatore esperto giunge alla soluzione in soli 4 passaggi.
Questo programma è un semplice esempio della vasta serie di "giochi di scambio", creati e sperimentati dagli appassionati. È possibile sviluppare algoritmi per lo scambio su una lista a partire da entrambe le sue estremità, o per il riordino di una griglia anziché di una sola linea. Oppure, al posto dei numeri, si possono usare blocchi colorati. Infine, si potrebbe inserire un algoritmo di "soccorso", per aiutare i giocatori in difficoltà.

REVERSE

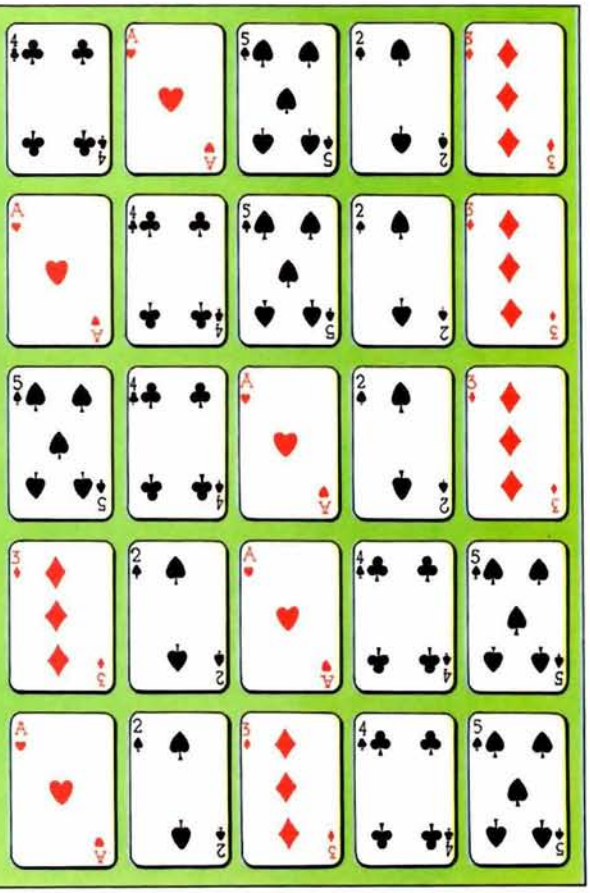
```
 20 DIM a(20)
 30 CLS: PRINT "Reverse !"
 40 INPUT "Quanti numeri ? ";n
 50 IF n<0 OR n>20 OR n<>INT n THEN GO TO 30
 60 REM Crea la lista
 70 FOR i=1 TO n: LET a(i)=i: NEXT i
 80 RANDOMIZE
 90 FOR i=1 TO n
100 LET r=INT(RND*n+1)
110 LET x=a(r): LET a(r)=a(i): LET a(i)=x
120 NEXT i
130 LET t=1
135 REM Visualizza la lista
140 CLS: PRINT "Passaggio ";t;":La lista e':": PRINT
150 FOR i=1 TO n: PRINT a(i);" ";: NEXT i
152 REM Verifica del risultato
154 LET i=1
156 IF a(i)=i THEN LET i=i+1: IF i<=n THEN
    GO TO 156
158 IF i>n THEN GO TO 230
159 REM Esegue un tentativo
160 PRINT: PRINT: INPUT "Scambio ?";r
170 IF r<>INT r OR r<0 OR r>n THEN GO TO 140
175 REM Scambia r
180 LET t=t+1
190 FOR i=1 TO INT (r/2)
200 LET x=a(i): LET a(i)=a(r-i+1): LET a(r-i+1)=X
210 NEXT i
220 GO TO 140
230 REM Lista riordinata!
240 PRINT: PRINT: PRINT "Hai terminato in  ";t;"
    passaggi"
250 PRINT: INPUT "Giochi ancora (s/n) ? ";a$
260 IF a$="S" OR a$="s" THEN RUN
270 CLS: STOP
```

BASIC & BASIC

Sul Commodore 64 e sul Vic-20, sostituire RANDOMIZE con XX=RND(−TI), RND∗N con RND(1)∗N e CLS con PRINT CHR$(147).
Sul Micro BBC, cancellare la linea 80 e sostituire R=INT(RND∗N+1) con R=RND(N).
Sull'Oric-1 e sull'Atmos, cancellare la linea 80 e sostituire RND∗N con RND(1)∗N

SCAMBIO DOPO SCAMBIO

L'obiettivo di *Reverse* è il riordino di una lista, invertendo in successione alcuni gruppi di elementi al suo interno. Ciascun gruppo è composto da un numero di elementi indicato dal giocatore ed i gruppi devono essere invertiti partendo sempre dall'elemento più a sinistra. Nell'esempio in figura la sequenza dei gruppi scambiati è 2-3-5-3 e si sviluppa nel seguente modo: "Scambia tra loro le prime due carte sulla sinistra, poi le prime tre... e così via"



La Sharp

Una grande ditta giapponese, produttrice di una vasta gamma di apparecchiature

La Sharp Corporation ha sempre partecipato attivamente alle principali innovazioni tecnologiche, ma il suo primo prodotto di successo è stato un articolo piuttosto semplice e curioso: il lapis a pulsante "Ever Sharp". Il suo ideatore, Tokuji Hayakawa, fondò la Sharp nel 1915 per produrre la propria invenzione ma, già nel 1925, la ditta si occupava di elettronica, fabbricando radio a galena. Il primo ingresso ufficiale, nel mercato mondiale dell'elettronica, avvenne dopo la Seconda Guerra Mondiale, quando la Sharp si dedicò alla costruzione di televisori e di altri elettrodomestici. Nel 1960 la Sharp estese i propri interessi al mercato delle apparecchiature per ufficio, producendo una serie di calcolatrici da tavolo.

Oggi la Sharp è un "colosso" multinazionale ed è suddivisa in sei gruppi di produzione, con 34 impianti di fabbricazione in 30 paesi, oltre che in Giappone.

Uno dei primi computer della Sharp è stato l'MZ80K, lanciato nel 1981, al quale sono seguiti l'MZ380A e l'MZ380B, tutti dotati di monitor e unità a cassette incorporati. Questi apparecchi venduti come computer gestionali, incontrarono un discreto successo anche nel settore degli home computer. Un originale slogan pubblicitario li definiva "macchine pulite", per sottolineare la voluta mancanza di un linguaggio "residente" in ROM.

Questa scelta presentava il vantaggio di poter caricare in memoria da una cassetta vari linguaggi, compreso il sistema operativo CP/M della Digital Research.

Agli inizi del 1983, la Sharp intraprese l'esportazione dell'intera gamma dei suoi home computer e apparecchi gestionali sul mercato europeo, introducendo anche il computer gestionale MZ-3541 ed il computer tascabile PC-1500. Il successo di quest'ultimo ha spinto la Sharp a lanciare un secondo computer tascabile, il PC-1251.

Il primo home computer Sharp, per il mercato europeo, fu l'MZ-711, una versione appositamente modificata della serie giapponese MZ-700, in cui venne eliminato l'immenso set di caratteri giapponesi presenti nella versione originale, destinando l'area di memoria rimasta libera ad una serie di facilitazioni grafiche addizionali. La configurazione standard dell'MZ-711 è dotata di un registratore di dati incorporato e di un apposito alloggio per la stampante/plotter (opzionale).

Nel maggio 1984, la Sharp ha presentato il PC-1500A, una versione potenziata del PC-1500. Il nuovo modello possiede 8,9 Kbyte di RAM, espandibili fino a 24 Kbyte. Nell'autunno del 1984, la Sharp ha lanciato il tascabile PC-1350, dotato di capacità grafiche e di un display di 4 linee, ed il sistema "Sharpwriter", nato da una combinazione tra la macchina da scrivere elettrica Sharp ZX-401 ed il microcomputer MZ-3541: la stampante viene qui usata come tastiera/stampante, collegata al computer tramite un'interfaccia RS232.

I progetti della Compagnia mirano allo sviluppo di apparecchi di tipo tascabile, che stanno incontrando un crescente successo di mercato. Tuttavia, ciò non significa che la Sharp abbia intenzione di trascurare il settore degli home computer che, grazie agli apparecchi della serie MZ-700, le ha fruttato ingenti guadagni. Da qualche tempo è stata creata una sezione dedicata specificatamente alle apparecchiature da ufficio e ciò fa supporre che la ditta giapponese intenda estendere la propria presenza anche in quest'area del mercato.

Infine, la Sharp sta realizzando un nuovo grosso impianto, in Gran Bretagna, per l'assemblaggio di videoregistratori, che dovrebbe produrre circa 60000 apparecchi per il mercato europeo.

La Sharp non sembra aver pianificato alcuna particolare strategia per quanto riguarda la propria partecipazione al lancio del sistema MSX, progettato da alcune compagnie giapponesi ed americane.

Il Gruppo "Industrial Instruments"

Le nuove scoperte tecnologiche realizzate dal centro ingegneristico della Sharp vengono poi sperimentate ed applicate nell'impianto Industrial Instruments Group (IIG), situato a Yamato-Koriyama-shi, in Giappone. Qui vengono progettati quasi tutti gli apparecchi della Sharp (compresi i personal computer e le calcolatrici tascabili)



FINANCIAL MICRO TIMES

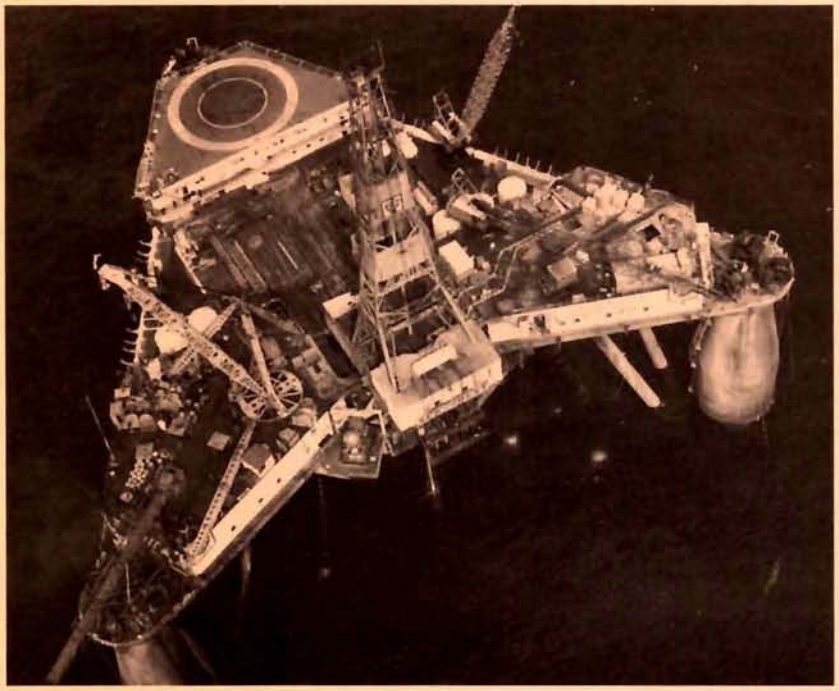
I "business games"



Nel mondo degli affari...

Molte persone aspirano a poter amministrare da soli i propri affari, ma, anche nel caso di piccole imprese, i problemi che ne derivano sono talvolta notevoli. Le difficoltà aumentano ancor più quando si tratta della gestione finanziaria di una società. Queste pagine comunque, non sono dedicate al vero software commerciale, ma ad una serie di giochi di "simulazione finanziaria", che riproducono abbastanza realisticamente problemi e difficoltà tipici delle imprese di maggiori dimensioni. Il più noto di questi giochi è *Monopoly*, una versione modificata del famoso gioco da tavolo. I giochi finanziari si adattano perfettamente ai computer, perché non implicano una vera perdita di beni o denaro e il computer può eseguire tutti i calcoli necessari.
Birrerie, linee aeree, aziende agricole, case automobilistiche, sono solo alcuni esempi delle società simulate da questi giochi.
Ciascun tipo di società deve affrontare specifiche difficoltà: ad esempio, il costo del carburante è molto importante per una compagnia aerea, mentre problemi come il crescere dell'inflazione o gli alti tassi d'interesse sono comuni a tutte.
Indipendentemente dal tipo di gioco scelto, i programmi di simulazione commerciale funzionano più o meno allo stesso modo. Il giocatore "si trasforma" in direttore di una società, e, all'inizio del gioco, entra in possesso di una certa quantità di denaro. Nel programma *Corn Cropper*, nel quale il giocatore dirige una fattoria, la cassa iniziale ammonta appena a 120 milioni, ma se si è a capo di una compagnia petrolifera, come nel gioco *Dallas*, 2.000 miliardi saranno appena sufficienti per cominciare.
All'inizio il giocatore deve investire una parte del capitale in "beni immobili": aeroplani, fabbriche d'automobili, terreni, ecc., mediante i quali potrà avviare la produzione di beni e servizi ed iniziare ad accumulare un certo reddito. Ovviamente, lo sviluppo del gioco è piuttosto complesso ed il successo o il fallimento della compagnia dipendono da molti fattori.
Il realismo delle simulazioni varia da programma a programma. Alcuni giochi, come *Dallas*, puntano quasi esclusivamente al divertimento, ma altri sono molto più realistici: il programma *Corn Cropper* tiene conto degli agenti atmosferici (sole, pioggia, ecc.), delle condizioni ideali per la coltivazione del grano, dei salari dei dipendenti e del costo degli attrezzi agricoli, tutti fattori importanti per l'amministrazione di una moderna azienda agricola.
I giocatori che aspirassero a qualcosa di più impegnativo della conduzione di un'attività commerciale possono cimentarsi nel governo di una nazione! "1984" è un gioco che riproduce l'economia della Gran Bretagna ed assegna al giocatore la carica di Primo Ministro.
Per simulare l'economia nazionale, gli economisti si servono di un "modello": non un modello fisico, ma un complesso diagramma che mostra come tutte le componenti economiche di una nazione interagiscano tra loro. Quello del programma "1984" è un modello semplificato e comprende solo 5 sezioni: il governo, il sistema bancario, la popolazione, l'apparato industriale ed il resto del mondo. Nonostante ciò, esistono almeno 30 interconnessioni tra le varie sezioni: il governo finanzia le industrie, la popolazione paga le tasse al governo e così via.
Il gioco *AMODEL*, che "gira" su un computer Sperry 1100 di grandi dimensioni, adotta un modello economico reale (adoperato dal governo britannico per fare previsioni sull'economia nazionale). Come è facile supporre, questo modello è assai più complesso del precedente: per formulare le previsioni, il programma elabora circa 1100 diverse variabili economiche: il tasso d'inflazione, la disoccupazione ecc., calcolate in base a dati effettivi della situazione economica degli ultimi 10 anni. Questi dati occupano, da soli, circa 250 Kbyte di memoria e quando il programma gira a pieno regime sono necessari circa 8 Megabyte. Questo sofisticato modello viene usato, soprattutto, per prevedere il tasso di disoccupazione, il livello delle importazioni e delle esportazioni, il tasso di cambio del dollaro e molti altri indici economici. Per dare un'idea della sua complessità, ricordiamo che per calcolare il tasso d'inflazione dell'anno successivo sono necessari due o tre minuti di elaborazione: può sembrare un tempo ragionevole, ma va tenuto presente che il microprocessore elabora circa 10 milioni di istruzioni in codice macchina al secondo! Per la stampa della previsione occorrono dai 15 ai 20 minuti.
I giochi di simulazione per computer offrono una vasta gamma di opportunità, mettendo a dura prova l'abilità dei giocatori.

FINANCIAL MICRO TIMES



ASSOCIATED PRESS

DALLAS

***Dallas* è un gioco d'affari in cui il giocatore è posto a capo di una compagnia petrolifera. Iniziando il gioco con una cassa di 2.000 miliardi di lire, egli deve riuscire ad accumulare una cifra doppia per sconfiggere la sua più agguerrita rivale, la Euing Associates (il nome è volutamente diverso da quello della nota serie televisiva).**

Sullo schermo compare una mappa quadrettata del Texas e, via via che il gioco procede, il giocatore cerca di vincere le gare d'appalto per la "concessione" in ciascun quadratino. Ottenuta la concessione, il giocatore può effettuare una perforazione di prova per individuare un giacimento petrolifero. Infine, se il tentativo ha successo, può impiantare un vero e proprio pozzo petrolifero ed iniziare l'estrazione.
Se la compagnia si espone troppo dal punto di vista finanziario, ad esempio comprando troppe concessioni prima di poter contare su un sicuro rientro di capitali, il giocatore è costretto a chiedere un prestito alla banca ed a pagarne gli interessi. Se però il prestito supera i 40 miliardi di lire, la compagnia corre il rischio di essere acquistata dalla Euing Associates.
Questo gioco, prodotto dalla Cases Computer Simulations, è disponibile per micro quali lo Spectrum, il Micro BBC, l'Oric e l'Electron.

AIRLINE

In questa simulazione, il giocatore è il presidente della L-AIR, una compagnia aerea privata con capitale iniziale di 7,5 miliardi di lire. Se in capo a sette anni (simulati, s'intende!), egli riesce a decuplicare il capitale iniziale, può acquistare la 'British Airways' (nome della nota compagnia di bandiera britannica) e vincere così il gioco.

All'inizio di ogni anno finanziario, il giocatore deve decidere quanti nuovi aerei acquistare, in base a previsioni sul numero di passeggeri: il primo anno non ha denaro sufficiente neppure per un aereo (che costa 25 miliardi), quindi lo deve prendere in affitto. Negli anni successivi, il giocatore deve decidere, in base ai costi di nolo ed ai tassi d'interesse, se gli convenga affittare o comprare un aereo.
Deve anche valutare di quanti elementi sarà composto l'equipaggio ed il livello di comodità dell'apparecchio. Se le scelte si rivelano troppo basse, la compagnia mancherà di personale ed alcuni voli dovranno essere soppressi. In caso contrario, il giocatore corre il rischio di spendere troppo denaro. Di tanto in tanto, una serie di messaggi telex attraversa lo schermo del computer, tenendo informata la compagnia sugli eventuali aumenti di prezzo del carburante o, ad esempio, sull'ottenimento di una licenza per una nuova rotta di navigazione.
Alla fine dell'anno un bilancio finanziario ed una relazione del presidente permettono di conoscere la situazione economica complessiva della L-AIR. Se le perdite del primo anno superano i 25 miliardi di lire, la compagnia viene liquidata.
Anche il gioco *Airline* viene prodotto dalla Cases Computer Simulations in versioni per Micro BBC, Spectrum, Oric e Electron.



ASSOCIATED PRESS

FINANCIAL MICRO TIMES

1984



ASSOCIATED PRESS

In questo gioco, il giocatore interpreta il ruolo di Primo Ministro (britannico) nell'anno 1984. Il suo obiettivo, è quello di mantenere la carica il più a lungo possibile e tale risultato si raggiunge tentando di controllare l'intera economia del paese, pur senza perdere popolarità.

Come primo atto del mandato, il giocatore deve prendere importanti decisioni finanziarie, basandosi su 8 indicatori economici (il tasso d'inflazione, la disoccupazione, e via dicendo). All'inizio di ogni anno, un grafico mostra l'andamento dei vari indici nell'anno precedente. La prima decisione riguarda la definizione del tasso minimo d'interesse bancario, che determina il tasso d'interesse per l'anno in corso. Poi il primo ministro deve contrattare gli aumenti di paga per la pubblica amministrazione e per i dipendenti delle imprese pubbliche e private. Se gli aumenti non seguono una ben precisa logica economica, al giocatore viene richiesto di rimettere il mandato!
Quindi, il giocatore deve decidere quanto assegnare ai vari ministeri e dicasteri governativi e, infine, presentare le proprie previsioni di bilancio, relative anche a modifiche o aumenti di tassazione. Alla fine di ogni anno del mandato quinquennale, il programma effettua un sondaggio d'opinione per verificare la popolarità delle scelte fatte, che contribuisce in modo determinante ad una conferma della nomina, decretata alla fine del gioco.
La Incentive Software produce due versioni del gioco *1984*: per lo Spectrum ed il Micro BBC.

CORN CROPPER

Prodotto dalla Cases Computer Simulations per lo Spectrum, il Micro BBC e l'Electron, il gioco *Corn Cropper* simula la conduzione di una grande azienda agricola per la produzione di grano.

Inizialmente, al giocatore viene assegnato un capitale di 120 milioni di lire col quale amministrare una fattoria per un periodo di 5 anni (non si tratta di tempo reale, ovviamente!). Per vincere, alla scadenza del contratto, il capitale dovrà essere di almeno 600 milioni.
All'inizio, l'azienda dispone di 30 acri di terra incolta.
Il giocatore dovrà quindi acquistare sementi, affittare un trattore, assumere dei lavoratori e versare un certo canone per l'irrigazione, se la pioggia non è sufficiente.
All'inizio di ciascun mese il programma fornirà alcune previsioni metereologiche (sulla temperatura e sulle precipitazioni previste), utili per decidere se e quando seminare.
Col trascorrere dei mesi, il giocatore può consultare un "rapporto sullo stato del raccolto", una specie di calendario che lo informa se i campi seminati sono pronti per la mietitura. Per questa operazione è necessario assumere manodopera in più, affinché il raccolto avvenga nel minor tempo possibile.
Nonostante le precauzioni prese, il raccolto può correre alcuni rischi imprevisti: può essere mangiato dai topi o attaccato dagli insetti oppure danneggiato dal gelo.
Per aumentare la produzione, si può ricorrere all'uso di fertilizzanti, ma il momento per farlo va scelto con grande cura.
Via via che il gioco procede, il giocatore deve cercare di accumulare denaro, per investirlo nell'acquisto di altri terreni e di nuove sementi.
È molto difficile accrescere il capitale oltre i 250 milioni, anche perché la banca non consente di andare "in rosso". Come ultima risorsa il giocatore potrà vendere una parte dei terreni dell'azienda per aumentare la propria liquidità.



ASSOCIATED PRESS



Obiettivo: la "poltrona del Presidente"

Questi "giochi d'affari" per microcomputer usano tecniche di simulazione piuttosto sofisticate e permettono al giocatore di assumere vari ruoli, quali quello di presidente di una compagnia aerea, di una società petrolifera o di una azienda agricola o addirittura di Primo Ministro, al numero 10 di Downing Street. Il loro realismo dipende da elementi di controllo introdotti nel sistema e dalla complessità dei modelli adottati

IAN McKINNELL

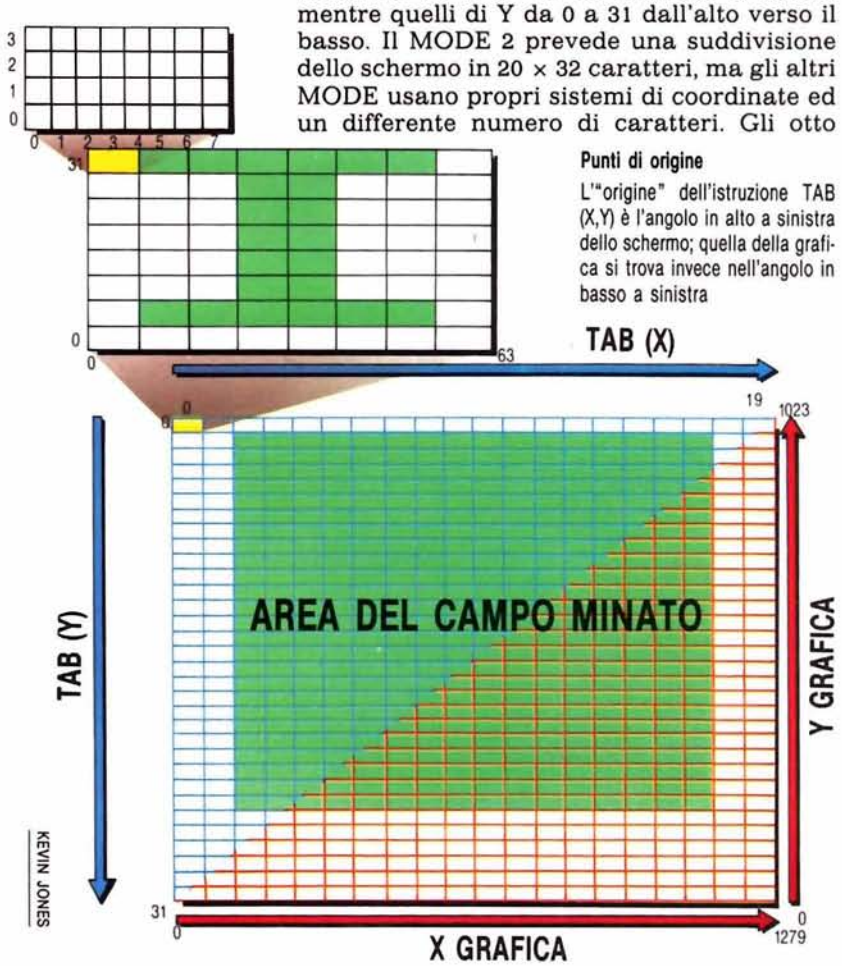
Campo visuale

La seconda parte di un gioco grafico per i computer Acorn

Nella prima parte del gioco, usando il MODE 5 del BBC, abbiamo definito un'area, chiamata "campo minato", al cui interno si svolgerà il gioco. Dopo aver creato l'immagine delle mine, del rivelatore e dell'assistente, abbiamo sviluppato le procedure per la disposizione casuale delle mine all'interno del campo minato e collocato il rivelatore e l'assistente nelle loro posizioni di partenza. Per rendere lo scenario del gioco più interessante, possiamo usare i comandi di alta risoluzione MOVE e DRAW per tracciare il perimetro del campo.
Sui modelli BBC e sull'Electron, l'uso combinato di grafica ad alta risoluzione e di caratteri a bassa risoluzione presenta alcuni problemi, poiché i due modi grafici si basano su sistemi di coordinate diversi. Abbiamo già parlato dello schermo a bassa risoluzione, quando lo abbiamo usato per disporvi le varie figure del gioco con il comando PRINT TAB(X,Y). In questo sistema, l'"origine" (cioè il punto con coordinate X e Y uguali a 0) è l'angolo in alto a sinistra ed i valori di X crescono da 0 a 19 da sinistra a destra mentre quelli di Y da 0 a 31 dall'alto verso il basso. Il MODE 2 prevede una suddivisione dello schermo in 20 × 32 caratteri, ma gli altri MODE usano propri sistemi di coordinate ed un differente numero di caratteri. Gli otto MODE del BBC e dell'Electron offrono tre diversi livelli di risoluzione (640 × 256, 320 × 256 e 160 × 256), tutti basati su uno stesso sistema di coordinate. Quindi le immagini grafiche progettate per uno degli otto modi si possono usare anche con tutti gli altri.
Il sistema di coordinate BBC/Electron tratta lo schermo come se avesse una risoluzione pari a 1280 × 1024. Questa è naturalmente la massima risoluzione dei due apparecchi che sembra sia stata progettata per i futuri sviluppi del BBC. Le coordinate sono espresse come numeri compresi tra 0 e 1279 in senso orizzontale e tra 0 e 1023 in senso verticale. Il BASIC del BBC le converte automaticamente nei valori adatti alla risoluzione di schermo adoperata, tra le tre disponibili.
Come abbiamo già detto, la fusione di grafica e caratteri presenta alcune difficoltà: l'origine del sistema di coordinate grafiche si trova in basso a sinistra sullo schermo, mentre quella del sistema per i caratteri è posta in alto a sinistra.
La risoluzione di 160 × 256 pixel del MODE 5 è in stretta relazione col numero di caratteri visualizzabili. Siccome ogni carattere viene costruito all'interno di una griglia di 8 × 8 pixel e lo schermo del MODE 5 è di 20 × 32 caratteri, la lunghezza di una riga pixel dovrà essere di 8 × 20 = 160 pixel e quella di una colonna pixel di 32 × 8 = 256 pixel. Si rammenti che un pixel è la più piccola area luminosa che può essere accesa o spenta in modo indipendente sullo schermo. Nel sistema ad alta risoluzione, ci sono 1.280 coordinate sull'asse orizzontale X e 1.024 sull'asse verticale Y: dividendo adesso il primo valore per il numero dei pixel di una riga pixel otteniamo 1280/160 = 8, mentre la divisione del secondo per il numero dei pixel di una colonna pixel dà 1024/256 = 4. Ciò significa che a ciascun pixel corrispondono 8 coordinate sull'asse X e 4 sull'asse Y e che, per "accenderlo" (con PLOT) è sufficiente indicare un'unica coppia di coordinate (X,Y) delle molte possibili. La figura mostra come si possa usare un'intera gamma di coordinate, per "accendere" o "spegnere" un particolare pixel. Le coordinate (7,3), (0,0) o (5,2), per esempio, si riferiscono tutte ad uno stesso pixel.
Vediamo adesso come sia possibile collegare la posizione dei caratteri alle coordinate, per l'alta risoluzione. Se ogni pixel dell'asse orizzontale equivale a 8 coordinate o "unità" X, allora la larghezza di un carattere è di 64 unità. Analogamente, se ogni pixel dell'asse verticale equivale a 4 coordinate o unità Y, l'altezza di un carattere è pari a 32 unità. Pertanto, possiamo calcolare la posizione di

Pixel... in campo!

Nel MODE 5, la griglia per la creazione dei caratteri è un rettangolo la cui base ed altezza sono, rispettivamente, di 64 e di 32 unità grafiche di alta risoluzione. Quindi, ciascun pixel della griglia misura 8 x 4 unità. I comandi PLOT o DRAW, relativi ad una qualsiasi di queste unità, accendono l'intero pixel: PLOT 7,3 sarà perciò equivalente a PLOT 0,0 o PLOT 5,2 ecc...



Punti di origine

L'"origine" dell'istruzione TAB (X,Y) è l'angolo in alto a sinistra dello schermo; quella della grafica si trova invece nell'angolo in basso a sinistra

qualsiasi punto sullo schermo in termini di coordinate per l'alta risoluzione, e usare poi i comandi MOVE e DRAW.
In base a quanto appena detto, le coordinate dell'angolo inferiore sinistro e di quello superiore destro del campo minato risultano rispettivamente: 120,188 e 1152,992. Tutte le altre coordinate del perimetro fanno riferimento a questi due punti.
La procedura seguente disegna un bordo attorno al perimetro del campo minato. GCOL 0,1 seleziona il colore logico da usare per la grafica: il primo numero definisce il tipo di tracciamento (discusso più avanti), mentre il secondo seleziona il colore. Nel MODE 5, il colore logico 1 è normalmente il rosso. Il comando MOVE sposta il cursore grafico dall'origine all'angolo in basso a sinistra del bordo. I successivi comandi DRAW tracciano linee rette sullo schermo, dalla posizione "corrente" del cursore al punto indicato nel comando.

```
2470 DEF PROCdisegnabordo
2480 GCOL 0,1
2490 MOVE 120,188
2500 DRAW 120,992
2510 DRAW 1152,992
2520 DRAW 1152,188
2530 DRAW 120,188
2540 ENDPROC
```

Il temporizzatore interno

Il BBC e l'Electron possiedono un temporizzatore interno a cui si può facilmente accedere, dal BASIC, tramite la variabile riservata TIME. Il contenuto di TIME (il tempo trascorso dall'ultimo azzeramento di detta variabile) è espresso in centesimi di secondo. Pertanto, la procedura "marcatempo" visualizza la parola "Tempo", il suo valore iniziale ed azzera la variabile TIME. Questa procedura viene chiamata durante la routine di "preparazione" e serve per far partire il marcatempo del gioco.

```
2640 DEF PROCmarcatempo
2650 PRINT TAB(2,27)"Tempo     02:00"
2660 TIME=0
2670 ENDPROC
```

Durante il ciclo principale del programma, il tempo che compare sullo schermo deve essere costantemente aggiornato. Per visualizzare il tempo in secondi occorre dividere il contenuto della variabile TIME per 100, convertire il risultato in secondi ed infine riportarlo sullo schermo. I comandi DIV e MOD, del BASIC BBC, consentono di visualizzare il tempo anche in minuti e secondi: TIME DIV 100 fornisce il numero dei secondi, sottoforma di numero intero, mentre (TIME DIV 100) MOD 60 calcola i secondi da 0 a 59, per poi ripartire da 0. Infatti, il comando MOD 60 fornisce la cifra di riporto ottenuta dividendo il numero totale dei secondi per 60. Quindi, ad esempio, (63/60) dà come risultato 1 con riporto di 3. (63/60) MOD 60 fornisce quindi il valore 3. I minuti si possono facilmente isolare e visualizzare usando l'istruzione (TIME DIV 6000) MOD 60. La seguente procedura viene usata per conteggiare il tempo durante il gioco:

```
2900 DEF PROCaggiornatempo
2910 sec$=STR$(((12100-TIME)DIV 100)MOD 60)
2920 min$=STR$(((12100-TIME)DIV 6000)MOD 60)
2930 REM **AGGIUNGE ZERI DAVANTI AL NUMERO**
2940 sec$=LEFT$(zero$, 2-LEN(sec$))+sec$
2950 min$=LEFT$(zero$, 2-LEN(min$))+min$
2960 time$=min$+":"+sec$
2970 PRINT TAB(11,27);time$
2980 ENDPROC
```

Con questa procedura abbiamo fatto un ulteriore passo avanti: il tempo, oltre ad essere diviso in minuti e secondi, viene in effetti conteggiato "a scalare" (da 2 minuti fino a 0). Inoltre, una breve routine fa sì che la rappresentazione dei minuti e dei secondi sia sempre a due cifre, inserendo, quando è necessario, degli "0" non significativi.
Le due procedure che seguono permettono di completare lo scenario del gioco. Inizialmente, il giocatore dispone di quattro "vite": sull'ultima riga dello schermo dovranno comparire, quindi, le "vite" ancora a disposizione (ogni vita è rappresentata da un'immagine dell'"assistente"). La variabile "conto", posta inizialmente a 1, tiene il conto, appunto, delle vite perse durante il gioco.

```
2690 PROCassistente
2700 men$=CHR$(226)+CHR$(226)+CHR$(226)
2710 conto=1
2720 COLOUR 1
2730 PRINT TAB(2,30);men$
2740 COLOUR 2
2750 ENDPROC
```

L'ultima procedura di preparazione dello schermo inizializza e visualizza i punteggi. Il valore di maxpti$, ossia del punteggio massimo, non viene qui modificato, in quanto la procedura è richiamata tutte le volte che si inizia un nuovo gioco, mentre invece il punteggio massimo va azzerato soltanto all'inizio del programma.

```
2770 PROCazzerapunteggio
2780 punti=0: punti$="00000"
2790 PRINT TAB (2,28) "Punteggio     00000"
2800 PRINT TAB (2,29) "Punteggio massimo"; maxpti$
2810 ENDPROC
```

Avendo realizzato tutte le procedure di preparazione dello schermo, possiamo ora creare una procedura, di livello più alto, che le richiami tutte. Nella fase precedente, le procedure venivano chiamate direttamente da un breve programma principale. Cancelliamo ora quelle linee per sostituirle con le seguenti:

```
1880 PROCpreparazione
1890 COLOUR 2
1900 flagfine=0
1910 PROCinizializza
1920 PROCcreafig
1940 PROCmettimine(40)
1950 PROCdisegnabordo
1960 PROCmarcatempo
1970 PROCazzerapunteggio
1980 PROCassistente
1990 PROCstampacaratteri
2000 ENDPROC
```

Possiamo adesso scrivere un breve programma principale per chiamare la procedura "preparazione" e per aggiornare il punteggio mediante un ciclo iterativo REPEAT...UNTIL. Aggiungiamo le seguenti linee al nostro programma:

```
10 REM ****PROGRAMMA DI CHIAMATA****
20 maxpti$ = "00000"
30 PROCpreparazione
40 REPEAT
50 PROCaggiornatempo
60 UNTIL TIME>12099
70 END
```

La Brother EP-44

Una macchina da scrivere elettronica, ma con molte doti in più

La Brother EP-44 pesa circa Kg 2,5 e funziona a batterie: è quindi una vera e propria "portatile". Tuttavia, un piccolo adattatore (opzionale) permette di collegarla alla corrente di rete. Sulla tastiera, oltre ai consueti tasti di una macchina da scrivere, ci sono quattro tasti di controllo per il "word processing" e sette tasti di calcolo. Ma la caratteristica più interessante dell'apparecchio è certamente il display a cristalli liquidi (LCD) che permette di visualizzare 15 caratteri di testo già compilato, prima di stamparlo.

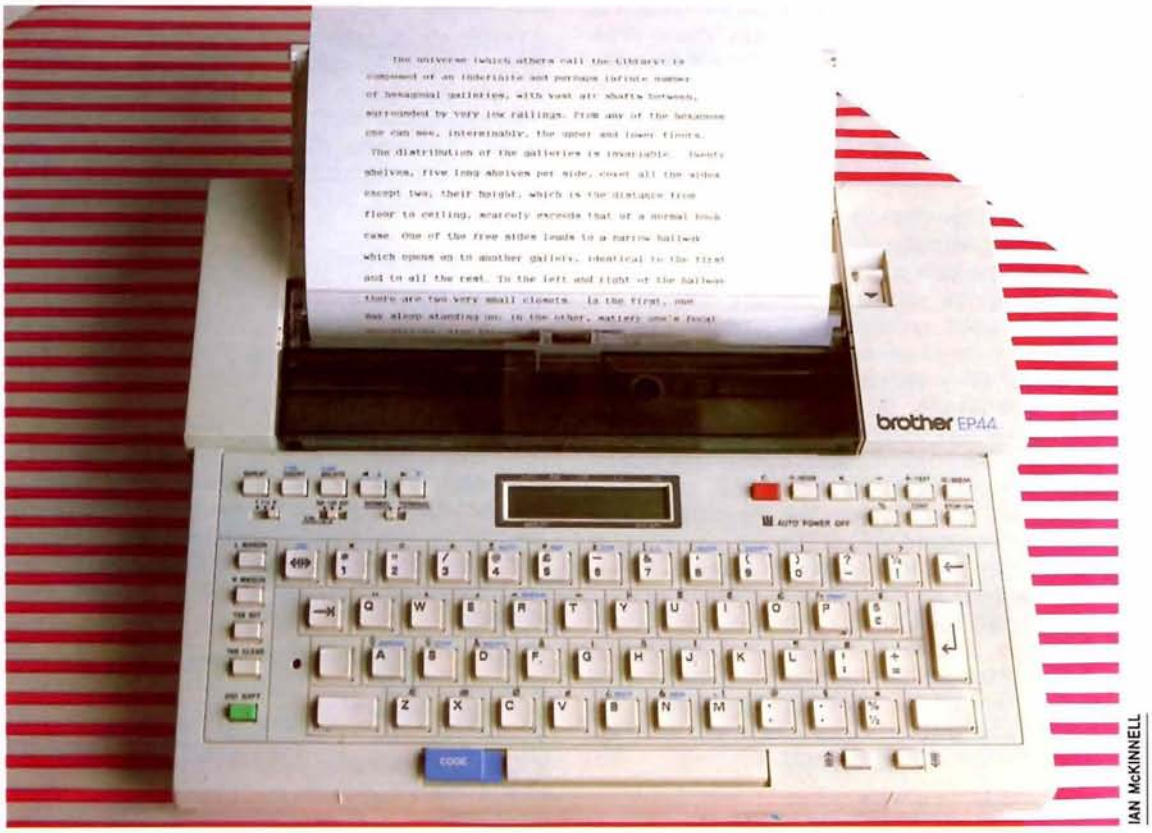
Se la EP-44 viene usata come semplice macchina da scrivere, il commutatore a slitta Print Mode Selector viene spostato sul modo DP (ossia: "Direct Print"). In questo modo il testo compare sulla "finestra" LCD e, contemporaneamente, viene stampato su carta. L'uso dei fermi di tabulazione, delle interlinee e dei margini è identico a quello di una comune macchina da scrivere. L'apparecchio possiede però un insolito sistema di trascinamento della carta: invece di ruotare a mano il rullo portacarta, l'utente preme uno dei due pulsanti che servono per muovere la carta verso l'alto o il basso. Le prestazioni della tastiera sono discrete, anche se leggermente inferiori agli standard di una macchina da scrivere elettronica. La EP-44 si serve di una stampante termica ed impiega una matrice di 24 × 18 punti per "bruciare" i caratteri sull'apposita carta termosensibile: i risultati sono di ottima qualità. Un'altra stampante della famiglia Brother, la EP-22, utilizza una matrice di dimensioni minori, a discapito della qualità, che risulta assai inferiore.

Il principale inconveniente delle stampanti termiche è la bassa velocità di stampa: nel caso della EP-44, meno di 16 caratteri per secondo. Quando la Brother viene usata come macchina da scrivere non ci sono problemi ma, quando funziona come stampante per computer la lentezza di stampa diviene significativa. Infatti, una buona stampante a matrice d'aghi è capace di stampare testi ad una velocità 10 volte superiore! L'altro inconveniente di questo sistema di stampa è che viene adoperata una speciale carta termica, costosa e fastidiosamente lucida. Il nastro termico può stampare anche su carta normale, ma i risultati sono sensibilmente inferiori.

Se il commutatore a slitta viene spostato sul modo CP ("Correction Print") il testo scritto compare sul display LCD, ma trascorre un breve intervallo di tempo prima che venga stampato. In pratica, il testo viene stampato sulla carta con un "ritardo" di quindici caratteri rispetto a quello visualizzato sul display,

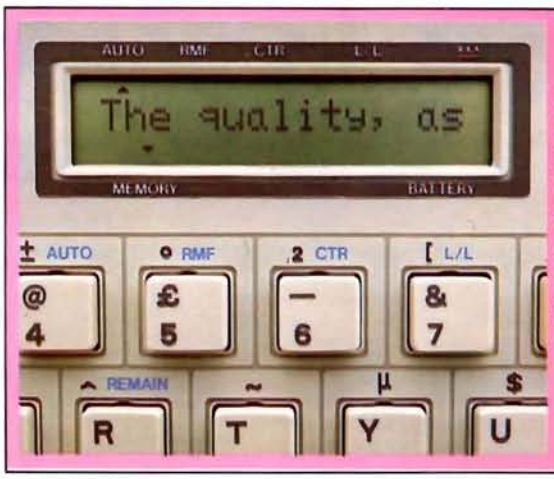
La tastiera

La tastiera, che adotta solo in parte la normale disposizione QWERTY, è composta da 51 tasti leggermente sagomati e piacevolmente scorrevoli. Ci sono due tasti, dotati di "fermo" per le maiuscole, un tasto per i simboli ed uno di funzione. I tasti dei caratteri sono multifunzione, consentendo l'uso di una serie di accenti e caratteri stranieri. Gli altri 21 tasti controllano la stampa, la calcolatrice e le funzioni di memoria, la definizione dei margini dei testi, i tabulatori ed il trascinamento della carta



IAN McKINNELL

consentendo all'utente di correggere uno degli ultimi quindici caratteri. La correzione del testo sul visualizzatore LCD si esegue posizionando un piccolo cursore, tramite due tasti appositi, sul carattere da correggere. La EP-44 possiede anche un modo chiamato "Linea dopo Linea", che permette di esaminare, ed eventualmente correggere, un'intera linea alla volta: in questo modo il display LCD si comporta come una "finestra" che viene fatta scorrere sulla linea di testo mediante i tasti del cursore. Nel modo "Linea dopo Linea" la riga di testo viene stampata battendo il tasto di "ritorno carrello".
Chi è solito usare word processor, non ha l'abitudine di battere il "ritorno carrello" alla fine di ogni riga, poiché il testo viene impaginato automaticamente: la EP-44 possiede anche un modo "Auto Return" che passa automaticamente ad una nuova linea quando la barra spaziatrice viene battuta in una delle ultime sei posizioni della linea.
Nell'apparecchio è incorporata una calcolatrice, che permette di eseguire semplici operazioni sul display LCD, senza interferire col testo da stampare. Se, ad esempio, si sta battendo una fattura di vendita ed occorre calcolare l'IVA o lo sconto da applicare, si può usare la calcolatrice e, successivamente, inserire i risultati nella fattura.
Molte delle funzioni finora citate sono presenti su molte macchine da scrivere elettroniche, ma l'EP-44 può essere usata anche come word processor. Possiede, infatti, 3,5 Kbyte di memoria, nella quale memorizzare testi. Questa memoria può sembrare piccola, se paragonata a quella della maggior parte degli home computer, ma è in grado di accogliere un documento di circa tre pagine. I comandi di controllo per le funzioni di word processing vengono impartiti tramite i normali tasti della macchina da scrivere, purché si premano assieme ad uno speciale tasto blu, marcato con "CODE". Quindi, per inserire in memoria un nuovo testo, si dovranno battere contemporaneamente i tasti CODE e "N" (per Nuovo testo). Questa operazione funziona nei modi Direct Print e Correction Print.
Il testo caricato in memoria può essere redatto usando il display LCD ed i tasti per il cursore. Questi ultimi muovono la finestra LCD nelle quattro direzioni, permettendo quindi di esaminare l'intero testo e di inserire, cancellare o correggere qualsiasi carattere. Non è tuttavia consentito il trasferimento di una o più parole sulla linea successiva qualora la nuova linea redatta sia troppo lunga: ciò significa che non è possibile "riformattare" il documento. Il testo caricato in memoria, comunque, può essere modificato senza doverlo ribattere per intero. Se, ad esempio, si vuol inviare una stessa lettera a diversi destinatari, si potrà modificare il testo in memoria inserendovi via via un diverso nome ed indirizzo e battendo assieme i tasti CODE e "P" (che sta per Print) e stampare così la lettera. La memoria conserva il testo anche quando l'apparecchio viene spento.
Sollevando una piccola piastrina laterale, si ha accesso ad una porta seriale RS232. Spostando un apposito commutatore a slitta in posizione "Terminal", il Brother si trasforma in una stampante per computer, ma sono necessarie alcune prove, per selezionare la giusta velocità di trasmissione (baud) e gli altri parametri specifici per il tipo di computer collegato. La procedura di selezione è abbastanza semplice: il display LCD visualizza ciclicamente le varie velocità di trasmissione e gli altri parametri e l'utente deve premere il tasto MODE, quando sul display appare la registrazione adatta al computer usato. Le scelte fatte vengono conservate anche quando l'apparecchio viene spento.
Sfortunatamente, l'EP-44 non può usare moduli continui tipo "fanfold" ed i singoli fogli di carta devono essere inseriti a mano. La qualità della sua stampa è comunque migliore di quella delle stampanti a matrice d'aghi. Nel modo "Terminal", l'EP-44 può trasmettere e ricevere dati e, collegata ad un modem, permette all'utente di comunicare via telefono con altri home computer o con apparecchi di maggiori dimensioni, purché usino lo stesso protocollo di comunicazione.
La Brother EP-44, nonostante le ridotte dimensioni, si dimostra un apparecchio versatile e interessante. Grazie al prezzo relativamente contenuto e all'alto livello delle sue prestazioni, l'EP-44 risponde perfettamente alle esigenze di quei possessori di home computer attratti dalla sua duplice funzione di stampante e di macchina da scrivere.



LCD MULTIUSO

Il display a cristalli liquidi (LCD) dell'EP-44 visualizza 15 caratteri, 5 indicatori sui modi di stampa ed uno sul livello di carica delle batterie. Un semplice word processor permette la redazione di intere righe di testo sul piccolo display, prima di passare alla stampa. Il contrasto dei caratteri può essere regolato in base alla luminosità dell'ambiente esterno

IAN McKINNELL

Print-Out

```
Any smooth-finish typing
paper can be used with the
ribbon in place, but the
paper-feed cannot accept
carbons.  The print
quality is good on both
types of paper.
Text can be
  Centred & Underlined
foreign (Æø¿¢µ¶), and
sub- or super-scripted
(X2=N2+T0).
```

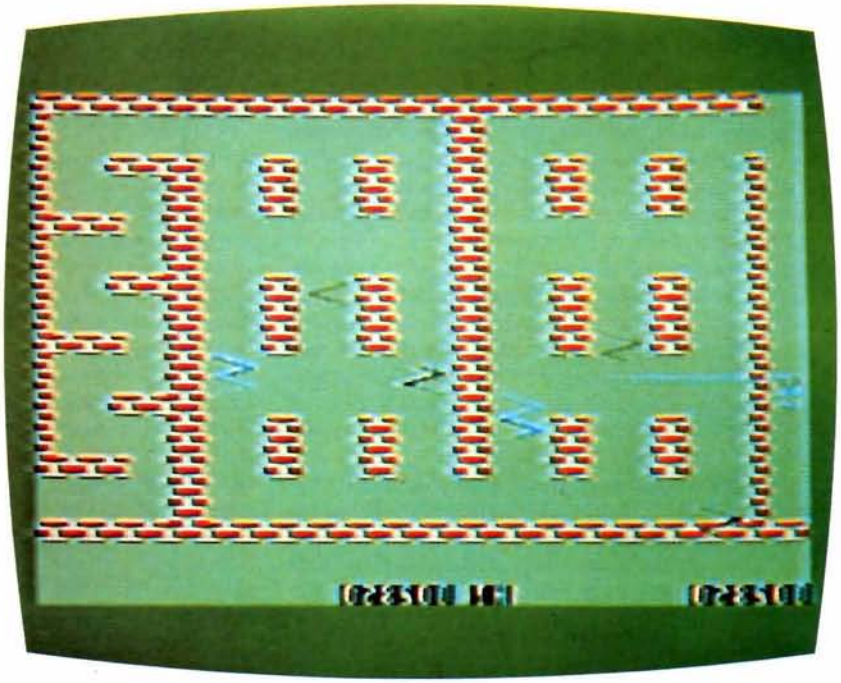
La qualità di stampa

La stampante termica, benché adoperi una matrice di aghi, produce una stampa di ottima qualità, ma la lentezza del processo di stampa si fa sentire (specialmente usando la EP-44 come stampante per un computer). È anche previsto il funzionamento con carta normale (senza carta carbone, però). Si noti la possibilità di usare, nel testo, caratteri speciali, selezionati con la semplice pressione di un tasto

Software in vetrina

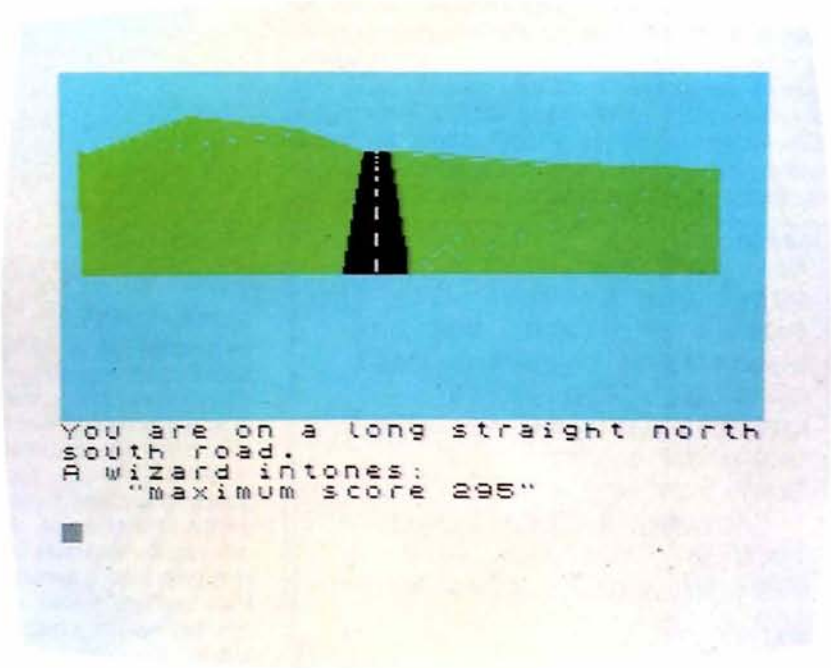
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Ecco due programmi che escono un poco dalla consuetudine: nel primo caso si tratta di una serie di "prove", sul Commodore 64, da superare per diventare "spie qualificate"; nel secondo di uno strumento per costruire le proprie avventure, con tanto di grafica, sullo Spectrum.



Spy school per il Commodore 64

È il primo gioco per questo apparecchio scritto da una software house di recente costituzione: la Dimension 21. Il giocatore può scegliere fra quattro "prove", a due livelli di difficoltà, da portare a termine con successo, per divenire spia in piena regola (di gioco, s'intende!). In una delle prove, *Sniper*, il giocatore è un "cecchino", col compito di abbattere tutti gli elicotteri che gli ronzano attorno. In *Sabotage* l'obiettivo è la distruzione di una base nemica. *Bergulary* è un'esercitazione di... furto (!) ed *Escape*, infine, è paragonabile ad una "fuga a ostacoli". Tutto sommato, è un bel po' di divertimento per un solo programma!



Dungeon Builder per lo Spectrum

Questo programma è un solido blocco di ben 15 Kbyte in codice macchina, che consente all'utente di disegnare gli scenari per i propri giochi d'avventura. Il prodotto finale può essere eseguito in modo del tutto indipendente dal programma "generatore". Si tratta di uno strumento molto duttile, orientato particolarmente agli effetti grafici a colori. L'esecuzione è disinvolta e priva di noiose attese, anche se i risultati non sono del tutto immediati. Senza dubbio *Dungeon Builder* ha attirato e continuerà ad attirare le simpatie di coloro che amano cimentarsi nella creazione di nuove avventure (e, perché no?, di venderle).

Dalla carta allo schermo

Apparecchi che facilitano il trasferimento di disegni nel computer

I "tracciatori digitali" (o "digital tracer") sono strumenti che consentono di inviare un disegno, una fotografia o un'immagine allo schermo (e quindi alla memoria) di un computer, con un semplice lavoro di ricalco. La facilità di ciò dipende soprattutto dal software di corredo all'apparecchio. Qui vengono esaminati quattro "pantografi" digitali: tre per il Micro BBC ed uno per lo Spectrum.

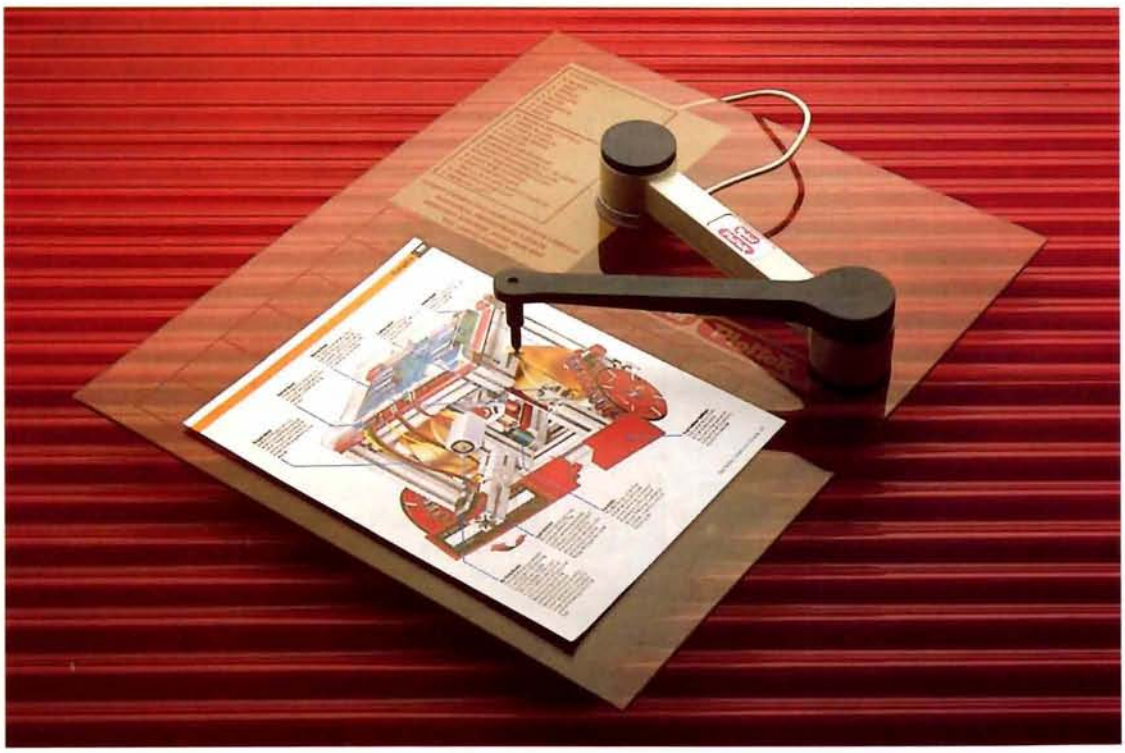
Il principio di funzionamento è pressoché identico in quasi tutti i tracciatori e si basa su due elementi (bracci) collegati da uno snodo: ad una delle estremità è posto un "puntatore" o un "mirino", la cui posizione è inviata al computer tramite segnali elettrici, che variano a seconda degli spostamenti sul disegno. Il computer trasforma i segnali in impulsi digitali e se ne serve per riprodurre l'immagine sullo schermo. Tutti i tracciatori digitali sono corredati da un software operativo, che offre svariate opzioni quali, ad esempio, la selezione dei colori da usare per le linee. Il software dei modelli per il BBC, invece, prevede l'uso di un solo colore, ma un'ampia scelta della risoluzione: i progettisti del software sono costretti a privilegiare l'una o l'altra caratteristica, a causa del limitato spazio di memoria disponibile.

Il Robot Plotter, prodotto dalla Robot Computer Development, è il più interessante dei quattro apparecchi presi in esame. Possiede una base trasparente in perspex, dotata di un reticolato, che indica la posizione dei pixel: il disegno o la fotografia da riprodurre possono quindi essere inseriti sotto di essa. Il braccio grafico, composto da un segmento metallico e da uno plastico, è molto robusto e viene fissato all'angolo superiore destro della base. Il suo puntatore assomiglia a un piccolo lapis e poggia sulla faccia superiore della superficie di perspex: purtroppo la sua conformazione rende difficile vedere con chiarezza il disegno che si vuol riprodurre.

Il Robot Plotter è corredato da un nastro a cassetta che, oltre a contenere il software operativo per il Micro BBC, include anche alcuni programmi di utilità per tracciare cerchi, rettangoli e linee.

Il programma principale immagazzina in memoria tutte le immagini come sequenze di brevi linee: questo metodo permette all'utente di cancellare una linea senza danneggiare quelle circostanti. Un'immagine complessa come quella in figura occupa una discreta quantità di memoria, quindi il sistema spesso usa l'intera capacità del BBC. Grazie a questa tecnica di memorizzazione, però, le immagini video possono essere trasferite con facilità da un apparecchio ad un altro.

Anche il braccio grafico Digigraph ha una



Il "Robot Plotter"
L'immagine da riprodurre viene collocata sotto la base in perpex del Robot Plotter. Il software in dotazione comprende routine per tracciare cerchi e rettangoli



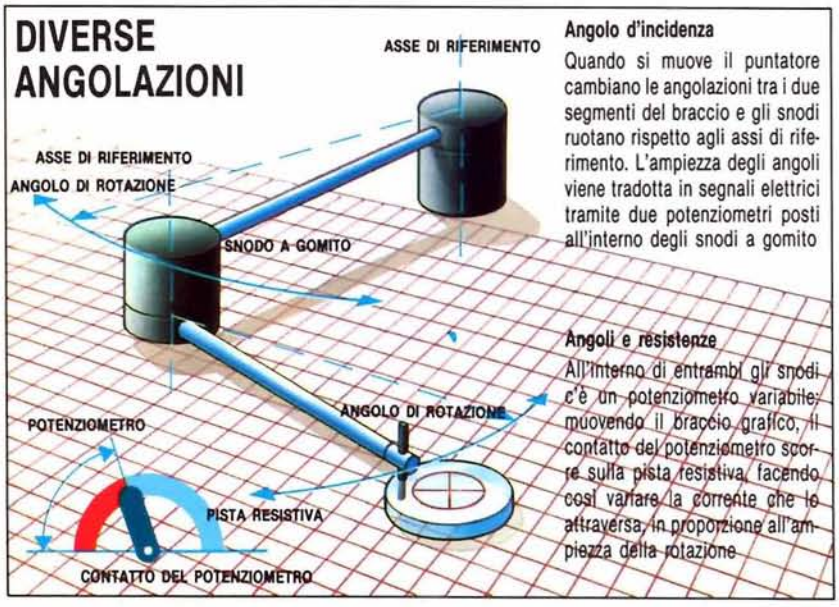
Il "Digigraph"
Fabbricato in alluminio e materiali plastici, il sistema Digigraph poggia su una base di legno laminato e possiede alcuni tracciati grafici a scopo di esercitazione

struttura robusta: sulla base (una larga tavoletta di legno laminato) è riprodotta una griglia rossa, sulla quale viene appoggiato il disegno da riprodurre. I due elementi del braccio sono di alluminio e il puntatore è costituito da un dischetto trasparente in perspex. Il Digigraph è il più facile da usare dei quattro apparecchi esaminati: il movimento del braccio è lineare, e il puntatore trasparente consente una perfetta visione dell'immagine su cui si lavora. Il software di corredo sembra meno sofisticato di quello del Robot Plotter, ma offre le stesse facilitazioni. I movimenti del braccio non sono memorizzati come istruzioni separate, bensì vengono riprodotti direttamente sullo schermo; le immagini così ottenute vengono memorizzate e richiamate in memoria, salvando e caricando la memoria video. Questo criterio rende meno facile il lavoro di riproduzione, ma consente di risparmiare memoria: qualunque sia la complessità del disegno, infatti, il consumo di memoria resta costante. Il corredo del sistema Digigraph comprende anche alcuni eccellenti manuali ed alcuni campioni di tracciato, sui quali è consigliabile far pratica prima di intraprendere il lavoro di riproduzione. Il software operativo può essere venduto sia su cassette, sia su dischetto.

Il braccio grafico della RD Labs viene venduto in due formati, uno per lo Spectrum e l'altro per il BBC, entrambi in versione tipo "scatola di montaggio". Il braccio, fatto di materiale plastico, è infatti privo di una base propria, ma può essere fissato (tramite una piastrina biadesiva) su qualsiasi superficie piana. La meccanica è scorrevole e precisa, benché il puntatore, a forma di croce, si dimostri poco funzionale.

La versione BBC comprende un manuale e una cassetta di software. Le immagini vengono memorizzate con la stessa tecnica del Robot Plotter, il che comporta identici problemi di memoria. Il software del sistema è leggermente più complesso e possiede alcune routine per tracciare cerchi, rettangoli e linee. Inoltre, viene fornito un programma di animazione, basato sulla capacità del BBC di ridefinire la "tavolozza" dei colori, che permette di animare semplici sequenze d'immagini. Una cassetta dimostrativa illustra le possibilità grafiche offerte dall'apparecchio.

La versione per lo Spectrum è meno costosa della precedente e include un convertitore analogico/digitale (il Micro BBC ne ha uno incorporato). Disegnare in modo continuo è possibile, ma il software non consente una gran rapidità. Inoltre, quando vengono tracciate linee che si estendono oltre i confini





Il "Digital tracer" della RD Labs
La versione per il Micro BBC è fabbricata in materiale plastico e viene fissata con una piastrina biadesiva ad una qualsiasi superficie piana



INTERFACCIA PER L'RD SPECTRUM

Il tracciatore grafico RD, per lo Spectrum, è identico a quello per il Micro BBC ma viene collegato al computer mediante una speciale interfaccia, che funge da convertitore analogico/digitale (questo dispositivo, nel BBC, è incorporato di serie)

dello schermo, il software non visualizza messaggi d'errore, come invece accade nei programmi Spectrum. Nella versione Spectrum, le linee curve tendono a "raddrizzarsi" sullo schermo, essendo tracciate con una serie di brevi trattini (colpa della media risoluzione dello Spectrum, comunque).
Per alcuni lavori specialistici, è possibile scrivere programmi che "leggono" le varie coordinate di tracciamento direttamente dal braccio grafico. Sullo Spectrum, a differenza del Micro BBC, è difficile realizzare questa operazione perché è sprovvisto di una porta per il joystick e di adeguati comandi BASIC. Per i possessori di Spectrum la scelta è limitata al braccio grafico RD Labs, mentre per il micro BBC la scelta, come abbiamo visto, è più ampia: l'RD Labs (il meno costoso, ma anche il più delicato), il Robot Plotter (assai robusto e dotato di ottimo software, ma di uso più complesso) e il Digigraph, che ha un software meno sofisticato del precedente, ma è più facile da usare.

IMMAGINI ALTERNATIVE

I tracciatori grafici, grazie all'ampia gamma di colori e di opzioni grafiche di cui dispongono, consentono di creare piacevoli immagini ad alta risoluzione: tuttavia il loro impiego richiede esperienza ed una grande attenzione

I programmi diagnostici

Una simulazione che illustra i principi dell'"Intelligenza Artificiale"

PROGRAMMA DIAGNOSTICO

```
90 CLEAR 500 : REM RISERVA SPAZIO PER LE STRINGHE
100 GOSUB 2000:          REM ** INIZIALIZZA **
200 FOR L=1 TO 2*LT STEP 2
300 PRINT O$: INPUT I$
400 GOSUB 3000:          REM ** ANALISI **
500 NEXT L
1000 PRINT TAB(5);"-----    LA SEDUTA E' FINITA    -------"
1100 PRINT TAB(4);"----- PREMI UN TASTO PER CONTINUARE---"
1200 A$=INKEY$: IF A$="" THEN GOTO 1200
1300 GOSUB 5000:         REM ** TRASCRIZIONE **
1900 END
1999 REM ****************************************
2000 REM **    SUBROUTINE DI INIZIALIZZAZIONE    **
2001 REM ****************************************
2050 LT=10: AN=10: T9=500: EX=2*LT
2100 DIM R$(AN): DIM H$(2*LT)
2200 DATA "SI...","PARLANE ANCORA...","CONTINUA...",
     "E POI...","QUINDI..."
2250 DATA "QUESTO E' IMPORTANTE...","PERCHE' TI INTERESSA...?"
2300 DATA  "SPIEGATI  MEGLIO...","PERCHE' AFFERMI QUESTO...?"
2350 DATA "COSA C'E' CHE NON VA..."
2500 FOR K=1 TO AN: READ R$(K): NEXT K
2600 PRINT CHR$(12): O$="SALVE, CHE PROBLEMA HAI..?"
2950 RETURN
2999 REM ****************************************
3000 REM **         SUBROUTINE DI ANALISI        **
3001 REM ****************************************
3100 IF I$="ARRIVEDERCI" THEN EX=L-1:  L=2*LT: RETURN
3200 H$(L)=O$: H$(L+1)=I$
3300 R9=INT(AN*RND(1)+1): IF R9=R0 THEN GOTO 3300
3400 O$=R$(R9): R0=R9
3950 RETURN
4999 REM ****************************************
5000 REM **      SUBROUTINE DI TRASCRIZIONE      **
5001 REM ****************************************
5050 PRINT CHR$(12): PRINT TAB(10);"**TRASCRIZIONE**"
5100 FOR K=1 TO EX STEP 2
5150 PRINT TAB(5);H$(K)
5200 PRINT H$(K+1)
5300 FOR D=1 TO T9: NEXT D
5400 NEXT K
5900 RETURN
```

Negli anni quaranta, Alan Turing, uno dei pionieri dell'informatica, formulò una famosa ipotesi sull'intelligenza artificiale: una macchina si sarebbe potuta considerare "intelligente", quando, in un dialogo con un uomo, questi non si sarebbe accorto di parlare con un oggetto meccanico. Tutt'oggi, però, non esistono ancora computer che abbiano superato questa prova, benché molti programmi per computer riescano, con astuti accorgimenti, ad indirizzare la conversazione dell'uomo nella direzione da essi scelta, simulando una specie di "intelligenza". In molti casi, però, è l'uomo che non si accorge dei subdoli tranelli tesi dal computer e continua, in effetti, a parlare da solo. Quest'affermazione può sembrare incredibile, ma è sorprendentemente vera! Ripensando a una piacevole conversazione avuta con un amico, probabilmente ci si accorgerà che abbiamo parlato quasi sempre noi e che il nostro interlocutore è intervenuto solo con brevi e concise frasi "di collegamento" o "di incitamento". Questo è il principio su cui poggiano le principali tecniche di psicoterapia.
Il programma presentato in queste pagine si-



mula il comportamento di un terapeuta che non partecipa "direttamente" alla conversazione, ma interviene con frasi come: "DIMMI DI PIÙ..." o "PERCHÉ È COSÌ IMPORTANTE...?". Il programma mantiene una registrazione della conversazione, ripetendone alcune frasi dopo un certo numero di scambi, fino a quando il "paziente" non digita la parola "ARRIVEDERCI".

Un programma può essere sviluppato in direzione dell'intelligenza artificiale, mettendolo in grado di analizzare le frasi del dialogante e scegliere un'adeguata risposta. La versione qui presentata esegue un'operazione simile alla riga 3100, quando viene battuta la parola "ARRIVEDERCI". Potremmo estendere questa capacità di analisi a parole come "SI" e "NO" a cui potrebbero seguire brevi frasi di risposta come "PERCHÉ?" o "PERCHÉ NO?". Quindi, il programma potrebbe controllare se il dialogante ripete una particolare risposta e, in tal caso, replicare in modo conseguente o porre fine alla seduta. Inoltre, potrebbe verificare se è stata posta una domanda (la frase termina con un punto interrogativo) o se si tratta di un'esclamazione (un punto esclamativo), e intervenire con frasi del tipo: "PERCHÉ DOMANDI QUESTO...?" o "PERCHÉ TI SEI EMOZIONATO...?". Queste sono tutte strategie molto efficaci, che richiedono un minimo tempo di elaborazione e non implicano alcun tentativo reale di capire il senso di quanto detto dal dialogante.

Addirittura, potremmo preparare una tabella di parole chiave ed introdurre una specifica risposta per ogni parola, ricavandola da una seconda tabella. La scelta delle parole chiave e delle risposte dipende dal tipo di conversazione che ci si attende: l'effetto della risposta scelta, nella successiva parte della "seduta", può fornire alcune indicazioni sul subconscio dell'interlocutore. Il limite di questo metodo (e di tutti quelli che prevedono la ricerca di parole in un brano di conversazione) è il tempo impiegato per analizzare anche una breve frase. La relativa velocità del BASIC è in questo caso un fattore limitante, quindi una qualsiasi procedura di analisi andrebbe scritta in codice macchina, cosicché anche i metodi di analisi più semplici darebbero risultati sorprendenti e permetterebbero di comprendere meglio i problemi che i computer della quinta generazione devono risolvere.

L'analisi delle parole usate dal dialogante può essere condotta in molti modi: la tecnica più interessante è probabilmente l'"analisi sintattica", ossia la scomposizione di una frase nelle sue componenti di base, come il sostantivo, il verbo o il pronome. La sua programmazione è piuttosto complessa, in quanto poggia su un insieme di regole grammaticali e sintattiche e richiede una serie di tabelle di pronomi, preposizioni, congiunzioni ecc. È certamente più semplice individuare la parola più lunga della risposta e chiedere all'utente di spiegare le sensazioni che prova riflettendo su di essa: la parola più lunga è spesso la più importante di una breve frase. In altri casi, il programma sceglierà a caso tra le parole più lunghe di 5 lettere, ad esempio, oppure userà quelle che seguono parole come "IO", "MIO", "TU" o "TUO".

Tutte queste tecniche danno una chiara idea dell'immensa complessità del linguaggio e della sua analisi, e permettono anche di cogliere l'inadeguatezza del BASIC come strumento analitico. Ciò incita ad approfondire altri linguaggi di programmazione, quali il LISP ed il PROLOG, che, essendo molto più strutturati, meglio si adattano all'analisi del linguaggio e dei concetti in esso contenuti.

Aggiungere queste linee al primo programma per la verifica SI/NO e per l'individuazione della parola più lunga

```
3420 IF  LEFT$(I$,3)="SI" THEN  O$="COSA
     TI RENDE COSI' SICURO..?": RETURN
3450 IF LEFT$(I$,2)="NO" THEN O$="PERCHE' NO..?"
3500 Z$=RIGHT$(I$,1)
3520 IF Z$="?" THEN O$="PERCHE' ME LO CHIEDI..?"
3550 IF Z$="!" THEN O$="PERCHE' TI TURBA
     TANTO..?"
3600 IF R9<AN/4 THEN GOSUB 4000
3950 RETURN
3999 REM ****************************************
4000 REM **SUBROUTINE PER LA PAROLA PIU' LUNGA**
4001 REM ****************************************
4050 W=1: H=1: WL=1: S=1
4100 I$=I$+" ":L9=LEN(I$)
4120 FOR C=1 TO L9:FOR P=C TO L9
4150 Z$=MID$(I$,P,1)
4170 IF Z$=" " THEN W=P-C:H=C:C=P:P=L9
4200 NEXT P
4220 IF W>WL THEN WL=W: S=H
4250 NEXT C
4270 O$="CHE SIGNIFICA " + MID$(I$,S,WL) +
     " PER TE.."
4450 RETURN
```

```
**TRASCRIZIONE**

SALVE, CHE PROBLEMA HAI..?
NON HO AMICI
PERCHE' NO..?
PER LA MIA DIFFICOLTA' NEL
  COMUNICARE
CHE SIGNIFICA DIFFICOLTA'
  PER TE?..
PENSO SEMPRE DI NON PIACERE
  AGLI ALTRI
SPIEGATI MEGLIO..
NON SO!
PERCHE' TI TURBA TANTO..?
NON MI CAPISCI NEMMENO TU?
PERCHE' ME LO CHIEDI..?
```

Invito alla riflessione

Ecco un esempio di "seduta" col programma qui presentato. Si noti come il tono impersonale delle risposte fornite dal computer simuli efficacemente il comportamento distaccato di un analista



Basic & Basic

Questo programma è scritto in Microsoft BASIC. Gli utenti dello Spectrum dovranno inserire LET in tutte le istruzioni di assegnazione e modificare i valori di TAB

Spectrum

Sostituire DIM R$(AN): DIM H$(2*LT) con DIM R$(AN,30): DIM H$(2*LT,100)
Sostituire LEFT$(I$,3) con I$(TO 3) e LEFT$(I$,2) con I$(TO 2)
Sostituire RIGHT$(I$,1) con I$(LEN(I$))
Sostituire MID$(I$,P,1) con I$(P)
Sostituire MID$(I$,S,WL) con I$(S TO S+WL—1)
Sostituire INT(AN+RND(1)+1) con INT(AN+RND+1)

Commodore Vic-20 e 64

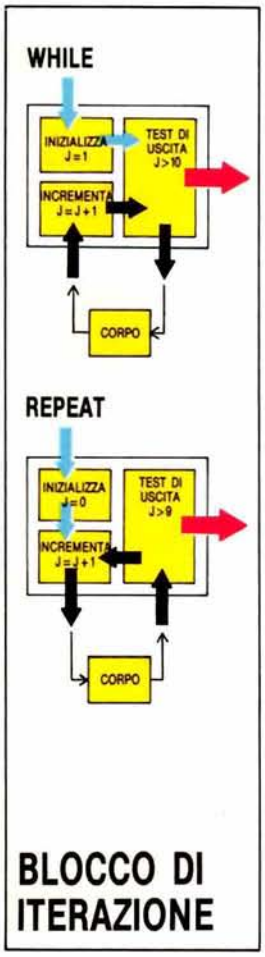
Sostituire PRINT CHR$(12) con PRINT CHR$(147)
Sostituire INT(AN*RND(1)+1) con INT(RND(1)*AN+1)
Sostituire A$=INKEY$ con GET A$

Micro BBC

Sostituire A$=INKEY$ con A$=INKEY$(0)
Sostituire INT(AN*RND(1)+1) con RND(AN)

Cicli e iterazioni

Utili varianti ai simboli classici dei diagrammi di flusso



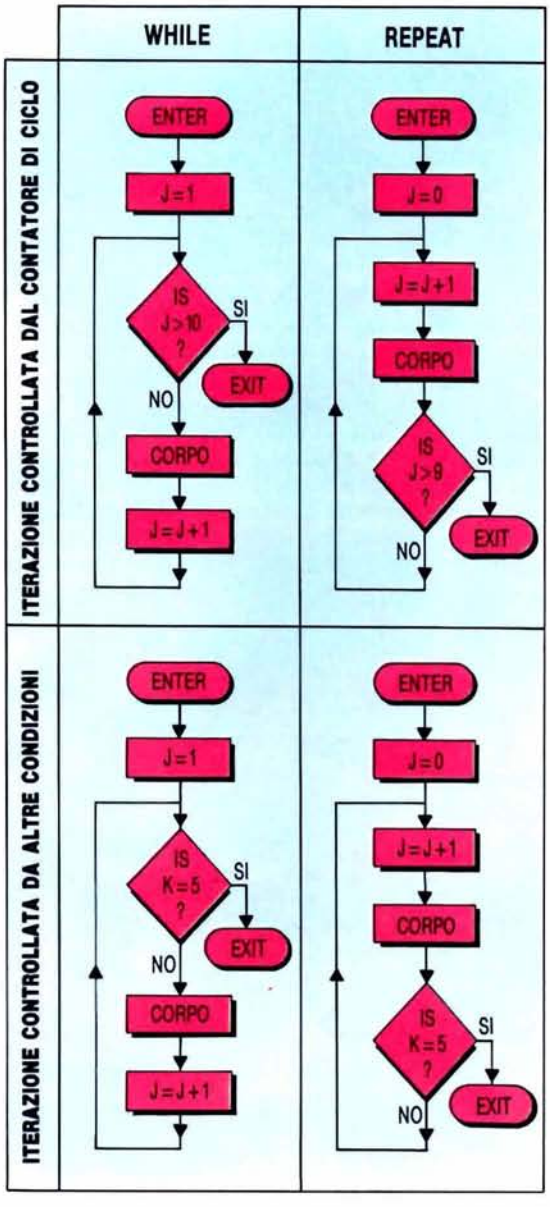
I cicli iterativi o iterazioni rappresentano una delle strutture di base dei linguaggi di programmazione. Abbiamo già visto che le iterazioni vengono usate negli algoritmi tutte le volte che una particolare verifica implica ripetute deviazioni del flusso di controllo su un percorso "chiuso", ripetendo più volte la stessa verifica e quindi una stessa porzione di codice. Tuttavia, questa è una descrizione generica e piuttosto limitata. Poiché le iterazioni sono strutture primitive indispensabili, che impegnano approssimativamente il 60 per cento dell'attività totale del microprocessore, è estremamente importante studiarle in modo approfondito. Occorre prestare attenzione ai loro effetti sulla struttura generale del programma (o algoritmo) e alle varie tecniche usate per realizzarle e classificarle. Le iterazioni vengono spesso suddivise in due classi, a seconda della loro somiglianza con le due strutture iterative REPEAT...UNTIL e WHILE...ENDWHILE, tipiche dei linguaggi di alto livello (entrambe sono disponibili in PASCAL, mentre la forma REPEAT esiste soltanto nei dialetti BASIC del BBC e dell'Oric). I due tipi di iterazione differiscono per la posizione del test di uscita dal ciclo: in un ciclo REPEAT il test di verifica è posto alla fine del ciclo, mentre in un ciclo WHILE il test si trova all'inizio. Quindi il "corpo" dei cicli REPEAT verrà sempre eseguito almeno una volta, mentre ciò non è vero per i cicli WHILE. Questa prima differenza può essere rilevata con chiarezza dai diagrammi di flusso.

Un'altra tecnica per classificare le iterazioni si basa sulla variabile impiegata come contatore di ciclo. Infatti, alcune iterazioni la usano nel test di uscita dal ciclo, mentre in altre, l'uscita dal ciclo è controllata da verifiche di diverso genere. Purtroppo, questa diversità è assai poco evidente nei comuni diagrammi di flusso lineari, usati indistintamente per rappresentare sia le iterazioni, sia i semplici "salti". Spesso si rende necessario esaminare in dettaglio l'algoritmo per distinguerli l'uno dall'altro.

Tuttavia, esiste una particolare struttura, detta "blocco di iterazione", che rappresenta con chiarezza l'inizio di un ciclo ed elimina ogni possibile confusione tra salto e ciclo. Essa è composta da tre blocchi collegati: il primo contiene l'inizializzazione del contatore, il secondo la procedura che lo incrementa ed il terzo il test di uscita dal ciclo. Entrambe le iterazioni REPEAT e WHILE possono essere rappresentate in questo modo, ma si possono facilmente distinguere dal diverso flusso di controllo che collega i blocchi. Il terzo blocco indica invece se il test di uscita dal ciclo è controllato dal contatore o meno.

Nei cicli REPEAT il flusso di controllo è il seguente: "inizializza-corpo-verifica-corpo-verifica" mentre nei cicli WHILE è: "inizializza-verifica-corpo-verifica-corpo". Infatti, in entrambi i diagrammi di flusso, il corpo del ciclo iterativo sembra essere "sospeso" tra le linee di controllo che escono e rientrano nel blocco di iterazione.

CLASSIFICAZIONE



Snooker

L'acquisto di un biliardo costituisce una spesa non trascurabile, quindi...

Per chi abbia giocato a biliardo almeno una volta, l'idea di "computerizzare" questo gioco sembrerà uno sproposito: per i professionisti si tratterà di una vera e propria eresia! Com'è possibile quantificare e programmare la delicata curvatura di un colpo ad effetto e le impercettibili differenze tra due tavoli da biliardo a prima vista identici? Il tentativo diventa poi assurdo quando si dispone dei soli 16 K di memoria di uno Spectrum! Tuttavia, il gioco del biliardo può essere pensato come una semplice applicazione di una legge di fisica (quella sulla conservazione del movimento) e un computer non trova certo difficoltà nel risolvere le equazioni che da essa derivano.

Lo *Snooker* della Visions mostra un tavolo da biliardo visto dall'alto sul quale si trovano le palle, rappresentate da cerchi colorati. Per preparare un colpo, il giocatore deve posizionare una specie di mirino sul "tavolo", in modo da definire la traiettoria che la palla bianca dovrà percorrere. La maggiore o minore forza del colpo dipende dal lasso di tempo trascorso premendo il tasto di lancio. Bastano pochi colpi per accorgersi che il gioco non è del tutto privo di inconvenienti. Il primo è che il computer (sia esso uno Spectrum, un BBC o un Commodore) non può calcolare in modo realistico gli angoli di impatto o l'attrito del "panno" per cui i risultati sullo schermo sono piuttosto mediocri. Molti colpi, pertanto, assumono traiettorie imprevedibili (come del resto accade anche su un vero tavolo da biliardo), ma altri non possono proprio essere giocati. Per compensare questa imprecisione, le buche del tavolo hanno dimensioni sensibilmente maggiori che nella realtà.

Il secondo inconveniente, forse più fastidioso del primo, dipende dal fatto che il tavolo da gioco è visto dall'alto. Ciò significa che il giocatore, nella preparazione di un colpo, non può prendere la mira servendosi della "stecca". Inoltre, siccome lo schermo ha sempre una leggera incurvatura, la valutazione delle angolazioni di tiro ne risente doppiamente. I primi colpi, quindi, saranno a dir poco disastrosi. Tuttavia, una volta individuati ed accettati questi limiti, lo Snooker diventa un interessante gioco di strategia e il giocatore comincia ad apprezzare alcune ingegnose caratteristiche di programmazione quali la possibilità, ad esempio, di dare il "giro" alla palla, benché i colpi di questo tipo non siano sempre precisi.

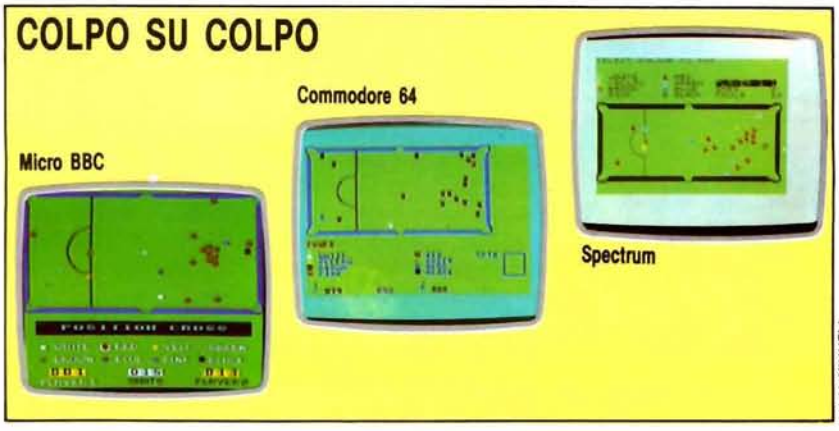
Le tre versioni del gioco rispecchiano le diversità e i punti deboli dei tre apparecchi usati. Sorprendentemente, il programma occupa poca memoria e viene caricato rapidamente da tutti e tre i micro, anche dal Commodore la cui lentezza di caricamento è ormai leggendaria. Quindi, la qualità di ciascuna versione dipende non tanto dalla capacità di memoria, quanto dalle particolari caratteristiche grafiche dell'apparecchio.

La versione per lo Spectrum è la meno incisiva: il tavolo da biliardo è piccolo e i colori delle palle non sono realistici. La versione Commodore è migliore: il tavolo e i colori sono più curati e la dimensione delle palle è perfettamente in scala. I punteggi sono affiancati da due immagini stilizzate di giocatori ed il programma possiede una breve dimostrazione alla quale si può accedere solo dopo aver giocato almeno una partita. La versione BBC è la migliore delle tre: il tavolo occupa quasi tutto lo schermo e la procedura di puntamento è molto più precisa.

Il gioco prevede la partecipazione di uno o due giocatori, ma la prima opzione viene di solito usata per partite d'allenamento. L'impressione generale che se ne ricava è più che accettabile, nonostante sorga il dubbio che gli autori avrebbero potuto dar maggior risalto alle potenzialità del gioco se avessero dato meno importanza al risparmio di memoria.

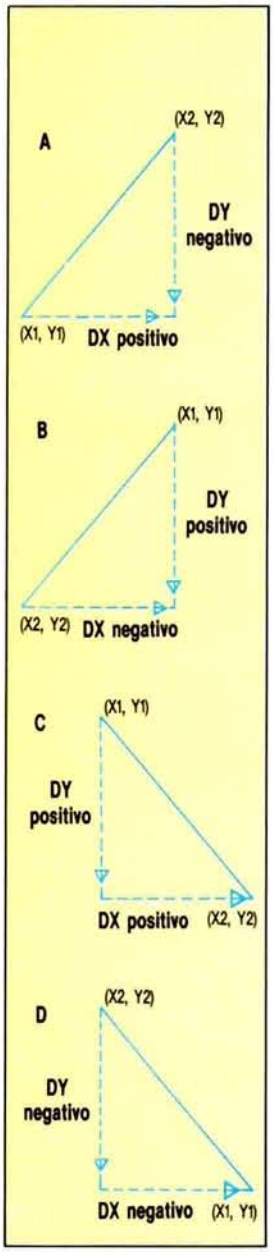
Snooker: per il Commodore 64
per lo Spectrum a 16 K
per il Micro BBC
Editori: Visions (Software Factory) Ltd, GB
Autore: Tim Bell (adattamento per il BBC di Andy Williams)
Joystick: Opzionale
Formato: Cassetta

Le caratteristiche essenziali del biliardo sono la precisione e la forza dei colpi. Il gioco "Snooker" della Visions possiede un "mirino" che viene posizionato sul tavolo da biliardo, mentre la forza del colpo dipende dalla durata della pressione esercitata sul tasto di lancio



Piani inclinati

Tracciare linee diagonali con precisione sul Commodore 64



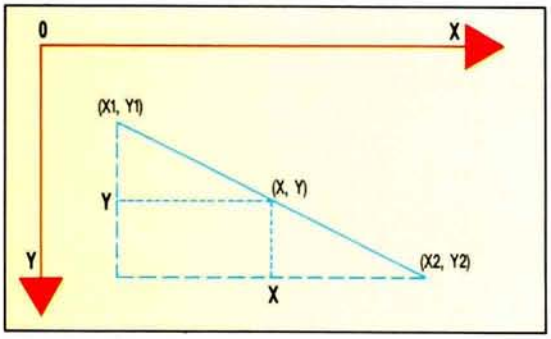
Inclinazione Uno

Quando il valore assoluto dell'inclinazione (Y2−Y1)/(X2−X1) è maggiore di uno, conviene mantenere DX positivo, usando uno dei seguenti algoritmi:

A) Ricalcolare −(DY) e decrementare Y, partendo da (X1,Y1) fino a raggiungere Y2
B) Scambiare (X1,Y1) con (X2,Y2) e ricalcolare
C) Incrementare Y, partendo da (X1,Y1), fino a raggiungere Y2
D) Scambiare (X1,Y1) con (X2,Y2) e ricalcolare

Il modo più ovvio per tracciare una linea da un punto ad un altro dello schermo è calcolarne l'inclinazione usando le coordinate dei due punti estremi. Quindi, partendo da un'estremità della linea, possiamo tracciare tutti i punti successivi, incrementando la coordinata X e calcolando il corrispondente valore della coordinata Y con una semplice equazione, fino a raggiungere il punto finale. Esaminiamo in dettaglio le procedure matematiche coinvolte in questa operazione.

Se indichiamo con (X1,Y1) e (X2,Y2) le coordinate dei due punti estremi del segmento, allora l'inclinazione G è uguale a (Y2-Y1)/(X2-X1). Il diagramma mostra una linea inclinata ed un suo punto interno di coordinate (X,Y):



L'espressione (Y−Y1)/(X−X1) permette di calcolare l'inclinazione tra il punto iniziale e quello di coordinate (X,Y), che è uguale, ovviamente, all'inclinazione G dell'intera linea. Uguagliando questi due elementi e sviluppando l'equazione, si ottiene un'espressione matematica per il valore della coordinata Y di qualsiasi punto della linea:

$$(Y-Y1)/(X-X1)=G$$
$$(Y-Y1)=G(X-X1)$$
$$Y=Y1+G(X-X1)$$

Se la differenza tra X ed X1 è uguale ad uno (ad esempio, se abbiamo incrementato il valore X di uno), allora potremo semplificare l'espressione a:

$$Y=Y1+G$$

Quindi, partendo dal punto estremo (X1,Y1) e incrementando X, possiamo calcolare tutti i valori successivi di Y, semplicemente aggiungendo ripetutamente il valore dell'inclinazione della linea. In BASIC, è possibile perfezionare questo procedimento usando il programma Plotsub (vedere a pag. 817), per tracciare tutti i punti della linea. Tuttavia questo metodo presenta alcune difficoltà:

- Se X2 è minore di X1, dovremo decrementare X invece di incrementarlo.
- Se il valore dell'inclinazione è maggiore di uno, la linea tracciata non apparirà "continua" in quanto, per G maggiore di uno, Y cresce di un valore superiore all'unità, ad ogni incremento di X.
- Le inclinazioni negative causano complicazioni, se occorre implementarle in codice macchina. Per rappresentare le variazioni negative di X e Y si potrebbe usare il "complemento a due", ma siccome le coordinate Y possono variare da 0 a 199 e quelle X da 0 a 319, bisognerebbe effettuare il complemento a due di valori a due byte.
- Non si possono tracciare linee verticali perché il calcolo delle loro inclinazioni implica divisioni per zero.

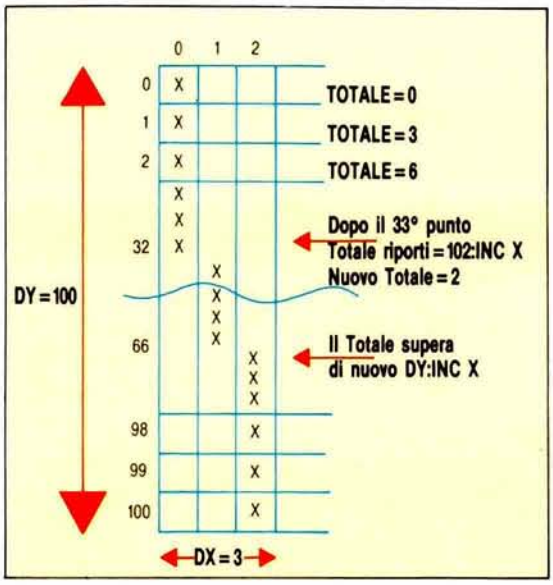
Perfezionamenti

La prima difficoltà si supera facilmente, scambiando i due punti prima di eseguire qualsiasi calcolo.

Per tracciare una linea continua, occorre incrementare il valore della coordinata X e calcolare Y, se l'inclinazione è minore di uno, mentre, se è maggiore di uno, occorre incrementare Y e calcolare X. La biforcazione del problema, conseguente alla condizione (G>1?), ci consente di evitare l'impiego di divisioni e del complemento a due di valori a due byte. Abbiamo già esaminato la procedura per tracciare una linea quando l'inclinazione è minore di uno: vediamo adesso come procedere quando l'inclinazione risulta maggiore di uno. Poniamo DX=X2−X1 e DY=Y2−Y1. Confrontando i valori di DX e DY, è possibile determinare se l'inclinazione è maggiore di uno: ciò si verifica se DX è più piccolo di DY. Volendo evitare che DX e DY assumano valori negativi, la procedura diventa più complessa. Si possono verificare quattro casi (mostrati nel grafico a sinistra) nei quali l'inclinazione è maggiore di uno. Nel nostro programma dobbiamo innanzi tutto scoprire quale di questi casi si verifica e quindi adottare le debite procedure:

Caso a) Ricalcolare DX come Y1-Y2. Partendo da (X1,Y1), decrementiamo Y fino a raggiungere Y2.
Caso b) Si scambiano i punti e si riparte.
Caso c) Si parte da (X1,Y1), incrementando Y fino a raggiungere Y2.
Caso d) Si scambiano i punti e si riparte.

All'inizio del programma, calcoleremo DX e DY: se si ottiene un risultato negativo, è possibile selezionare i bit di una data locazione o registro in modo conforme al risultato ottenuto. Il bit 0 di questo registro (che chiameremo REGNEG) segnala che DY è negativo, mentre il bit 1 indica che DX è negativo. Dopo questi



calcoli, REGNEG può essere sottoposto a verifica per determinare quale delle quattro procedure applicare:

Caso	a	b	c	d
Valore di REGNEG	1	2	0	3
Equivalente binario	01	10	00	11

Qualunque procedura si adotti, alla fine dovremo disegnare la linea mediante un ciclo iterativo che parta da (X1,Y1), decrementi o incrementi Y in modo appropriato e calcoli il corrispondente valore di X, prima di passare il controllo a Plotsub, per tracciare la linea. Il ciclo termina quando Y raggiunge Y2.
Modificando ancora una volta l'espressione originale, otterremo l'espressione per calcolare X da Y:

$$(Y-Y1)/(X-X1)=G$$
$$1/(X-X1)=G/(Y-Y1)$$
$$(X-X1)=(Y-Y1)/G$$
$$X=X1+(Y-Y1)/G$$

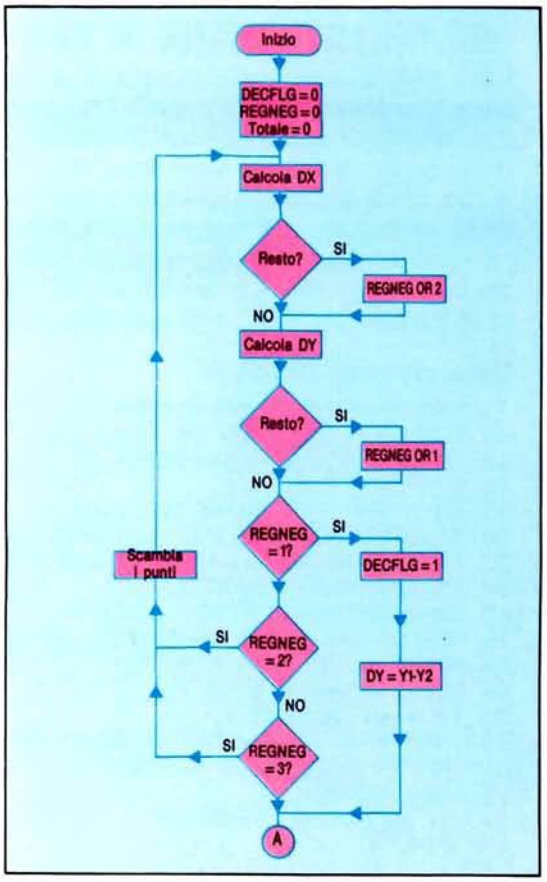
quindi, dato che (Y−Y1) è uguale ad uno, si ha che X=X1+1/G, ma poiché G=DY/DX, l'espressione precedente diventa X=X1+DX/DY. Tuttavia, se l'inclinazione è maggiore di uno, allora DY è sempre maggiore di DX, quindi il risultato della divisione è sempre 0, con resto DX. Se vogliamo mantenere un totale progressivo dei resti, via via che il ciclo si sviluppa, dovremo incrementare X ogni volta che la somma dei resti supera DY e subito dopo sottrarre il valore di DY dal totale.
Questa tecnica può sembrare complicata, ma possiamo esaminarne un'applicazione pratica considerando lo schema in alto a sinistra, nel quale (X1,Y1)=(0,0) e (X2,Y2)=(2,100). La linea risultante è composta da tre trattini verticali e appare piuttosto inclinata. Si possono ottenere linee meno inclinate, purché composte da un maggior numero di trattini più brevi e, quando il valore dell'inclinazione sarà esattamente uno, ogni barretta sarà rappresentata da un pixel e la linea avrà l'aspetto di una diagonale perfetta.

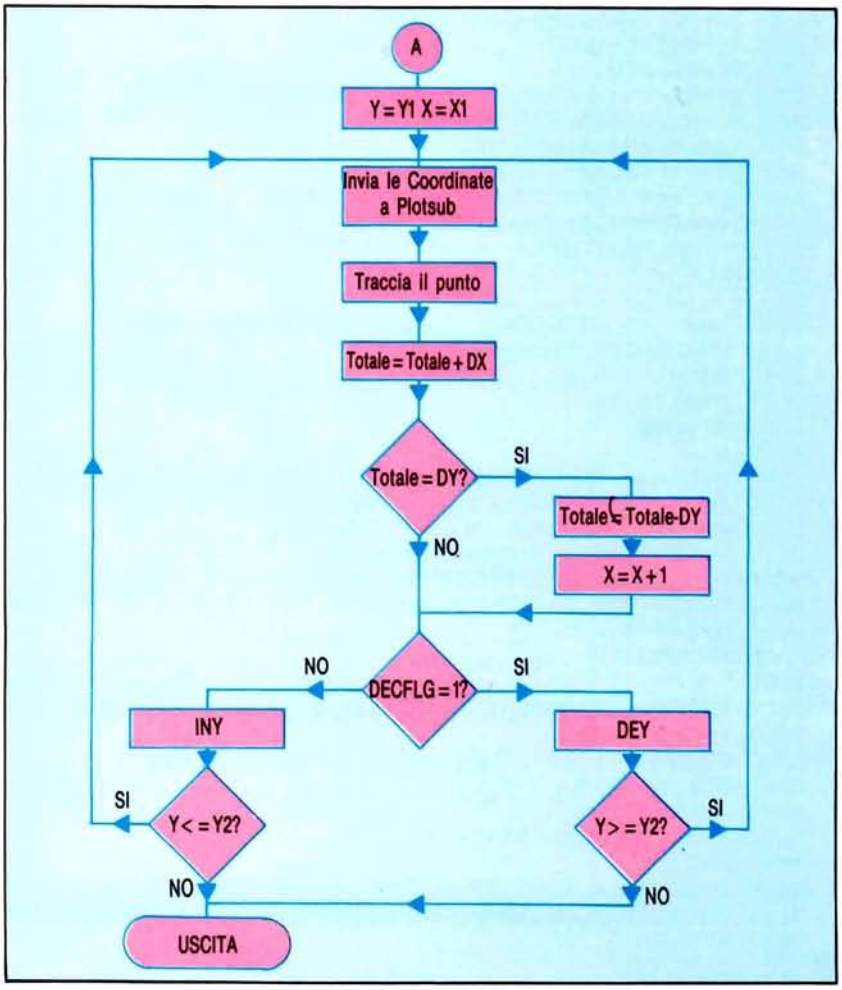
Primo diagramma di flusso

Il diagramma a sinistra chiarisce come sia possibile tracciare una linea orizzontale o verticale composta da un campo di punti discreti: le linee con diversa angolazione devono però essere formate da una serie di trattini, disposti in successione. La linea qui proposta è formata da tre "gradini" e congiunge i punti di coordinate (0,0) e (2,100)

Linee di controllo

Il diagramma di flusso a destra mostra come Lineasub tratti i quattro casi in cui l'inclinazione è maggiore di uno. Le variabili TOTALE e DECFLG vengono definite nel diagramma di flusso superiore e vengono poi usate in quello inferiore, che traccia effettivamente la linea. La procedura di tracciamento inizia dal punto (X1,Y1) e, dato che DX viene mantenuto positivo, decrementa o incrementa Y a seconda dello stato di DECFLG





USO DELLA ROUTINE IN BASIC

Innanzi tutto, si devono caricare Lineasub e Plotsub nel modo mostrato dal Programma Dimostrativo BASIC. Abbiamo chiamato i nostri programmi finali in codice oggetto PLOTSUB.HEX e LINEASUB.HEX. Dopo aver caricato le routine in codice macchina, il programma dimostrativo chiede all'utente di introdurre le coordinate di due punti, verifica se l'inclinazione è maggiore di uno e, infine, esegue Plotsub per disabilitare l'alta risoluzione e selezionare il colore. La subroutine alla linea 2000 cambia le coordinate di X in Byte Basso e Byte Alto, prima di inserire (con POKE) questi valori nelle locazioni selezionate in Lineasub per i punti di inizio e fine linea. La subroutine alla linea 3000 visualizza i valori delle locazioni usate da Lineasub per le variabili: questa procedura può rappresentare un comodo metodo di debugging per programmi in codice macchina. Invece di scrivere il codice sorgente di LINEASUB e assemblarlo in codice macchina, il programma Caricatore del Codice Macchina inserisce lo stesso programma in memoria leggendo (READ) una serie di valori DATA ed inserendoli (POKE) in memoria. Per caricare LINEASUB, occorre eseguire questo programma Caricatore.

Programma Dimostrativo BASIC

```
10 REM ********************
12 REM **   LINESUB 64   **
13 REM ********************
14 :
15 DN=8:REM PER CASSETTA DN=1
20 IF A=0 THENA=1:LOAD"PLOTSUB.HEX",DN,1
30 IF A=1 THENA=2:LOAD"LINESUB.HEX",DN,1
50 INPUT"PRIMO PUNTO";X1,Y1
60 INPUT"SECONDO PUNTO";X2,Y2
70 GOSUB 1000: REM ABILITA L'ALTA RISOLUZIONE
80 GOSUB 2000: REM LINESUB
90 GETA$:IFA$=""THEN 90
95 IF A$=" "THEN200
100 REM *** RIPULISCE LO SCHERMO ***
110 POKE49408,0:SYS 49422
120 GOSUB3000
125 GETA$:IFA$=""THEN125
127 GOTO50
140 :
200 REM *** DISEGNA UN TRIANGOLO ***
205 XA=30:YA=10:XB=310:YB=98
210 XC=90:YC=180
220 GOSUB1000
230 X1=XA:Y1=YA:X2=XB:Y2=YB:GOSUB2000
240 X1=XC:Y1=YC:GOSUB2000
250 X2=XA:Y2=YA:GOSUB2000
255 GETA$:IFA$="" THEN 255
260 REM *** RIPULISCE LO SCERMO ***
270 POKE49408,0:SYS 49422
275 PRINTCHR$(147)
280 END
290 :
1000 REM *** ABILITA L'ALTA RISOLUZIONE ***
1010 POKE49408,1:POKE49409,1
1015 POKE49410,1
1020 SYS 49422
1030 RETURN
1040 :
2000 REM *** INTRODUCE LINEASUB ***
2010 MHI=INT(X1/256):MLO=X1-256*MHI
2020 NHI=INT(X2/256):NLO=X2-256*NHI
2030 POKE49920,MLO:POKE49921,MHI
2040 POKE49922,NLO:POKE49923,NHI
2050 POKE49924,Y1:POKE49925,Y2
2060 SYS 49934
2070 RETURN
2080 :
3000 REM *** STAMPA I VALORI DELLE LOCAZIONI ***
3001 RESTORE
3002 PRINTCHR$(147):REM CANCELLA LO SCHERMO
3005 FORI=0TO13
3010 READA$
3020 PRINTA$,PEEK(49920+I)
3030 NEXT I
3040 DATA X1LO,XIHI,X2LO,X2HI,Y1,Y2,DXLO
3050 DATA DXHI,DY,TEMP,TOTLO,TOTHI,REGNEG,DECFLG
3060 RETURN
```

Caricatore del Codice Macchina

```
10 FOR I=49934 TO 50371
20 READA:CC=CC+A
30 POKEI,A:NEXT
50 READ A:IFCC<>A THEN PRINT "ERRORE
   DI CHECKSUM":END
100 DATA72,138,72,152,72,169,0,141,13
110 DATA195,141,12,195,141,10,195,141
120 DATA11,195,173,2,195,56,237,0,195
130 DATA141,6,195,173,3,195,237,1,195
140 DATA141,7,195,16,8,173,12,195,9,2
150 DATA141,12,195,173,5,195,56,237,4
160 DATA195,141,8,195,176,8,173,12,195
170 DATA9,1,141,12,195,173,12,195,201
180 DATA1,240,20,201,2,208,6,32,140
190 DATA196,76,19,195,201,3,208,19,32
200 DATA140,196,76,19,195,173,4,195,56
210 DATA237,5,195,141,8,195,238,13,195
220 DATA173,6,195,24,105,1,141,6,195
230 DATA173,7,195,105,0,141,7,195,238
240 DATA8,195,173,4,195,168,173,7,195
250 DATA201,1,240,115,173,6,195,205,8
260 DATA195,176,107,173,0,195,141,3
270 DATA193,173,1,195,141,4,193,152
280 DATA141,5,193,32,131,193,173,10
290 DATA195,24,109,6,195,176,8,141,10
300 DATA195,205,8,195,144,24,56,237,8
310 DATA195,141,10,195,173,0,195,24
320 DATA105,1,141,0,195,173,1,195,105
330 DATA0,141,1,195,173,13,195,201,1
340 DATA208,31,136,204,5,195,240,3,76
350 DATA161,195,152,141,5,193,173,0
360 DATA195,141,3,193,173,1,195,141,4
370 DATA193,32,131,193,76,134,196,200
380 DATA204,5,195,144,152,76,134,196
390 DATA173,0,195,141,3,193,173,1,195
400 DATA141,4,193,152,141,5,193,32,131
410 DATA193,173,10,195,24,109,8,195
420 DATA141,10,195,173,11,195,105,0
430 DATA141,11,195,173,10,195,56,237,6
440 DATA195,141,10,195,173,11,195,237
450 DATA7,195,141,11,195,48,15,173,13
460 DATA195,201,1,240,4,200,76,104,196
470 DATA136,76,104,196,173,10,195,24
480 DATA109,6,195,141,10,195,173,11
490 DATA195,109,7,195,141,11,195,173,0
500 DATA195,24,105,1,141,0,195,173,1
510 DATA195,105,0,141,1,195,205,3,195
520 DATA208,142,173,0,195,205,2,195
530 DATA208,134,104,168,104,170,104,96
540 DATA173,2,195,141,9,195,173,0,195
550 DATA141,2,195,173,9,195,141,0,195
560 DATA173,3,195,141,9,195,173,1,195
570 DATA141,3,195,173,9,195,141,1,195
580 DATA173,5,195,141,9,195,173,4,195
590 DATA141,5,195,173,9,195,141,4,195
600 DATA96,230
610 DATA50794:REM*CHECKSUM*
```

Listato Assembly

```
                ;**********************
                ;**********************
                ;**                  **
                ;**  LINESUB 64      **
                ;**                  **
                ;**********************
                ;**********************
                ;
                PLTSUB = $C183
                XLO    = $C103
                XHI    = $C104
                YLO    = $C105
                *      = $C300
                ;
                ;**** VARIABILI LINEASUB ****
                ;
                X1LO   ***+1
                X1HI   ***+1
                X2LO   ***+1
                X2HI   ***+1
                Y1     ***+1
                Y2     ***+1
                DXLO   ***+1
                DXHI   ***+1
                DY     ***+1
                TEMP   ***+1
                TOTLO  ***+1
                TOTHI  ***+1
                REGNEG ***+1
                DECFLG ***+1
                ;
                ;**** APPOGGIA I REGISTRI SULLO STACK ****
                ;
48              PHA
8A              TXA
48              PHA
98              TYA
48              PHA
                ;
                ;**** INIZIALIZZA LE VARIABILI ****
                ;
                START
A9 00           LDA #$00
8D 0D C3        STA DECFLG
8D 0C C3        STA REGNEG
8D 0A C3        STA TOTLO
8D 0B C3        STA TOTHI
                ;
                ;**** CALCOLA DX ****
AD 02 C3        LDA X2LO
38              SEC
ED 00 C3        SBC X1LO
8D 06 C3        STA DXLO
AD 03 C3        LDA X2HI
4C B6 C4        JMP FINISH
                ;
                NODEY
C8              INY
CC 05 C3        CPY Y2
90 98           BCC PLOT
                ;
4C B6 C4        JMP FINISH
                ;**** INCLINAZIONI MINORI DI UNO ****
                ;
                GLESS
AD 00 C3        LDA X1LO
8D 03 C1        STA XLO
AD 01 C3        LDA X1HI
8D 04 C1        STA XHI
98              TYA
8D 05 C1        STA YLO
20 83 C1        JSR PLTSUB
                ;
AD DA C3        LDA TOTLO
18              CLC
6D 08 C3        ADC DY          ; TOTALE=TOTALE+DY
8D 0A C3        STA TOTLO
AD 0B C3        LDA TOHI
69 00           ADC #$00        ; SOMMA IL RIPORTO
8D 0B C3        STA TOTHI
                ;
AD 0A C3        LDA TOTLO
38              SEC
ED 06 C3        SBC DXLO
8D 0A C3        STA TOTLO       ; TOTALE=TOTALE-DX
AD 0B C3        LDA TOTHI
ED 07 C3        SBC DXHI
8D 0B C3        STA TOTHI
30 0F           BMI ADD         ; RISULTATO NEGATIVO
                ;
                ;
AD 0D C3        LDA DECFLG
C9 01           CMP #$01
F0 04           BEQ DECY
C8              INY
4C 68 C4        JMP PAST
                DECY
88              DEY
4C 68 C4        JMP PAST
                ;
                ADD
AD 0A C3        LDA TOTLO
18              CLC
6D 06 C3        ADC DXLO        ; SOMMA DX
8D 0A C3        STA TOTLO
AD 0B C3        LDA TOTHI
6D 07 C3        ADC DXHI
8D 0B C3        STA TOTHI
ED 01 C3        SBC X1HI
8D 07 C3        STA DXHI
10 0B           BPL NBRMDX
AD 0C C3        LDA #REGNEG
09 02           ORA #$02        ; FLAG DX NEGATIVO
8D 0C C3        STA REGNEG
                ;
                ;**** CALCOLA DY ****
                ;
                NBRMDX
AD 05 C3        LDA Y2
38              SEC
ED 04 C3        SBC Y1
8D 08 C3        STA DY
B0 08           BCS NBRMDY
AD 0C C3        LDA REGNEG
09 01           ORA #$01        ; FLAG DY NEGATIVO
8D 0C C3        STA REGNEG
                ;
                NBRMDY
                ;**** VERIFICA LO STATO DI REGNEG ****
                ;
AD 0C C3        LDA REGNEG
C9 01           CMP #$01
F0 14           BEQ NEG1
                ;
C9 02           CMP #$02
D0 06           BNE FALSE
20 BC C4        JSR SWOP
4C 13 C3        JMP START
                ;
                FALSE
C9 03           CMP #$03
D0 13           BNE NEGO
20 BC C4        JSR SWOP
4C 13 C3        JMP START
                ;
                NEG1
AD 04 C3        LDA Y1
38              SEC
ED 05 C3        SBC Y2          ; RICALCOLA DY COME Y1-Y2
8D 08 C3        STA DY
EE 0D C3        INC DECFLG      ; SELEZIONA DECFLEG
                ;
                NEGO
AD 06 C3        LDA DXLO
18              CLC
69 01           ADC #$01        ; SOMMA 1 A DX
8D 06 C3        STA DXLO
AD 07 C3        LDA DXHI
69 00           ADC #$00        ; SOMMA IL RIPORTO
8D 07 C3        STA DXHI
EE 08 C3        INC DY          ; SOMMA 1 A DY
AD 04 C3        LDA Y1
A8              TAY             ; PONE Y=Y1
                ;
                ;**** VERIFICA PER DX>=DY ****
                ;
AD 07 C3        LDA DXHI
C9 01           CMP #$01
F0 73           BEQ GLESS
AD 06 C3        LDA DXLO
CD 08 C3        CMP DY
B0 6B           BCS GLESS
                ;
                ;**** TRACCIA UN PUNTO ****
                ;
                PLOT
AD 00 C3        LDA X1LO
8D 03 C1        STA XLO         ; FORNISCE IL BYTE BASSO DI X
AD 01 C3        LDA X1HI
8D 04 C1        STA XHI         ; FORNISCE IL BYTE ALTO DI X
98              TYA
8D 05 C1        STA YLO         ; FORNISCE Y
20 83 C1        JSR PLTSUB
AD 0A C3        LDA TOTLO
18              CLC
6D 06 C3        ADC DXLO        ; TOTALE=TOTALE+DX
B0 08           BCS DOINX       ; LA SOMMA SUPERA 1 BYTE?
8D 0A C3        STA TOTLO
CD 08 C3        CMP DY
90 18           BCC NOINX       ; IL TOTALE HA RAGGIUNTO DY
                DOINX
38              SEC
ED 08 C3        SBC DY          ; TOTALE=TOTALE-DY
8D 0A C3        STA TOTLO
AD 00 C3        LDA X1LO
18              CLC
69 01           ADC #$01
8D 00 C3        STA X1LO        ; INCREMENTA X1
AD 01 C3        LDA X1HI
69 00           ADC #$00
8D 01 C3        STA X1HI
                ;
                NOINX
AD 0D C3        LDA DECFLG
C9 01           CMP #$01
D0 1F           BNE NODEY       ; DECFLG=1?
88              DEY
CC 05 C3        CPY Y2
F0 03           BEQ FINAL       ; SE Y=Y2 TRACCIA  L'ULTIMO PUNTO
4C A1 C3        JMP PLOT
                FINAL           ; TRACCIA L'ULTIMO PUNTO
98              TYA
8D 05 C1        STA YLO
AD 00 C3        LDA X1LO
8D 03 C1        STA XLO
AD 01 C3        LDA X1HI
8D 04 C1        STA XHI
20 83 C1        JSR PLTSUB
                ;
                PAST
AD 00 C3        LDA X1LO
18              CLC
69 01           ADC #$01
8D 00 C3        STA X1LO        ; INCREMENTA X
AD 01 C3        LDA X1HI
69 00           ADC #$00
8D 01 C3        STA X1HI        ; SOMMA IL RIPORTO
                ;
CD 03 C3        CMP X2HI
D0 BE           BNE GLESS
AD 00 C3        LDA X1LO        ; X HA RAGGIUNTO X2?
CD 02 C3        CMP X2LO
D0 B6           BNE GLESS
                ;
                ;**** RECUPERA I REGISTRI DALLO STACK ****
                ;
                FINISH
68              PLA
A8              TAY
68              PLA
AA              TAX
68              PLA
                ;
60              RTS
                ;**** FINE DEL PROGRAMMA ****
                ;
                ;**** SUBROUTINE PER SCAMBIARE I PUNTI ****
                ;
                SWOP
AD 02 C3        LDA X2LO
8D 09 C3        STA TEMP
AD 00 C3        LDA X1LO
8D 02 C3        STA X2LO
AD 09 C3        LDA TEMP        ; SCAMBIA I BYTE BASSI DI X
8D 00 C3        STA X1LO
                ;
AD 03 C3        LDA X2HI
8D 09 C3        STA TEMP
AD 01 C3        LDA X1HI
8D 03 C3        STA X2HI
AD 09 C3        LDA TEMP        ; SCAMBIA I BYTE ALTI DI X
8D 01 C3        STA X1HI
                ;
AD 05 C3        LDA Y2
8D 09 C3        STA TEMP
AD 04 C3        LDA Y1
8D 05 C3        STA Y2
AD 09 C3        LDA TEMP        ; SCAMBIA Y
8D 04 C3        STA Y1
                ;
60              RTS
                ;
                ;**** FINE DELLA SUBROUTINE ****
```

ROUTINE LINEASUB IN CODICE MACCHINA

La routine in codice macchina traccia linee con inclinazioni maggiori, uguali o minori di uno. Non ci sono problemi con le linee verticali perché, usando il metodo che abbiamo descritto, non compaiono divisioni per zero. Per linee con inclinazioni minori di uno, valgono le regole viste in precedenza. Come già abbiamo detto, la routine esegue un test di verifica per determinare se DX è maggiore di DY (ovvero se l'inclinazione è maggiore di uno).

- La routine usa due byte per ciascuna coordinata X perché i valori di X possono superare 255. Y invece non può superare 199 e può quindi essere contenuto in un solo byte.
- Le locazioni impiegate da Plotsub per immagazzinare le coordinate dei punti, vengono definite all'inizio del listato Assembly assieme all'indirizzo di partenza di Plotsub.
- I bit 1 e 0 di REGNEG vengono selezionati in base ai risultati dei calcoli di DX e DY. Per accedere a ciascun bit separatamente, senza toccare gli altri, viene usata l'istruzione logica ORA.
- Per conservare i valori delle coordinate Y a singolo byte, durante la procedura di tracciamento, la routine si serve del registro Y. I valori contenuti nel registro non vengono modificati dalla routine Plotsub.

N.B. Per poter usare Plotsub assieme a Lineasub, si dovrà modificare il listato del codice sorgente (vedere a pag. 819), inserendo un'istruzione di salto tra SBC REMX e TAX ed il Programma Dimostrativo aggiungendo le seguenti linee:

```
5010 FOR I=HRSFLG TO
     HRSFLG+314: READ A
5380 DATA 169,7,237,8,193,240,6,
     170,14,13,193
5410 DATA 38944: REM
     "CHECKSUM"
```

La Artic Computing

Un altro classico esempio di software house letteralmente nata dal nulla

Richard Turner, un giovane studente squattrinato, iniziò a scrivere software nel 1981 ed i suoi primi programmi di gioco furono *Battleship* e *Trek* per il computer ZX80. Per superare le limitazioni imposte da questo apparecchio (lo ZX80 cancella lo schermo ogni volta che qualcosa viene spostato su di esso), Turner decise di dedicarsi allo sviluppo di giochi di strategia, che, non implicando necessariamente l'animazione grafica, potessero dare migliori risultati.

Il primo grande successo commerciale furono gli *Scacchi ZX*, che vennero lanciati sul mercato nell'estate del 1981.

La Artic Computing assunse l'attuale assetto commerciale verso la fine di quell'anno, ma andò incontro a notevoli problemi quando Turner accettò una borsa di studio dalla Ford Motor Company, per la facoltà di Ingegneria Elettronica dell'Imperial College, a Londra. I suoi studi durarono solo un anno, alla fine del quale egli decise di interromperli per breve tempo e si dedicò al rilancio della compagnia.

Da allora Turner non ha più rimesso piede all'Università!

La prima idea di fondare l'Artic gli era venuta quando abitava ancora, coi propri genitori, a Hull. Ma quando, nel giugno dell'83, il catalogo dei programmi offerti dalla compagnia raggiunse i 93 titoli, Turner decise di trasferirsi negli attuali uffici di Brandesburton.

Da allora la Artic ha assunto nuovo personale: oggi è composta da 15 impiegati, di cui tre si occupano esclusivamente delle ordinazioni telefoniche, e 5 programmatori a tempo pieno.

La Artic sta progettando di impiantare una propria catena di negozi in tutta la Gran Bretagna, sotto il nome di Artic Software Station, che venderà, oltre al proprio software, anche prodotti della concorrenza. Una prima "stazione" è già stata aperta ad Acton, un quartiere di Londra, nel luglio del 1984. L'originalità di questo punto di vendita è che non si trova all'interno di un grosso centro commerciale, ma dista qualche chilometro dalla grande area del West End di Londra. Jeff Raggett, il direttore londinese della Artic, ha giustificato questa scelta dichiarando che il costo degli affitti nelle aree ad alto insediamento commerciale è divenuto troppo oneroso per negozi con un giro d'affari pari a quello del nuovo negozio Artic. Inoltre, la sua posizione decentrata permette di avere un rapporto più continuativo e personale con i clienti.

La Artic ha introdotto in commercio un'altra novità: i "cofanetti" espositivi che possono contenere fino a 64 cassette ciascuno. Si tratta di nuovi tipi di raccoglitori, venduti a tutti gli operatori commerciali che trattano il software della Artic, che consentono di mostrare assieme, anche nei negozi più piccoli, l'intera gamma dei prodotti Artic. Pare che questi contenitori abbiano riscosso un notevole successo di pubblico.

La Artic si sta preparando seriamente anche per il mercato europeo e d'oltreoceano, tramite una particolare campagna commerciale. Per quanto riguarda gli Stati Uniti, la Artic ha già stipulato un contratto di scambio con due note software house americane, la Softsync e la International Publishing, per la distribuzione dei rispettivi prodotti nei paesi di appartenenza. Sino ad oggi, i giochi di maggior successo della Artic sono stati il *Bear Bovver* (di cui sono state vendute oltre 40.000 cassette), il *Galaxianus* ed il *Gobbleman*.

L'"ultimo nato" è invece un gioco per lo Spectrum chiamato *World Cup*, che in soli tre mesi ha venduto oltre 5.000 copie.

I giochi... artici!

Ecco alcuni dei giochi della gamma Artic. Tra di essi, il vendutissimo *Bear Bovver* ed il recente *World Cup*, una versione per lo Spectrum della famosissima Coppa del Mondo di Calcio



IAN McKINNELL



Il mondo in un computer

La marca di un computer non rivela la provenienza dei suoi componenti

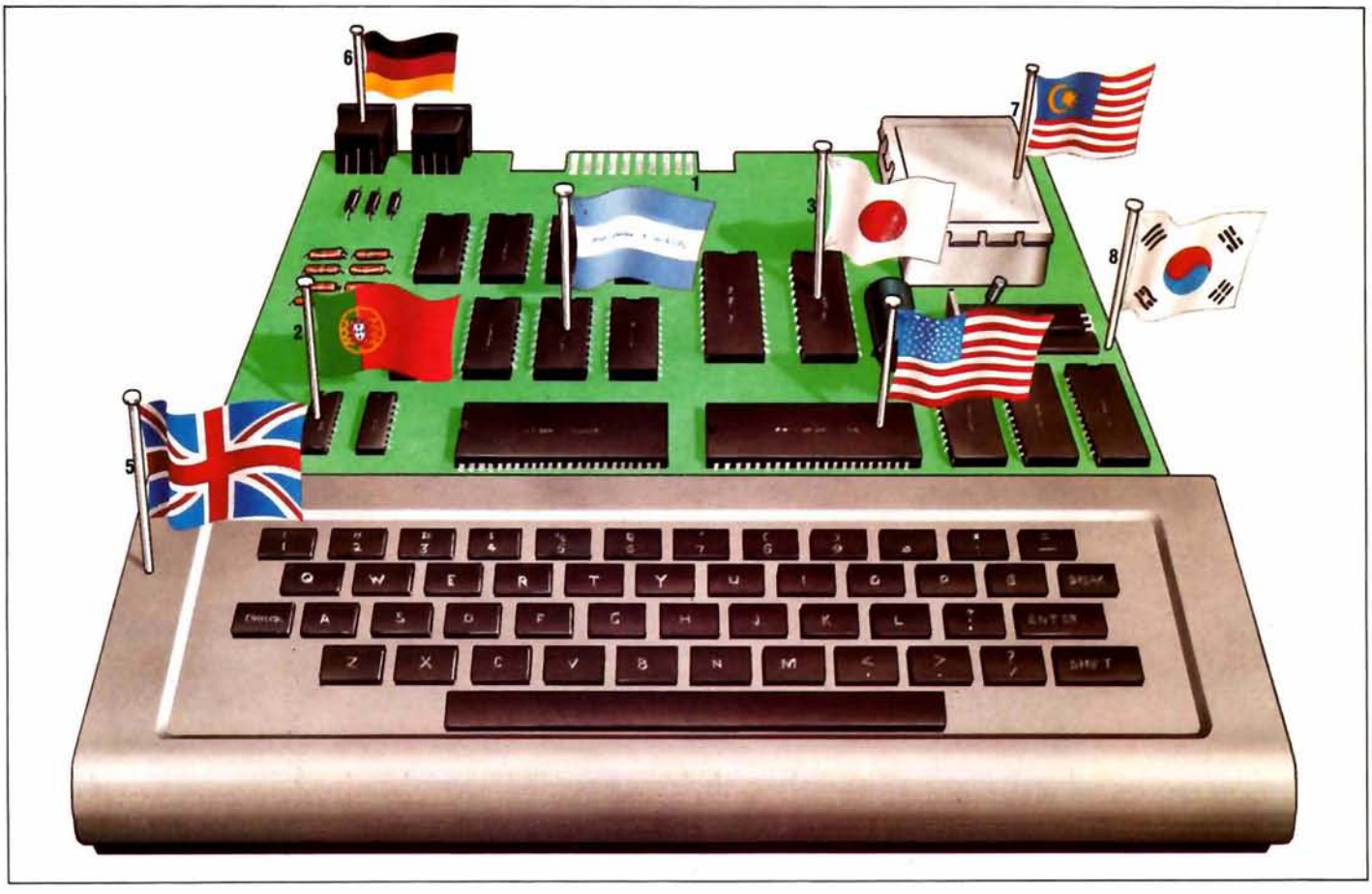
La fabbricazione dei micro è un vero "intrigo internazionale". L'Amstrad CPC 249, ad esempio, viene costruito interamente in Corea e buona parte del Micro BBC della Acorn viene assemblata a Hong Kong. La Sinclair si è sempre limitata esclusivamente alla ricerca e alla progettazione, subappaltando poi ad altre compagnie la fabbricazione dei vari componenti e l'assemblaggio finale.

Queste scelte sono dovute alla necessità di mantenere più bassi possibile i costi di produzione e, quindi, i prezzi degli apparecchi. Sotto quest'ottica, dunque, le schede a circuiti stampati devono essere semplici e di piccole dimensioni, oltre a prevedere un numero limitato di chip (non per il loro prezzo, ma perché l'impiego di numerosi chip rende i circuiti più complessi, riducendo l'affidabilità dell'apparecchio).

Questa è la ragione per cui gran parte degli home computer più comuni si avvale dell'ULA (Uncommitted Logical Array). L'ULA è un chip piuttosto costoso da progettare e produrre ma può sostituire, sulla scheda, fino ad una dozzina di chip di minori dimensioni. La maggior parte dei microchip viene fabbricata nella cosiddetta 'Silicon Valley', in California, il più importante centro mondiale per la fabbricazione di questi componenti. I chip, dopo essere stati costruiti, vengono racchiusi in piccoli contenitori di plastica o ceramica: questa fase della fabbricazione tuttavia, non richiede tecnologie altrettanto intensive ed avanzate (vedere a pag. 121) e viene quasi sempre affidata a compagnie estere, considerando l'alto costo della manodopera in USA. Terminata la fase di progettazione di una scheda, inizia la ricerca di un subappaltatore. Anche la fabbricazione di una scheda a circuiti stampati, richiede un notevole investimento di capitali in attrezzature ed impianti, ma esistono numerose industrie specializzate proprio in questo ramo. Dopo numerosi passaggi, basandosi su disegni "in trasparente" dei circuiti, il fabbricante produce la scheda, sulla quale si trovano le piste in rame a cui verranno poi saldati i vari componenti elettronici.

Intrigo internazionale

Questo immaginario microcomputer è stato fabbricato con componenti prodotti in varie parti del mondo. I chip sono stati prodotti in: El Salvador (1), Portogallo (2), Giappone (3) e USA (4). La custodia, la tastiera e l'assemblaggio finale sono stati realizzati in Gran Bretagna (5) e i vari connettori in Germania Ovest (6). Infine, il modulatore RF è prodotto in Malesia (7) e la scheda del circuito stampato in Corea (8)



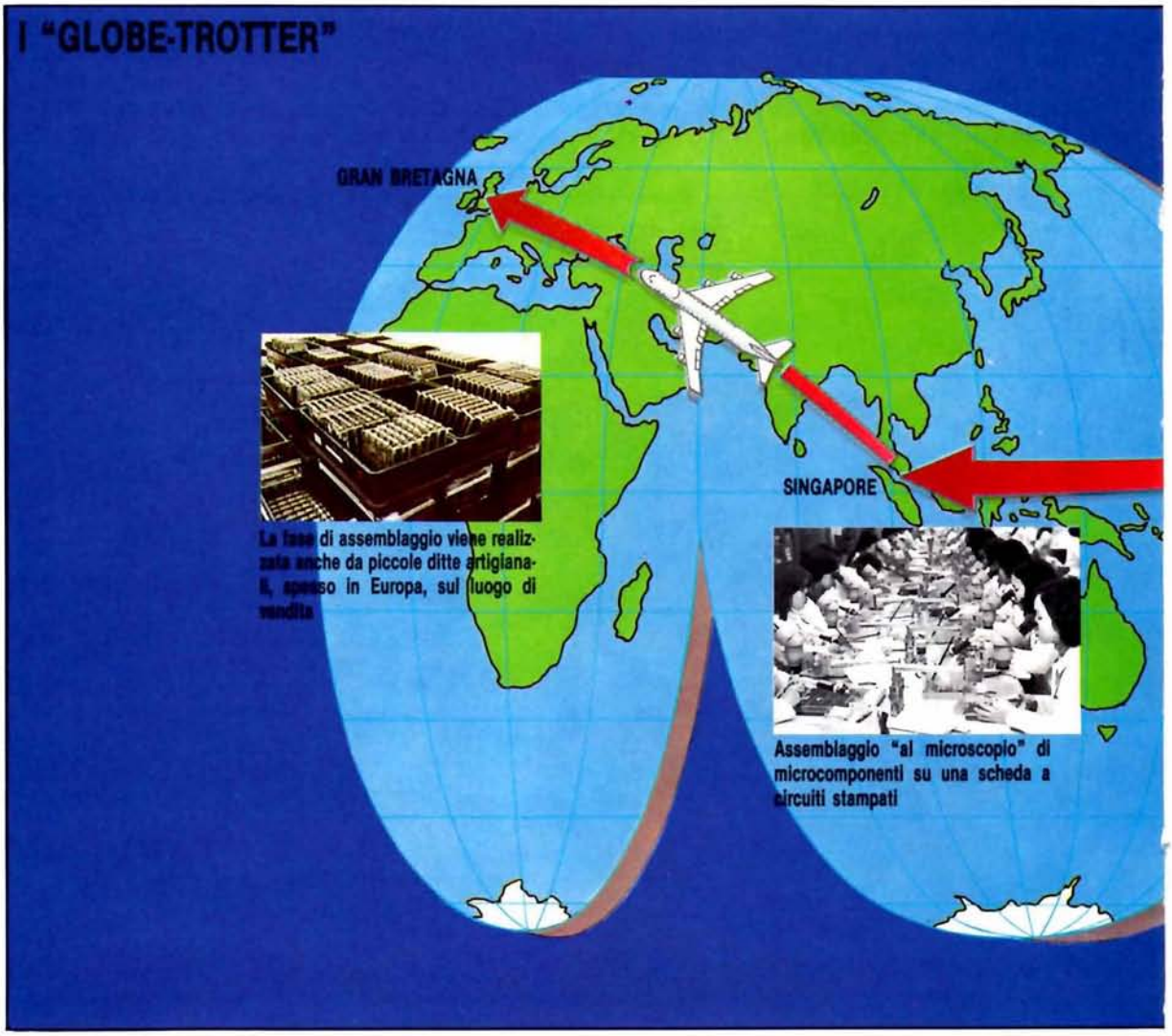
STEVE CROSS

Anche nella progettazione delle schede va tenuto conto dei costi di produzione. Infatti, è possibile ottenere schede "multistrati" (equivalenti a più circuiti stampati sulla stessa scheda, separati da materiale isolante), ma questo metodo viene quasi sempre scartato dai progettisti perché troppo caro. Invece, un requisito molto importante per le schede dei computer, è quello della placcatura dei fori nei quali verranno poi saldati i componenti, che garantisce un perfetto collegamento elettrico fra componenti stessi e circuito e, eventualmente, tra le stesse piste in rame situate sui due lati della scheda.
Tutti gli altri componenti, ossia alimentatori, modulatori, connettori, tastiere, e anche l'involucro plastico dell'apparecchio (che richiede dispendiose apparecchiature di stampaggio), vengono acquistati da vari produttori in tutti i paesi del mondo ed inviati ad un nuovo subappaltatore per l'assemblaggio finale.
Questa fase può essere eseguita in due modi: con linee di assemblaggio automatiche, oppure manualmente a seconda del costo della manodopera. La prima tecnica viene di solito adottata negli USA, in Europa e in Giappone, la seconda in Corea, a Singapore e a Hong Kong. Le linee di assemblaggio automatico sono costituite da "robot" che inseriscono sulla scheda i vari componenti, prelevandoli da apposite "bandoliere" o caricatori. Ai tecnici, che controllano le linee, è affidato il compito di rimpiazzare i caricatori vuoti.
Qualunque sia il sistema usato, tutti i componenti vengono inseriti nella scheda in modo che i piedini di connessione sporgano al di sotto di essa. Le schede vengono quindi passate in una "saldatrice a onda", che ricopre la parte sporgente dei piedini in stagno fuso. Questo, per un effetto di capillarità, risale lungo i fori e collega i componenti ai circuiti stampati della faccia superiore.
A questo punto la scheda è pronta: dopo il collaudo, viene inserita nella custodia del micro, spedita ai depositi attraverso le reti di distribuzione, e venduta. Queste ultime fasi sono più complesse di quanto si creda ed implicano problemi di vario genere.

Linee di distribuzione

Il primo problema è quello dei tempi: tutte le parti del computer, provenienti dai diversi produttori, devono arrivare contemporaneamente, in un'unica località, per l'assemblag-

In giro per il mondo
Questa schematica cartina del mondo illustra gli spostamenti dei componenti dei microprocessori nelle varie fasi di assemblaggio



gio. Il responsabile di questo settore è uno degli anelli principali della catena di produzione, poiché deve essere in grado di acquistare le varie parti al minor prezzo possibile, e di programmare e sincronizzare l'arrivo delle forniture. Tutti i fabbricanti di computer ben conoscono i disastrosi effetti sulla produzione e sulle vendite, causati dal ritardo anche di un solo componente! Un secondo problema è il costo delle forniture. Ad esempio, anche una piccola differenza di poche decine di lire nel prezzo di un connettore, può portare ad un aumento dei costi di produzione tale da riflettersi negativamente sulla vendita del micro.

L'assemblaggio di un computer, sia esso automatico o manuale, è un'altra area soggetta ad inconvenienti: può accadere che i componenti vengano inseriti sulla scheda dal di sotto o in posizione invertita, oppure che vengano del tutto omessi, danneggiando irreparabilmente il prodotto finito. Può anche accadere che la saldatrice a flusso non riesca a saldare il piedino di un chip oppure che un'intera fornitura non abbia le dovute caratteristiche tecniche.

Tutto ciò dimostra l'importanza dei collaudi preliminari tanto dei singoli componenti, quanto della scheda finita. Molte ditte di assemblaggio per micro svolgono saltuari controlli sui componenti in arrivo, ma tutte eseguono collaudi più o meno sofisticati sul prodotto finito. Queste verifiche sono costose e vengono eseguite da computer molto potenti.

Un'altra pratica diffusa, per garantire il funzionamento dei prodotti, è la verifica effettuata da un incaricato della ditta produttrice, nella stessa sede del cliente (in questo caso il rivenditore).

Il subappaltatore che fabbrica l'Oric ha ideato un metodo di verifica molto originale: tutti gli apparecchi finiti vengono pesati prima della spedizione. Un peso inferiore allo standard, infatti, segnala inequivocabilmente l'omissione di qualche componente in fase di assemblaggio.

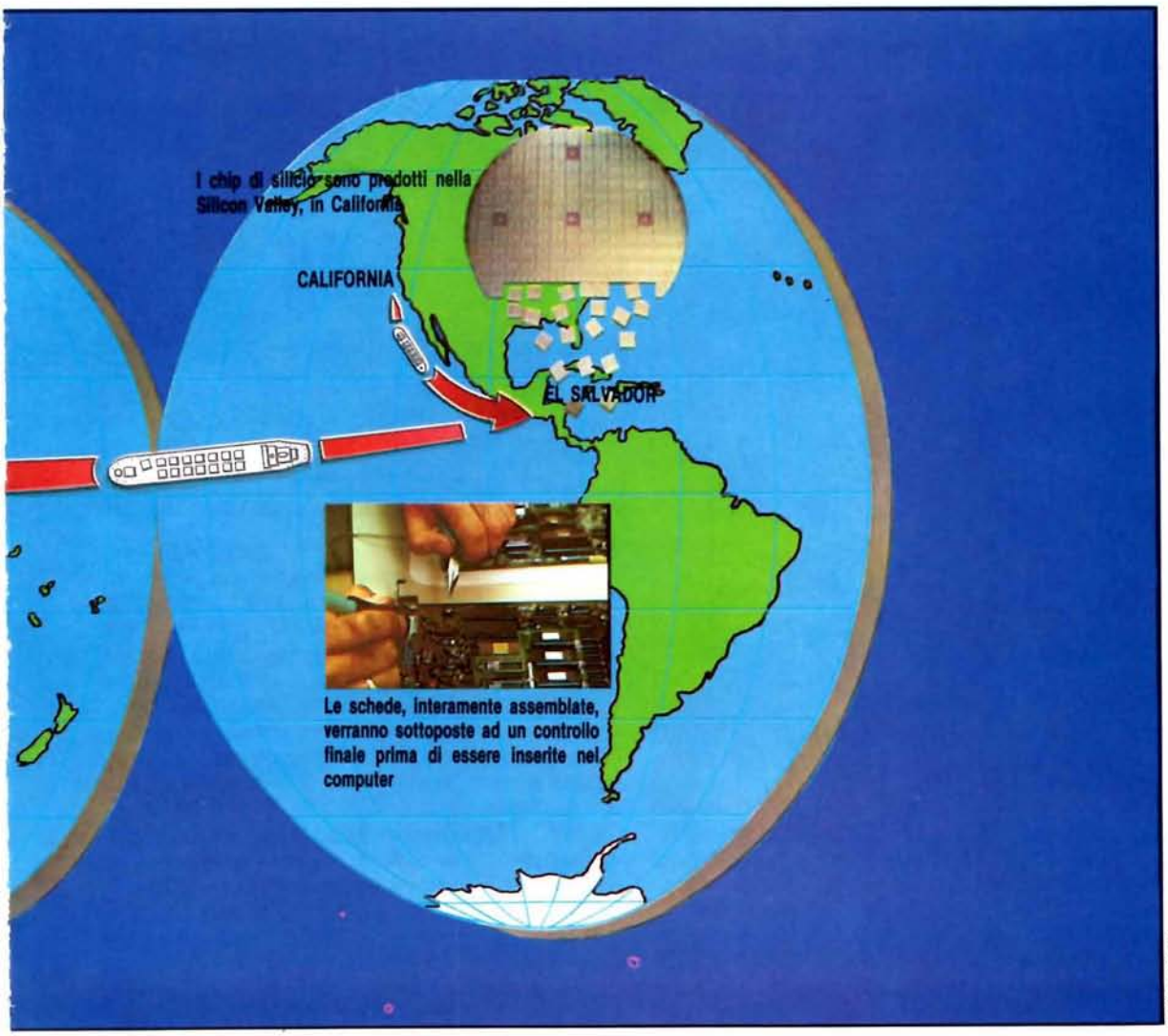
Ma il test di funzionamento definitivo si fa solo collegando l'apparecchio ad un televisore.

I fabbricanti di computer gestionali spesso adottano "test d'immersione" o "di rodaggio" di uno o due giorni: per tutto questo tempo il computer esegue le proprie routine interne o software di prova per verificare che tutto funzioni a dovere.



Un "taglio ai costi"

Il trasporto delle varie parti di un computer, da un paese ad un altro, ha come obiettivo la diminuzione dei costi di produzione e quindi, dei prezzi al dettaglio. Subappaltando la produzione dei componenti a ditte che operano in paesi nei quali il costo della manodopera è relativamente basso, i fabbricanti riescono a ridurre in modo significativo le loro spese. Oggi, tuttavia, grazie ai recenti progressi nel campo dell'automazione, anche i paesi più industrializzati riescono a produrre internamente i loro prodotti a costi decisamente concorrenziali. L'Atmos della Oric, ad esempio, viene completamente fabbricato in Gran Bretagna



KEVIN JONES

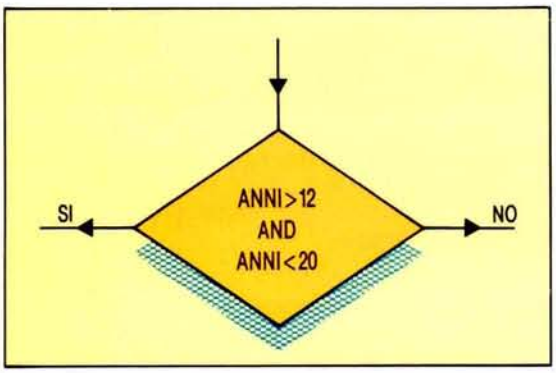
Mosse decisive

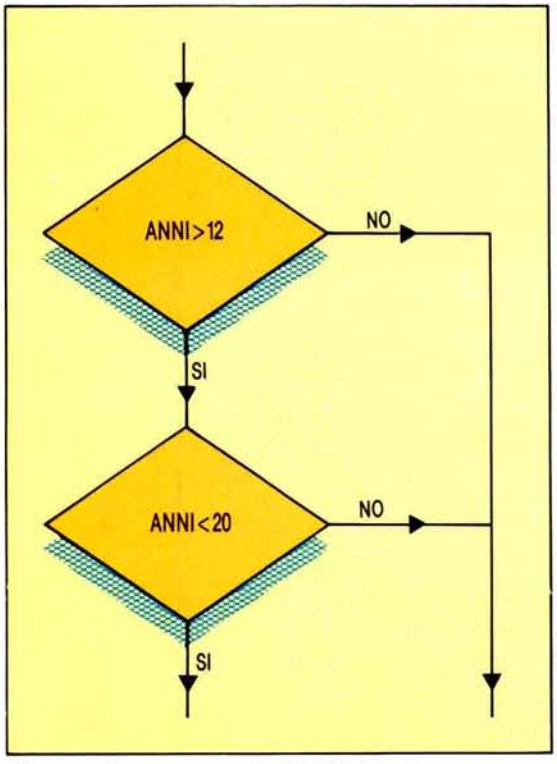
Le diramazioni complesse sottoforma di diagrammi e tavole decisionali

Nei programmi, spesso vengono usate espressioni multi-decisionali quali:

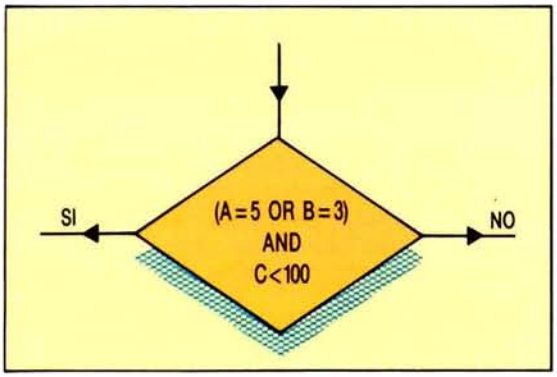
```
IF ANNI>12 AND ANNI<20 THEN STATO="TEENAGER"
```

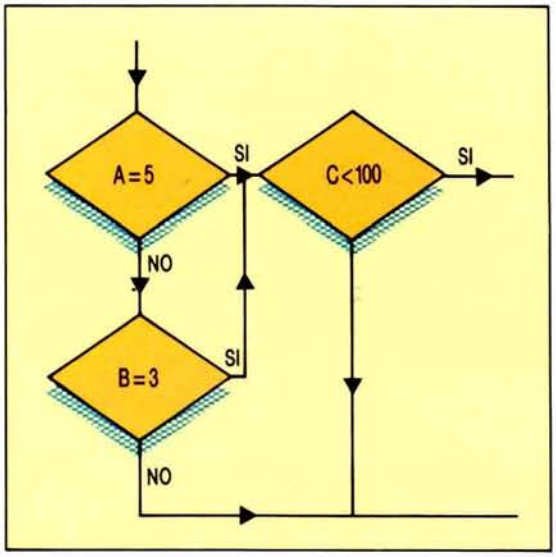
Gli algoritmi con istruzioni di questo tipo risultano però più facilmente comprensibili se il blocco multi-decisionale viene suddiviso nelle decisioni che lo compongono:





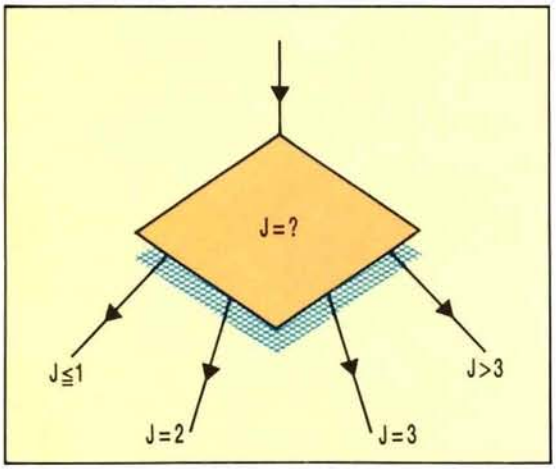
Il precedente esempio BASIC è stato rappresentato in forma di diagramma a blocchi o di flusso per evidenziare come la versione "composta" appaia meno chiara di quella "estesa". Nell'esempio seguente, formato da tre decisioni, il diagramma a tre blocchi decisionali è senza dubbio più comprensibile di quello a singolo blocco multi-decisionale. Inoltre, vengono evidenziate le analogie con le regole dell'algebra Booleana, che permette di costruire complicati circuiti combinando assieme semplici porte logiche.

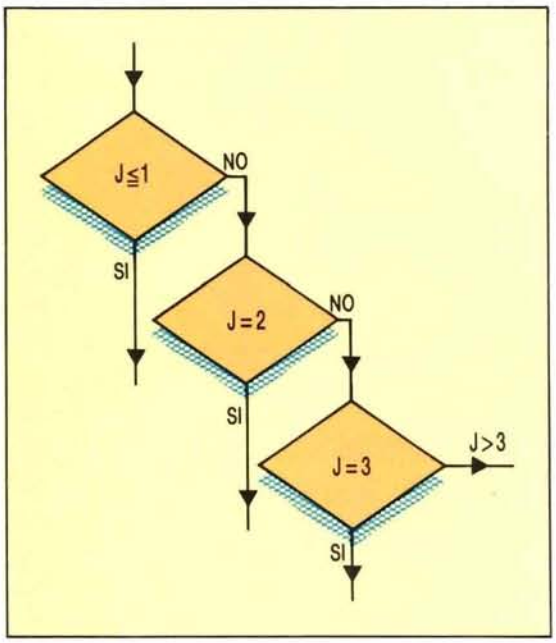




Negli esempi visti, tutte le decisioni sono esclusivamente di tipo binario, ma è abbastanza frequente che un algoritmo implichi decisioni con più di due uscite possibili. Ad esempio, quando in un programma si immette un carattere da tastiera, per indicare una delle opzioni di un menu, la procedura deve saltare ad una tra diverse subroutine possibili. A questo scopo, molti linguaggi di programmazione dispongono di istruzioni di salto multiplo, quali CASE...OF in PASCAL e ON...GOTO o ON...GOSUB in BASIC. Le regole su cui si basano le decisioni binarie sono valide anche per quelle multiple: il flusso logico in uscita da un blocco decisionale può seguire un solo percorso e tutte le uscite devono essere etichettate. In altre parole, le linee di flusso in uscita coprono tutte le possibilità e ciascuna esclude le altre. Una decisione mul-

tipla può essere rappresentata, come nell'esempio che segue, da una serie di percorsi di uscita, che originano da uno stesso blocco di decisione. Questa tecnica viene usata di rado: più spesso il blocco multi-decisionale viene suddiviso in decisioni binarie.
Le decisioni multiple si possono rappresentare come una sequenza di decisioni binarie.





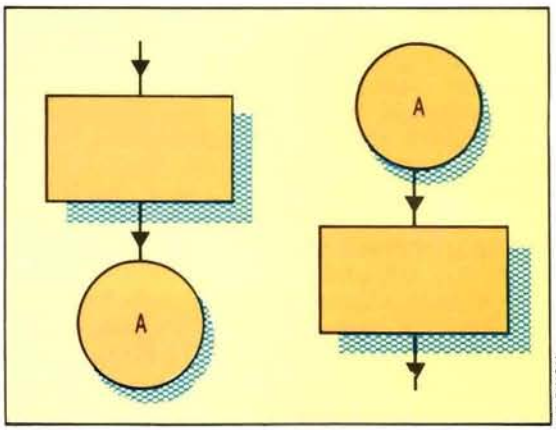
Di fronte a decisioni multiple, anziché il normale diagramma di flusso, conviene usare una "tabella decisionale". Quella che segue rappresenta una serie di regole per la determinazione delle decisioni ed è suddivisa in quattro sezioni: la descrizione delle condizioni contemplate dalle regole, la descrizione degli effetti delle varie decisioni, una griglia superiore per la combinazione tra condizioni e regole ed una inferiore che mostra gli effetti associati a ciascuna regola. Nella griglia "condizioni/regole", le celle contengono i valori delle variabili mentre, in quella "effetti/regole", il segno di "visto" indica l'effetto ottenuto ed il valore percentuale rappresenta il parametro assegnato a quel particolare effetto. La regola 4, ad esempio, va così interpretata: "Se il livello di gioco è 2, il giocatore è un "principiante" e viene premuto il pulsante di sparo, allora attivare le difese a raffica 'cieca' e ridurre del due per cento il livello energetico".

	REGOLE							
CONDIZIONI	1	2	3	4	5	6	7	8
PULSANTE DI "FUOCO" AZIONATO	✓	✗	✗	✗	✓	✗	✓	✗
LIVELLO DEL GIOCO	1	1	2	2	1	1	2	2
LIVELLO DEL GIOCATORE	P	P	P	P	E	E	E	E
EFFETTI								
AZIONE EVASIVA DELL'ALIENO							✓	✓
DIFESE A RAFFICA CIECA			✓	✓			✓	✓
RIDUZIONE LIVELLO ENERGIA	1%	1%	2%	2%	2%	2%	4%	4%

P = Principiante

E = Esperto

Spesso, le tabelle decisionali vengono adoperate per combinare decisioni semplici, formulandone così di complesse. Espresse in forma più elementare di quella qui illustrata, esse sono esattamente equivalenti alle tavole della verità usate per le porte logiche.
Un'ultima raccomandazione sull'uso dei diagrammi di flusso: è sempre consigliabile scrivere il diagramma su un unico foglio, evitando così di dover perdere tempo a voltare le pagine per seguirne lo sviluppo. Se ciò non è possibile, lo si potrà suddividere in algoritmi di minori dimensioni. Si rammenti che ogni algoritmo può essere usato, come singola istruzione, da molti altri algoritmi. In questo modo ogni routine di un programma potrà essere inserita, come un unico blocco di elaborazione, all'interno del diagramma di flusso dell'intero programma, anche se la routine considerata si serve di altre routine, che a loro volta ne usano altre e così via.
Se il diagramma di flusso non entra in una sola pagina, lo si potrà interrompere in un punto adeguato (ad esempio, una decisione) ed usare un cerchio contenente un simbolo d'identificazione del punto nella pagina seguente da cui riprende il flusso di controllo (indicato da un secondo cerchio contenente lo stesso simbolo). Una soluzione alternativa è considerare la parte eccedente come un algoritmo separato, riferirsi ad essa in un blocco di elaborazione e rappresentarla con un diagramma di flusso proprio.



Il Prism VTX5000

Un accessorio che offre al piccolo Spectrum la possibilità di ampie comunicazioni



Fin dagli inizi, il database della Prestel ha cercato di attirare nuovi utenti; ma l'impianto di collegamento era piuttosto costoso ed il volume di informazioni offerte dal database non superava poi di molto quello di un comune quotidiano. Tuttavia, con un modem ed un software adeguati, quasi tutti gli home computer possono accedere al Prestel. La Prism ha deciso di sfruttare questa attraente opportunità di mercato, creando il Micronet, un servizio interno al Prestel dedicato a notizie ed informazioni sui microcomputer. Ha quindi messo in vendita un modem ed un pacchetto software, che permettono di accedere al Micronet (quindi al Prestel), riscuotendo un immediato successo. Il numero di utenti e la gamma di servizi informativi a loro disposizione è in costante aumento.

Con la decisione di produrre uno strumento per collegare lo Spectrum della Sinclair al Micronet, la Prism ha accettato una difficile sfida poiché lo Spectrum non è particolarmente adatto a questo tipo di applicazioni: è sprovvisto di una porta RS232 o di un'interfaccia seriale e non può quindi collegarsi ai comuni modem; ha un display di 24 righe per 32 colonne, mentre il Prestel prescrive un formato 24×40 ed un complicato sistema grafico "teletext". La Prism ha creato un'unità progettata appositamente per questa applicazione: il modem Prism VTX5000.

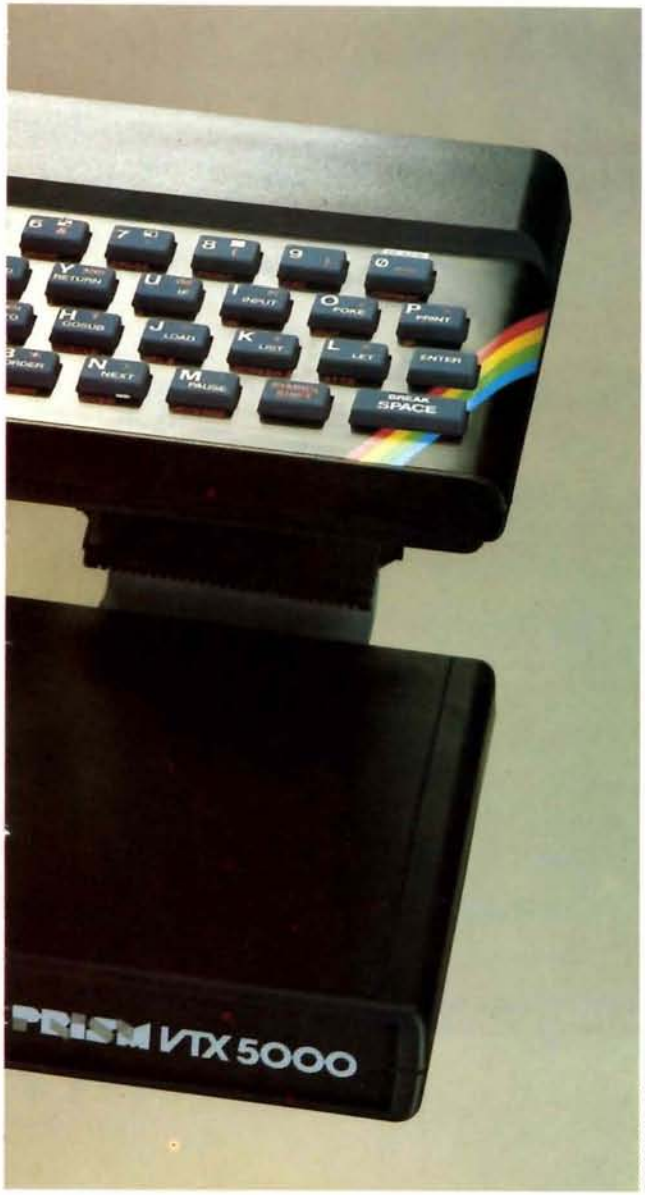
Questo apparecchio contiene tutte le interfacce necessarie, un modem a innesto diretto con velocità di trasmissione di 1200/75 baud ed il software occorrente, che non solo fornisce tutte le funzioni standard per accedere al Prestel ma usa le capacità grafiche dello Spectrum per simulare un vero video teletext 40×24. Con una spesa molto ragionevole, gli utenti dello Spectrum possono quindi acquistare l'hardware necessario per collegarsi al Micronet.

Il VTX5000 può essere collocato sotto lo Spectrum, collegandolo all'apposito connettore per espansioni. Il cavetto a nastro è dotato di una terza presa, che permette di collegare al sistema altre periferiche come una stampante o il Microdrive. L'unità viene connessa direttamente alla linea telefonica ed è quindi priva di accoppiatore acustico: questa particolarità garantisce una ricezione molto più affidabile.

Per istallare il Prism VTX5000, però, l'impianto telefonico deve possedere i nuovi tipi di presa recentemente introdotti in Gran Bretagna e negli USA: si dovrà staccare la spina del telefono, sostituirla con quella del modem ed infine collegare il telefono al modem. Con questa tecnica è sufficiente un'unica presa telefonica per mettere in funzione l'intero sistema. Una volta collegato, il telefono funziona normalmente.

All'accensione, lo Spectrum esegue automaticamente il software Micronet, residente nella ROM del modem (senza doverlo caricare ogni volta). Il pacchetto Micronet è del tutto simile a quelli di corredo agli altri micro e lo si può utilizzare anche su apparecchi di tipo diverso. Il software viene controllato tramite una serie di menu e non presenta particolari difficoltà d'uso.

La prima opzione del menu è quella di "logon" (allacciamento): l'utente deve digitare sulla tastiera il proprio numero d'identità, a 10 cifre, assegnatogli dal Prestel. Questo numero viene memorizzato dal computer per tutto il tempo che l'apparecchio resta acceso, perciò non occorre impostarlo nuovamente, anche se l'utente richiama più volte il Prestel. Successivamente, occorre formare il numero telefonico di uno dei quattro computer che contengono il database Prestel. Ricevuto il



CHRIS STEVENS

segnale di risposta, si agisce sull'interruttore "Line" del modem e si rimette a posto la cornetta telefonica. D'ora in poi, lo Spectrum è in contatto con il Prestel. In questa fase, il sistema impedisce a chiunque di usare quel particolare codice Prestel e permette all'utente di modificarlo, ogniqualvolta lo ritenga necessario, per mantenerlo segreto.

Avuto accesso al database, il Prism può essere usato come un qualsiasi altro adattatore Prestel. Per passare da una pagina all'altra del database si dovrebbero battere i tasti (*) e (#), ma sullo Spectrum si usano, rispettivamente, Enter e Symbol Shift. Le altre funzioni offerte dal software Micronet possono essere richiamate in qualsiasi momento. È quindi possibile copiare le pagine del database (per registrarle su nastro o stamparle) oppure trasferire interi programmi dal sistema direttamente nella memoria dello Spectrum. Le pagine di "telesoftware" del Micronet offrono due tipi di programmi: quelli gratuiti e quelli a pagamento (generalmente, i primi sono semplici programmi di svago).

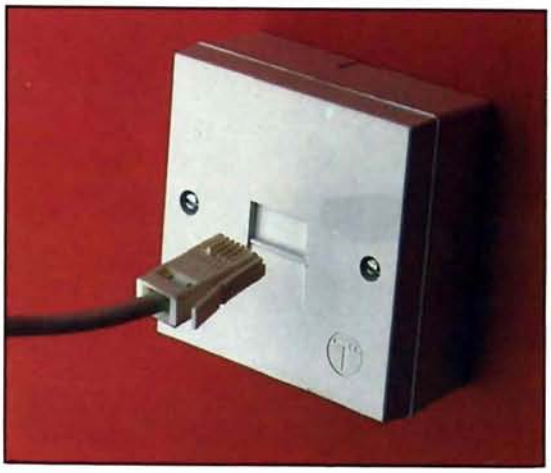
Il Micronet ed il Prestel oggi offrono un'immensa quantità di informazioni: hanno pagine di documentazione tecnica, giochi, pagine letterarie, comunicati pubblicitari e persino barzellette! Il Prestel consente anche lo scambio di posta elettronica tra gli utenti abituali del sistema. Il Micronet ha un rivale: il *Viewfax*, una specie di gazzettino che, in un'altra sezione del Prestel, offre un'ironica rubrica per microcomputer condotta da un misterioso "MicroGnomo".

Le potenzialità informative del Prestel sono destinate a crescere in futuro, ma alcuni degli inconvenienti iniziali traspaiono ancora, soprattutto nel Micronet: alcune pagine d'informazione sono superate e, talvolta, la procedura per passare da una pagina all'altra è poco chiara.

Il Prestel permette anche di comunicare direttamente con altri utenti Spectrum ed il VTX5000 consente di collegare via telefono due computer Spectrum per scambi di messaggi e programmi: la velocità di trasmissione è di 1200 baud. I primi modem venivano venduti senza il necessario software, ma oggi ogni apparecchio è dotato di una "cassetta", che facilita questo scambio di informazioni. Ovviamente, il software Spectrum consente di comunicare solo con altri Spectrum dotati di modem VTX5000.

Purtroppo i numerosi modelli di modem in commercio impiegano standard di comunicazione diversi, alcuni dei quali risultano "incomprensibili" al VTX5000. Forse il limite più grosso di questo modem è la mancanza di uno standard, che permetta di accedere alle reti informative create da appassionati di tutto il mondo. La velocità di trasmissione usata per queste comunicazioni è di 300 baud, troppo lenta per il Prism VTX5000!

I possessori di Spectrum, che volessero addentrarsi nel campo delle comunicazioni via computer, possono trarre notevoli vantaggi dall'acquisto di un'interfaccia RS232 (nel caso specifico, l'Interfaccia 1 dello ZX Sinclair) e di un modem polivalente (di uso generico). Tuttavia, questa scelta implica la stesura di un software proprio e la realizzazione di un impianto specifico. Il VTX5000, quindi, è uno strumento ideale per chi si limiti ad usare il Prestel.



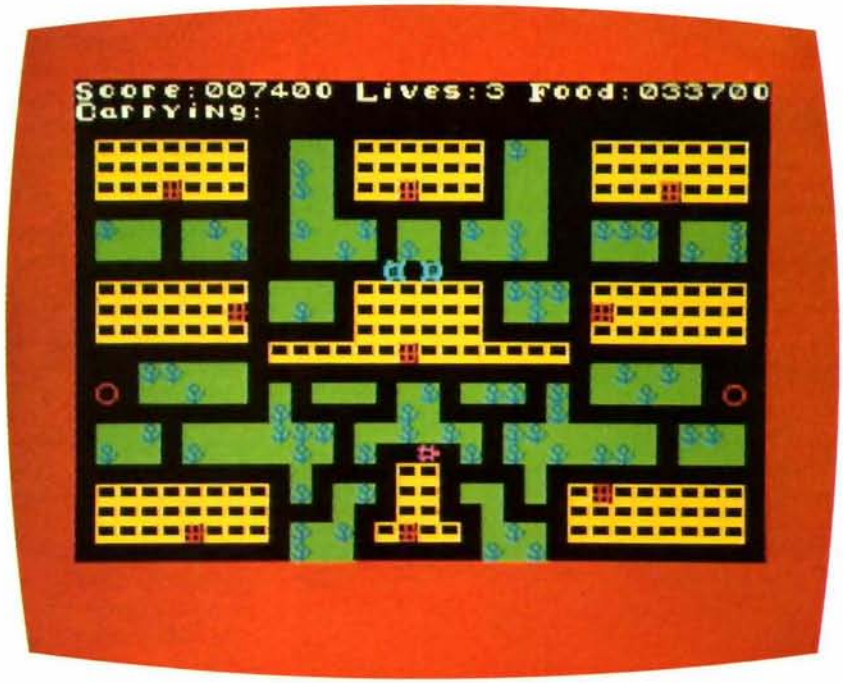
Presa telefonica modulare
Ecco le nuove prese telefoniche, installate in USA ed in Gran Bretagna. La presa accoglie uno spinotto a sezione quadrata di materiale plastico

Collegamento diretto
Il VTX5000 si innesta direttamente allo Spectrum, senza bisogno di altre interfacce. L'apparecchio telefonico viene collegato al VTX5000, a sua volta innestato sulla linea telefonica. Questo modem, però, funziona solo con le nuove prese modulari (tipo USA) e non con le spine telefoniche di vecchia concezione ancora molto diffuse in Europa

Software in vetrina

Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Nella vastissima gamma di giochi per il Commodore e per lo Spectrum la scelta si fa sempre più difficile, se non si ha almeno una vaga idea del tema e della qualità dei programmi. Ecco quindi qualche accenno ad altri due giochi coi quali divertirsi su questi computer.



Les Flics per lo Spectrum

Come si nota subito dallo schermo qui riprodotto, il gioco non si preannuncia come originalissimo: si richiama, infatti, ad una struttura abbastanza classica nei giochi tipo "Arcade". Lo scopo è quello di rubare il diamante "Purple Puma" (Puma purpureo), ma prima occorre "raccogliere" vari oggetti e comprenderne l'utilità. Nel frattempo si devono evitare le auto della polizia, i gendarmi e "Kaolin the Chef" (Kaolin il Cuoco). Evidentemente basato sulle avventure della Pantera Rosa (viene addirittura "suonato" il notissimo motivetto), il gioco sfrutta in modo molto appropriato la grafica dello Spectrum, ma rischia di annoiare dopo un po', essendo piuttosto ripetitivo. Il programma è prodotto dalla PSS (GB).



Tales of the Arabian Nights per il Commodore 64

In questo caso, lo spunto è fornito dai racconti di *Mille e una notte*. Gli scenari sono esotici ed il protagonista, il principe Imrahl, è impegnato nella ricerca della principessa Anita, per liberarla dal feroce sultano Saladino. Ogni successiva scena rappresenta una nuova "notte" durante la quale portare a termine un'impresa. In genere, si tratta di scovare delle brocche, su ognuna delle quali è incisa una lettera (tutte assieme formano la parola "Arabian"). Ovviamente, le cose sono rese più difficili dalla presenza di feroci animali, soldati e potenti cannoni, che mettono in pericolo la "vita" del principe (alias, il giocatore). Un fatto degno di nota è l'abilissimo tentativo, da parte dei programmatori della Interceptor (che produce il gioco), di sintetizzare alcune parole, basandosi soltanto sul generatore audio del computer: la voce assomiglia un poco ad un coro di rane gracidanti, ma è pur sempre comprensibile.

L'Amstrad CPC 464

Anatomia di un microcomputer a diffusione europea: tecnologie a confronto

Il computer Amstrad viene prodotto in due versioni, entrambe basate sulla stessa unità centrale, ma dotate di un monitor monocromo in un caso e a colori (standard RGB) nell'altro. L'alimentatore per il computer si trova all'interno del monitor ed il cavetto di connessione tra questo ed il computer serve sia per la corrente sia per i segnali video. Poiché l'Amstrad ha un registratore a cassette incorporato, è sufficiente un solo cavo per alimentare tutto il sistema.
Nel monitor monocromo lo schermo è di colore verde e le immagini sono molto luminose e nitide: ideali per attività gestionali e per la redazione di testi. Occasionalmente, l'immagine subisce qualche oscillazione (probabilmente dovuta alla vicinanza dell'alimentatore al tubo catodico). Il monitor a colori possiede invece una risoluzione media: la grafica a più colori dell'Amstrad viene riprodotta egregiamente, mentre non si può dire altrettanto dei testi visualizzati su 80 colonne.
Al monitor monocromo è abbinata una tastiera completa (tipo macchina da scrivere), dotata di una tastierina numerica separata. Tutti i tasti possono essere riprogrammati, per poter usufruire di caratteri diversi da quelli indicati. Inoltre, la tastierina può comportarsi come un set di tasti di funzione programmabili, ai quali è possibile abbinare stringhe lunghe fino a 32 caratteri. Ad esempio, un tasto può essere programmato per caricare oppure per listare un programma o, ancora, per cancellare lo schermo.
Il registratore a cassette è un apparecchio piuttosto standard: le operazioni di avvio e di arresto, nel caricamento e nel salvataggio dei programmi, vengono controllate dal computer. L'Amstrad possiede due velocità di caricamento, una a 1.000 l'altra a 2.000 baud, selezionabili tramite software. Quando carica un programma, il computer rileva automaticamente la corretta velocità di registrazione e mette a punto il sistema in modo adeguato. Normalmente, la velocità è fissata in 1.000 baud, ma anche quella a 2.000 baud, che offre ampi margini di sicurezza, viene spesso utilizzata nel software commerciale, prodotto per l'Amstrad.
Oltre a questi dispositivi interni, l'Amstrad può contare su un'ampia gamma di periferiche: attraverso un connettore (piuttosto rudimentale, a dire il vero!) il computer può essere collegato ad una stampante Centronics. Come memoria di massa esterna, è previsto l'impiego di unità a dischetti (non ancora sul mercato) e, tra non molto, il sistema potrà



CHRIS STEVENS

Opzioni monitor
La versione più economica dell'Amstrad possiede un monitor monocromo, mentre la più cara ha un monitor a colori (non è possibile adoperare il computer col normale televisore di casa). Le immagini sui due monitor appartengono a due diversi programmi: al "Wordhang", che è una versione del noto gioco "Hangman" e all'"Admiral Graf Spee", un gioco navale sulla Seconda Guerra Mondiale

disporre del linguaggio LOGO e del sistema operativo gestionale CP/M. L'apparecchio possiede una sola porta per joystick (tipo Atari), ma, poiché in molti programmi di gioco ne occorrono due, l'Amstrad è dotato di una coppia di joystick collegabili contemporaneamente nell'unica porta.

All'interno dell'apparecchio è montato un altoparlante (completo di controllo del volume), ma i segnali audio possono essere inviati anche ad un amplificatore esterno, che meglio si riproduce l'effetto stereofonico ottenibile con le tre "voci" separate di cui è dotato l'apparecchio: una voce viene convogliata all'altoparlante destro, una al sinistro e la terza viene egualmente ripartita tra i due. Con un programma adeguato, è possibile riprodurre suoni stereofonici di grande effetto (ad esempio, il rombo di un'astronave che attraversa la stanza).

Nonostante la complessità del sistema, il BASIC permette di programmare l'emissione dei suoni con molta semplicità: non solo è possibile iniziare e chiudere l'emissione di una nota ad un volume ed una tonalità prefissati, ma si può anche controllarne l'inviluppo. L'inviluppo del volume può essere programmato per imitare il suono di un pianoforte o di un telefono, mentre quello relativo alla tonalità può essere controllato in modo indipendente per emettere effetti come il suono di una sirena o il fischio di un treno. Tutte le voci, infine, possono essere fuse a "rumori" per simulare realistiche esplosioni e colpi di arma da fuoco.

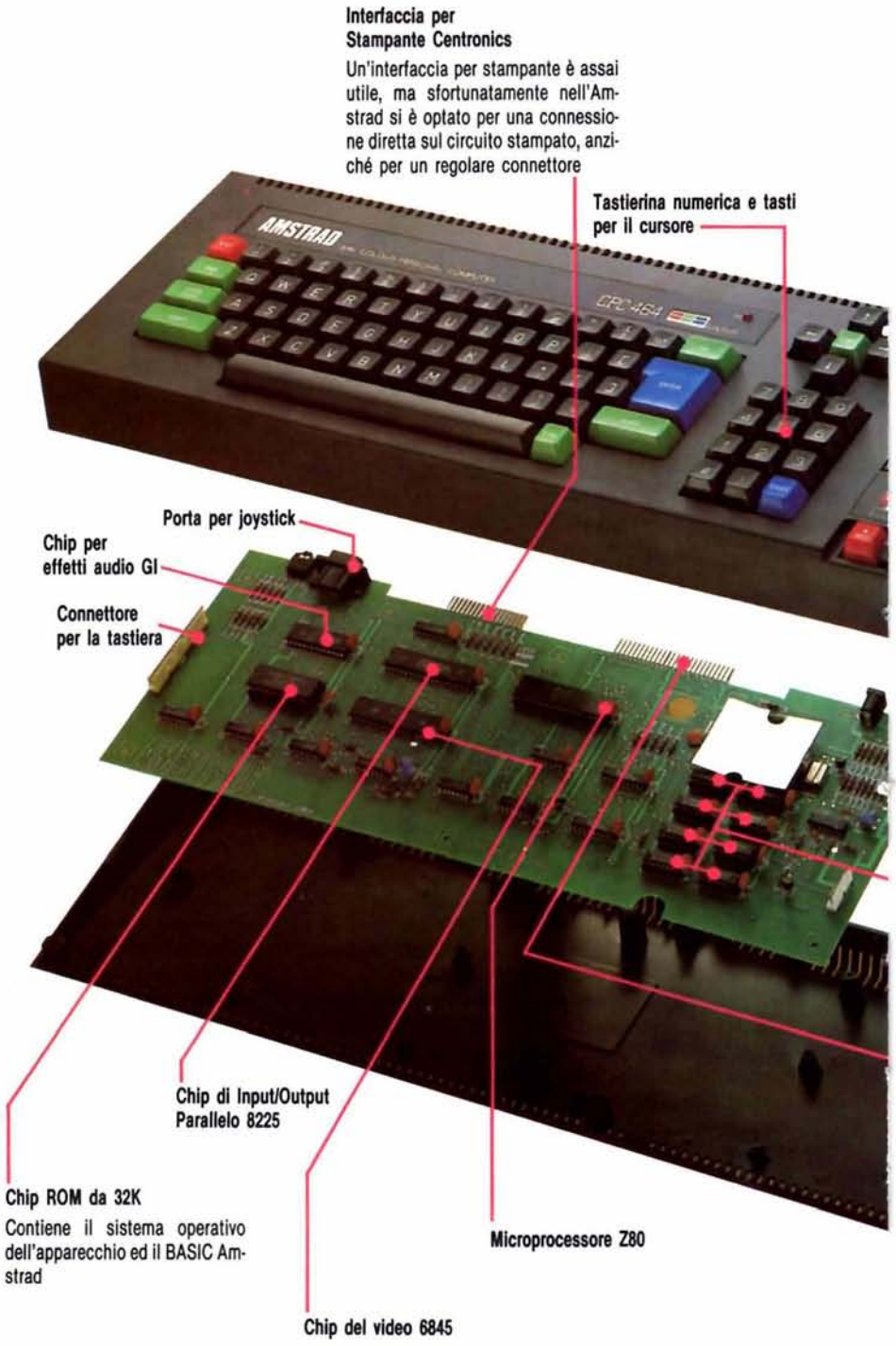
Comunque, la caratteristica che più colpisce nell'Amstrad è la grafica. L'apparecchio dispone di tre modi grafici, ciascuno dotato di una propria gamma di caratteri e di colori visualizzabili contemporaneamente sullo schermo. Tutti e tre i modi usano gli stessi 16 Kbyte di memoria, ma esiste un preciso rapporto tra il numero dei colori, la risoluzione ed il formato del testo.

Il modo di massima risoluzione consente di usare soltanto una coppia di colori alla volta, uno per lo sfondo ed uno per le immagini in primo piano su un video a 80 colonne, ma con un'impressionante risoluzione grafica di 640 × 200 pixel. All'altro estremo, il modo a 20 colonne può visualizzare simultaneamente fino a 16 colori. Il terzo modo grafico possiede invece un formato video a 40 colonne e 4 colori.

I colori dei vari modi grafici vengono scelti da una "tavolozza" di 27 colori, che comprende varie tonalità di rosso, verde, blu, ecc. Indipendentemente dal modo selezionato, il colore del contorno, sullo schermo, può essere scelto direttamente dalla tavolozza. Qualunque colore può essere fatto "lampeggiare", alternandolo ad un altro, a diverse velocità.

La grafica dell'Amstrad si adatta bene alle tecniche di animazione, e la totale assenza di sprite viene in parte compensata dalla buona gestione dello schermo, consentita dal BASIC.

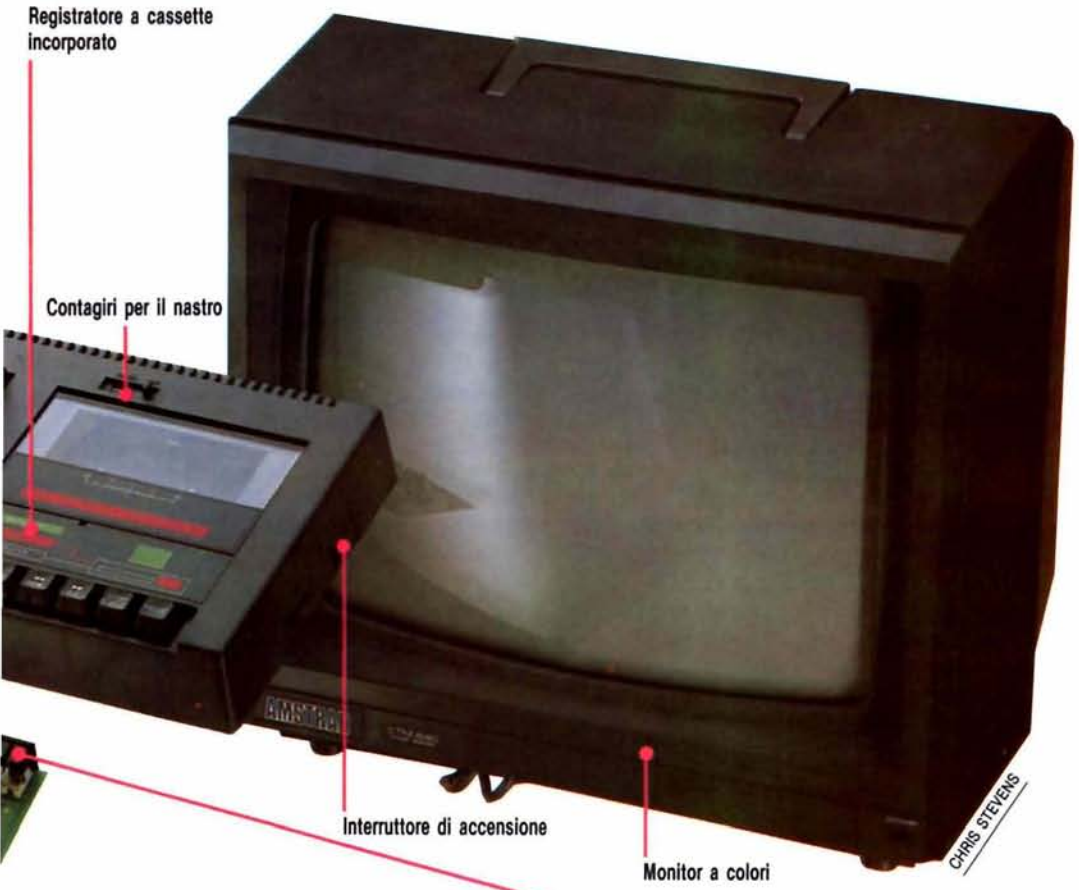
I 16 Kbyte di memoria, interamente dedicati al video grafico, non incidono sulla quantità





I JOYSTICK

L'Amstrad ha una sola porta per joystick, adatta ai modelli che adottano lo standard Atari. Nonostante ciò, la stessa Amstrad produce un proprio joystick, dotato di una presa che permette di collegarne un secondo. I segnali emessi da quest'ultimo vengono ritrasmessi al computer dal joystick principale



Uscita per il monitor
A questa porta sono disponibili due tipi di segnale: uno per monitor monocromo e l'altro per monitor a colori RGB. Entrambi i monitor, quindi, vengono collegati a questa porta

Porta d'espansione
La più interessante tra le periferiche collegabili a questa porta è l'unità a dischetti, che trasforma l'Amstrad in un apparecchio funzionante sotto il sistema operativo CP/M

AMSTRAD CPC 464

DIMENSIONI

Tastiera/registratore:
mm. 565 x 170 x 70
Monitor a colori:
mm. 380 x 350 x 350

CPU

Z80

MEMORIA

64 Kbyte di RAM (42 Kbyte dei quali per i programmi BASIC); 32 Kbyte di ROM

VIDEO

Funzionamento in tre "modi" (grafici/testuali):
risoluzione 640 x 200 (2 colori)
risoluzione 320 x 200 (4 colori)
risoluzione 160 x 200 (16 colori)
I colori, 27 in tutto, si scelgono da una "tavolozza"

INTERFACCE

Per joystick, per stampante tipo Centronics, bus d'espansione (per unità a dischetti), uscita audio (stereo), uscita per monitor

LINGUAGGI FORNITI

BASIC (incorporato), PASCAL (su cassetta)

TASTIERA

Tipo macchina da scrivere, con 74 tasti e tastierina numerica separata. Tutti i tasti sono riprogrammabili

DOCUMENTAZIONE

Il manuale di guida per i principianti è sufficientemente chiaro ma redatto in lingua inglese! È anche disponibile un manuale di consultazione per il BASIC ed una guida tecnica all'apparecchio

PRO

Il registratore incorporato ed il monitor a colori fanno dell'Amstrad un home computer molto compatto. L'ampia memoria disponibile è più che sufficiente, la grafica è ottima, come anche la generazione di effetti audio

CONTRO

Il registratore a cassette non sembra di costruzione molto robusta. I comandi grafici del BASIC non consentono (almeno nella versione attuale) di sfruttare al massimo le notevoli capacità offerte dall'ottimo hardware

UNITÀ A DISCHETTI

La Amstrad sta producendo un'unità a dischetti per il CPC 464. Nel prezzo sarà compreso il sistema operativo CP/M, indispensabile per la maggior parte dei programmi gestionali



di memoria a disposizione per i programmi dell'utente: la RAM video e la ROM del BASIC, infatti, occupano la stessa area nella mappa di memoria del processore.
Un apposito chip abilita l'una o l'altra a seconda delle necessità, quindi per i programmi BASIC e per i dati rimane sempre libera un'area di 42 Kbyte.
Il BASIC usato sull'Amstrad è uno dei dialetti più sofisticati e ben si integra con l'eccellente hardware per la grafica. Il sistema possiede molte utili funzioni che facilitano la creazione di immagini sullo schermo. L'origine delle coordinate grafiche può essere trasferita dall'angolo in basso a sinistra in qualsiasi altro punto del video. È inoltre possibile aprire sullo schermo una finestra grafica per restringere le operazioni grafiche ad un'area utente. Per di più, è possibile ampliare la RAM standard (64 Kbyte), intervenendo opportunamente sulle "pagine" della mappa di memoria.
La caratteristica più originale del BASIC Amstrad riguarda la gestione degli "interrupt". Molti micro consentono a chi programma in codice macchina di utilizzare il sistema degli interrupt per controllare le routine in codice macchina: il BASIC Amstrad fa un ulteriore passo avanti, permettendo di usare gli interrupt dall'interno di un programma BASIC. In BASIC, il comando AFTER introduce una data pausa prima di rinviare un programma ad una particolare subroutine, ed il comando EVERY ripete la stessa operazione più volte. Questa avanzata funzione permette di sviluppare con facilità programmi in



LA "AMSOFT"

La divisione software della Amstrad, denominata Amsoft, ha compiuto una notevole "performance", sviluppando numerosi programmi per l'Amstrad, prima ancora del suo lancio sul mercato. La produzione è stata orientata soprattutto sui giochi, modificando versioni originariamente scritte per altri apparecchi. Oggi molte Case di software stanno lavorando su programmi adatti all'Amstrad, quindi in futuro la scelta sarà piuttosto vasta. I giochi qui raffigurati sono (da sinistra): "Roland on the ropes" e "Roland in the caves", prodotti dalla Indescomp, e "Oh Mummy" creato dalla Gem Software

CHRIS STEVENS

limitata. Il modo testo prevede fino ad 8 diverse finestre sulle quali visualizzare di volta in volta i testi scritti. Ad esempio, una finestra può contenere una serie di domande, mentre su un'altra vengono visualizzate le risposte.
Purtroppo la grafica BASIC è priva di una funzione per colorare un'area dello schermo: mancano i comandi per tracciare figure "piene" o per riempire di colore l'interno di una figura già disegnata. L'unico modo per ottenere un'immagine colorata è tracciare una fitta serie di linee l'una vicina all'altra, ma i risultati sono mediocri. Tuttavia, dato che il BASIC si trova su ROM, può darsi che venga in futuro prodotta una versione che elimini le attuali carenze. In modo simile, il sistema potrebbe essere arricchito con nuovi linguaggi quali il PASCAL, il FORTH ed il LOGO, senza per questo intaccare i 42 Kbyte di RAM tempo reale: da quelli per la raccolta di dati, ai giochi stile "Arcade".
L'Amstrad è uno dei pochissimi home computer che adottino un monitor, anziché un comune televisore.
Tuttavia, chi ha acquistato la versione a monitor monocromo, può occasionalmente aver bisogno di usare un monitor a colori: in tal caso l'apparecchio deve essere interfacciato tramite uno speciale adattatore video ma, poiché il sistema viene alimentato attraverso il monitor monocromo, sarà necessario collegare contemporaneamente entrambi i monitor all'Amstrad.
La stupenda grafica e l'affidabilità dell'hardware, l'avanzato dialetto BASIC e la sua potenziale espansione (il tutto ad un prezzo relativamente contenuto), fanno dell'Amstrad CPC 464 uno dei più sofisticati home computer oggi sul mercato.

Il Sabre Wulf

Un altro gioco di successo, prodotto dalla Ultimate, Play the Game

Poiché le vendite dei giochi d'azione stile "Arcade" stanno già subendo una flessione, la tendenza delle case di software è orientata alla produzione di giochi che uniscano elementi di strategia a spunti avventurosi. *Sabre Wulf*, della "Ultimate, Play The Game", segue appunto questa nuova formula.
Spesso i giochi stile "Adventure", nei quali il giocatore deve guidare il protagonista attraverso le più disparate situazioni e risolvere rebus strada facendo, presentano tediosi "tempi morti". L'azione rimane in sospeso per lunghi periodi, mentre il giocatore cerca di trovare una risposta ad un quesito apparentemente irresolubile. I giochi stile "Arcade", d'altra parte, non spingono certo alla riflessione e richiedono esclusivamente una buona prontezza di riflessi e agilità nelle dita.
Sabre Wulf cerca di combinare le caratteristiche migliori di entrambi i generi. Fondamentalmente, si tratta di un labirinto, ambientato in una giungla, con vaghi richiami al notissimo *Pac Man*, un classico gioco "Arcade". Il giocatore guida il protagonista dell'azione attraverso un complicatissimo labirinto, cercando di evitare gli aggressori e impadronendosi di tesori, strada facendo, per aumentare il punteggio. L'obiettivo del gioco è ritrovare i quattro frammenti di un prezioso amuleto magico.
La giungla è raffigurata in modo stupendo e presenta forse il miglior scenario grafico mai programmato per lo Spectrum. Gli animali, le piante, le montagne, le caverne ed i tesori sono ricchi di particolari e tutta la scenografia richiama alla memoria, addirittura, i più famosi quadri di Rousseau. Il labirinto è molto intricato ed è già un buon risultato sopravvivere per il 20% dell'intero sviluppo del gioco.
La maggior parte degli "aggressori" nemici viene eliminata con un semplice e rapido colpo di spada del protagonista, purché egli si trovi di fronte ad essi. Sotto questo aspetto, *Sabre Wulf* assomiglia a *Pac Man*. Per alcuni avversari, però, occorrono armi speciali, che vengono scoperte via via che ci si addentra nel labirinto. Altri "oggetti" hanno il pregio di aumentare il tempo di gioco, fatto estremamente utile in *Sabre Wulf*, poiché solo i giocatori più esperti riescono a sopravvivere a lungo. Alcune figure, come le orchidee in fiore, possono aiutare o ostacolare (temporaneamente) il giocatore; a seconda del loro colore, possono rendere il protagonista immune da pericoli, trasformarlo momentaneamente in una pianta, raddoppiare la sua velocità o addirittura invertire gli effetti dei tasti di controllo! *Sabre Wulf* è un gioco ben progettato, anche se presenta alcuni dei limiti comuni a tutti i programmi di gioco: il suono, ad esempio, inizialmente piacevole e ben articolato, col passar del tempo diviene fastidioso e, purtroppo, non è possibile eliminarlo. Sebbene venga presentato come un gioco per due contendenti, *Sabre Wulf* è in realtà un gioco a singolo giocatore con due tabelle segnapunti, e prevede l'"Albo della Fama", tipico dei giochi Ultimate, per i punteggi massimi. L'inserimento dei nominativi avviene secondo le regole più classiche dei giochi "Arcade": il giocatore che realizza un alto punteggio immette le proprie iniziali usando i controlli di movimento (joystick o tastiera).
Le software house si stanno accorgendo (finalmente!), che allo Spectrum si possono oggi collegare una vasta gamma di tastiere e la Ultimate prevede l'uso di alcune di esse nei propri programmi. Tuttavia, il controllo da tastiera, nel *Sabre Wulf*, è piuttosto difficoltoso, in quanto vengono adoperati i tasti Q, W, E, R e T, disposti sulla stessa fila della tastiera, per guidare i movimenti e per vibrare i colpi di spada.
A parte queste piccole imperfezioni, *Sabre Wulf* è un eccellente gioco e mantiene un perfetto equilibrio tra strategia ed azione. Il suo successo di pubblico è destinato a crescere nel tempo e già dimostra di essere un degno successore dei precedenti prodotti della Ultimate.

Sabre Wulf: per lo Spectrum da 48 Kbyte
Editori: Ashby Computers e Graphics Ltd (GB)
Autori: Ultimate, Play The Game
Joystick: Kempston, Interface 2 e tasti cursore
Formato: Cassetta



LIZ HEANEY

La pagina di presentazione
L'illustrazione mostra l'immagine di presentazione del "Sabre Wulf", così come appare sullo schermo. Lo stesso disegno compare sulla fascetta della cassetta.
Il "Sabre Wulf", è un gioco a labirinto, dotato di una grande varietà di personaggi originali. I sentieri del labirinto sono contornati da immagini di una giungla intricata, con uno stile che assomiglia a quello dei dipinti di Rousseau

Rotta di collisione

La terza parte dello sviluppo di un gioco grafico per il BBC

Nel BASIC BBC, per rilevare la pressione di un tasto, esistono non meno di 4 diversi comandi, la cui scelta, dipende dal risultato che si vuole ottenere. INKEY$ e INKEY vengono usati, di solito, per introdurre una breve pausa, durante la quale l'operatore può premere un tasto, prima di riprendere l'esecuzione del programma. GET$ e GET, invece, interrompono l'esecuzione del programma, finché non viene battuto un tasto, come nel frequente caso in cui il programma attende una risposta dell'operatore ad una domanda del tipo "Un'altra partita S/N?". Con GET$ e GET il programma si arresta in attesa che venga premuta una "S" o una "N" (le uniche lettere "riconosciute"). Possiamo quindi usare un ciclo per ripetere (REPEAT) l'istruzione GET finché (UNTIL) la risposta non è "S" oppure "N":

```
1000 PRINT "UN'ALTRA PARTITA (S/N?)"
1010 REPEAT
1020 A$=GET$
1030 UNTIL A$="S" OR a$="N"
1040 Ecc...
```

Usando GET$ e INKEY$, il tasto battuto viene interpretato come una stringa alfanumerica di un solo carattere, come nell'esempio precedente. Usando GET e INKEY, invece, alla pressione del tasto viene associato un valore numerico. Questi comandi, pertanto, consentono di controllare quei tasti a cui non corrisponde alcun carattere, come Return o i tasti per il cursore. L'istruzione *FX4,1 può essere utilizzata per ottenere i codici ASCII (decimali) dei tasti per il cursore. Essi sono:

Cursore a sinistra	136
Cursore a destra	137
Cursore verso il basso	138
Cursore verso l'alto	139

Poniamo che il nostro programma debba accettare soltanto gli input dei tasti che muovono il cursore a destra e a sinistra. Il seguente frammento usa INKEY per fermare l'esecuzione del programma, per 1/4 di secondo, in attesa di un input:

```
1000 *FX4,1: REM ABILITA IL MODO ASCII PER IL CURSORE
1010 REPEAT
1020   A$=INKEY(25)
1030 UNTIL A=136 OR A=137
1040 *FX 4,0: RIPORTA IL CURSORE AL MODO EDIT
```

Il parametro 25, alla linea 1020, comunica al computer di fermare l'esecuzione del programma per 25 centesimi di secondo.
Le istruzioni viste non controllano direttamente la tastiera, ma, al contrario, operano su una particolare zona di memoria, chiamata "buffer" della tastiera. Questa è un'area di immagazzinamento temporaneo, nella quale vengono depositati i codici generati dalla tastiera: l'ultimo carattere battuto viene inserito all'estremità finale della coda d'attesa, mentre il processore estrae uno o più caratteri dall'altra estremità. Ne deriva che tutti i caratteri digitati vengono registrati in questo buffer, dove attendono di essere elaborati dal processore. Poiché, generalmente, i comandi INKEY, GET, INKEY$ e GET$ esaminano la prima posizione della sequenza, non è possibile sapere quanto tempo un carattere sia stato ad aspettare nel buffer. Nei giochi controllati da tastiera, questo fatto può provocare una certa lentezza nelle risposte perché il programma deve elaborare tutti i caratteri che, eventualmente, precedono l'ultimo battuto dal giocatore. Ecco perché, premendo ripetutamente un tasto (per muovere il cursore in una data direzione) e poi il tasto che lo sposta nella direzione opposta, spesso accade che trascorra un certo lasso di tempo prima che il cursore si "accorga" del cambiamento di direzione.
Questo inconveniente può essere risolto in due modi: o svuotando il buffer della tastiera con il comando *FX15,1, immediatamente prima di sottoporlo a verifica, oppure usando una variazione del comando INKEY. Come abbiamo già detto, INKEY() introduce nel programma una pausa, la cui durata è indicata dal numero posto tra parentesi, durante la quale attende la battuta di un tasto, prima di riavviare il programma. Tuttavia, se inseriamo un numero negativo tra le parentesi che seguono il comando, INKEY passa a controllare direttamente la tastiera e non il suo buffer. Per questo motivo, a ciascun tasto è stato assegnato un numero di riferimento negativo (vedere la tabella a pagina 275 del Manuale per l'Utente, versione originale). Il nostro programma prevede l'uso dei tasti del cursore per controllare i movimenti delle figure. I valori negativi dei tasti da usare con INKEY sono:

Cursore a sinistra	−26
Cursore a destra	−122
Cursore verso il basso	−42
Cursore verso l'alto	−58

La seguente procedura utilizza INKEY per controllare direttamente, in successione, i 4 tasti per il cursore. Se viene premuto uno dei tasti, l'esecuzione del programma salta alla proce-

dura "movimento". A quest'ultima vengono passati anche due parametri che indicano la direzione nella quale dovrà essere mosso il carattere che rappresenta il rivelatore di mine.

```
3000 DEF PROCcontrollotasti
3010 REM ** SU ? **
3020 IF INKEY(-58)=-1 THEN PROCmovimento(0,-1)
3030 REM ** GIU' ? **
3040 IF INKEY(-42)=-1 THEN PROCmovimento(0,1)
3050 REM ** A DESTRA ? **
3060 IF INKEY(-122)=-1 THEN PROCmovimento(1,0)
3070 REM ** A SINISTRA ? **
3080 IF INKEY(-26)=-1 THEN PROCmovimento(-1,0)
3090 ENDPROC
```

La procedura "movimento"

Si tratta della procedura "centrale" del programma. Essa controlla gli spostamenti dei caratteri che rappresentano il rivelatore e l'assistente, individuando anche le collisioni con le mine. Esaminiamo per prima la parte della procedura che controlla i movimenti dei caratteri.

La procedura "controllotastiera" passa alla procedura "movimento" due parametri (registrati al suo interno nelle variabili deltax e deltay) che indicano i nuovi valori da assegnare alle coordinate x e y del rivelatore di mine. Se, ad esempio, viene battuto il tasto per "cursore verso l'alto", alla routine "movimento" vengono forniti i valori 0 e —1. Le istruzioni xdet = xdet + deltax e ydet = ydet + deltay aggiornano le coordinate del rivelatore. Nel caso appena visto, 0 viene sommato a xdet e —1 a ydet. Questa operazione comporterebbe uno spostamento verso il basso di una posizione ma, poiché l'origine delle coordinate per la posizione dei caratteri si trova nell'angolo in alto a sinistra dello schermo, i valori di y crescono dall'alto verso il basso. Pertanto, la sottrazione ydet—1 porta, in effetti, ad uno spostamento verso l'alto di una posizione. Questa tecnica di aggiornamento delle coordinate vale anche per gli spostamenti nelle altre tre direzioni. Ottenere spostamenti diagonali non comporta grossi problemi: se passiamo i valori (1,—1) alla procedura "movimento", il rivelatore si sposterà diagonalmente, di una cella in alto e di una cella a destra. Tuttavia, per guidare da tastiera i movimenti diagonali, occorre introdurre una nuova serie di tasti di controllo. Ecco il listato della procedura "movimento":

```
3220 DEF PROCmovimento(deltax,deltay)
3230 ** CANCELLA LE VECCHIE POSIZIONI **
3240 COLOUR 1
3250 PRINT TAB(xdet,ydet);" "
3260 PRINT TAB(xman,yman);" "
3270 REM ** MUOVE IL RIVELATORE **
3280 xdet=xdet+deltax
3290 ydet=ydet+deltay
3300 ** VERIFICA PERIMETRO DI GIOCO **
3310 IF xdet>17 THEN xdet=17
3320 IF ydet>25 THEN ydet=25
3330 IF xdet<2 THEN xdet=2
3340 IF ydet<1 THEN ydet=1
3350 REM ** CALCOLA LE COORDINATE DELL'ASSISTENTE **
3360 xman=19-xdet
3370 yman=26-ydet
3380 PROCconversione(xman,yman)
3390 IF POINT(xgraph,ygraph)=2 THEN
     PROCesplosione(xgraph,ygraph)
3400 PROCconversione(xdet,ydet)
3410 IF POINT(xgraph,ygraph)=2 THEN PROCminatrovata
3420 PROCstampacaratteri
3430 ENDPROC
```

Prima di cambiare le coordinate del rivelatore, si devono cancellare dallo schermo le precedenti immagini del rivelatore e dell'assistente. Le linee 3250 e 3260 usano i vecchi valori di xdet, ydet, xman e yman per coprire i vecchi caratteri con degli spazi vuoti. Poiché i nuovi caratteri vengono riprodotti in rosso (codice di colore logico 1), è necessario usare il comando di colore (linea 3240) per porre ad 1 il "corrente" colore dello sfondo. Le linee 3280 e 3290 aggiornano le coordinate del rivelatore. Prima di visualizzarlo nella sua nuova posizione, la procedura controlla che le coordinate non si trovino al di fuori dell'area che racchiude il campo minato. I limiti superiori ed inferiori di xdet e ydet vengono verificati nelle linee 3310-3340. Queste stesse linee provvedono a "bloccare" il rivelatore sul margine del campo, finché non viene mosso nella direzione opposta. Come alternativa, sarebbe stato possibile creare un effetto "rotatorio", facendo sparire il rivelatore quando questo raggiunge il margine destro del campo minato per farlo riapparire, ad esempio, dalla parte opposta. Per ottenere questo effetto, si dovrà modificare la linea 3310 in:

```
3310 IF xdet>17 THEN xdet=2
```

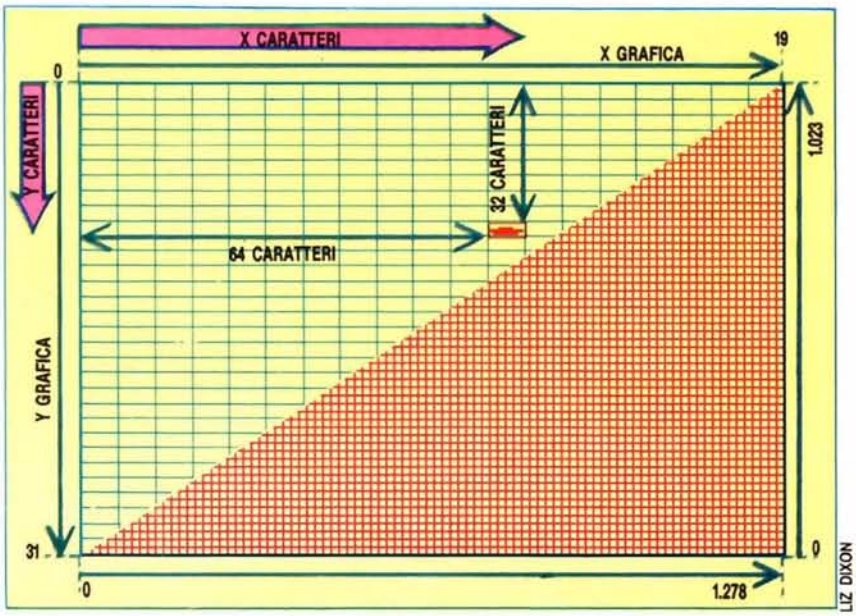
Modificando in modo simile anche le altre tre linee si otterrà l'effetto rotatorio su tutti i margini del campo minato.

Una delle regole del gioco è che l'assistente deve sempre imitare i movimenti compiuti dal rivelatore per individuare e distruggere le mine. Quindi, le coordinate dell'assistente devono essere automaticamente aggiornate in base a quelle del rivelatore: questa operazione viene eseguita dalle linee 3360 e 3370. Osserviamo la relazione esistente tra le coordinate x (xman = 19—xdet). All'inizio del gioco xdet è 2 e xman è 17. Se il rivelatore viene spostato di una posizione a destra, il valore di xdet diviene 3. Usando la formula precedente, xman sarà dato da 19-3 = 16; quindi l'assistente si muoverà di una posizione a sinistra. Se xdet viene spostato nuovamente a destra (xdet = 4), xman diverrà 15 e si muoverà ancora a sinistra di una posizione, e così via. Le coordinate y si comportano allo stesso modo.

Conversione delle coordinate

Il gioco "fonde" la grafica ad alta risoluzione con il modo testuale del BBC/Electron. Questa tecnica offre alcuni vantaggi, ma implica la combinazione di due diversi sistemi di coordinate, uno per la grafica e l'altro per il testo. Il BBC/Electron possiede parecchi formati testo di diverso tipo, ciascuno con un proprio sistema di coordinate. Il gioco usa il MODE 5, che ha uno schermo di 20 x 32 caratteri, illustrato nella parte superiore sinistra del diagramma. L'apparecchio presenta anche tre diverse risoluzioni grafiche che adottano uno stesso sistema di coordinate di 1280 x 1024 (visibile nella parte inferiore destra del diagramma).

Il programma impiega il sistema di coordinate di alta risoluzione, per "leggere" i punti dello schermo di bassa risoluzione: quindi le coordinate di carattere, vanno convertite in coordinate di alta risoluzione. A questo scopo, la coordinata orizzontale (XCHAR) viene moltiplicata per 64 e quella verticale (YCHAR) per 32. Però, le coordinate dello schermo testuale hanno origine nell'angolo superiore sinistro dello schermo e si sviluppano verso il basso, mentre quelle grafiche originano nell'angolo inferiore sinistro e procedono verso l'alto. Questo problema viene risolto sottraendo 32*YCHAR da 1023



L'ultima operazione da compiere, prima di visualizzare il rivelatore e l'assistente, è verificare se l'uno o l'altro siano stati spostati su una cella di carattere già occupata da una mina. Il BASIC del BBC consente di controllare se in un qualsiasi punto dello schermo è stato acceso un pixel di un dato colore, e l'istruzione POINT(X,Y) fornisce il codice di colore del pixel con coordinate (X,Y); questa istruzione può quindi essere utilizzata per scoprire se il colore della cella sulla quale ci siamo spostati è verde (cioè se la cella contiene una mina). Tuttavia, c'è un problema: POINT(X,Y) usa il sistema di coordinate di alta risoluzione per indicare il pixel esaminato. Volendo adoperare questo comando nel gioco, dobbiamo convertire le coordinate del carattere in coordinate grafiche. Il punto della cella più facile da determinare dovrebbe essere il suo centro. In precedenza abbiamo calcolato che, nel MODE 5, una cella di carattere è un piccolo rettangolo di 64×32 unità grafiche. Moltiplicando xchar per 64, otterremo la coordinata xgraph del lato della cella in questione; sommando poi ad xgraph il numero 32, avremo la coordinata x del centro della cella. Il calcolo di ygraph è un po' più complicato, poiché i due sistemi di coordinate hanno origini opposte. In cima allo schermo ygraph è 1023. Quindi, spostandoci verso il basso, 32*ychar ci porterà sul margine superiore della cella: sommando 16 al risultato e sottraendo il tutto da 1023, otterremo la coordinata y del centro della cella. La seguente procedura converte le coordinate di carattere in coordinate grafiche:

```
3720 DEF PROCconversione(xchar,ychar)
3730 xgraph=64*xchar+32
3740 ygraph=1023-(32*ychar+16)
3750 ENDPROC
```

Possiamo comprendere meglio l'importanza dello scambio di parametri tra procedure, tornando ad esaminare la procedura "movimento". Al suo interno, la procedura "conversione" viene usata due volte: la prima, per convertire le coordinate dell'assistente in coordinate grafiche, che verranno poi usate alla linea 3390 per verificare la presenza del colore verde. (Si rammenti che, sebbene le coordinate dell'assistente siano state aggiornate, il carattere che lo rappresenta non è ancora stato visualizzato nella nuova posizione). Se il pixel è verde, il programma salta ad un'altra procedura, che visualizza un'esplosione. La routine di conversione è usata una seconda volta, alla linea 3400, per calcolare le coordinate di carattere del rivelatore di mine. La linea 3410 controlla se la sua nuova posizione è già occupata da una mina e, in caso positivo, chiama la procedura "trovatamina". Infine viene chiamata la procedura "stampacaratteri", che visualizza il rivelatore e l'assistente nella loro nuova posizione.
Poiché la procedura "esplosione" verrà discussa più avanti, per ora la sostituiremo con la seguente procedura "di comodo":

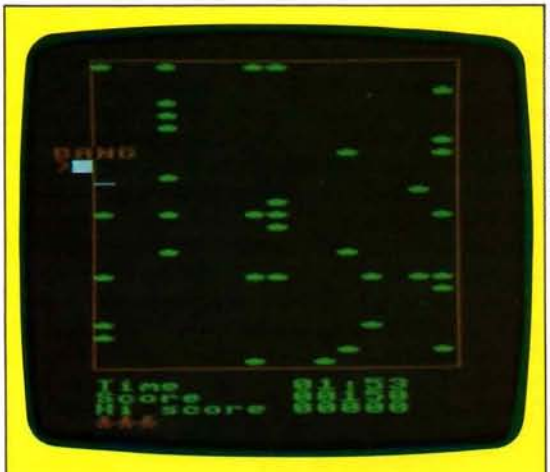
```
3550 DEF PROCesplosione (x-esplos,yesplos)
3560 PRINT "BANG"
3570 END
3580 ENDPROC
```

Esaminiamo adesso la procedura "trovatamina", chiamata quando il rivelatore si sposta su una cella occupata da una mina. Può essere divertente associare un suono al ritrovamento di una mina (questa routine verrà descritta più avanti) ma, per adesso, è sufficiente che l'istruzione SOUND produca un segnale di alta intensità. Il compito principale della procedura è quello di incrementare il punteggio del giocatore. Il programma adopera due variabili per il punteggio, di cui la prima è una variabile numerica che viene incrementata partendo da 150. Dato che il punteggio è sempre un numero a 5 cifre, dobbiamo aggiungere alla sua sinistra degli zero non significativi. Per far ciò, occorre prima convertire il valore numerico del punteggio in una variabile stringa, quindi utilizzare le tecniche di manipolazione delle stringhe (già viste a pag. 884) per aggiungere gli zero necessari. La procedura completa sarà:

```
3770 DEF PROCminatrovata
3780 REM ** SEGNALE SONORO **
3790 SOUND 2,-15,170,3
3800 REM ** INCREMENTO DEL PUNTEGGIO **
3810 COLOUR 2
3820 score=score+150
3830 score$=STR$(score)
3840 score$=LEFT$(zero$,5-LEN(score$))+score$
3850 PRINT TAB(11,28);score$
3860 ENDPROC
```

A pagina 885 abbiamo presentato un breve programma di chiamata per le procedure scritte fino a quel momento; ad esse possiamo ora aggiungere le nuove procedure. L'unica modifica da apportare, affinché il programma funzioni, è una chiamata alla procedura "controllotasti" dall'interno del corpo principale del programma. Si dovrà quindi inserire la seguente linea temporanea:

55 PROCcontrollotasti



IAN McKINNELL / LIZ HEANEY

"BANG" SUL BERSAGLIO

A questo punto del gioco "Campo minato" il programma dispone a caso le mine sullo schermo, crea le immagini del rivelatore e dell'assistente, definisce le tabelle di punteggio e il movimento. Poiché il programma non è completo, quando l'assistente incappa in una mina, sullo schermo compare la scritta "BANG". La procedura per visualizzare l'esplosione viene realizzata più avanti, assieme ad un particolare effetto sonoro. Provando il gioco così com'è, può anche darsi che compaiano messaggi d'errore sullo schermo, causati dalla struttura incompleta del programma

Prontezza di riflessi

Un semplice programma di gioco, che misura il tempo di reazione ad un particolare stimolo

Piacerebbe a tutti avere riflessi da campioni, e spesso si è portati a pensare che la nostra reazione ad un particolare evento sia istantanea. Tuttavia, la rapidità di risposta ad uno stimolo è, in media, di un terzo di secondo: un lasso di tempo abbastanza lungo, se si considera che un'auto da corsa può percorrere 10 metri in questo breve intervallo! I tempi di reazione dipendono da molti fattori: l'uso di alcolici, la stanchezza o una malattia li condizionano negativamente ed in modo significativo.

Questo programma è stato progettato per permettere a chiunque di verificare i propri riflessi e, per rendere la cosa più divertente, vengono calcolati i tempi di reazione medi sulla base di 5 prove.

Il giocatore impersona un pilota d'astronave con l'incarico di consegnare, urgentemente, alcuni medicinali ad una colonia che si trova in una cintura di asteroidi. È di vitale importanza che i farmaci arrivino in tempo, ma l'astronave deve fermarsi ogni volta che rischia una collisione.

Per arrestare la nave il giocatore deve premere la barra spaziatrice della tastiera; evitando l'asteroide, l'astronave può riprendere la corsa. Il gioco visualizza il tempo intercorso tra la comparsa sul video dell'asteroide e la battuta della barra.

La media dei tempi di reazione viene calcolata dopo che il giocatore avrà fermato la nave per 5 volte. Per accertarsi che non ci siano stati imbrogli, il programma controlla che il giocatore non abbia tenuto la barra costantemente abbassata: in questo caso i motori si fermeranno ed il computer produrrà un suono di ammonimento. Quando un asteroide appare sullo schermo, il radar dell'astronave emette un breve segnale acustico. Come esercizio di programmazione, proviamo a modificare il programma in modo da ricevere di volta in volta un segnale acustico o luminoso, ma mai entrambi contemporaneamente. Una modifica finale potrebbe prevedere la scelta tra più opzioni di questo genere, visualizzate sullo schermo. Questa avvicinerebbe il programma più ad un test di valutazione, che ad un vero e proprio indicatore dei tempi di reazione.

TEMPO DI REAZIONE

```
 10 REM *PER LO SPECTRUM*
 20 INK 7: PAPER 0: BORDER 0
 30 FOR T=0 TO 31: READ A
 40 POKE USR "A"+T,A: NEXT T
 60 DATA 123,231,255,255,245,233,123,100
 70 DATA 34,187,208,49,86,96,178,205
 80 DATA 67,234,45,123,198,255,29,245
 90 DATA 78,100,245,60,50,160,245,189
150 LET M=0: LET A=0: LET V=0: DIM R(5): CLS
220 FOR G=1 TO 5
250 FOR P=1 TO RND*150+50
260 PLOT INT (RND*250), INT (RND*160)
270 IF INKEY$<>" " THEN GO TO 280
275 BEEP .1,-3: GO TO 265
280 NEXT P
290 FOR A=144 TO 147
320 PRINT AT RND*20, RND*30;CHR$ (A)
330 NEXT A
350 BEEP.05,15: LET C=0
360 LET C=C+1: LET A$=INKEY$
410 IF A$<>" " THEN GO TO 355
420 LET R(G)=C/66+.05
430 PRINT AT 19,0;"HAI IMPIEGATO ";INT (R(G)*100)/100;
440 PRINT "SECONDI PER FERMARE "
445 PRINT "I MOTORI"
450 LET M=M+R(G)
453 FOR J=1 TO 300: NEXTJ
455 NEXT G
460 LET A=M/5: CLS
480 PRINT "MEDIA DEI TEMPI DOPO 5 PROVE"
490 PRINT "IL TUO TEMPO DI REAZIONE E' DI ";INT (A*100)/100
495 PRINT " SECONDI "
500 FOR G=1 TO 5: LET V=V+ABS (R(G)-A)
510 NEXT G
520 PRINT : PRINT "IL TUO TEMPO DI REAZIONE E' VARIATO DEL "
530 PRINT INT (V*20/A); "  PER CENTO"
540 PRINT : PRINT "VUOI FARE UN'ALTRA PROVA ? (S/N)"
550 LET R$=INKEY$: IF R$="s" THEN GO TO 150
560 IF R$<>"n" THEN GO TO 550
570 INK 0: PAPER 7: BORDER 7:CLS


 10 REM *PER IL MICRO BBC E L'ELECTRON*
 20 MODE 1: CLS: VDU23;8202;0;0;0: DIM REACT(5)
 30 REM *DEFINIZIONE CARATTERI (ASTEROIDI)*
 40 FOR J=0 TO 3
 50 DVU 23,237+J,78,100,245,60,50,160,245,189
 60 VDU 23,233+J,89,67,34,156,123,200,256,29
 70 NEXT J
 80 SOMMA=0: MEDIA=0: VARIAZIONE=0: CLS
 90 FOR PROVA=1 TO 5
100 REM *DISEGNA LE STELLE*
110 FOR RITARDO=1 TO 100+RND(300)
120 PLOT 69, RND(1200), RND(1012)
130 IF INKEY(-99)=0 THEN 150 ELSE
140 SOUND 1,-15,5,1: GOTO 130
150 NEXT RITARDO
160 REM *TRACCIA PASSAGGIO ASTEROIDI*
170 FOR ASTEROIDE=233 TO 240
180 X=RND(28): Y=RND(28)
190 PRINT TAB(X,Y);CHR$(ASTEROIDE)
200 NEXT ASTEROIDE
210 SOUND 2,-15,150,3
220 REM *VERIFICA BATTUTA BARRA*
230 TIME=0
240 IF INKEY(-99)=0 THEN 240
250 REAZIONE(PROVA)=TIME/100
260 PRINT TAB(3,26);"HAI IMPIEGATO ";
270 PRINT;REAZIONE(PROVA);" SECONDI ";
280 PRINT TAB(3,28);"PER FERMARE I MOTORI"
290 FOR I=1 TO 500;NEXT I
300 SOMMA=SOMMA+REAZIONE(PROVA)
310 NEXT PROVA
320 FOR I=1 TO 3000;NEXT I
330 REM *CALCOLA I TEMPI DI RISPOSTA*
340 MEDIA=SOMMA/5
350 FOR PROVA=1 TO 5
360 VARIAZIONE=VARIAZIONE+ABS(REAZIONE(PROVA)-MEDIA)
370 NEXT PROVA
380 CLS: PRINT: PRINT
390 PRINT "MEDIA DEI TEMPI DOPO 5 PROVE"
400 PRINT "IL TUO TEMPO DI REAZIONE E' DI ";MEDIA;" SECONDI"
410 PRINT: PRINT "IL TUO TEMPO DI REAZIONE E' VARIATO DI UN "
420 PRINT INT(VARIAZIONE*20/MEDIA);" PER CENTO "
430 PRINT: PRINT "VUOI FARE UN'ALTRA PROVA S/N ?"
440 R$=INKEY$(0): IF R$="S" THEN 80
450 IF R$<>"N" THEN 440 ELSE MODE 4
```



MIKE CLOEWS

Un cerchio di luce

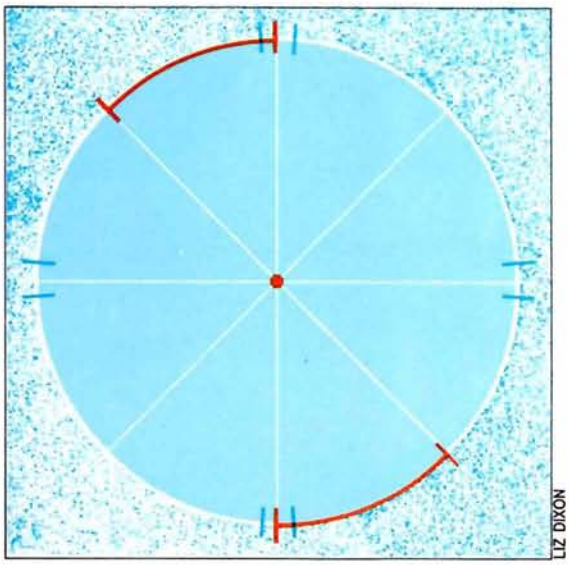
Ecco una routine in codice macchina per disegnare cerchi sul Micro BBC

Disegnare un cerchio può sembrare un'operazione semplice, ma trovare un'equazione matematica per tracciare velocemente i punti che lo compongono è un esercizio decisamente più difficile. Il metodo più semplice per tracciare un cerchio prevede l'impiego delle funzioni SIN (seno) e COS (coseno), come in questo esempio:

```
DEF PROCCERCHIO (XORG, YORG,R)
  MOVE XORG+R, YORG
  FOR THETA=0 TO 2*PI STEP PI/32
    X=R*COS(THETA)
    Y=R*SIN(THETA)
    DRAW X+XORG,Y+YORG
  NEXT THETA
ENDPROC
```

Poiché questa procedura usa le funzioni SIN e COS, occorre un certo tempo per eseguire i calcoli, pertanto il tracciamento del cerchio è piuttosto lento. Tuttavia l'operazione può risultare notevolmente più rapida se usiamo il seguente algoritmo, ricavato dalla geometria elementare e dal calcolo differenziale:

```
  3 MODE 1
  5 PROCCIRCONFERENZA (500, 600, 200)
  7 END
 10 DEF PROCCIRCONFERENZA (XORG,YORG,R)
 20 Y=R
 30 FOR X=1 TO Y*. 707
 40 Y=Y-X/Y
 50 PROCPUNTI (X, Y)
 60 NEXT
 70 ENDPROC
 80 DEF PROCPUNTI (X, Y)
 90 PLOT 69,XORG+X,YORG+Y
100 PLOT 69,XORG—X,YORG+Y
110 PLOT 69,XORG—X,YORG—Y
120 PLOT 69,XORG+X,YORG—Y
130 PLOT 69,XORG+Y,YORG+X
140 PLOT 69,XORG—Y,YORG+X
150 PLOT 69,XORG—Y,YORG—X
160 PLOT 69,XORG+Y,YORG—X
170 ENDPROC
```



LIZ DIXON

Immagini speculari

Per disegnare il cerchio, il programma in codice macchina usa un'equazione scelta appositamente per velocizzare l'operazione di tracciamento. Tuttavia, il programma risulterà ancora lento, poiché deve calcolare ciascun punto della circonferenza. Ma siccome la metà superiore di un cerchio è un'immagine speculare di quella inferiore, basta calcolare l'equazione per il semicerchio superiore e poi riprodurre la sua immagine speculare inferiore. Lo stesso principio vale per la metà sinistra del cerchio e per la sua immagine speculare destra. In realtà, il programma calcola un arco che è solo un ottavo di circonferenza. I punti che lo compongono vengono poi trasferiti per sette volte sulle rimanenti sezioni di circonferenza in modo da disegnare un cerchio intero

Questa routine disegna un cerchio tracciando contemporaneamente le sue 8 sezioni: l'operazione risulta così molto più veloce. L'algoritmo appena visto è migliore del precedente, in quanto non deve calcolare i valori del seno e del coseno per ogni punto della circonferenza; ma poiché esegue un'operazione di divisione per ogni punto, la velocità di tracciamento è pur sempre lenta.
Al contrario, il seguente algoritmo non richiede alcuna complicata operazione matematica:

```
 10 MODE 4
 20 PNUM=69
 30 PROCCERCHIO (500,600,200)
 40 END
 50 :
 60 DEF PROCCERCHIO (X,Y,R)
 70 VDU29,X;Y;:REM DEFINISCE L'ORIGINE DELLE
    COORDINATE GRAFICHE
 80 X=0:Y=R:D=3—2*R:REM VARIABILI
 90 REPEAT
100 PROCTRACCIA
110 IF  D<0:  D=D+4*X+6:  ELSE  D=D+4*(X—Y)+
    10: Y=Y—4
120 X=X+4
130 UNTIL X>Y: ENDPROC
140 :
150 DEF PROCTRACCIA
160 PLOT PNUM,X,Y
170 PLOT PNUM,Y,X
180 PLOT PNUM,Y,—X
190 PLOT PNUM,—X,Y
200 PLOT PNUM,—X,—Y
210 PLOT PNUM,—Y,—X
220 PLOT PNUM,—Y,X
230 PLOT PNUM,X,—Y
240 ENDPROC
```

Questo è noto come algoritmo di Breshen. È molto più veloce perché deve eseguire solo addizioni, sottrazioni e moltiplicazioni per 2 o 4 (entrambe possono essere implementate tramite istruzioni per lo scorrimento di bit). Nel nostro programma in codice macchina viene usato proprio questo algoritmo.
Per permettere al programma di lavorare con tutti i modi grafici, abbiamo usato il comando

CERCHI OLIMPICI BBC

Questa routine può essere usata sul BBC e sull'Electron per disegnare i cinque cerchi olimpici. Per eseguire questo programma in codice macchina occorre assegnare i valori desiderati alle tre variabili intere (X%,Y% e R%) e chiamare la routine con il comando: CALL CIRCLE,X%, Y%,R%. Questa traccerà una circonferenza di raggio R%, con centro in X%,Y%. Si noti che la routine trasporta l'origine delle coordinate grafiche al centro della circonferenza. Per riportarle nella posizione originale, si usi il comando VDU29,0;0;. Sfortunatamente l'istruzione CALL non consente di usare le espressioni come parametri, quindi X%, Y% e R% possono essere solo variabili. Per superare questa difficoltà è stata usata la procedura PROCCERCHIO (vedere il listato)

```
 30PNUM=69
 40S%=2
 50OSWRCH=&FFEE
 60DIM CODE% 600
 70DIM D% 12
 80X=D%+2
 90Y=D%+4
100D=D%+6
110N1=D%+8
120N2=D%+10
130PROCCOMPILA
140MODE4
150PROCOLIMPICI
160END
170#
180DEF PROCCOMPILA
190FOR I%=0 TO S% STEP S%
200K%=P%
210P%=CODE%
220[OPT I%
230.CERCHIO
240JSR INIT
250:
260.LOOP
270JSR COMPXY:BMI DOIT:JSR CPLOT:RTS
280.DOIT
290JSR CPLOT
300LDA D+1:BPL D_POSITIVO
310:
320.D_NEGATIVO
330JSR DNEG
340JSR SOMMA_4_A_X
350JMP LOOP
360:
370.D_POSITIVO
380JSR DPOS
390JSR SOMMA_4_A_X
400JMP LOOP
410:
420.INIT
430LDY #8
440.L7
450LDA &601,Y:STA &80,Y
460DEY:BPL L7
470INY
480LDA (&80),Y:STA X
490LDA (&83),Y:STA Y
500LDA (&86),Y:STA D
510INY
520LDA (&80),Y:STA X+1
530LDA (&83),Y:STA Y+1
540LDA (&86),Y:STA D+1
550LDA #29:STA D%+1
560LDA #0:STA D%
570JSR PSTR
580LDA #25:STA D%
590LDA #PNUM:STA D%+1
600JSR SETD
610RTS
620:
630.COMPXY
640LDA X:STA N1:LDA X+1:STA N1+1
650LDA Y:STA N2:LDA Y+1:STA N2+1
660JSR SOTTR
670LDA N1+1
680RTS
690:
700.CPLOT
710LDX #4
720.L2
730JSR P2
740DEX:BNE L2
750RTS
760:
770.DNEG
780LDA X:STA N1:LDA X+1:STA N1+1
790JSR PER4
800LDA #6:STA N2:LDA #0:STA N2+1
810JSR SOMMA
820LDA D:STA N2:LDA D+1:STA N2+1
830JSR SOMMA
840LDA N1:STA D:LDA N1+1:STA D+1
850RTS
860:
870.DPOS
880LDA X:STA N1:LDA X+1:STA N1+1
890LDA Y:STA N2:LDA Y+1:STA N2+1
900JSR SOTTR
910JSR PER4
920LDA #10:STA N2:LDA #0:STA N2+1
930JSR SOMMA
940LDA D:STA N2:LDA D+1:STA N2+1
950JSR SOMMA
960LDA N1:STA D:LDA N1+1:STA D+1
970JSR SOTTR_4_DA_Y
980RTS
990:
1000.SOMMA_4_A_X
1010LDA #4:STA N1:LDA #0:STA N1+1
1020LDA X:STA N2:LDA X+1:STA N2+1
1030JSR SOMMA
1040LDA N1:STA X:LDA N1+1:STA X+1
1050RST
1060:
1070.SOTTR_4_DA_Y
1080LDA#4:STA N2:LDA #0:STA N2+1
1090LDA Y:STA N1:LDA Y+1:STA N1+1
1100JSR SOTTR
1110LDA N1:STA Y:LDA N1+1:STA Y+1
1120RTS
1130.SETD
1140LDA #0:STA X:STA X+1
1150LDA D:STA Y:LDA D+1:STA Y+1
1160ASL D:ROL D+1
1170LDA #3:STA N1:LDA #0:STA N1+1
1180LDA D:STA N2:LDA D+1:STA N2+1
1190JSR SOTTR
1200LDA N1:STA D:LDA N1+1:STA D+1
1210RTS
1220:
1230.P2
1240JSR PSTR
1250JSR SCAMBIAXY
1260JSR PSTR
1270JSR NEGY
1280RTS
1290:
1300.PER4
1310ASL N1:ROL N1+1
1320ASL N1:ROL N1+1
1330RTS
1340:
1350.SOMMA
1360CLC
1370LDA N1:ADC N2:STA N1
1380LDA N1+1:ADC N2+1:STA N1+1
1390RTS
1400:
1410.SOTTR:\ (N1=N1-N2)
1420SEC
1430LDA N1:SBC N2:STA N1
1440LDA N1+1:SBC N2+I:STA N1+1
1450RTS
1460:
1470.PSTR
1480LDY #250
1490.L1
1500LDA D%-250,Y
1510JSR OSWRCH
1520INY
1530BNE L1
1540RTS
1550:
1560.SCAMBIAXY
1570LDA X:PHA:LDA X+1:PHA
1580LDA Y:STA X:LDA Y+1:STA X+1
1590PLA:STA Y+1:PLA:STA Y
1600RTS
1610:
1620.NEGY
1630LDA #0:STA N1:STA N1+1
1640LDA Y:STA N2
1650LDA Y+1:STA N2+1
1660JSR SOTTR
1670LDA N1:STA Y
1680LDA N1+1:STA Y+1
1690RTS
1700]
1710NEXT
1720ENDPROC
1730#
1740DEF PROCCERCHIO(P1%,P2%,P3%)
1750CALL CERCHIO,P1%,P2%,P3%
1760ENDPROC
1770#
1780DEF PROCOLIMPICI
1790PROCCERCHIO(300,600,150)
1800PROCCERCHIO(650,600,150)
1810PROCCERCHIO(1000,600,150)
1820PROCCERCHIO(475,450,150)
1830PROCCERCHIO(825,450,150)
1840VDU29,0;0;
1850MOVE100,250
1860DRAW100,800
1870DRAW1200,800
1880DRAW1200,250
1890DRAW100,250
1900ENDPROC
```

VDU 25 per tracciare i punti della circonferenza, anche se ciò compromette la velocità del programma. La suddivisione del programma in routine "indipendenti" ne favorisce la comprensione: per non creare confusione abbiamo usato solo tecniche semplici e lineari, anche se questo riduce la velocità di esecuzione. Nonostante ciò, per tracciare un cerchio con raggio di 300 unità, il programma in codice macchina impiega 52 centesimi di secondo, mentre la versione BASIC impiega ben 1,9 secondi.

La routine può essere migliorata cambiando il valore di PNUM da 69 a 5. In questo caso il programma traccerà delle linee al posto dei punti ed il risultato finale sarà un disco colorato, anziché una circonferenza.

Un effetto collaterale di questa modifica è la comparsa di linee non desiderate; per eliminare questo inconveniente, si dovrà aggiungere la seguente istruzione:

```
1745 VDU29,0;0;:MOVEP1%,P2%n
```

che in Assembler corrisponde alle istruzioni:

```
575 LDA#25:JSR OSWRCH:LDA#4:JSR
    OSWRCH:LDA#0:JSR OSWRCH:LDA#0:JSR
    OSWRCH:LDA#0:JSR OSWRCH:LDA#0:JSR
    OSWRCH
```

Il programma può essere ulteriormente migliorato, introducendo comandi per il tracciamento di archi ed ellissi.

La Texas Instruments

Nel 1958, un ingegnere della Texas inventò il primo circuito integrato. Da allora...



Il presidente della TI

J. Fred Bucy, attuale Presidente e Capo dell'Ufficio Esecutivo della TI, ha occupato questa carica nell'Aprile del 1976

Presentato nel 1978, l'home computer TI99/4A si basava sul microprocessore a 16 bit TMS990 della Texas Instruments ed aveva una memoria RAM di 16 Kbyte. Possedeva una gamma di 16 colori e tre canali di suono. Nonostante l'ottimo successo ottenuto, il computer è uscito di scena, vittima della durissima guerra dei prezzi, scatenatasi negli Stati Uniti tra il 1982 ed il 1983. La TI corse ai ripari abbassando ripetutamente il prezzo dell'apparechio ma, alla fine dell'83, fu costretta a cessarne la produzione.

Richard Mann, responsabile delle pubbliche relazioni della TI per il mercato Europeo, dichiarò in quel frangente che il computer aveva incontrato il favore del pubblico, ma che aveva fatto perdere all'azienda centinaia di milioni di dollari. Egli affermò anche la TI avrebbe in seguito puntato quasi esclusivamente sulla vendita di apparecchi dal costo molto più elevato, per il mercato dei computer professionali.

La gamma dei prodotti della Texas Instruments oggi spazia dai giocattoli elettronici per bambini come lo *Speak and Spell*, ai microcomputer e alle sofisticate apparecchiature di misurazione sismica. La TI possiede quindici impianti industriali sparsi per il mondo e realizza un fatturato annuo di oltre 4 miliardi di dollari.

La Texas Instruments fu fondata nel 1930 da due scienziati, John Karchner e Eugen McDermott, con il nome di Geophisical Service Incorporated. Karchner aveva maturato l'idea di usare l'eco prodotto dalle onde sonore inviate nel sottosuolo, per calcolare la profondità degli strati geologici; la GSI decise di stabilire la propria sede a Dallas, nel Texas, per poter vendere le apparecchiature prodotte alle compagnie petrolifere.

Durante gli anni '30, la GSI crebbe sviluppando nuove tecniche ed attrezzature di misurazione sismica e, nella Seconda Guerra Mondiale, i suoi strumenti di rilevamento furono impiegati per l'individuazione dei sottomarini. Questo nuovo campo di applicazioni spinse la compagnia a creare un settore separato, per la ricerca e la produzione; tale settore collaterale si estese talmente che, nel 1951, fu deciso di renderlo indipendente dalla casa madre. La nuova compagnia fu chiamata Texas Instruments.

Nell'anno seguente, la TI ottenne dai Laboratori Bell, una licenza per la produzione di un componente elettronico di recente invenzione, il transistor, e nel 1954 produsse la prima radio a transistor del mondo. Nel 1958 Jack Kilby, un ingegnere impiegato nella compagnia, inventò il circuito integrato e la TI è ancora oggi all'avanguardia nella ricerca in questo campo. Nel 1958 Kilby collaborò anche alla realizzazione del primo calcolatore elettronico tascabile del mondo.

Il computer TI Professional, lanciato sul mercato nel Gennaio del 1983, si basa su un microprocessore 8088 e può quindi usare i sistemi operativi CPM-86 e MS-DOS. Sempre nell'83 la TI ha presentato il computer gestionale portatile Texas Instruments ed il micro "da viaggio" CC40.

Oggi, la compagnia produce anche una grande varietà di componenti elettronici ed un chip RAM a 64 Kbyte, di cui la TI dichiara di essere una delle maggiori produttrici nel mondo. I chip TI sono presenti quasi in tutti gli home computer, compreso il Commodore 64 e lo Spectrum della Sinclair.

Come la maggior parte dei produttori di chip, la Texas Instruments ha sviluppato una serie di processori a 16 bit, tra cui il TMS9900 usato nell'home computer TI99/A4.

Questo originale processore non possiede registri interni, che invece sono stati inseriti in una speciale "memoria di lavoro" esterna alla CPU. È un dispositivo estremamente potente, ma i suoi limiti strutturali non permettono di utilizzarlo pienamente. La TI sta realizzando un microprocessore a 16 bit, il TMS99000 e confida in un grande successo per questo nuovo prodotto.

La casa madre della TI

Gran parte delle apparecchiature prodotte dalla Texas Instruments vengono fabbricate nel moderno impianto a Dallas, nel Texas (USA)



Il compressore di dati

Un apparecchio minuscolo, ma con una memoria formidabile

L'Organiser della Psion ha il peso e le dimensioni di un rasoio elettrico e, a prima vista, ne ha anche l'apparenza. La tastiera, dotata di 36 piccoli pulsanti, comprende tutte le lettere dell'alfabeto (l'Organiser usa solo lettere maiuscole) e diversi tasti di funzione. Agendo sul tasto Shift si abilitano la tastiera numerica (da 0 a 9), alcune funzioni matematiche ed i segni d'interpunzione. Sopra la tastiera c'è un display a cristalli liquidi da 16 caratteri. Il contrasto del display è regolabile, per meglio adattarlo a qualsiasi angolazione di lettura. I caratteri sono formati da una matrice di 5 × 8 punti.

Posteriormente, l'apparecchio possiede due connettori ai quali vengono collegati i "datapack" (le memorie per l'immagazzinamento a lungo termine delle informazioni). L'Organiser è corredato di un datapack da 8 Kbyte, ma ne è disponibile un altro da 16 Kbyte.

Quando viene premuto il tasto ON, posto in alto a sinistra sulla tastiera, sul display compaiono l'ora (0-24) e la data. Le quattro funzioni principali dell'Organiser vengono richiamate battendo un solo tasto. Come spiegato sul manuale d'uso, non è necessario scrivere per intero i comandi per queste funzioni. Esse sono:

- SAVE: è possibile inserire da tastiera sul display registrazioni lunghe fino a 200 caratteri, curarne la redazione con i due controlli per il cursore (sinistro e destro) e col tasto Delete, quindi salvare (SAVE) il testo finale su uno dei due datapack.
- FIND: Questa funzione permette di recuperare le registrazioni da entrambi i datapack, ma le stringhe ricercate non devono però superare i 15 caratteri. La stringa viene scritta sul display e l'apparecchio la confronta con le registrazioni immagazzinate. Il

Registrazione veloce

Lo Psion Organiser, usato come computer tascabile dotato di un potente database, offre un sistema di registrazione di dati rapido ed efficiente. Grazie alle funzioni di calcolo incorporate e alla capacità di immagazzinare e recuperare con rapidità le informazioni, l'Organiser può gestire una serie di funzioni di solito disponibili solo sui microcomputer da tavolo o su quelli da viaggio



IAN McKINNELL

comando FIND è molto veloce: il tempo medio di ricerca, in un database di 21 Kbyte, è di circa 8 secondi. Le registrazioni che contengono la stringa cercata sono visualizzate in successione, seguendo l'ordine di individuazione. Per esaminare la registrazione per intero, si può farle scorrere in entrambe le direzioni, tramite i tasti del cursore.

Nella seguente dimostrazione è stato usato il datapack Psion *Guida ai ristoranti di Londra*, un database di 16 Kbyte che contiene i nomi di 105 ristoranti, ciascuno corredato di indirizzo, numero telefonico, nome della più vicina stazione della metropolitana, tipo di cucina, orario d'apertura, listino prezzi e di una generica valutazione di qualità. Una ricerca basata sulla stringa ARB ha dato il seguente risultato: il nome di un ristorante chiamato *Barbarellas* e quello di un ristorante situato in prossimità della stazione della metropolitana di "Marble Arch". La velocità e la funzionalità di questa funzione di ricerca si dimostrano addirittura superiori a quelle di molti programmi di database per microcomputer.

● ERASE: Serve per cancellare una registrazione, dopo averla richiamata dal datapack. Sulla tastiera dell'Organiser non esiste il tasto di funzione ERASE: questo comando può essere eseguito solo dopo aver usato l'istruzione FIND. Con la registrazione visualizzata sul display, l'utente preme il tasto MODE per abilitare la funzione ERASE. Poi, premendo il tasto Execute, si ottiene la cancellazione della registrazione dalla memoria. Tuttavia, lo spazio di memoria liberato non può essere usato per una nuova registrazione, pertanto il comando rende soltanto la registrazione inaccessibile alla CPU. Le proprietà delle memorie datapack verranno trattate più avanti.

● CALC: Consente di usare l'Organiser come una calcolatrice a 4 funzioni (addizione, sottrazione, moltiplicazione e divisione). I risultati possono avere fino a 7 cifre significative, mentre per i numeri molto grandi o molto

TABELLA COMPARATIVA

L'Organiser della Psion viene qui confrontato con un sofisticato sistema a registrazione manuale per indirizzi/appuntamenti/appunti

	PSION	SISTEMA MANUALE FILOFAX
DIMENSIONI	mm. 142 x 78 x 29,3	mm. 180 x 12 x 30
PESO	g. 225	g. 200
DISPLAY	16 caratteri LCD	Quantità di scritto introdotta manualmente dall'utente
MEMORIZZAZIONE	10.900 caratteri in una rampak da 8 Kbyte	Fino a 200 pagine
METODO DI RICERCA	Qualsiasi stringa di caratteri lunga fino a 15 caratteri	Manuale, su indice alfabetico
CARATTERISTICHE OPZIONALI	Rampak da 8 o 16 Kbyte, archivi scientifici, matematici, per ingegneri, per la contabilità	Pagine per agenda, per rubrica, per contabilità, mappe stradali, blocco notes e fogli musicali
VARIE	Indica l'ora e la data. Utilizzabile come calcolatrice	Fornito di elegante custodia in pelle

Basta una mano...

Sulla tastiera Psion, le lettere sono disposte in ordine alfabetico. Questa sistemazione poco razionale può creare qualche problema a chi è abituato alle tastiere tipo QWERTY. Il tasto Shift abilita la tastiera numerica, le funzioni della calcolatrice, i movimenti del cursore ed i comandi per il database



KEVIN JONES

piccoli viene usata la notazione scientifica. Anche in questo caso, non esiste un tasto CALC sulla tastiera dell'apparecchio: la funzione viene abilitata premendo il tasto MODE in un qualsiasi momento, purché non successivo ad una ricerca con FIND.
Il prezzo relativamente basso della macchina si intona alla sua semplicità e facilità d'uso. Inoltre, la tecnologia a basso consumo impiegata nell'Organiser consente di alimentarlo con un'unica pila a 9 volt, che ha una durata media di oltre sei mesi. I circuiti si basano su chip CMOS (semiconduttori complementari metallo-ossido), mentre i datapack sono chip EPROM (memorie di sola lettura programmabili e cancellabili): entrambi i dispositivi presentano livelli di consumo energetico estremamente bassi. Fino a poco tempo fa, l'elevato costo di questi chip avrebbe costretto la ditta produttrice, la Psion, a tenere alto il prezzo dell'apparecchio.
Il cuore dell'Organiser è un processore Hitachi 6301X a 8 bit, che gira a 0,93 MHz e viene controllato da una ROM a 4 Kbyte. Inoltre, ci sono 2 Kbyte di RAM per l'inserimento dei dati, le informazioni video e l'area di lavoro della calcolatrice. I datapack EPROM innestati sul retro dell'apparecchio equivalgono alla memoria RAM dei comuni microcomputer. La EPROM può essere considerata come una RAM "a memoria perenne", vale a dire che i dati registrati non possono più essere cancellati, ma si possono richiamare come se si trattasse di una comune ROM; è quindi una ROM "programmabile". La funzione ERASE dell'Organiser in realtà non libera aree di memoria della EPROM, bensì "marca" le registrazioni per segnalarne la cancellazione al processore. Quindi, un datapack "pieno" (anche di dati cancellati) non può ricevere nuove registrazioni. Il solo modo per cancellare i dati memorizzati e riportare il chip al suo stato originale è esporlo ad un intenso fascio di raggi ultravioletti: questa operazione viene svolta da un secondo strumento della Psion, chiamato Psion Formatter. I chip datapack a 8 e a 16 Kbyte possono essere "riformattati" fino a 100 volte.
Rispetto alla RAM, la EPROM offre il vantaggio di poter conservare i dati memorizzati senza consumare corrente. La RAM è invece "volatile" e deve essere continuamente alimentata. Inoltre, i chip EPROM sono molto affidabili ed il loro costo medio "per byte" è circa 1/5 di quello per un byte di RAM. Le EPROM vengono comunemente impiegate dai fabbricanti di computer nei "prototipi" dei nuovi apparecchi, poiché, in caso di errori software, possono essere riprogrammate e riutilizzate, mentre una ROM programmata in modo errato deve essere gettata via.
L'Organiser può essere collegato ad altri computer mediante un'interfaccia RS232 (opzionale), da innestare ad uno dei due connettori per datapack, che permette di trasmettere e ricevere dati ad una velocità di circa 9600 baud. Tramite il software di corredo all'interfaccia, si possono selezionare diversi protocolli di trasmissione. Ad esempio, è possibile formattare la lunghezza della pagina e le dimensioni della riga da inviare ad una stampante.
L'Organiser è l'apparecchio ideale per tutte le attività che richiedono l'uso di un potente computer portatile. Può essere usato per annotare i risultati di un esperimento o come agenda d'appuntamenti, anche se la tastiera consente una limitata velocità di battitura (le lettere sono disposte in ordine alfabetico). I migliori risultati si hanno nell'archiviazione di informazioni correlate tra loro: il listino prezzi di un commerciante, ad esempio, oppure la bibliografia per una piccola biblioteca o, ancora, gli indirizzi ed i numeri telefonici di una rubrica. Infine, l'apparecchio può dare ottimi risultati abbinato ad un programma per computer, nella registrazione immediata di dati sperimentali o nel riordino di informazioni destinate ad elaborazioni successive.

COMPUTER TASCABILI

Negli ultimi anni, alcune famose industrie elettroniche hanno lanciato sul mercato vari modelli di computer tascabili. Questi apparecchi hanno avuto un certo successo, ma sono rimasti molto al di sotto dei livelli di vendita raggiunti dalle calcolatrici tascabili. Alcuni computer tascabili non sono altro che calcolatrici presentate con maggior enfasi: possono essere programmati soltanto in speciali linguaggi "esclusivi" e sono dedicati soprattutto ad ingegneri e scienziati, che possono aver necessità di eseguire complicati calcoli in qualsiasi momento e situazione. In un secondo momento, l'industria ha prodotto computer tascabili programmabili in BASIC, più adatti ad impieghi differenziati. Per tali apparecchi sono stati scritti programmi gestionali di vario genere.
Tuttavia, i computer tascabili non si sono mai diffusi su larga scala. Soprattutto si è sentita la mancanza di software non specialistico, che permettesse alla gente comune di usare i "tascabili" per le proprie attività giornaliere. Le capacità di memoria si sono dimostrate troppo limitate, anche per semplici rubriche o elenchi di libri e le dimensioni eccessivamente ridotte delle tastiere hanno spinto molti utenti a riprendere in mano carta e penna per annotare indirizzi o appunti. L'unico vero vantaggio offerto dalle versioni computerizzate è la possibilità di eseguire ricerche sofisticate su una serie di dati, ma non tutti i computer tascabili dispongono di questa funzione



Hewlett Packard 41C

È una calcolatrice programmabile molto sofisticata. Viene prodotta in tre modelli: HP-41C, HP-41CV e HP-41CX. Il 41CX ha 1/2 K di memoria, gli altri modelli 2 K. Tutti e tre gli apparecchi possiedono una tastiera alfabetica, che può essere usata solo per la programmazione della calcolatrice, e non per la redazione di testi



Sharp PC 1500



Sharp PC 1251

La Sharp produce due computer tascabili: il PC 1251 ed il PC 1500. Il primo è il più piccolo apparecchio di questo tipo sul mercato, il secondo è il più potente. Entrambi vengono programmati in BASIC ed il PC 1500 dispone anche di comandi aggiuntivi. Hanno 4 Kbyte di memoria, ma quella del PC 1500 può essere ampliata fino a 20 Kbyte

Texas Instruments TU-66

Fondamentalmente, si tratta di una calcolatrice programmabile, ideale per scienziati e matematici. Non è in grado di elaborare testi: si dimostra quindi inadatta ad applicazioni di carattere generale. La memoria è molto ridotta: solo 1/2 Kbyte!





Casio FX700P

Questo è uno dei numerosi computer tascabili, programmabili in BASIC, prodotti dalla Casio. Ha una memoria di 2 Kbyte ed una ridottissima tastiera alfabetica. Una particolare versione della macchina possiede una piccola stampante incorporata

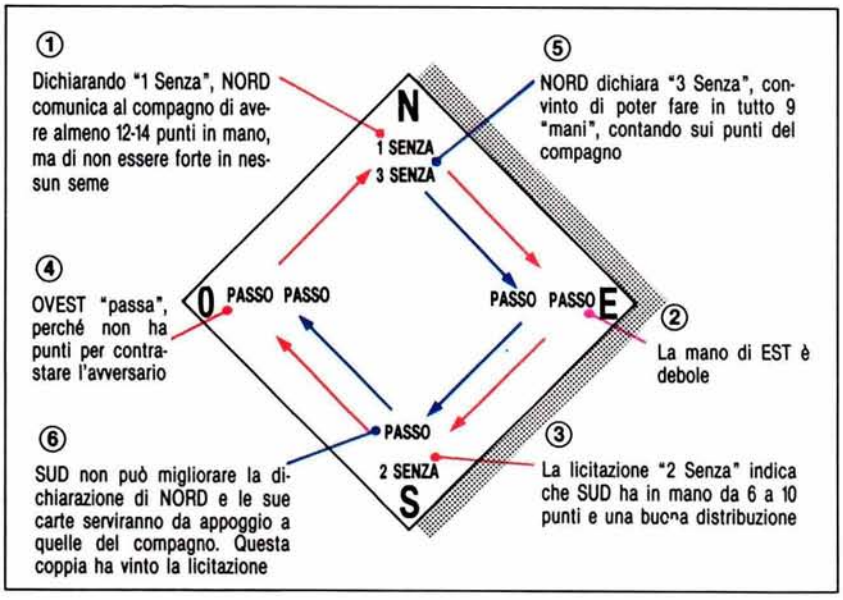
Un bridgista d'eccezione

Il software che mette in grado il computer di insegnare e giocare a "bridge"

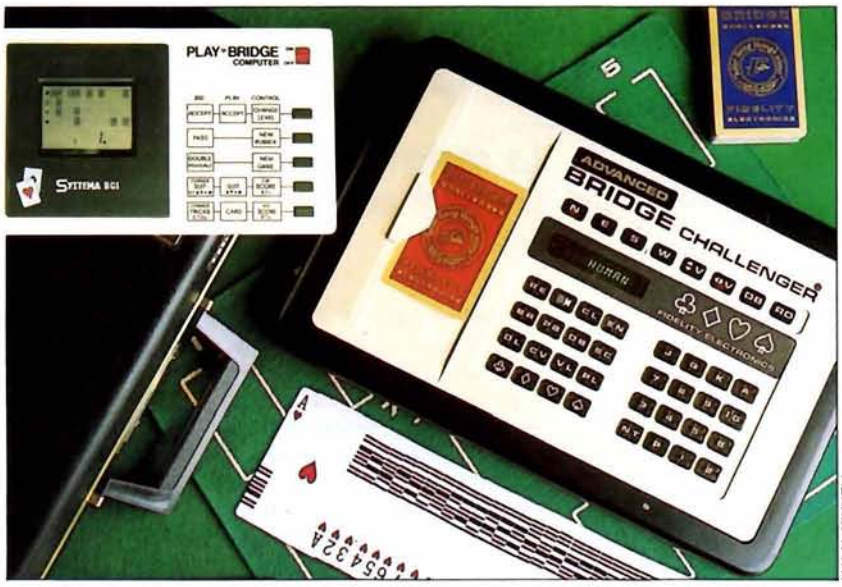
I programmi di bridge per computer si dividono in due categorie: i pacchetti di apprendimento e quelli di gioco.
Il mercato offre un discreto numero di programmi didattici e il migliore, in termini di progettazione e presentazione grafica, è certamente il package *Bridgemaster*, di cui sono disponibili versioni per lo Spectrum, il BBC Modello B, l'Electron ed il Commodore 64. Il *Bridgemaster* è stato progettato da Terence Reese, un ex-campione del mondo (oggi consulente di bridge per i quotidiani inglesi Observer e Standard), autore di numerosi libri sul bridge.
Il principio su cui si basano tutti i programmi didattici di bridge è che il principiante deve esercitarsi su di una serie di partite o "mani" prestabilite dallo stesso programma. Questa è anche la principale differenza tra i package di apprendimento e quelli di gioco, nei quali le mani vengono invece generate in modo casuale.
Il programma "conosce" tutte le mani che verranno proposte ed è quindi in grado di guidare il giocatore, passo dopo passo, attraverso le molte regole e le "convenzioni" del gioco.
Il *Bridgemaster* dà un'eccellente idea dei risultati ottenibili con un buon package didattico.
Il corso presenta due programmi dedicati rispettivamente ai principianti e ai giocatori di medie capacità. Il primo si chiama *Complete Learning Package For The Beginner At Bridge* (Corso completo di apprendimento per principianti del Bridge), il secondo *Expert Bridge* (Bridge per esperti). Il package per principianti è composto da due cassette, due nastri di commento, un breve manuale di istruzioni ed il *Begin Bridge* di Reese, un "tascabile" pubblicato dalla Penguin, non direttamente collegato al programma. I nastri di commento forniscono al principiante tutte le informazioni necessarie.
Nel bridge su computer, i quattro giocatori vengono indicati con i termini Nord, Sud, Est, Ovest, secondo la convenzione standard. Il gioco inizia con la distribuzione delle carte, seguita dalla cosiddetta "dichiarazione", nella quale ciascun giocatore, pur senza rivelare le proprie carte agli altri, ne dichiara la forza, usando particolari convenzioni. La forza di una mano è determinata dal numero di carte di alto valore, dal numero di carte di uno stesso seme, o da una combinazione di questi due fattori. La figura a sinistra mostra un esempio di dichiarazione. La dichiarazione vincente fissa il "contratto", ossia le "prese" che il giocatore si impegna a fare nella mano. Dopo la dichiarazione, inizia il gioco della carta. Durante il gioco, il giocatore che ha vinto la dichiarazione viene detto "dichiarante" ed il suo compagno (il giocatore di fronte) diventa il "morto".
Le carte del morto vengono stese, in vista, sul tavolo da gioco. Il dichiarante guadagnerà o perderà punti a seconda che riesca o meno a realizzare il numero di prese dichiarate nel contratto. Tutti i programmi didattici per bridge prevedono la fase della dichiarazione, ma solo alcuni permettono al giocatore di aggiudicarsi il contratto. Nel *Bridgemaster*, il giocatore ha la possibilità di fare la sua dichiarazione dopo aver esaminato le proprie carte, visualizzate sullo schermo. È poi la volta del computer, che comunica la dichiarazione che avrebbe fatto con quella stessa mano. Il programma di Reese, come molti altri programmi d'apprendimento, è strutturato in modo tale che il computer accetti solo quella dichiarazione o quella giocata prevista dal progettista del programma. Reese riconosce che questa tecnica talvolta porta ad un tipo di gioco del tutto prevedibile, in quanto le scelte del giocatore vengono rifiutate, favorendo così uno sviluppo predeterminato del gioco.
Dopo aver mostrato la mano di carte e la dichiarazione, il programma si occupa del gioco della carta. Il metodo più comune per illustrare questa fase, prevede la comparsa sullo schermo di un quadrato i cui angoli sono indicati dalle lettere N, E, S, W. Le mani

Dichiarazione vincente

Nella dichiarazione, i valori delle carte sono i seguenti: Asso-4, Re-3, Donna-2, Jack-1. Le scritte in rosso indicano la prima dichiarazione, quelle in blu la seconda e conclusiva



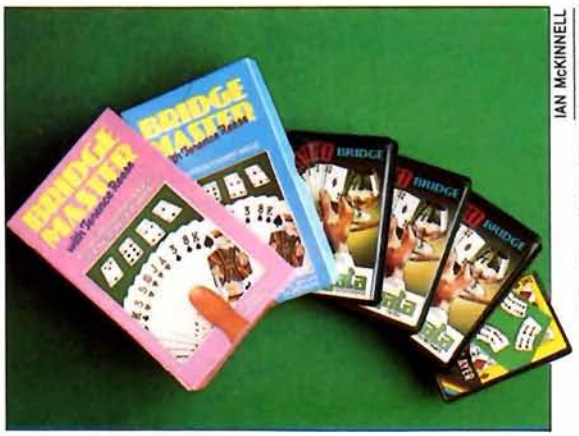
dei giocatori Nord e Sud vengono visualizzate per intero e le loro carte vengono disposte per seme (nell'ordine: picche, cuori, quadri e fiori), con il valore di ciascuna visualizzato a fianco.
Di solito, al giocatore viene assegnata la posizione Sud e tanto Nord quanto Sud possono diventare il morto. In entrambi i casi, è sempre Sud che gioca la mano anche se la dichiarazione è stata vinta da Nord. In questo il programma si differenzia dal vero bridge, nel quale un giocatore può passare un'intera serata senza aggiudicarsi una dichiarazione!
Nei programmi didattici questa diversità non ha rilevanza ed influisce solo sui sistemi di gioco difensivo. Queste tecniche vengono impiegate quando si vuol impedire alla coppia avversaria di realizzare il contratto (cioè il numero di prese dichiarate). In un programma di gioco, le implicazioni sono molto più complesse: nessun programma di questo tipo giocherà una mano nella quale Est o Ovest si siano aggiudicati la dichiarazione. Se il giocatore perde volutamente la dichiarazione (ad esempio nel discreto programma di gioco prodotto dalla CP Software per lo Spectrum a 48 Kbyte) lasciandola ad Est o ad Ovest, il programma visualizza un messaggio in cui comunica di non essere stato progettato per giocare in tali condizioni ed invita il giocatore a premere il tasto R per generare una nuova mano.
A dichiarazione conclusa, il programma passa al gioco della carta. Chi ha vinto la dichiarazione, cioè il computer per Est e Ovest oppure il giocatore per Nord e Sud, gioca una carta. Le carte di Est e Ovest compaiono sullo schermo una alla volta, come accade in una vera partita.
I programmi di bridge per computer possiedono un certo numero di utili funzioni. Il *Bridgemaster*, ad esempio, consente al giocatore di scegliere tra quattro opzioni a seconda del suo livello di conoscenza del gioco. P assegna al giocatore il controllo del gioco degli angoli N e S (sotto la guida di Reese); A abilita il livello "autogioco" dove il giocatore segue una mano giocata automaticamente dal computer; H visualizza le mani di tutti e quattro i giocatori; infine D seleziona una diversa distribuzione delle carte. Questa è una serie di opzioni standard presente in quasi tutti i programmi di bridge.
La scelta "Autogioco" si dimostra particolarmente utile abbinata ai nastri di commento di Reese poiché permette al giocatore di seguire lo sviluppo di una partita, senza essere distratto dalla scelta della carta. Tutte le mani possono essere giocate più volte. I programmi di gioco possiedono opzioni simili, ma non consentono la ripetizione della dichiarazione né di giocare la mano dopo che la dichiarazione ha assegnato al giocatore il ruolo di morto.
Abbiamo esaminato alcuni dei principi alla base dei package didattici, ma il mercato offre anche un certo numero di programmi di perfezionamento per giocatori esperti.



IAN McKINNELL

Computer "dedicati"
Questi due apparecchi sono computer "da bridge". Il primo (a destra) è un modello da tavolo della Fidelity Electronics, chiamato Advanced Bridge Challenger, ed è dedicato agli appassionati di bridge: il suo programma di gioco richiede buone doti tecniche e grande concentrazione. Il principale difetto di questo apparecchio è il piccolo display a soli 8 caratteri. La seconda macchina (a sinistra) è il Systema Play-Bridge, alimentato a pile. Per imparare ad usarlo bastano pochi minuti; le sue caratteristiche non sono molto avanzate, ma è divertente e facile da usare

L'*Expert Bridge*, che fa parte del *Bridgemaster*, è un buon programma di perfezionamento che approfondisce le tecniche di gioco per le mani molto forti, le dichiarazioni di "slam", le "compressioni" ed i "tagli".
Un altro buon programma di perfezionamento viene prodotto dalla Alligata Software in versioni per l'Oric, per il BBC Modello B, per l'Electron e per il Commodore 64. Il programma richiede una discreta conoscenza delle tecniche di gioco ed è sprovvisto di manuali, quindi gli utenti con scarse doti di memoria dovranno avere sotto mano carta e penna per prendere appunti.
Buono è anche il pacchetto della CP Software, che però non consente al computer di giocare come dichiarante e quindi non permette di esercitarsi nelle tecniche di difesa.
I programmi di bridge, siano essi didattici o di gioco, sono uno strumento ideale per imparare a giocare e a sfruttare nel modo migliore le proprie carte. Il bridge è un gioco che richiede molte doti: buona memoria, capacità analitica, intuizione e buona intesa con il compagno, cautela nelle dichiarazioni ed una notevole freddezza. I pacchetti di cui abbiamo parlato non simulano in modo perfetto una partita di bridge tra quattro giocatori, tuttavia permettono di sviluppare sicurezza ed elasticità di gioco.

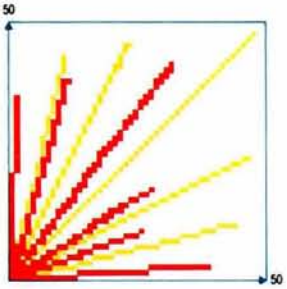


IAN McKINNELL

Ampia scelta
Molti sono i "pacchetti per bridge" oggi disponibili: dai programmi didattici ai giochi che mettono a dura prova le capacità dei professionisti. La figura mostra (da sinistra a destra): il "Bridgemaster", disponibile su cassetta per lo Spectrum, lo ZX81, il BBC Modello B, l'Electron ed il Commodore 64; il "Bridge" della Alligata Software, in versione su cassetta per l'Oric, il BBC Modello B, l'Electron ed il Commodore 64; il "Bridge Player", prodotto su cassetta dalla CP Software per lo Spectrum a 48 Kbyte

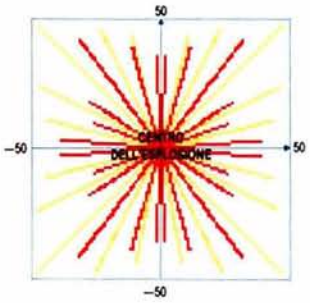
Scoppi e tuoni

Le sezioni di programma per accompagnare le esplosioni con effetti grafici e audio



Esplosione

L'immagine dell'esplosione viene creata generando linee di lunghezza casuale, che distano dal centro da —50 a 50 unità. Il programma crea soltanto uno dei quattro quadranti dell'esplosione, riproducendolo poi in modo speculare sugli altri tre quadranti



Il Micro BBC e l'Electron possiedono una "tavolozza" di 16 colori utili nella creazione dell'"effetto esplosione". In realtà, i colori sono solo 8, in quanto negli altri casi si tratta di variazioni ottenute alternando due colori diversi, il che avviene ogni mezzo secondo, ma il ritmo del lampeggiamento si può variare mediante due comandi FX. Il comando *FX9 definisce il tempo di visualizzazione del primo colore della coppia, e viene misurato in cinquantesimi di secondo. Quindi *FX9,20 indicherà un tempo di visualizzazione del primo colore di 2/5 di secondo. Il secondo comando FX, ossia *FX10, agisce analogamente, ma sul secondo colore della coppia. Cambiando uno o entrambi i tempi di visualizzazione, si potranno ottenere effetti cromatici "guizzanti".
L'effetto esplosione viene creato tracciando una serie di brevi linee di diverso colore a partire da un punto centrale. Per migliorare l'effetto generale, si potranno far lampeggiare anche le mine e i caratteri del punteggio, ai quali in precedenza avevamo dato il colore logico 2 (selezionato sul verde). Col seguente comando si ripristina il colore logico 2, dopo averne selezionato uno a caso tra le 16 combinazioni possibili:

```
VDU 19,2,RND(15),0,0,0
```

RND(15) seleziona un numero intero compreso tra 1 e 15: il valore 0, che corrisponde al nero (il colore dello sfondo) non viene mai scelto. I colori impiegati per tracciare le linee dell'esplosione vengono selezionati in modo casuale col comando GCOL 0,RND(3). Occorre quindi un ciclo iterativo per tracciare una serie di linee originanti dal centro dell'esplosione, in una gamma di colori selezionati a caso. Finora abbiamo preso in considerazione solamente i comandi di alta risoluzione MOVE e DRAW; DRAW (X,Y) traccia sempre una linea fino ad un punto con coordinate "assolute" (X,Y). Esiste però un secondo gruppo di comandi di tracciamento in alta risoluzione che permette, tra l'altro, di congiungere punti "relativi" tra loro con una linea. Il comando PLOT K,X,Y consente di tracciare linee tra punti relativi o assoluti, a seconda del valore assegnato a K. La tabella, qui accanto, illustra alcune delle possibili variazioni che si possono ottenere con PLOT: PLOT4 e PLOT5 corrispondono ai comandi MOVE e DRAW. Nell'effetto esplosione, useremo PLOT1 per tracciare le linee relative al centro dell'esplosione che, tuttavia, va definito prima di eseguire questa operazione.
In precedenza, nel richiamare una versione "fittizia" o "di comodo" di PROCesplosione, abbiamo passato i valori di xgraph e ygraph alle coordinate xesplos e yesplos. Queste non giocavano alcun ruolo nella procedura fittizia, ma ora vengono adoperate per definire il centro dell'esplosione. Se la procedura viene chiamata dalla linea 3390 di PROCmovimento (vedere a pag. 915), xgraph e ygraph saranno le coordinate grafiche del centro della cella di carattere che contiene la mina. Quindi, l'istruzione MOVExesplos, yesplos sposterà il cursore grafico sul punto che si vuol indicare come centro dell'esplosione. Il motivo per cui abbiamo passato le coordinate ad un'altra coppia di variabili da usare all'interno della procedura dell'esplosione, invece di usare più semplicemente xgraph e ygraph, è che questa procedura verrà chiamata anche da un'altra parte del programma. Quindi le coordinate del centro dell'esplosione dovranno venire specificate da una seconda coppia di variabili, ovvero xesplos e yesplos. Questo trasferimento di parametri rende la procedura dell'esplosione molto più flessibile.
Dopo aver posizionato (con MOVE) il cursore grafico al centro dell'esplosione, possiamo tracciare una linea in una direzione casuale, con lunghezza massima pari, ad esempio, a 50 unità. La sequenza

```
PLOT 1,RND(50),RND(50)
```

però, non è sufficiente per portare a termine l'intero lavoro, poiché non definisce coordinate negative (partendo dal centro dell'esplosione, una coordinata x negativa traccia una linea verso sinistra, mentre una coordinata y negativa produce una linea diretta verso il basso). L'uso ripetuto di questo comando traccerà una serie di linee solo nel quadrante ove entrambe le coordinate sono positive, ossia quello superiore destro del sistema di assi Cartesiani, che ha come origine il centro dell'esplosione.
Per introdurre valori negativi (e quindi tracciare linee in tutte le direzioni), dovremo sostituire RND(50) con RND(100)—50. Poiché il valore più alto di RND(100) è 100, il valore massimo che si ha da RND(100)—50 è: 100—50 = 50. D'altra parte, il valore minimo di RND(100) è 1, quindi quello minimo di RND(100)—50 è: 1—50 = —49.

K = 0	Movimento relativo all'ultimo punto
K = 1	Traccia linea fino posizione relativa nel colore di primo piano
K = 2	Traccia linea fino posizione relativa nel colore logico inverso
K = 3	Traccia linea fino posizione relativa nel colore di sfondo
K = 4	Movimento fino posizione assoluta
K = 5	Traccia linea fino posizione assoluta nel colore di primo piano
K = 6	Traccia linea fino posizione assoluta nel colore logico inverso
K = 7	Traccia linea fino posizione assoluta nel colore di sfondo

Riportando, di volta in volta, il cursore al centro dell'esplosione (con MOVE) e tracciando nuovamente una linea relativa in qualunque direzione e colore, otterremo l'immagine completa dell'esplosione.
Occupiamoci adesso degli effetti sonori da abbinare alla comparsa dell'esplosione sullo schermo. Usando i comandi SOUND e ENVELOPE del BASIC BBC, possiamo creare suoni discretamente sofisticati. SOUND può generare sia note musicali (o anche interi accordi) sia il "rumore", necessario per effetti sonori speciali. Il comando ENVELOPE, abbinato a SOUND, serve per modulare il suono emesso, consentendo di imitare gli strumenti musicali.
Al comando SOUND sono associati 4 parametri, che definiscono le caratteristiche dell'accordo emesso:

SOUND C,A,P,D

• C indica il numero di canale ed i valori ammessi sono compresi tra 0 e 3. I canali da 1 a 3 producono note musicali, mentre il canale 0 serve per effetti speciali. Nel nostro programma usiamo proprio quest'ultimo canale.
• A è l'ampiezza o "volume" del suono emesso. I valori per il volume sono compresi tra −15 (volume massimo) e 0 (azzeramento del suono). I valori positivi di A selezionano la modulazione del suono e vengono usati assieme al comando ENVELOPE.
• P indica di solito l'altezza della nota, ossia quanto essa sia "grave" o "acuta".
• D definisce la durata della nota, con valori compresi tra 0 e 254. Il tempo di durata viene conteggiato in ventesimi di secondo: per suonare una nota per un secondo, quindi, il valore di D va posto a 20. Se a D viene assegnato al valore −1, il suono si protrae all'infinito o finché non si interviene con altri comandi per fermarlo.
All'effetto sonoro per l'esplosione abbiamo assegnato il volume massimo, espresso dal valore −15. La determinazione dell'altezza del suono è più complessa. Quando usiamo il canale 0, P assume valori compresi tra 0 e 7 che danno i seguenti risultati:

P=0	Suono ad alta frequenza
P=1	Suono a media frequenza
P=2	Suono a bassa frequenza
P=3	Suono, la cui frequenza è selezionata dal canale 1
P=4	Rumore ad alta frequenza
P=5	Rumore a media frequenza
P=6	Rumore a bassa frequenza
P=7	Rumore, la cui frequenza è selezionata dal canale 1

Il suono del primo gruppo di quattro forme d'onda (0-3) assomiglia a quello di un "cicalino", mentre le forme d'onda per il rumore (4-7) producono un fruscio simile a quello emesso da un televisore fuori sintonia. Il comando SOUND usato nel programma è:

SOUND 0,−15,4,40

che produce un sibilo di alta intensità ed a volume massimo, della durata di 2 secondi (40/20).

Quello che segue è il listato completo della procedura:

```
3550 DEF PROCesplosione(xesplos,yesplos)
3560 REM ** EFFETTO SONORO **
3570 SOUND 0,-15,4,40
3580 REM ** SELEZIONA RITMO LAMPEGGIAMENTO **
3590 *FX9,20
3600 *FX10,50
3610 FOR I=1 TO 100
3620 MOVE xesplos,yesplos
3630 VDU19,2,RND(15),0,0,0
3640 GCOL 0,RND(3)
3650 PLOT 1,RND(100)-50,RND(100)-50
3660 NEXT I
3670 PROCripristino
3680 ENDPROC
```

Dopo un'esplosione, il programma deve compiere alcune operazioni:
• Ridurre il numero di "vite" da giocare e verificare se tutte sono state usate o meno.
• Cancellare dal video l'esplosione.
• Riportare il rivelatore e l'assistente nella loro posizione originaria.
La prima operazione viene eseguita adoperando un contatore, incrementato ogni volta che la procedura viene chiamata. Se, dopo l'incremento, il valore del contatore è superiore a quattro, la variabile endflag viene posta a 1 per segnalare che il gioco è terminato. Eseguita la verifica di fine gioco, il programma deve cancellare l'esplosione. Questa operazione può essere realizzata con varie tecniche, ad esempio stampando spazi vuoti sull'area dell'esplosione. Nel nostro caso il metodo impiegato cancella lo schermo, colloca nuovamente le mine e visualizza il tempo e i punteggi. Per esser precisi, il programma deve ridisporre la stessa quantità di mine presente prima dell'esplosione. Il loro numero viene calcolato direttamente dal punteggio: poiché ogni mina vale 150 punti e all'inizio del gioco le mine erano 50, dividendo il punteggio per 150 si ottiene il numero delle mine già individuate che, sottratto a 50, darà il numero di quelle da reinserire sullo schermo. Questo valore viene passato alla procedura "mettimine".
Il rivelatore e l'assistente vengono riportati in posizione di partenza inizializzando le loro coordinate (ossia richiamando la procedura "inizializzavariabili"), quindi chiamando la procedura "stampacaratteri". Aggiungiamo al programma il seguente listato, relativo alla procedura "ripristino", assieme a quello della procedura "esplosione".

```
3880 DEF PROCripristino
3890 conto=conto+1
3900 IF conto>4 THEN endflag=1: ENDPROC
3910 CLS
3920 VDU19,2,2,0,0,0
3930 COLOUR 2
3940 PROCinizializzavariabili
3950 minerimaste=50-punti/150
3960 PROCdisponemine(minerimaste)
3970 PROCdisegnabordo
3980 PRINTTAB(2,27);"TEMPO"
3990 PRINTTAB(2,28)"PUNTEGGIO"
4000 PRINTTAB(18,28)punti$
4010 PRINTTAB(2,29)"PUNTEGGIO MAX."
4020 PRINTTAB(18,29)puntimax$
4030 uominirimasti$=LEFT$(men$,4-conto)
4040 COLOUR 1
4050 PRINTTAB(2,30);uominirimasti$;"
4060 COLOUR 2
4070 PROCstampacaratteri
4080 ENDPROC
```

Adesso è necessario introdurre la seguente modifica nel programma provvisorio di chiamata (vedere a pag. 885):

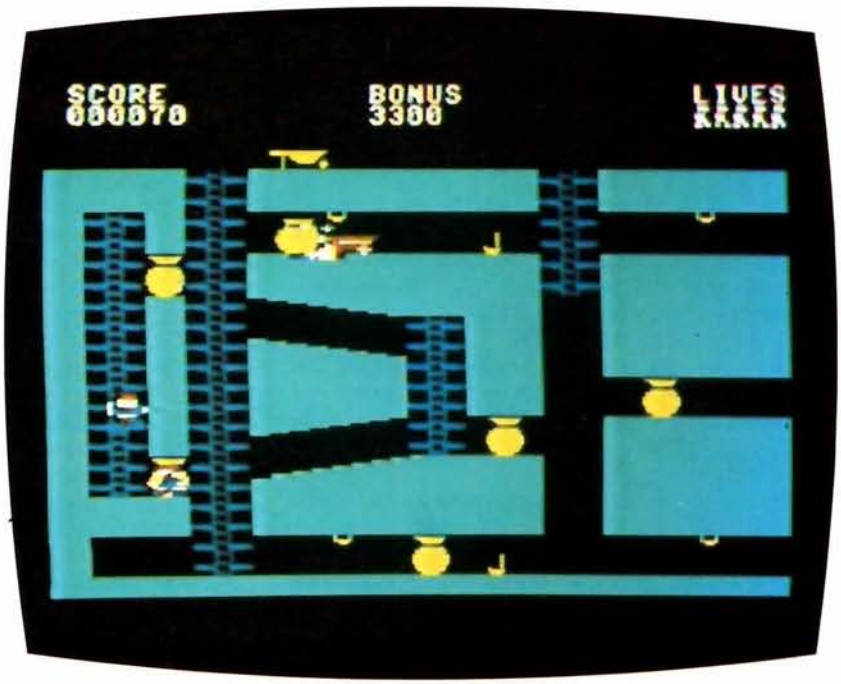
60 UNTIL TIME>12099 OR endflag=1

KEVIN JONES

Software in vetrina

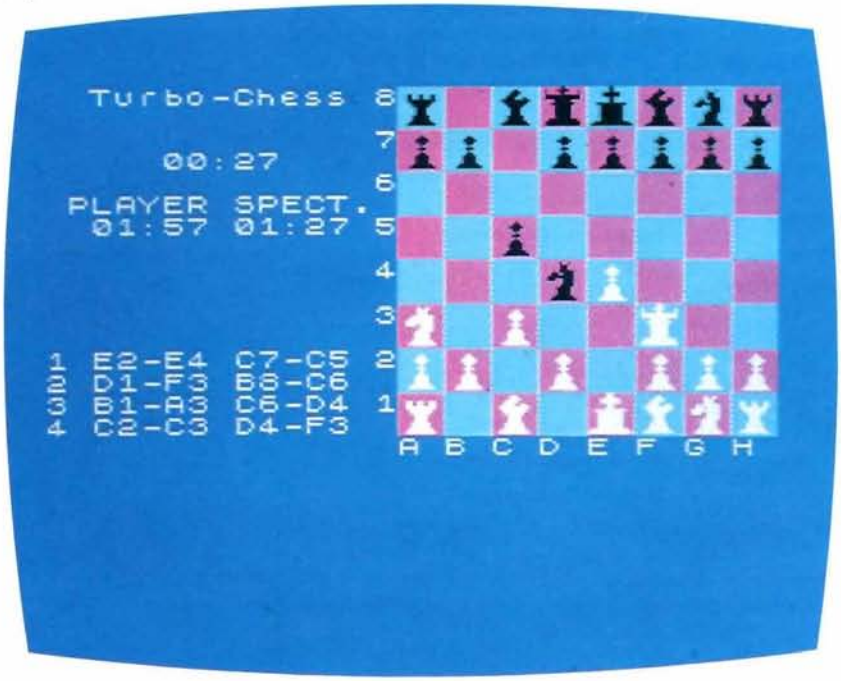
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

È questa la volta di due programmi appartenenti a generi completamente diversi: uno segue le tracce dell'ormai leggendario *Pac Man*, l'altro invece costituisce una sfida per gli scacchisti più accaniti.



Gilligan's Gold per il Commodore 64

Benché questo gioco sia basato su di un labirinto, le numerose variazioni aggiuntive fanno sì che esso si discosti dai tradizionali giochetti tipo *Pac Man*. L'ambiente è una miniera, nella quale "Gilligan", il protagonista (ossia il giocatore), deve raccogliere sacchi pieni d'oro e caricarli sulla propria carriola. Tutto ciò sembra semplice, ma non mancano i colpi di scena: la miniera è pericolosa e vi si aggirano banditi armati, carrelli in "corsa libera" ed altre diavolerie. Sparse, sul soffitto della miniera, ci sono delle maniglie, utili per saltare nei carrelli e spostarsi così da una parte all'altra. Il tempo a disposizione è molto breve: 4000 unità, ma per ogni sacco d'oro conquistato (è il caso di dirlo!) si guadagnano altre 4000 unità. Si tratta dunque di un gioco molto spassoso, prodotto dalla Interceptor Micros.



Turbo Chess per lo Spectrum

I programmi per giocare a scacchi contro un computer non si contano più. Non sempre si tratta di software di gran successo (c'è sempre qualche necessario compromesso tra varietà di gioco e memoria disponibile), ma il *Turbo Chess* prodotto dalla Kerlan sembra avere più di una carta in regola. Esso offre numerosi livelli di gioco, da selezionare in base alle capacità del giocatore, oltre alla possibilità di memorizzare una partita in qualsiasi momento, per riprendere il gioco in seguito. Il programma, nonostante la limitata memoria dello Spectrum, possiede notevoli routine di analisi ed una grafica molto più che decente. Per gli appassionati di scacchi, questo è un programma che non può mancare dalla collezione.

L'interfaccia Acorn Plus 1

Un apparecchio per portare le prestazioni dell'Electron al livello di quelle del BBC

L'Electron è stato ideato come versione ridotta del Micro BBC. I progettisti hanno mantenuto l'ottimo BASIC BBC ed il sistema operativo, ma, per non creare un duplicato e per diminuirne il prezzo, hanno eliminato molte delle interfacce che rendono il Micro BBC così versatile. L'Electron, pertanto, possiede un'unica porta d'interfaccia per cassetta, due prese per monitor (RGB o video composito), un modulatore di uscita per collegarsi ad un apparecchio televisivo ed una presa di alimentazione. L'interfaccia Plus 1 aggiunge all'apparecchio standard una porta parallela per stampante, un connettore per joystick e due slot per cartucce ROM "a innesto". Queste caratteristiche permettono di adoperare sull'Electron i giochi stile "arcade", mentre la porta per stampante è utile a chi occorre la stampa di listati oppure a chi intenda usare software per l'elaborazione di testi.

L'installazione dell'unità di espansione Plus 1 è molto semplice: l'interfaccia viene innestata sul connettore piatto che sporge dalla parte posteriore e viene fissata all'apparecchio tramite due bulloncini. Attraverso il connettore passano sia i dati, sia la corrente di alimentazione per l'interfaccia. Il software che controlla la nuova interfaccia è contenuto in un chip ROM, interno alla stessa unità.

La presa per la stampante è una porta Centronics parallela di tipo standard. Per attivare la stampante, occorre digitare sulla tastiera il comando VDU2. Il comando VDU3 serve per disattivarla. Entrambi i comandi sono identici a quelli usati sul Micro BBC. Analogamente, VDU1 permette di inviare i caratteri alla stampante: anche questo è un comando standard del BASIC BBC. I joystick vengono collegati alla porta analogica, una presa tipo-D a 15 piedini. Il tipo di collegamento è identico a quello usato per la porta analogica del Micro BBC, consentendo così l'impiego di qualsiasi joystick adatto per il BBC. L'interfaccia Plus 1 può controllare fino a quattro segnali analogici, provenienti da quattro strumenti di controllo per giochi (paddle), oppure da due joystick. Ogni joystick, infatti, produce due segnali: uno per i movimenti verso l'alto ed il basso, l'altro per i movimenti orizzontali.

Il processo di conversione dei segnali da analogici a digitali viene effettuato da un apposito chip di conversione "analogico/digitale". Nel caso dell'Electron, si tratta di un ADCO844, meno complesso di quello adottato dal Micro BBC. Quindi l'Electron non risponde ai movimenti del joystick con la stessa precisione di un computer BBC. Nei programmi di gioco, dove il joystick svolge esclusivamente la funzione di interruttore a 4 canali, non si hanno problemi. Tuttavia, se il joystick viene abbinato ad un programma grafico, ad esempio per disegnare "a mano libera" sullo schermo, sull'Electron si ottengono movimenti meno lineari ed immagini molto meno particolareggiate.

All'accensione dell'Electron Plus 1, viene automaticamente caricato il software contenuto nella cartuccia inserita nella slot ROM anteriore. Per bloccare questa procedura, basta premere il tasto Escape. Il sistema di registrazione su cartucce ROM è simile a quello su cassette (anche se è molto più veloce) ed accetta quindi comandi come *CAT, LOAD e CHAIN. Per usare software su cassette, con la nuova interfaccia Plus 1 collegata, occorre usare il comando *TAPE, oppure estrarre la cartuccia dopo aver spento l'apparecchio. Oltre che per i giochi e per il BASIC, il formato cartuccia è stato adottato anche per altri linguaggi di programmazione ma, diversamente dalle ROM di gioco, queste si sostituiscono alla ROM del BASIC.

Purtroppo, quando all'Electron viene collegata l'interfaccia Plus 1, la maggior parte degli accessori del computer produce fastidiosi effetti collaterali: errori di caricamento, ad esempio, quando vengono usati programmi registrati su cassetta, specialmente con i programmi che contengono "file" di dati (c'è chi ha trovato problemi nel caricare sull'Electron espanso alcuni degli stessi programmi della cassetta di corredo all'Electron versione

Una macchina... in sviluppo

L'Electron è stato inizialmente progettato come versione economica del Micro BBC. È infatti privo di molte delle interfacce del BBC, ma offre "a buon mercato" la stessa ottima grafica e l'eccellente BASIC. In seguito, la Acorn ha prodotto l'interfaccia Plus 1, che amplia le possibilità di comunicazione dell'Electron

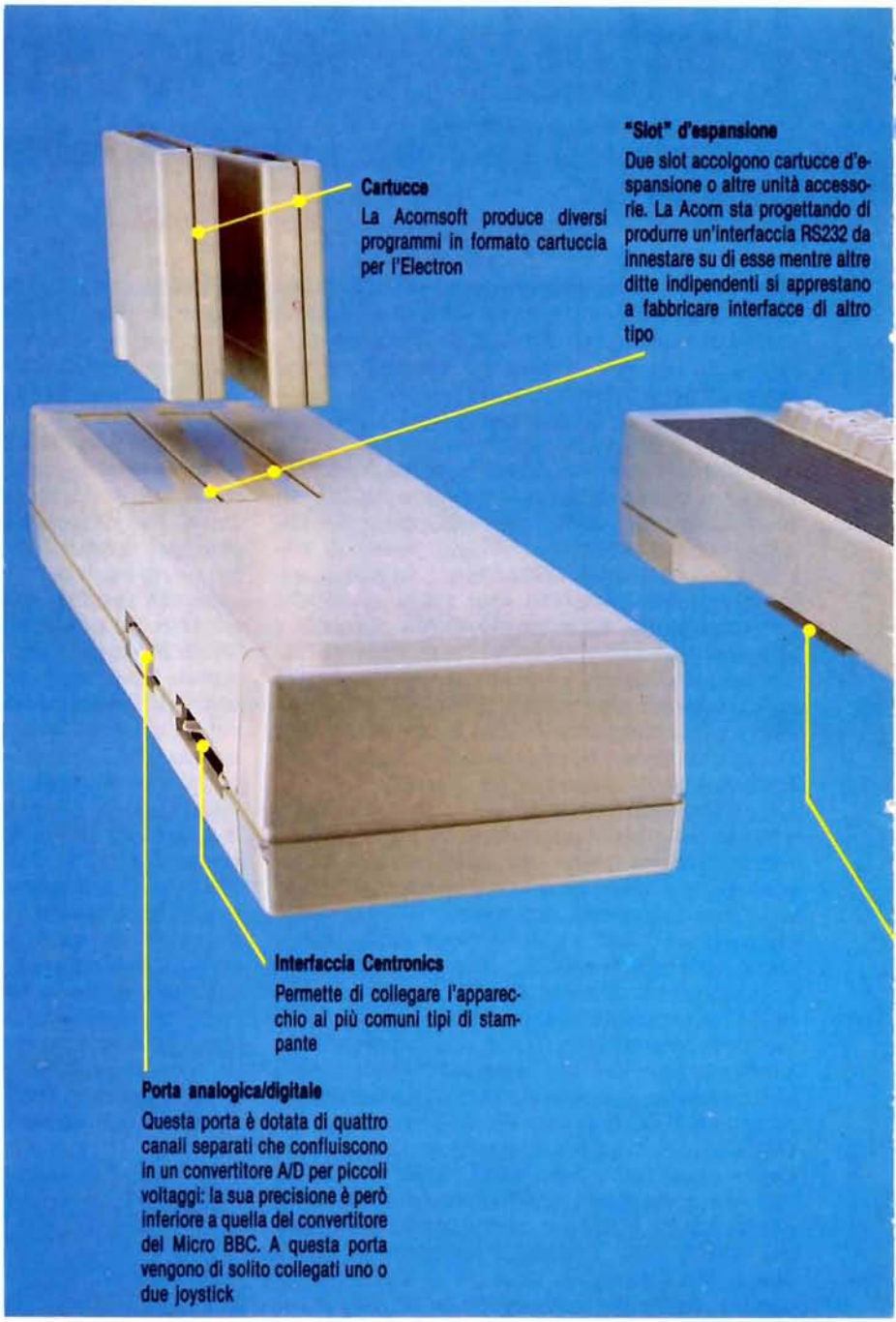


CHRIS STEVENS



I magnifici sei

Le cartucce dell'Electron offrono notevoli vantaggi rispetto ai normali nastri a cassetta perché il loro caricamento in memoria è molto più veloce (appena 2 secondi!). Inizialmente, sono state prodotte solo sei cartucce: quattro programmi di gioco, uno di didattica ed una versione del noto linguaggio LISP



"base"!). Comunque, tramite una sequenza di comandi operativi, è possibile "far credere" all'Electron che l'interfaccia Plus 1 non sia collegata ed ottenere quindi un funzionamento regolare. Questa tecnica viene appena accennata nel manuale d'uso e potrebbe confondere le idee agli utenti meno esperti.

Su tutti i computer della Acorn, battendo * HELP, compare una lista delle varie ROM dell'apparecchio. Usando questo comando sull'Electron con interfaccia Plus 1, si viene a sapere che la ROM principale del sistema operativo è OS 1.00, a cui segue il messaggio "Expansion 1.00 ADC/Printer/RS423", che si riferisce alla ROM interna all'unità d'espansione Plus 1. L'informazione più curiosa è l'ultima: RS423. Questa sigla indica un'interfaccia seriale standard che né l'Electron né l'interfaccia Plus 1 possiedono. Questo disguido è dovuto al fatto che la Acorn sta progettando di dotare l'Electron di un'interfaccia seriale RS423.

La principale differenza tra i due dialetti BASIC è che l'Electron è privo del MODE 7, cioè di grafica compatibile con il sistema teletext. Questo MODE viene spesso usato nei programmi del Micro BBC per i titoli e le pagine di istruzioni, poiché permette di economizzare memoria. Sull'Electron, non essendo previste le varie opzioni "viewdata" del MODE 7, si ottengono sullo schermo immagini prive di senso. La maggior parte dei programmi di gioco, però, usa gli altri MODE a disposizione, quindi il problema può essere, almeno in parte, aggirato.

Una seconda differenza tra i due apparecchi riguarda i comandi SOUND ed ENVELOPE. L'Electron possiede un solo canale audio, mentre il Micro BBC ne ha addirittura quattro. Inoltre, il comando ENVELOPE dell'Electron interviene sulla tonalità del suono e non sul suo volume. Nonostante ciò, i comandi sono tra loro compatibili ed i programmi in cui compaiono possono essere usati su entrambi gli apparecchi; ovviamente i suoni generati dall'Electron saranno notevolmente diversi.

La tastiera dell'Electron si dimostra più "sensibile" di quella del Micro BBC, ma possiede un numero minore di tasti. Quindi sull'Electron (come ad esempio sullo Spectrum Sinclair) la maggior parte dei tasti svolge due o tre funzioni diverse. Il tasto Shift serve per selezionare le minuscole o le maiuscole, ma premendo assieme il tasto Function e la "L", ad esempio, si ottiene il comando LIST del BASIC, mentre la combinazione dei tasti Control ed "L" ripulisce lo schermo. Forse sarebbe stato meglio conservare i 10 tasti di funzione (quelli colorati in rosso), presenti sul Micro BBC, riprogrammabili dall'utente. L'Electron usa come tasti di funzione i tasti numerici, abbinati al tasto Function.

I due apparecchi differiscono tra loro soprattutto nel campo delle interfacce. Il Micro BBC ne possiede un numero superiore a qualsiasi altro home computer e può quindi collegarsi a svariate periferiche: modem, stampanti, unità per dischetti, un processore aggiuntivo (Z80) e

"Snapper"
Lo Snapper della Acornsoft è una buona versione del classico gioco "Pac Man". È in formato cartuccia e può essere giocato da tastiera o con un joystick. Ne esiste anche una versione di prezzo inferiore, su cassetta





Connettore d'espansione
Questo connettore piatto consente di collegare l'interfaccia Plus 1 all'Electron ed è l'unica porta d'espansione presente sull'apparecchio

perfino piccoli robot. Tutte queste periferiche richiedono chip e prese addizionali, che accrescono inevitabilmente i costi di produzione. La loro eliminazione ha portato ad una riduzione del numero di chip e del consumo di corrente: l'apparecchio risulta quindi più piccolo e meno costoso. Era appunto su questa filosofia che poggiava il progetto originale dell'Electron; il fatto che l'Acorn abbia poi introdotto l'Electron Plus 1 indica forse un ripensamento da parte della casa produttrice. L'Electron Plus 1 è meno versatile del Micro BBC ed offre minori possibilità d'espansione. Nonostante ciò, molti utenti lo preferiscono per il basso prezzo, ma soprattutto per la sua ottima grafica, per le funzioni di suono, per il BASIC strutturato e per la crescente disponibilità di software.

MICRO BBC MODELLO B

DIMENSIONI

mm. 75 x 340 x 410

CPU

6502 a 1,8 MHz

MEMORIA

32 Kbyte di RAM, 32 Kbyte di ROM

VIDEO

8 modi di visualizzazione. Risoluzione massima testuale: 80 x 32 caratteri, grafica: 640 x 256 pixel. Fino ad otto colori disponibili, più variazioni (lampeggiamento). Caratteri definibili da utente e grafica teletext

INTERFACCE

UHF per televisore, RGB e video composito per monitor. Interfaccia per registratore, porta RS423 e parallela Centronics. Porta analogica (per joystick, ecc.), connettori per ROM (software aggiuntivo). Interfaccia "Tube" (per secondo microprocessore), bus a 1 MHz. Le interfacce per dischetti e per la rete Econet sono opzionali. Porta e presa di corrente ausiliaria

LINGUAGGI DISPONIBILI

BASIC BBC (incluso), Assembler 6502 (incluso), LISP, FORTH, BCPL e PASCAL

TASTIERA

Tipo macchina da scrivere, con 62 tasti, più 10 di funzione, programmabili da utente

DOCUMENTAZIONE

I manuali del BBC sono eccellenti per l'utente già un po' pratico, ma forse leggermente oscuri per il principiante

PRO

Il BBC possiede una delle migliori versioni di BASIC per home computer, ha numerose interfacce, un'ottima grafica e si presta alla didattica

CONTRO

La gran quantità di comandi a disposizione (tutt'altro che un difetto, però!) può sconcertare un principiante

ACORN ELECTRON PLUS 1

DIMENSIONI

mm. 65 x 260 x 340 (con interfaccia Plus 1 inserita)

CPU

6502 a 1,8 MHz

MEMORIA

32 Kbyte di RAM, 32 Kbyte di ROM

VIDEO

7 modi di visualizzazione. Risoluzione massima testuale: 80 x 32 caratteri, grafica: 640 x 256 pixel. Fino ad otto colori, più variazioni (lampeggiamento). Caratteri definibili da utente

INTERFACCE

UHF per televisore, RGB e video composito per monitor. Interfaccia per registratore, porta parallela per stampante. Porta analogica, connettori per ROM (software aggiuntivo)

LINGUAGGI DISPONIBILI

BASIC BBC (incluso), Assembler 6502 (incluso), LISP (anche su cartuccia), S-PASCAL

TASTIERA

Tipo macchina da scrivere, con 56 tasti, 10 dei quali programmabili. L'immissione comandi BASIC si può fare con la pressione simultanea del tasto funzione e di uno dei normali tasti

DOCUMENTAZIONE

I manuali dell'Electron sono eccellenti anche per il principiante. Oltre al BASIC viene trattato anche il codice macchina del 6502

PRO

La grafica è ottima, come anche la qualità della tastiera. La versione di BASIC è molto buona

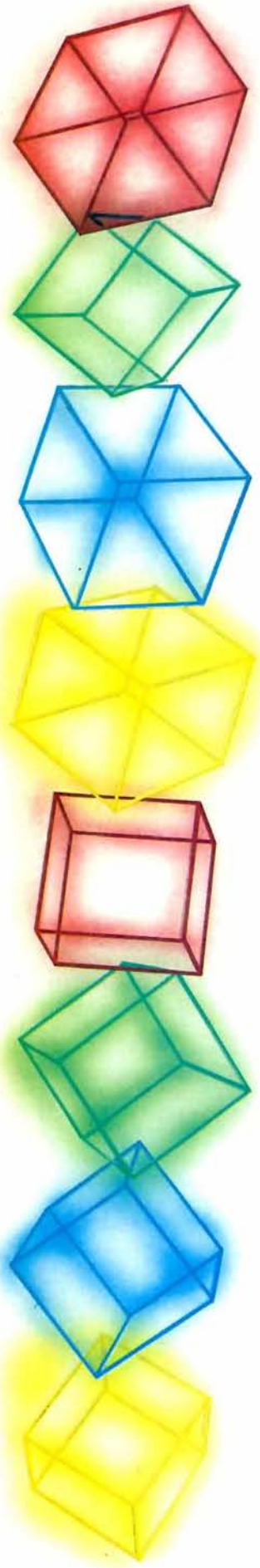
CONTRO

L'uso "multifunzione" dei tasti può creare qualche perplessità e difficoltà iniziali. Esiste un solo canale audio. Non esiste il modo di visualizzazione "teletext" e manca una porta di comunicazione





CHRIS STEVENS



Cubi in rotazione

I principi di base per produrre grafica animata sul proprio computer

Questo programma usa alcuni principi basilari della geometria per creare proiezioni prospettiche di figure geometriche, osservabili sotto qualsiasi angolo o distanza. Tutti i dati riguardanti le figure vengono memorizzati in istruzioni DATA del programma e si riferiscono alle coordinate tridimensionali di ogni punto terminale delle linee che compongono l'oggetto. Alle coordinate di ciascun punto, inoltre, viene associato un numero, per indicare se il punto costituisce l'inizio o la fine di una linea.

La trasformazione delle coordinate tridimensionali in valori bidimensionali specifici per i vari punti dello schermo implica operazioni matematiche semplici, ma piuttosto lunghe. Le coordinate x e y di ciascun punto (sul piano dello schermo televisivo) vengono divise per un fattore che indica la distanza dell'oggetto dall'osservatore. Quindi i risultati ottenuti vengono messi in scala secondo un parametro adeguato al sistema di coordinate del micro in uso. Variando il fattore di distanza, le dimensioni della figura cresceranno o diminuiranno, come se l'oggetto si avvicinasse o si allontanasse dall'osservatore.

Infine, è possibile usare una terza costante per cambiare gli effetti della proiezione prospettica: aumentandone il valore, la visione prospettica dell'oggetto cresce come se lo si guardasse attraverso una lente d'ingrandimento. Se, al contrario, il valore della costante diminuisce, si ottiene un apparente appiattimento dell'oggetto, simile a quello causato da un teleobiettivo.

Oltre a creare un'immagine prospettica dell'oggetto, il programma permette di ruotarlo per poterlo osservare sotto diverse angolazioni. La rotazione si basa su semplici equazioni trigonometriche. L'asse della figura viene ruotata dell'angolo desiderato per cui, quando ne viene fatta la proiezione, la nuova immagine appare ruotata.

Questo risultato si può ottenere ruotando sia l'asse y (il punto di vista sembra muoversi attorno all'oggetto), che l'asse x (il punto di vista si sposta dall'alto verso il basso o viceversa).

Sebbene il programma possa realizzare tutti questi effetti, le procedure su cui si basa sono sorprendentemente semplici. Il punto di vista dell'osservatore viene selezionato con i tasti numerici compresi tra 1 e 8 che rispettivamente provocano i seguenti spostamenti rispetto all'oggetto: a sinistra, a destra, in alto, in basso, di avvicinamento, di allontanamento, un aumento dell'effetto prospettico (occhio di pesce) ed infine una sua diminuizione (teleobiettivo).

Le coordinate tridimensionali della figura "corrente" vengono registrate in tre variabili numeriche: X, Y e Z. I cambiamenti di queste variabili e delle costanti, usati per le variazioni di prospettiva, vengono calcolati da una serie di subroutine. Il programma verifica costantemente la tastiera, per rilevare se viene premuto un tasto. In tal caso, cancella l'immagine sullo schermo (rintracciandola nel colore dello sfondo), chiama la routine che si occupa di elaborare la modifica richiesta ed infine traccia la nuova immagine.

La modifica della prospettiva viene realizzata dalla subroutine che disegna la figura. Questa esamina ciascun gruppo di coordinate tridimensionali, le trasforma in coordinate bidimensionali e traccia la nuova figura (modificando i punti iniziali e finali oppure tracciando le linee in base al quarto elemento dei dati di ciascun punto).

La formulazione dei dati da inserire nel programma per creare una nuova figura richiede un certo tempo, ma non presenta grosse difficoltà. Questi nuovi dati, quattro valori per ogni punto della figura, vengono immagazzinati in istruzioni poste alla fine del programma, mentre il numero totale dei punti viene definito nella prima linea. I dati del nostro programma creano un cubo con una linea diagonale su una faccia.

Ciascun gruppo di quattro valori indica, nell'ordine, il parametro di tracciamento e le coordinate x, y e z. I valori si possono calcolare pensando ad una rappresentazione tridimensionale dell'oggetto: immaginiamo di usare una penna e di portarci di volta in volta su tutti gli angoli della figura. Se spostiamo la nostra penna da un punto a un altro senza tracciare una linea, allora usiamo 4 come primo valore; in caso contrario si deve usare il valore 5. Sono stati scelti questi particolari valori perché rendono meno complicato il programma per il Micro BBC.

Le coordinate di origine (0,0) si trovano al centro dello schermo: conviene quindi considerare questo punto come "centro" dell'oggetto. L'asse x rappresenta l'asse orizzontale, l'asse y quello verticale, mentre l'asse z è quello perpendicolare allo schermo e la sua direzione positiva va verso l'interno di esso. Ad x, y e z si dovranno assegnare valori quanto più piccoli possibile. Definendo i valori iniziali della prospettiva e della distanza del punto di vista, si deve tener presente che un oggetto largo 10 riempie l'intero schermo. Questo valore potrà essere modificato cambiando il fattore di scala impiegato nella trasformazione prospettica. È consigliabile fare i primi tentativi con figure semplici, con

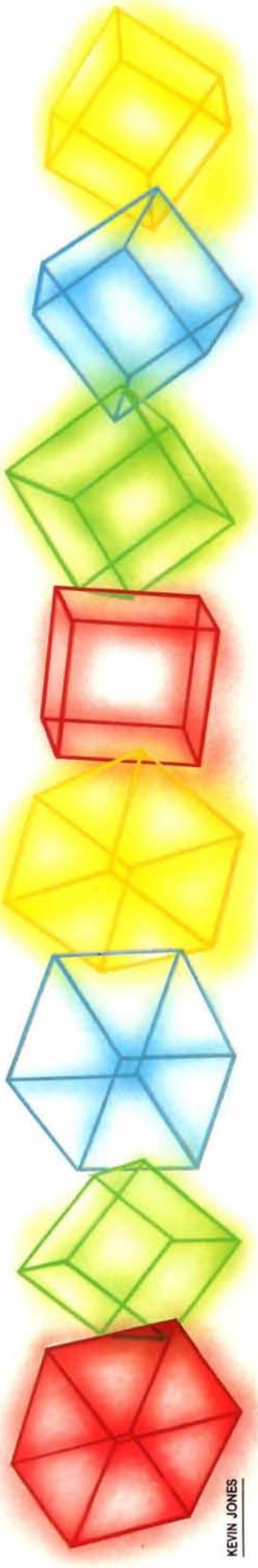
un ridotto numero di punti. Quando avremo imparato a progettare forme solide semplici (cubo, piramide, ecc.) ci potremo dedicare a figure più complesse.
Il programma può essere ampliato, per ottenere altri effetti speciali. Si potrebbe introdurre la traslazione di un oggetto rispetto all'origine delle coordinate, ma senza rotazione. Oppure inserire una routine che elimini le linee "nascoste" (ossia quelle parti che si trovano in secondo piano rispetto all'osservatore). Questo effetto renderebbe l'attuale immagine reticolare molto più realistica, ma richiede procedure matematiche piuttosto complesse che rallentano notevolmente il programma. Tuttavia, anche senza queste modifiche, si possono ottenere risultati spettacolari.

VERSIONE SPECTRUM

```
 10 LET N=16
 20 DIM P(50)
 21 DIM X(50)
 22 DIM Y(50)
 23 DIM Z(50)
 24 DIM A(50)
 25 DIM B(50)
 40 LET D=10: LET P=0.5
 50 LET SI=SIN 0.09: LET CO=COS 0.09
 60 FOR I=1 TO N
 70 READ P(I), X(I), Y(I), Z(I)
 80 NEXT I
 90 :
200 INVERSE 0: GO SUB  300
210 IF INKEY$<>"" THEN GO TO 210
211 IF INKEY$="" THEN GO TO 211
212 LET I$=INKEY$
230 INVERSE 1: GO SUB 300
240 IF I$="1" THEN GO SUB 1000
241 IF I$="2" THEN GO SUB 2000
242 IF I$="3" THEN GO SUB 3000
243 IF I$="4" THEN GO SUB 4000
244 IF I$="5" THEN GO SUB 5000
245 IF I$="6" THEN GO SUB 6000
246 IF I$="7" THEN GO SUB 7000
247 IF I$="8" THEN GO SUB 8000
250 GO TO 200
260 :
300 FOR I=1 TO N
310 LET A(I)=X(I)*300/(P*Z(I)+D):
    LET B(I)=Y(I)*300/(P*Z(I)+D)
320 NEXT I
330 FOR I=1 TO N
340 IF P(I)=4 THEN PLOT A(I)+128,B(I)+85
345 IF P(I)=5 THEN DRAW A(I)-A(I-1),B(I)-B(I-1)
350 NEXT I
360 RETURN
370 :
1000 FOR I=1 TO N
1010 LET X=X(I)*CO-Z(I)*SI
1020 LET Z=Z(I)*CO+X(I)*SI
1030 LET X(I)=X: LET Z(I)=Z
1040 NEXT I
1050 RETURN
1060 :
2000 FOR I=1 TO N
2010 LET X=X(I)*CO+Z(I)*SI
2020 LET Z=Z(I)*CO-X(I)*SI
2030 LET X(I)=X: LET Z(I)=Z
2040 NEXT I
2050 RETURN
2060 :
3000 FOR I=1 TO N
3010 LET Y=Y(I)*CO+Z(I)*SI
3020 LET Z=Z(I)*CO-Y(I)*SI
3030 LET Y(I)=Y: LET Z(I)=Z
3040 NEXT I
3050 RETURN
3060 :
4000 FOR I=1 TO N
4010 LET Y=Y(I)*CO-Z(I)*SI
4020 LET Z=Z(I)*CO+Y(I)*SI
4030 LET Y(I)=Y: LET Z(I)=Z
4040 NEXT I
4050 RETURN
4060 :
5000 LET D=D*0.9
5010 RETURN
5020 :
6000 LET D=D/0.9
6010 RETURN
6020 :
7000 LET P=P/0.9
7010 RETURN
7020 :
8000 LET P=P*0.9
8010 RETURN
8020:
9000 DATA 4,1,1,1, 5,1,1,-1, 5,-1,1,-1,
     5,-1,1,1, 5,1,1,1
9010 DATA 5,1,-1,1,  5,1,-1,-1, 5,-1,-1,-1,
     5,-1,-1,1, 5,1,-1,1
9020 DATA 4,1,-1,-1, 5,1,1,-1, 5,-1,-1,-1,
     5,-1,1,-1
9030 DATA 4,-1,1,1, 5,-1,-1,1
```

VERSIONE BBC

```
10 N=16
20 DIM P(50),X(50),Y(50),Z(50),A(50),B(50)
30 MODE 0: VDU 29,640;512;5
40 D=10: P=0.5
50 SI=SIN(0.09): CO=COS(0.09)
60 FOR I=1 TO N
70   READ P(I),X(I),Y(I),Z(I)
80   NEXT I
90 :
200 GCOL0,3: GOSUB 300
210 I$=GET$
220 V=VAL(I$)
230 GCOL0,0: GOSUB 300
240 ON V GOSUB 1000,2000,3000,4000,5000,
    6000,7000,8000 ELSE 150
250 GOTO 200
260 :
300 FOR I=1 TO N
310 A(I)=X(I)*1000/(P*Z(I)+D):
    B(I)=Y(I)*1000/(P*Z(I)+D)
320   NEXT I
330 FOR I=1 TO N
340   PLOT P(I),A(I),B(I)
350   NEXT I
360 RETURN
370 :
1000 FOR I=1 TO N
1010   X=X(I)*CO-Z(I)*SI
1020   Z=Z(I)*CO+X(I)*SI
1030   X(I)=X: Z(I)=Z
1040   NEXT I
1050 RETURN
1060 :
2000 FOR I=1 TO N
2010   X=X(I)*CO+Z(I)*SI
2020   Z=Z(I)*CO-X(I)*SI
2030   X(I)=X: Z(I)=Z
2040   NEXT I
2050 RETURN
2060 :
3000 FOR I=1 TO N
3010   Y=Y(I)*CO+Z(I)*SI
3020   Z=Z(I)*CO-Y(I)*SI
3030   Y(I)=Y: Z(I)=Z
3040   NEXT I
3050 RETURN
3060 :
4000 FOR I=1 TO N
4010   Y=Y(I)*CO-Z(I)*SI
4020   Z=Z(I)*CO+Y(I)*SI
4030   Y(I)=Y: Z(I)=Z
4040   NEXT I
4050 RETURN
4060 :
5000 D=D*.9
5010 RETURN
5020 :
6000 D=D/.9
6010 RETURN
6020 :
7000 P=P/.9
7010 RETURN
7020 :
8000 P=P*.9
8010 RETURN
8020 :
10000 DATA 4,1,1,1, 5,1,1,-1, 5,-1,1,-1,
      5,-1,1,1, 5,1,1,1
10010 DATA 5,1,-1,1, 5,1,-1,-1, 5,-1,-1,-1,
      5,-1,-1,1, 5,1,-1,1
10020 DATA 4,1,-1,-1, 5,1,1,-1, 5,-1,-1,-1,
      5,-1,1,-1,
10030 DATA 4,-1,1,1, 5,-1,-1,1
```



KEVIN JONES

Programmazione modulare

Semplici ma efficaci regole per migliorare lo stile di programmazione

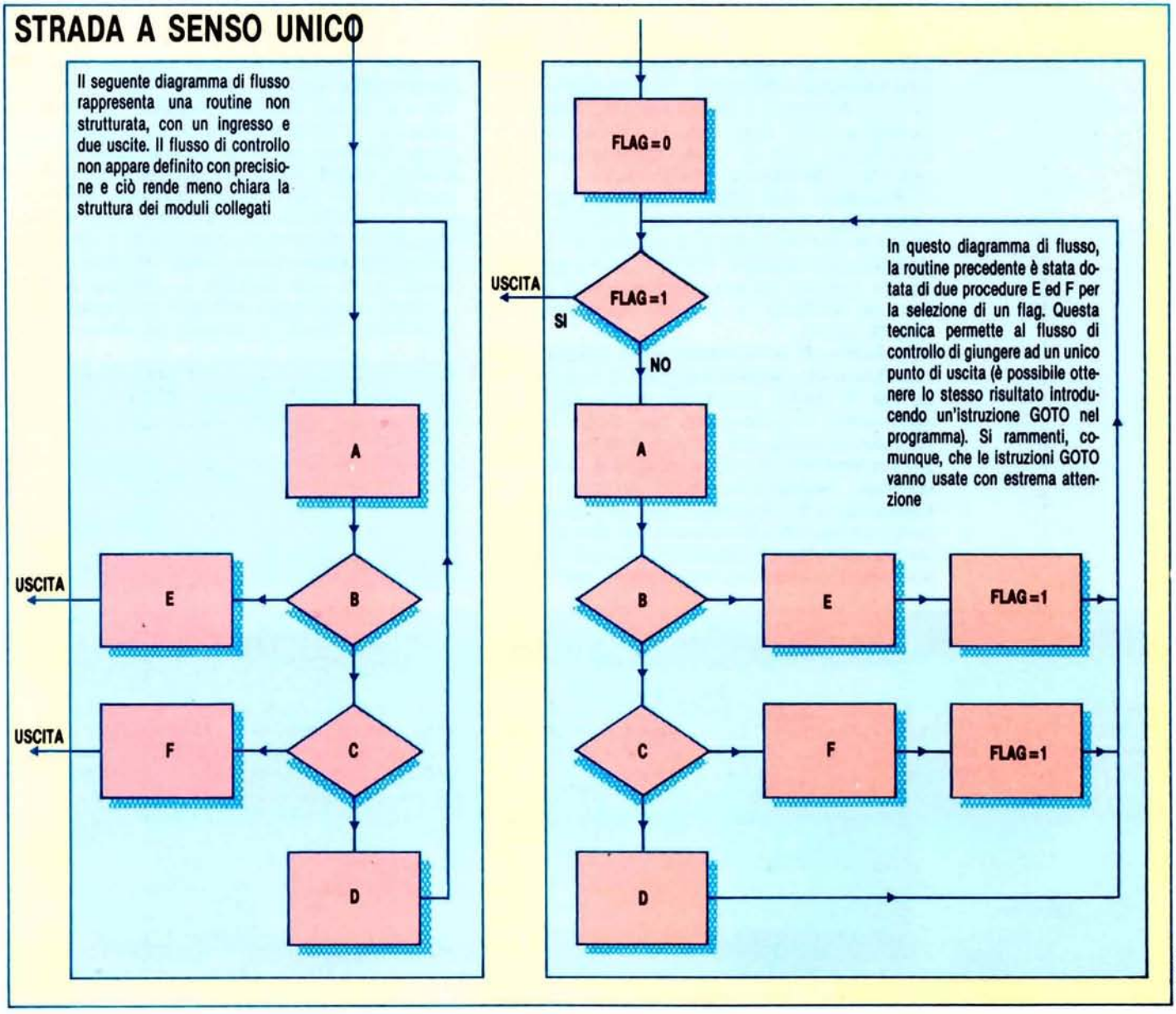
Un modulo è una parte di programma che svolge una particolare funzione. I punti di ingresso e di uscita, chiamati "interfacce" del modulo, vanno definiti con precisione. Inoltre, l'elaborazione eseguita all'interno di esso deve risultare indipendente dal resto del programma. Una volta redatto, un modulo va considerato come una "scatola nera": ci saranno informazioni che entrano e che escono dal modulo, ma le procedure d'elaborazione interna rimangono autonome.
I moduli così realizzati possono essere collegati tra loro per costruire un programma, senza che ci si debba preoccupare di come eseguono il loro lavoro. Il programmatore può scrivere una serie di moduli per poi utilizzarli successivamente o inserirli in programmi scritti da altri. Per trarre il massimo vantaggio dalla programmazione modulare strutturata, il programmatore deve prestare molta attenzione al flusso di controllo ed al flusso dei dati nel programma. Una semplice regola alla quale attenersi è la seguente: un modulo deve presentare una sola entrata ed una sola uscita. Ciò significa che il flusso di controllo all'interno del modulo deve iniziare da un preciso punto d'ingresso per poi raggiungere, indipendentemente dal numero di cicli e di salti eseguiti, una stessa uscita (pur giungendovi da percorsi diversi).
I moduli corrispondono, grosso modo, agli algoritmi di cui abbiamo già trattato. I linguaggi "strutturati", quale il PASCAL, permettono la creazione di subroutine, richiamabili mediante il nome loro assegnato, che usano variabili "interne" o "locali". Simili linguaggi inducono il programmatore a entrare o uscire da una routine (chiamata "procedura") servendosi esclusivamente dei suoi punti d'ingresso e di uscita.
In BASIC, usando la combinazione GOSUB...RETURN, è possibile richiamare una subroutine dal corpo principale del programma e, dopo la sua esecuzione, riportare il flusso di controllo sulla linea immediatamente successiva all'istruzione GOSUB. La GOSUB può passare il controllo a qualsiasi linea del programma senza alcuna limitazione. Due differenti GOSUB possono quindi rinviare a differenti linee di una subroutine che termina con un unico RETURN ma i risultati ottenuti possono essere completamente diversi. Similmente, una subroutine può contenere un numero illimitato di istruzioni RETURN.
È quindi necessario che il programmatore BASIC si imponga alcune regole di comportamento, ad esempio: assicurarsi che tutte le istruzioni GOSUB per una stessa subroutine rimandino al medesimo numero di linea e che ogni subroutine contenga un'unica istruzione RETURN. Il miglior modo di procedere è inserire nella prima linea di ogni subroutine un'istruzione REM contenente il nome della subroutine ed usare questa linea come punto d'ingresso. L'istruzione RETURN dovrà comparire nell'ultima linea della subroutine. Questa tecnica non è obbligatoria, ma rende il programma molto più comprensibile.

La regola del "goto"

Una particolare attenzione bisogna rivolgere al comando GOTO, che può danneggiare in modo irreparabile la struttura del programma. È buona regola attenersi al seguente principio: usare GOTO per passare il controllo esclusivamente ad una linea "interna" alla subroutine. Questa condizione elimina il rischio di saltare un'istruzione RETURN o di passare il controllo ad una RETURN non appropriata. Tuttavia, a volte può essere necessario uscire dalla routine senza eseguirne tutte le linee: in questo caso l'istruzione GOTO deve indicare la linea della subroutine che contiene l'istruzione RETURN.
L'uso di GOTO, all'interno di un ciclo, è ancora più pericoloso. Se il controllo "salta fuori" dal ciclo, il BASIC può non esserne informato e considerare il resto del programma come parte integrante! La regola da seguire è questa: quando ci si trova all'interno di un ciclo, nessuna istruzione GOTO deve riferirsi ad una linea esterna al ciclo stesso. Per chiudere un ciclo prima del previsto, si può usare un contatore di ciclo oppure verificare se una variabile ha raggiunto un dato valore. In questo caso, GOTO passerà il controllo alla linea di verifica (cioè una linea che contiene l'istruzione NEXT o WHILE). Anche in tal caso le istruzioni NEXT o WHILE devono trovarsi su una linea interna alla subroutine. Un limitato impiego delle istruzioni GOTO rende la struttura del programma molto più chiara.
Anche le istruzioni di salto condizionale comportano grossi rischi: le loro uscite possono indicare una linea esterna alla subroutine solo quando implicano la chiamata di una seconda subroutine. Il diagramma di flusso della routine permette di individuare più facilmente eventuali errori nei salti. Talvolta, un opportuno impiego di un "flag" consente di ridurre sensibilmente l'uso di istruzioni GOTO.
Il flusso dei dati che attraversa i moduli è molto simile a quello degli algoritmi (vedere a pag. 866). Poiché i moduli possono essere usati in modo indipendente, occorre collegarli in modo da limitare le loro interazioni ad un semplice scambio di dati. Il programma principale "passa" una serie di dati a un modulo

che li elabora e ritrasmette indietro i risultati ottenuti.
I dati che circolano all'interno del programma sono contenuti in variabili la cui libertà di movimento viene detta "scopo". Molti linguaggi di programmazione possono restringere lo scopo di una variabile a particolari subroutine. In PASCAL, le variabili usate in una subroutine (procedura) devono essere "dichiarate" per quella specifica procedura. Le variabili dichiarate per il programma principale sono chiamate "globali" e possono essere usate in qualsiasi parte del programma (anche all'interno dei suoi moduli), mentre le variabili dichiarate all'interno di una procedura sono dette "locali" ed hanno valore soltanto in quella particolare procedura.
Una variabile locale può avere un nome identico a quello di una variabile globale, ma l'uso di una delle due non influenza l'altra. Sfortunatamente, sono pochi i dialetti BASIC che consentono l'uso di variabili locali. Quindi, se si vogliono scrivere subroutine indipendenti, si devono in qualche modo simulare gli "effetti" prodotti dalle variabili locali.
Il modo più semplice per ottenere questo risultato è servirsi di nomi che eseguono compiti differenti. Alcune convenzioni già esistono e vengono abbondantemente usate dai programmatori. I nomi convenzionali, per identificare le variabili I, J e K, ad esempio, vengono usati per indicare contatori di ciclo e valori di indice.
Dopo aver redatto il diagramma di flusso di un programma, è abbastanza semplice numerare le subroutine o assegnare loro altri codici d'identificazione. Per usare una variabile globale come variabile locale all'interno di una subroutine, le si può aggiungere come prefisso il codice caratteristico di quella subroutine. Ad esempio, la routine 5 usa le variabili locali SUM5 e TOTAL5 per distinguerle da SUM12 e TOTAL12 della routine 12. Ci si deve però assicurare che il dialetto BASIC che si sta usando non prenda in esame solo i primi due caratteri della variabile!

STRADA A SENSO UNICO



LIZ DIXON

Il Valhalla

Un gioco d'avventura, arricchito da elementi della mitologia nordica

Valhalla possiede 81 scenari: 16 di questi, compreso lo sfuggente Valhalla della mitologia vichinga, si trovano nella regione di Asgard, 20 nella regione di Midgard ed i rimanenti 45 fanno parte di Hell. Ogni scena del gioco presenta 8 uscite ma mentre alcune possono risultare chiuse, altre si possono attraversare solo se muniti di adeguati oggetti magici. Il gioco è reso ancor più complicato dalla presenza di sentieri circolari, che collegano tra loro località distanti: se si imbocca il sentiero giusto, si può "saltare" ad una scena molto più avanti nel gioco, ma può anche accadere il contrario!

Il *Valhalla* ha un "cast" di 36 personaggi, suddivisi fra "buoni" e "cattivi": solo tre di essi possono comparire contemporaneamente sullo schermo. I personaggi interagiscono continuamente tra loro, si combattono, si scambiano cibo o vino, oppure scagliano oggetti a chi rimane "antipatico".

Il giocatore può entrare in contatto con i personaggi in qualsiasi momento, usando la tastiera per inserirsi nell'azione alla quale desidera partecipare. D'altra parte, potrebbe anche sedersi ed osservare ciò che accade ma così facendo, va di solito incontro a una rapida morte.

Non esiste un solo percorso per raggiungere l'ambita meta, quindi il giocatore può scegliere tra le molte strade possibili. Egli deve convincere i personaggi ad aiutarlo (non necessariamente deve chiedere assistenza ai personaggi buoni...). La grafica del *Valhalla* è discreta, soprattutto nella versione per il Commodore 64. Per ciascuna fase del gioco il programma visualizza uno scenario, in colori vivaci sullo Spectrum ed in tinte più sfumate sul Commodore. Poi appaiono i vari personaggi, che nella versione Spectrum sono raffigurati in bianco e nero, mentre sul Commodore appaiono più finemente disegnati, grazie all'alta risoluzione grafica ed alle maggiori capacità cromatiche dell'apparecchio. Infine, compaiono gli oggetti che corredano quella particolare scena: cibi, vino, gioielli, chiavi ed armi. Ancora una volta le immagini della versione Spectrum sono meno realistiche di quelle del Commodore.

I personaggi cominciano ad interpretare il loro ruolo e lo sviluppo dell'azione viene descritto da frasi che appaiono sulla parte inferiore dello schermo. A questo punto entra in azione il giocatore, che può intervenire nella trama in vario modo: può cercare di convincere una delle "divinità nordiche" a condividere con lui le sue ricchezze oppure può anche attaccare il dio con un'arma, per saggiarne la forza. Dopo un certo tempo il giocatore vorrà andare avanti nel gioco ed esplorare nuovi territori alla ricerca del favoloso Valhalla. Non è però facile impadronirsi degli oggetti magici che gli permetterebbero di entrare nel mitico Paradiso; dovrà quindi superare incredibili prove per portare a termine la sua impresa. L'eccessiva difficoltà di questa ricerca è forse il limite principale del gioco.

Valhalla: Per lo Spectrum a 48 K
Per il Commodore 64
Editori: Legend, (GB)
Autori: Graham Asher, Richard Edwards, Charles Goodwin, James Learmont, Jan Ostler, Andrew Owen, John Peel
Joystick: Non richiesto
Formato: Cassetta

GIORNO E NOTTE

Nel "Valhalla", come in molti recenti giochi, il passaggio dal giorno alla notte è rappresentato graficamente usando tinte leggermente più scure





LIZ HEANEY

I cerchi sul Commodore

Prosegue l'esame delle routine grafiche per questo home computer

Con un home computer è pressoché impossibile tracciare cerchi perfetti. L'esattezza dell'approssimazione dipende dal metodo usato e dalla lunghezza e complessità della routine finale. In BASIC, i programmi per tracciare cerchi di solito richiedono calcoli basati sulle funzioni di seno e coseno oppure su elevazioni al quadrato e su radici quadrate, per determinare le coordinate dei punti della circonferenza. Tuttavia, entrambi questi metodi presentano difficoltà quando si tenta di svilupparli in codice macchina.
Esaminiamo adesso un procedimento alternativo, particolarmente adatto ad una programmazione in codice macchina. Questo metodo considera il diametro di una circonferenza suddiviso in un certo numero di parti uguali, ciascuna di lunghezza W. Immaginiamo ora che da ciascuno dei punti di divisione parta un segmento perpendicolare che termina in un punto C, posto sulla circonferenza. Il diagramma illustra uno di tali segmenti, posto ad N divisioni dall'estremità sinistra del diametro AB. Congiungendo A, B e C, otteniamo i due triangoli rettangoli ACD e BCD, mostrati in figura.
Usando il Teorema di Pitagora, possiamo quindi scrivere la seguente espressione:

$AC^2 = AD^2 + CD^2$
$CB^2 = DB^2 + CD^2$

Sommando queste equazioni, si ottiene:

$AC^2 + CB^2 = AD^2 + DB^2 + 2CD^2$

Tuttavia, per una particolare proprietà delle circonferenze, anche il triangolo ABC è rettangolo. Quindi possiamo scrivere:

$AC^2 + CB^2 = AB^2$

Sostituendo AB^2 alla somma $AC^2 + CB^2$ dell'equazione precedente, si ha:

$AB^2 = AD^2 + DB^2 + 2CD^2$

dove CD corrisponde a "ysup" cioè la distanza tra il diametro e la circonferenza. AD è N × W ed AB è 2 × R, dove R rappresenta il raggio della circonferenza. Sostituendo i corrispondenti valori nell'equazione e riordinando i fattori otterremo:

$2ysup^2 = (2R)^2 - (NW)^2 - (2R - NW)^2$
$ysup = 2RNW - N^2W^2$

Suddividiamo per comodità il diametro in 64 parti uguali: allora W = 2 × R/64, che, ridotto, dà: W = R/32. Sostituendo W nell'equazione:

$ysup^2 = 2RNR/32 - N^2R^2/32^2$
$ysup^2 = R^2/32^2 \times (64N - N^2)$

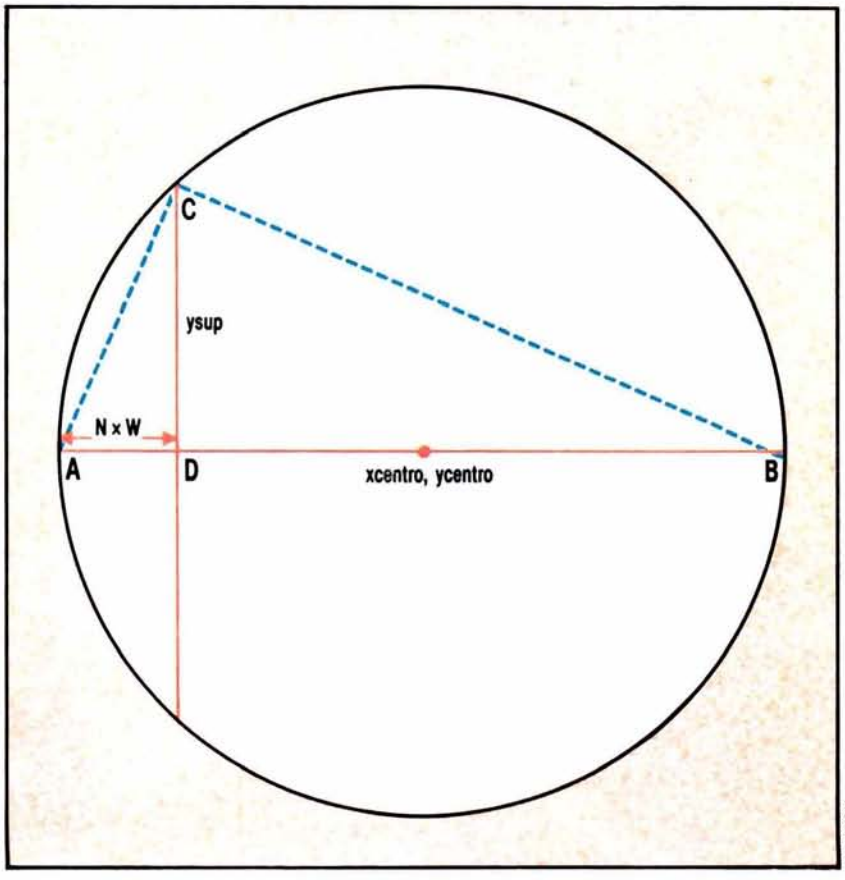
e calcolando la radice quadrata di entrambi gli elementi, otteniamo:

$ysup = R/32 \times SQR(64N - N^2)$

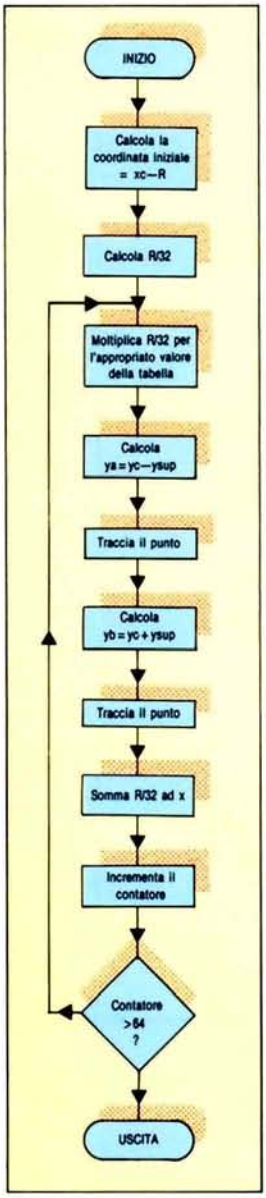
Partendo con x = xcentro − R ed incrementando di 64 "passi" uguali, ad ogni incremento la distanza della circonferenza dal diametro è data dall'ultima formula ottenuta, dove N indica il numero di passi. Sebbene questo risultato preveda il calcolo di una radice quadrata, possiamo notare che l'espressione sotto radice è indipendente dalle coordinate del centro e dal raggio della circonferenza. Possiamo quindi calcolare una tabella di valori per la funzione radice quadrata che fornirà una soluzione per ogni valore di N compreso tra 0 e 64. Questa tabella viene calcolata una sola volta e può essere incorporata nel programma come tabella "di consultazione".
La coordinata y assoluta per ciascun incremento di x, si ricava dall'equazione:

ya = ycentro − ysup

Infine, grazie alla simmetria del cerchio, possiamo calcolare le corrispondenti coordinate



LIZ DIXON



y per la metà inferiore del cerchio, usando la seguente formula:

yb = ycentro + ysup

Sviluppo della routine

Passiamo adesso a sviluppare in dettaglio la struttura della routine. Il diagramma di flusso mostra la procedura di calcolo per ciascun punto della circonferenza. Si noti che la routine diverrebbe più veloce se potesse eseguire una sola moltiplicazione per ogni punto tracciato. Comunque, anche nello stato attuale, la routine presenta due piccoli problemi. Innanzi tutto, non traccia la circonferenza con una linea continua, bensì con una serie di punti ravvicinati. Secondariamente, sebbene in BASIC essa produca cerchi con una buona definizione, quando viene tradotta in codice macchina è imprecisa.

Il primo problema si elimina usando la routine Lineasub (vedere a pag. 899), per collegare i punti con brevi linee ed ottenere una circonferenza continua. Il secondo è dovuto alle imprecisioni nel calcolo della radice quadrata, riportate nella tabella "di consultazione". Infatti, il calcolo di questi valori in BASIC e il loro inserimento (tramite POKE) in una serie di byte appositamente selezionati per la tabella, porta in realtà alla registrazione della sola parte intera di ciascun valore. Per ottenere cerchi tracciati con più cura, dovremo accrescere la precisione dei valori memorizzati in tabella. Il numero più grande da registrare è 32: possiamo quindi moltiplicare ogni numero per otto prima di immagazzinarlo, usando ancora un solo byte per ogni registrazione. Infatti 32 è l'unico numero che, moltiplicato per otto, non può essere contenuto in un solo byte. Per semplificare la routine, il programma approssima questo valore. Successivamente, la routine in codice macchina dividerà i valori in tabella per otto, eseguendo su di essi tre LSR (Scorrimenti Logici Verso Destra), conservando il resto della divisione per i calcoli successivi.

Il listato del codice sorgente riserva 65 byte all'inizio del programma per la tabella, assegnando un'etichetta al primo byte di essa. L'accesso ai valori registrati in tabella prevede un indirizzamento indicizzato. Il listato del codice sorgente viene inserito ed assemblato in memoria con la procedura abituale. Tuttavia, prima di salvare il codice assemblato, occorre inserire ed eseguire il seguente programma BASIC, che non danneggia il codice oggetto, in quanto si trova in una parte più alta della memoria. Questo programma calcola i 65 valori della tabella necessari al programma Circsub, li moltiplica per otto e deposita (tramite POKE) i risultati nell'area di memoria selezionata all'inizio del programma in codice macchina. A questo punto il codice macchina può essere salvato, ricordandosi di salvare assieme al codice oggetto anche la tabella che inizia all'indirizzo $C500. Fatto ciò, essa verrà caricata automaticamente tutte le volte che viene caricato il programma Circsub.

```
5 REM *** CREA LA TABELLA CIRCSUB ***
10 FOR N=0 TO 64
20 X=SQR(64*N-N^2)*8
25 X%=X: DF=X-X%
27 IF RE>=.5 THEN X%=X%+1
28 IF X%>255 THEN X%=255
29 POKE 50432+N,X%
30 NEXT N
```

Il seguente programma mostra come sia possibile usare Circsub partendo da un programma BASIC: l'utente deve soltanto indicare il raggio e le coordinate del centro. La subroutine alla linea 2000 separa le coordinate x del centro in LO-byte/HI-byte (basso/alto), inserisce (con POKE) in Circsub i valori indicati e, infine, esegue l'appropriata chiamata SYS. Notare che, poiché Circsub usa Lineasub che a sua volta richiama Plotsub (vedere a pag. 817), tutte e tre le subroutine devono essere caricate all'inizio. Il seguente programma traccia sullo schermo cerchi di raggio crescente.

```
   3 REM *** PROGRAMMA DI PROVA CIRCSUB ***
   5 DN=8: REM PER CASSETTA DN=1
  10 IF A=0 THEN A=1: LOAD "PLOTSUB.HEX",DN,1
  15 IF A=1 THEN A=2: LOAD "LINESUB.HEX",DN,1
  20 IF A=2 THEN A=3: LOAD "CIRCSUB.HEX",DN,1
 100 GOSUB 1000
 110 YC=100: R=2
 120 FOR XC=30 TO 210 STEP 7
 130 R=R+3
 140 GOSUB 2000
 150 NEXT
 160 GET A$: IF A$="" THEN 160
 170 GOSUB 3000
 180 END
1000 REM *** ABILITA L'ALTA RISOLUZIONE ***
1010 POKE 49408,1: POKE 49409,1
1020 POKE 49410,3
1030 SYS 49422
1040 RETURN
1050 :
2000 REM *** RICHIAMA CIRCSUB ***
2010 CHI=INT(CX/256): CLO=CX-256*CHI
2020 POKE 50497,CLO: POKE 50498,CHI
2030 POKE 50499,YC
2040 POKE 50500,R
2050 SYS 50521
2060 RETURN
2070 :
3000  REM *** DISABILITA  L'ALTA  RISOLUZIONE ***
3005 RESTORE
3007 PRINT CHR$(147): REM CANCELLA LO SCHERMO
3008 PRINT TAB(10) "VARIABILI CIRCSUB"
3009 PTINT
3010 POKE 49408,0: SYS 49422
3030 FOR I=50497 TO 50497+23 STEP 2
3035 READ A$
3040 PRINT TAB(2);A$,PEEK(I),
3045 READ A$
3047 PRINT A$, PEEK(I+1)
3050 NEXT
3060 :
3070 RETURN
5000 DATA XCTRLO,XCTRHI,YCTR, RAGGIO,INTR
5010 DATA REMR,TOTX,RESREM,RESLO,RESHI,OLDXLO
5020 DATA OLDXHI,NEWXLO,NEWXHI,OLDYA
5030 DATA OLDYB,NEWYA,NEWYB,XFLAG,YFLAG
5040 DATA OLDY,NEWY,INTTAB,REMTAB
```

Se non si possiede un assembler, il codice macchina, come per le altre subroutine di alta risoluzione del Commodore 64, può anche essere inserito mediante una serie di istruzioni DATA. Il seguente listato "Programma Loader in BASIC" permette di caricare in memoria Circsub. Si noti che i rispettivi caricatori (loader) BASIC per Lineasub e Circsub vanno letti ed eseguiti prima di caricare il programma dimostrativo. Poiché le tre routine si trovano già in memoria, si omettano le linee 10, 15 e 20 del precedente programma. Ricordiamo, infine, che il caricatore BASIC per Circsub contiene già la tabella, quindi non è necessario eseguire nuovamente il programma di creazione visto in precedenza.

Tra breve...

Fin qui, nella sezione "Linguaggio macchina", ci siamo dedicati agli home computer con microprocessore Z80 o 6502. C'è però un terzo microprocessore molto popolare, il chip 6809, adottato dai computer Dragon e Tandy Color. Più avanti verrà trattato anche il codice macchina di questo chip

PROGRAMMA "LOADER" IN BASIC

```
 10 REM *** CARICATORE BASIC PER CIRCSUB ***
 20 FORI=50432TO50432+498
 30 READA:POKEI,A
 40 CC=CC+A
 50 NEXT I
 60 READA:IFA<>CCTHENPRINT"CHECKSUM ERROR"
100 DATA0,63,89,108,123,137,149,159
110 DATA169,177,185,193,199,205,211
120 DATA216,221,226,230,233,237,240
130 DATA243,245,247,249,251,252,253
140 DATA254,255,255,255,255,255,254
150 DATA253,252,251,249,247,245,243
160 DATA240,237,233,230,226,221,216
170 DATA211,205,199,193,185,177,169
180 DATA159,149,137,123,108,89,63,0
190 DATA205,0,100,80,2,16,16,0,0,0,29
200 DATA1,31,1,100,100,100,100,0,0,120
210 DATA100,0,0,72,138,72,152,72,173
220 DATA68,197,41,31,141,70,197,173,68
230 DATA197,160,5,74,136,208,252,141
240 DATA69,197,173,65,197,56,237,68
250 DATA197,141,77,197,141,75,197,173
260 DATA66,197,233,0,141,78.197,141,76
270 DATA197,173,67,197,141,79,197,141
280 DATA80,197,169,0,170,141,71,197
290 DATA189,0,197,160,3,74,136,208,252
300 DATA141,87,197,189,0,197,41,7,141
310 DATA88,197,172,68,197,169,0,141,72
320 DATA197,141,73,197,141,74,197,173
330 DATA72,197,24,109,88,197,141,72
340 DATA197,201,8,144,23,56,233,8,141
350 DATA72,197,173,73,197,24,105,1,141
360 DATA73,197,173,74,197,105,0,141,74
370 DATA197,173,73,197,24,109,87,197
380 DATA141,73,197,173,74,197,105,0
390 DATA141,74,197,136,208,198,160,5
400 DATA78,74,197,110,73,197,110,72
410 DATA197,136,208,244,173,72,197,201
420 DATA16,144,9,173,73,197,24,105,1
430 DATA141,73,197,173,67,197,56,237
440 DATA73,197,141,81,197,173,81,197
450 DATA141,86,197,173,79,197,141,85
460 DATA197,32,165,198,173,67,197,24
470 DATA109,73,197,141,82,197,173,82
480 DATA197,141,86,197,173,80,197,141
490 DATA85,197,32,165,198,173,77,197
500 DATA141,75,197,173,78,197,141,76
510 DATA197,173,81,197,141,79,197,173
520 DATA82,197,141,80,197,173,71,197
530 DATA24,109,70,197,141,71,197,201
540 DATA32,144,26,173,71,197,56,233,32
550 DATA141,71,197,173,77,197,24,105,1
560 DATA141,77,197,173,76,197,105,0
570 DATA141,78,197,173,77,197,24,109
580 DATA69,197,141,77,197,173,78,197
590 DATA105,0,141,78,197,232,224,65
600 DATA240,3,76,153,197,104,168,104
610 DATA170,104,96,169,0,141,83,197
620 DATA141,84,197,173,75,197,205,77
630 DATA197,208,3,238,83,197,173,85
640 DATA197,205,86,197,208,3,238,84
650 DATA197,173,83,197,45,84,197,208
660 DATA39,173,75,197,141,0,195,173,76
670 DATA197,141,1,195,173,85,197,141,4
680 DATA195,173,77,197,141,2,195,173
690 DATA78,197,141,3,195,173,86,197
700 DATA141,5,195,32,14,195,96
710 DATA67223:REM*CHECKSUM*
```

PROGRAMMA CIRCSUB

```
;*********************
;*********************
;**                 **
;**   CIRCSUB 64    **
;**                 **
;*********************
;*********************
;  VARIABILI LINESUB
;
LINSUB = $C30E
X1LO   = $C300
X1HI   = $C301
X2LO   = $C302
X2HI   = $C303
Y1     = $C304
Y2     = $C305
*      = $C500
;
;  VARIABILI CIRCSUB
;
TABLE  ***65          ;TABELLA DI CONSULTAZIONE
XCTRLO ***1
XCTRHI ***1
YCTR   ***1
RADIUS ***1
INTR   ***1
REMR   ***1
TOTX   ***1
RESREM ***1
RESLO  ***1
RESHI  ***1
OLDXLO ***1
OLDXHI ***1
NEWXLO ***1
NEWXHI ***1
OLDYA  ***1
OLDYB  ***1
NEWYA  ***1
NEWYB  ***1
XFLAG  ***1
YFLAG  ***1
OLDY   ***1
NEWY   ***1
INTTAB ***1
REMTAB ***1
;
;  APPOGGIA REGISTRI SULLO STACK
;
       PHA
       TXA
       PHA
       TYA
       PHA
;
;  CALCOLA INT(R/32) E REMR
;
AD 44 C5   LDA RADIUS
29 1F      AND #$1F
8D 46 C5   STA REMR
AD 44 C5   LDA RADIUS
A0 05      LDY #$05
4A      LOOP1 LSR A
88         DEY
D0 FC      BNE LOOP1
8D 45 C5   STA INTR
;
;  CALCOLA COORDINATA X INIZIALE
;
AD 41 C5   LDA XCTRLO
38         SEC
ED 44 C5   SBC RADIUS
8D 4D C5   STA NEWXLO
8D 4B C5   STA OLDXLO
AD 42 C5   LDA XCTRHI
E9 00      SBC #$00
8D 4E C5   STA NEWXHI
8D 4C C5   STA OLDXHI
;
;  SELEZIONA VALORI INZIALI DI OLDY
;
AD 43 C5   LDA YCTR
8D 4F C5   STA OLDYA
8D 50 C5   STA OLDYB
;
;  AZZERA CONTATORE E TOTX
;
A9 00      LDA #$00
AA         TAX
8D 47 C5   STA TOTX
;
;  CALCOLA INTTAB E REMTAB
;
NEXTPT
BD 00 C5   LDA TABLE,X
A0 03      LDY #$03
LOOP2
4A         LSR A
88         DEY
D0 FC      BNE LOOP2
8D 57 C5   STA INTTAB
;
BD 00 C5   LDA TABLE,X
29 07      AND #$07
8D 58 C5   STA REMTAB
;
;  MOLTIPLICA TABELLA PER RAGGIO
;
AC 44 C5   LDY RADIUS
A9 00      LDA #$00
8D 48 C5   STA RESREM
8D 49 C5   STA RESLO      ;RISPOSTA ZERO
8D 4A C5   STA RESHI
;
AD 4B C5 AGAIN  LDA RESREM
18         CLC
6D 58 C5   ADC REMTAB
8D 4B C5   STA RESREM
C9 08      CMP #$08       ;>= 8?
90 17      BCC NOCARR
38         SEC
E9 08      SBC #$08       ;AZZERA REM
8D 4B C5   STA RESREM
AD 49 C5   LDA RESLO
18         CLC
69 01      ADC #$01       ;INCR. RISULTATO
8D 49 C5   STA RESLO
AD 4A C5   LDA RESHI
69 00      ADC #$00
8D 4A C5   STA RESHI
NOCARR
AD 49 C5   LDA RESLO
18         CLC
6D 57 C5   ADC INTTAB     ;SOMMA PARTE INT.
8D 49 C5   STA RESLO
AD 4A C5   LDA RESHI
69 00      ADC #$00
8D 4A C5   STA RESHI
88         DEY
D0 C6      BNE AGAIN
;
;  DIVIDI RISULTATO PER 32
;
A0 05      LDY #$05
LOOP3
4E 4A C5   LSR RESHI
6E 49 C5   ROR RESLO
6E 4B C5   ROR RESREM
88         DEY
D0 F4      BNE LOOP3
;
AD 4B C6   LDA RESREM
C9 10      CMP #$10       ;REM >= 16?
90 09      BCC NOCRRY
AD 49 C5   LDA RESLO
18         CLC
69 01      ADC #$01       ;SOMMA 1
8D 49 C5   STA RESLO
;
;  CALCOLA LA 1A COORDINATA Y
;
AD 43 C5   LDA YCTR
38         SEC
ED 49 C5   SBC RESLO
8D 51 C5   STA NEWYA
;
;  TRACCIA LINEA
;
AD 51 C5   LDA NEWYA
8D 56 C5   STA NEWY
AD 4F C5   LDA OLDYA
8D 55 C5   STA OLDY
20 A5 C6   JSR DRAW
;
;  CALCOLA 2A COORDINATA Y
;
AD 43 C5   LDA YCTR
18         CLC
6D 49 C5   ADC RESLO
8D 52 C5   STA NEWYB
;
;  TRACCIA LINEA
;
AD 52 C5   LDA NEWYB
8D 56 C5   STA NEWY
AD 50 C5   LDA OLDYB
8D 55 C5   STA OLDY
20 A5 C6   JSR DRAW
;
;  SCAMBIA VECCHIE COORDINATE CON NUOVE
;
AD 4D C5   LDA NEWXLO
8D 4B C5   STA OLDXLO
AD 4E C5   LDA NEWXHI
8D 4C C5   STA OLDXHI
AD 51 C5   LDA NEWYA
8D 4F C5   STA OLDYA
AD 52 C5   LDA NEWYB
8D 50 C5   STA OLDYB
;
;  MODIFICA COORDINATA X
;
AD 47 C5   LDA TOTX
18         CLC
6D 46 C5   ADC REMR
8D 47 C5   STA TOTX       ;SOMMA REMR AL TOT.
C9 20      CMP #$20       ;>= 32?
90 1A      BCC NOINC
AD 47 C5   LDA TOTX
38         SEC
E9 20      SBC #$20       ;AZZERA TOTX
8D 47 C5   STA TOTX
AD 4D C5   LDA NEWXLO
18         CLC
69 01      ADC #$01       ;INCR. COORDINATA X
8D 4D C5   STA NEWXLO
AD 4C C5   LDA OLDXHI
69 00      ADC #$00
8D 4E C5   STA NEWXHI
NOINC
AD 4D C5   LDA NEWXLO
18         CLC
6D 45 C5   ADC INTR
8D 4D C5   STA NEWXLO
AD 4E C5   LDA NEWXHI
69 00      ADC #$00
8D 4E C5   STA NEWXHI
;
;  INCREMENTA CONTATORE
;
E8         INX
E0 41      CPX #$41
F0 03      BEQ FINISH
4C 99 C5   JMP NEXTPT
;
;  RECUPERA REGISTRI DALLO STACK
;
FINE
68         PLA
A8         TAY
68         PLA
AA         TAX
68         PLA
60         RTS
;
;  VERIFICA SE PUNTO E' LO STESSO
;
DRAW
A9 00      LDA #$00
8D 53 C5   STA XFLAG      ;AZZERA FLAG
8D 54 C5   STA YFLAG
;
AD 4B C5   LDA OLDXLO
CD 4D C5   CMP NEWXLO
D0 03      BNE NOXFLG
NOXFLG
AD 55 C5   LDA OLDY
CD 56 C5   CMP NEWY
D0 03      BNE NOYFLG
EE 54 C5   INC YFLAG
NOYFLG
AD 53 C5   LDA XFLAG
2D 54 C5   AND YFLAG
D0 27      BNE NODRAW
;
;  TRACCIA LINEA
;
AD 4B C5   LDA OLDXLO
8D 00 C3   STA X1LO
AD 4C C5   LDA OLDXHI
8D 01 C3   STA X1HI
AD 55 C5   LDA OLDY
8D 04 C3   STA Y1
AD 4D C5   LDA NEWXLO
8D 02 C3   STA X2LO
AD 4E C5   LDA NEWXHI
8D 03 C3   STA X2HI
AD 56 C5   LDA NEWY
8D 05 C3   STA Y2
;
20 0E C3   JSR LINSUB
;
NODRAW
60         RTS
```

La Audiogenic

Inizialmente, questa ditta si occupava di duplicazione di cassette, poi...

La Audiogenic viene definita dal suo fondatore e amministratore delegato Martin Maynard come una "Casa di riproduzione e distribuzione". Maynard, che in precedenza aveva lavorato nell'industria musicale, fondò la Audiogenic a Reading (GB), agli inizi del 1970. In principio, la ditta lavorò come studio di registrazione e di riproduzione di nastri musicali. Nel 1978, la Southern Electrical Board le commissionò la riproduzione di una "cassetta dati" per computer. Le apparecchiature della Audiogenic furono quindi modificate, per rispondere alla crescente richiesta del mercato di nastri per computer e, poco tempo dopo, la compagnia firmò un contratto con la Commodore per la riproduzione del software per il microcomputer Pet.

Da quel momento, la Audiogenic ha assunto il controllo della commercializzazione e della distribuzione del software Commodore su cassetta e si è dedicata anche alla vendita di libri, riviste ed altri materiali legati ai prodotti Commodore.

Nel 1981, dopo il lancio del VIC-20, Maynard ottenne le licenze di vendita dei prodotti VIC sviluppati dalle Software House statunitensi. Il software prodotto sotto licenza americana rappresenta ancora oggi circa l'80% del listino Audiogenic e costituisce l'85-90% dell'intero fatturato della compagnia (valutato attorno ai 4 miliardi).

La scelta di puntare sulla riproduzione e vendita del software altrui, anziché alla produzione interna, ha dato ottimi risultati e Maynard afferma che la Audiogenic ha già al suo attivo oltre un milione di cassette. Egli ritiene che questo successo sia in gran parte dovuto alla vasta gamma di prodotti offerti dalla ditta ed alla notevole esperienza accumulata. La Audiogenic, infatti, si è sempre mossa con molta cautela sul mercato, offrendo soltanto prodotti software ampiamente sperimentati, in stridente contrasto con la spregiudicata politica commerciale adottata talvolta da molte software house.

Oggi la Audiogenic ha 25 dipendenti (il programmatore capo, David Smithson è l'autore del famoso pacchetto database *Magpie*) e impegna molto di più nella ricerca sul software di utilità per le proprie attività interne, che nella lucrosa vendita di prodotti software di gioco.

Maynard, infatti, è dell'opinione che benché i computer manterranno sempre una componente di intrattenimento, in futuro tuttavia il software avrà obiettivi ben più concreti e pratici.

Nonostante ciò, la Audiogenic non trascura il mercato dei giochi. Uno dei suoi prodotti di maggior successo è stato infatti *Motor Mania* e, ultimamente, ha lanciato *Alice in Videoland*, un ciclopico gioco di 90 Kbyte per il Commodore 64, dotato di 5 scenari di gioco, caricati in memoria da un disco via via che l'azione procede.

Dopo essersi trasferita in una sede più ampia, la Audiogenic ha installato nuove apparecchiature per la riproduzione di cassette. I programmi software vengono adesso riprodotti in serie su un nastro continuo, successivamente segmentato ed inserito nelle cassette. Questa tecnica è più rapida ed economica della precedente, che prevedeva la duplicazione su cassette vergini già confezionate.

In futuro, la Audiogenic intende rafforzare la propria politica commerciale basata sulla riproduzione e distribuzione di prodotti creati da altre software house britanniche e statunitensi.

La diffusione di sempre nuove periferiche hardware, inoltre, ha convinto la Audiogenic a commercializzare anche una piccola "lavagna grafica" fabbricata dalla Koala Technologies. In campo software, la Audiogenic presenterà fra breve una serie di cassette per gli apparecchi MSX ed il software per il nuovo home computer Commodore 16.



L'"anima" dell'Audiogenic
Martin Maynard ha fondato la compagnia nel 1970 come studio di registrazione musicale



Spazio di manovra
La Sede Centrale della Audiogenic si è trasferita nell'aprile dell'84 dal centro di Reading (GB) ad un nuovo e spazioso fabbricato a Sutton Park, appena fuori città

Il computer ci controlla?

I più recenti sviluppi nel settore del controllo computerizzato

L'impiego di grandi banche dati e di potenti elaboratori interessa oggi i più svariati campi della vita quotidiana: molti sono i vantaggi pratici di tali sistemi, ma c'è un rovescio di questa medaglia?

In molti casi, simili applicazioni hanno scopi di pubblico interesse e sono di vitale importanza per il disbrigo di pratiche nei vari enti governativi o di assistenza sociale. Inoltre, è possibile collegare tra loro sistemi diversi correlando le informazioni contenute nei rispettivi archivi. In Gran Bretagna, ad esempio, questa tecnica è stata impiegata con profitto per collegare l'ente preposto al rilascio delle patenti di guida e delle targhe automobilistiche a quello della "Sicurezza Sociale" (equivalente alla INPS italiana). Talvolta, però, questi sistemi di controllo, con funzione sociale, sono stati collegati anche ad apparati statali con funzioni sostanzialmente diverse, consentendo alle autorità di controllare molto più attentamente le attività di tutti i cittadini.

Per regolamentare questa nuova forma di potere, le forze sociali di molte nazioni del mondo hanno presentato varie proposte di legge. In Gran Bretagna, ad esempio, è stata varata una legge, detta "Data Protection Act", al fine di impedire un uso indiscriminato delle informazioni che metterebbe a repentaglio la libertà individuale. Molti concordano sul fatto che la legislazione in questo settore debba essere costantemente aggiornata, in risposta alla rapida evoluzione della tecnologia dei computer.

Molta della tecnologia usata nei sistemi di sicurezza e di sorveglianza computerizzata si basa sulla "ricognizione visiva", una tecnica che permette al computer di confrontare ciò che "vede", con una serie di immagini archiviate in memoria. Questi computer possiedono una capacità di memoria enorme ed una notevole potenza di elaborazione. Oggi, la possibilità di soddisfare questi requisiti ad un costo relativamente basso, consente una crescente diffusione di simili sistemi.

Un esempio molto significativo è quello del celebre ufficio di Polizia di New Scotland Yard, a Londra: il sistema computerizzato per l'archiviazione delle impronte digitali, il cui sviluppo ha richiesto 15 anni di studi, può oggi memorizzare 650.000 impronte e 100.000 "frammenti", ossia impronte parziali, trovate nei luoghi dove sono stati compiuti dei crimini. Il sistema confronta le impronte fornite dalla Polizia Scientifica con quelle dei propri archivi e ne rileva le somiglianze. L'elaboratore usato è un potente microcomputer Prime collegato ad un efficiente processore a matrice, con monitor di alta risoluzione e ad alcune telecamere. Nonostante tutto, il sistema riesce a esaminare e confrontare solo 200 o 300 impronte al giorno con le 650.000 memorizzate.

Sempre nel Regno Unito, un sistema di ricognizione visiva simile a questo è stato installato, a titolo sperimentale, su un viadotto autostradale e permette di osservare via telecamera e registrare visivamente i numeri di targa dei veicoli in transito su entrambe le corsie. L'elaboratore ad esso collegato confronta quindi le targhe con quelle memorizzate nello schedario delle auto "ricercate". Le informazioni ottenute vengono infine radiotrasmesse direttamente alle autopattuglie in servizio sull'autostrada, affinché procedano all'intercettazione delle auto che risultino sospette. In un primo momento, questa innovazione venne fatta passare sotto silenzio e divenne di dominio pubblico solo quando Steve Connor, un giornalista della rivista New Scientist, notò le telecamere ed indagò sul loro uso.

I progressi tecnologici nell'hardware dei microcomputer hanno favorito lo sviluppo di

Spionaggio industriale

Grazie alle loro ridottissime dimensioni, gli apparecchi spia possono essere celati dovunque, nei luoghi più ovvi come nei più inaccessibili. In quest'ufficio, una "microspia" potrebbe trovarsi nei seguenti posti:

1) Nel microfono della cornetta telefonica, nella base del telefono oppure nella lampada accanto all'apparecchio telefonico
2) Nella terra della pianta, dove potrebbe essere scambiata per un insetto, oppure sotto il vaso
3) Nel muro, confusa tra le viti dei pannelli o sotto un rivestimento di sughero
4) Dietro un quadro o al posto di uno dei chiodi che lo sostengono
5) Sotto il piano della scrivania oppure in uno dei cassetti

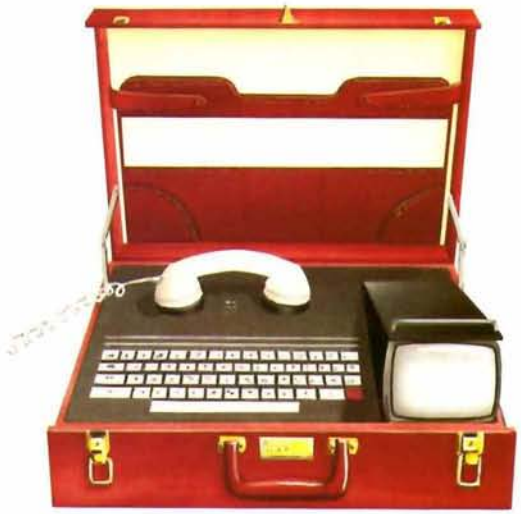
STEVE C



I "FERRI DEL MESTIERE"

Protezione computerizzata

Molti uomini d'affari o altre persone che hanno spesso a che fare con informazioni riservate, si difendono dallo spionaggio elettronico con apparecchiature simili a questo dispositivo computerizzato per la neutralizzazione delle intercettazioni telefoniche. Invece di parlare direttamente al telefono, l'utente batte sulla tastiera il messaggio che viene successivamente trasmesso in codice via telefono. Il messaggio viene ricevuto e decodificato sullo schermo di un apparecchio simile. Premendo un tasto, un sintetizzatore vocale incorporato può trasformare il messaggio in arrivo in discorso perfettamente comprensibile



Messaggio in codice

Questo strumento si basa sullo stesso principio del precedente e trasmette telefonicamente un messaggio in codice, che viene scritto a mano su un apposito ripiano sensibile. Il sistema permette di trasmettere anche la "firma" del mittente



Analizzatore di stress

Questo apparecchio è capace di rivelare il livello di stress di un individuo analizzando la sua voce: i risultati dell'analisi vengono immediatamente visualizzati su carta, in forma numerica. Questa sofisticata "macchina della verità" è anche in grado di scoprire se la persona sottoposta alla prova è tesa o semplicemente ansiosa



strumenti di sorveglianza sempre più piccoli, detti "microspie". La realizzazione di "chip" sempre più piccoli e più potenti ha reso possibile la produzione di sofisticate radio trasmittenti dalle dimensioni di un chicco di riso, sebbene dotate di sofisticati circuiti elettronici di controllo. Gli apparecchi di questo tipo "succhiano" corrente direttamente dagli apparecchi telefonici ai quali vengono applicati ed entrano in funzione solo quando qualcuno parla al telefono sotto controllo. Ma ci sono anche microspie a pila che, nascoste in qualsiasi punto di una stanza, possono captare le conversazioni che si svolgono nel raggio di alcuni metri, ritrasmettendole poi ad un apparecchio ricevente.

Un altro tipo di microspia, talmente sofisticato da far pensare ai più fantasiosi marchingegni alla James Bond, è capace di "sparare" un fascio laser su una finestra anche molto distante. Le vibrazioni dei vetri, prodotte dalla conversazione che si svolge all'interno della stanza, provocano delle interferenze nel raggio laser riflesso. Quest'ultimo viene captato da uno speciale apparecchio che, mediante un computer, è in grado di decodificare le interferenze e riprodurre l'intera conversazione.

In campo militare si hanno necessità diametralmente opposte: il principale obiettivo dei sistemi di sicurezza militari, infatti, non è tanto quello di scoprire le intenzioni degli avversari, bensì di evitare che le proprie attività vengano spiate. Anche in questo caso, la tecnologia dei chip e dei computer ha dato risultati eccezionali. Oggi, durante le operazioni militari, le trasmissioni radio avvengono su frequenze "saltellanti", la cui variabilità viene controllata da un computer in base ad un codice preprogrammato. Questa tecnica rende estremamente difficile l'intercettazione dei messaggi e, di conseguenza, le operazioni di disturbo.

I computer usati nei sistemi di sicurezza e di sorveglianza sono grossi elaboratori simili a quelli usati per la decodifica dei codici segreti da agenzie spionistiche come la CIA americana e la britannica National Security Agency, per non parlare delle Sezioni Speciali M15 ed M16.

Tra breve, grazie ai progressi della tecnologia hardware, la ricognizione visiva delle impronte digitali (e forse anche quella dei volti) diverrà una procedura completamente automatizzata e di basso costo. Probabilmente, in un prossimo futuro, tutte le auto della polizia verranno equipaggiate con microcomputer "di bordo", che permetteranno la ricezione istantanea di informazioni provenienti dai più diversi sistemi di sicurezza e sorveglianza (scuole, ospedali, uffici di polizia ecc.); per verificare l'identità di un individuo sospetto sarà sufficiente introdurre il suo tesserino di riconoscimento (tipo carta di credito) nell'apposito lettore del computer. L'elaboratore "leggerà" le impronte digitali o la fotografia e le confronterà con l'archivio centrale verificando l'identità del sospetto.

La visione peraltro interessante che si ricava da questa breve analisi pone inquietanti interrogativi. Il progresso tecnologico, infatti, ha ritmi travolgenti e difficilmente controllabili a breve scadenza: per non trovarsi di colpo in un mondo "alla Orwell", i provvedimenti vanno presi fin da ora.

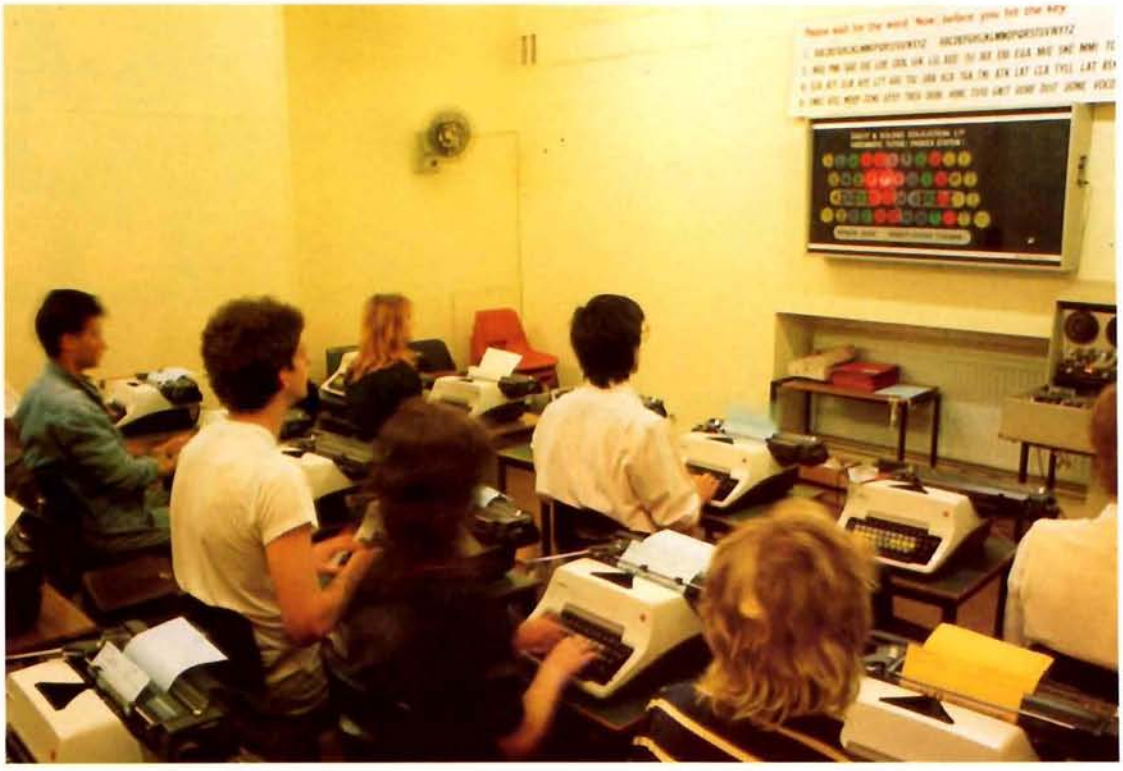
Un tocco di... tastiera

La dattilografia sta assumendo un'importanza sempre maggiore in un mondo popolato da computer

Chi, nel proprio lavoro, usa spesso macchine da scrivere o tastiere di computer, può risparmiare tempo e ridurre i propri errori con una buona tecnica dattilografica.
Abitualmente, i corsi di dattilografia vengono tenuti in aule appositamente attrezzate e prevedono diverse tecniche di insegnamento, ma oggi esistono in commercio vari programmi per computer per imparare la dattilografia direttamente sull'apparecchio. Per quanto riguarda la didattica, quattro sono i principi fondamentali da seguire per divenire dattilografi esperti:

- La padronanza della tastiera
- L'abitudine a tenere lo sguardo sullo schermo o sul foglio di carta
- La precisione
- La velocità di battitura

Queste doti si sviluppano gradualmente, con esercizi e con una pratica costante, fino a raggiungere un livello di discreta efficienza.
Il metodo di insegnamento più comune si basa sulla continua ripetizione di particolari sequenze di caratteri, per arrivare a muovere le dita sulla tastiera in modo automatico.
Il tipo di tastiera più diffusa sui computer è di derivazione anglosassone e viene chiamata "QWERTY", perché queste sono le prime 6 lettere a partire da sinistra sulla seconda linea di tasti. La disposizione dei simboli e delle lettere sulla tastiera risale a molti anni prima dell'avvento dei computer e, per finalità didattiche, le quattro file di tasti vengono suddivise in "sezione destra e sezione sinistra".
L'insegnamento della tecnica di battitura parte dalla padronanza degli 8 tasti della riga mediana (la terza dall'alto), chiamati "tasti di partenza", perché le dita partono e tornano sempre su di essi dopo una sequenza di battute.
I tasti di partenza per le quattro dita della mano sinistra sono: a, s, d, f; quelli per le quattro dita della mano destra sono; j, k, l, ; (punto e virgola). Dopo aver imparato ad adoperarli, il principiante passa all'uso degli altri tasti riferendosi sempre alla loro posizione rispetto ai tasti di partenza. I principianti rimangono sempre un po' sorpresi dall'abbondante uso che si fa del dito mignolo: nella battitura vengono usate tutte le dita ed il mignolo è quello che ha il maggior carico di lavoro. Appresa la posizione dei tasti, si passa alla "barra spaziatrice" da premere coi



TONY SLEEP

In posizione
La "Sight and Sound", una scuola di dattilografia con sedi in varie parti del mondo, offre corsi di dattilografia per principianti che danno ottimi risultati. In quest'immagine gli studenti si apprestano a battere una sequenza di lettere, guidati da una combinazione di stimoli visivi e sonori

pollici, al tasto SHIFT per le lettere maiuscole, ai tasti a due caratteri corrispondenti (minuscoli/maiuscoli o simboli).
Sfortunatamente per gli utenti di computer, la battitura dei tasti numerici viene piuttosto trascurata dalla maggior parte dei corsi e dei manuali di dattilografia, anche se non si differenzia molto da quella di tutti gli altri tasti. Le dita della mano sinistra, estendendosi sopra i tasti alfabetici, batteranno quelli numerici che vanno da 1 a 6; le dita della mano destra i tasti da 7 a 0 ed i seguenti della stessa riga.
Questa tecnica può essere usata anche sulle tastiere dotate di caratteri grafici: l'unica difficoltà da superare è ricordarsi la posizione dei vari simboli, per poterli battere con rapidità.
Mentre si impara a padroneggiare la tastiera, occorre anche abituarsi a tenere lo sguardo sullo schermo o sul foglio di carta. Questo è un aspetto fondamentale della dattilografia. Il metodo più usato contro l'abitudine di "sbirciare" continuamente la tastiera è quello di coprire i tasti con del nastro adesivo o con appositi cappucci di plastica. Con la tastiera ed il monitor di un computer si ottengono spesso risultati più rapidi: lo schermo si trova infatti all'altezza degli occhi e la tentazione di guardare verso il basso è minore. Inoltre, è possibile correggere immediatamente gli errori, abituandosi ad una maggiore precisione (il "terzo comandamento" del perfetto dattilografo).
La precisione dipende molto dalla pratica e dalla concentrazione. I tasti devono essere battuti centralmente con un colpo secco e deciso; con l'esperienza e con l'esercizio si impara a colpirli sempre con forza identica e con ritmo regolare.
Man mano ci si accorge di commettere sempre meno errori e che il ritmo, divenuto ormai istintivo, permette di accelerare la velocità di battitura.
La velocità, misurata con appositi test ed esercizi "a tempo", può essere aumentata solo quando si ha una perfetta padronanza delle tecniche appena descritte. Un dattilografo esperto riesce a scrivere più di 100 parole al minuto mentre, per un principiante, 30 parole al minuto sono già un discreto risultato.
Chi vuol imparare la dattilografia può contare su numerosi testi e manuali. Col passare del tempo, sono stati sviluppati nuovi metodi d'insegnamento, tutti basati più o meno sulle tecniche che abbiamo descritto. Pitman, una scuola di fama internazionale, pubblica ancor oggi un manuale la cui prima edizione risale a 35 anni fa, ma nella versione più recente, insegna la battitura di intere parole fin dalla prima lezione.
Poiché gli esercizi iniziali prevedono la battitura di parole brevi e di uso comune, la loro compitazione non richiede grossi sforzi e permette allo studente di concentrarsi sulla tecnica dattilografica.
La pubblicità del corso di dattilografia "Dico

I TASTI DI PARTENZA

L'illustrazione mostra la posizione dei tasti cosiddetti "di partenza" per la mano destra e sinistra ed i tasti controllati da ciascun dito. All'indice e al mignolo è assegnato il maggior numero di tasti, perché queste dita hanno maggior libertà di movimento. La tastiera tipo QWERTY, di derivazione anglosassone, va diffondendosi molto anche in Italia, poiché molti computer di provenienza estera adottano questa disposizione dei tasti. Ciò non toglie che molte case produttrici di computer prevedano, opzionalmente, la disposizione italiana classica dei tasti QZERTY



KEVIN JONES

Typing Course" afferma che sono bastate meno di 10 ore per insegnare a battere a macchina ad un bambino di undici anni. La tecnica impiegata trae spunto dagli studi dello psicologo americano B.F. Skinner e prevede l'uso di varie schede, ciascuna contenente una serie di domande: l'allievo inserisce la scheda nella macchina da scrivere e batte le relative risposte direttamente sulla tastiera.

Si tratta, quindi, di una nuova variazione del sistema tradizionale, nel quale il principiante deve copiare senza fare errori una serie di esercizi, corredati da disegni che mostrano la corretta posizione delle mani sulla tastiera.

Le più moderne tecniche didattiche prevedono l'impiego di "cassette" registrate e pannelli a luci intermittenti, che guidano le reazioni della vista, dell'udito e delle mani degli studenti.

Su di un pannello luminoso si accende una lettera e, contemporaneamente, una voce registrata sulla cassetta pronuncia la parola "Ora!" ritmando la velocità di battitura. Via via che il corso procede, cresce anche la velocità imposta agli studenti.

Queste tecniche vengono definite "di condizionamento" e gli stessi istruttori ammettono di non conoscerne a fondo il meccanismo di funzionamento.

Tuttavia, gli ottimi risultati ottenuti stanno a dimostrare che il sistema, qualunque ne sia il motivo, funziona!





SELEZIONE SOFTWARE

I programmi di dattilografia per computer possono presentarsi in forma di testo scritto o di gioco. Il software del primo genere imposta molte lezioni su esercizi di copiatura e di addestramento. I programmi di gioco invece, usano con abbondanza grafica, suono e immagini in rapido movimento. Per chi possieda computer con tastiere QWERTY, i termini anglosassoni non dovrebbe spaventare



Type Invaders

Editori: Carswell Computers
Formato: su disco o cassetta
Versioni: Micro BBC Modello B

Sostanzialmente, si tratta di un gioco, dedicato a chi ha già qualche conoscenza di dattilografia. Visualizza, nel più puro stile "space invaders", delle lettere alfabetiche che calano dall'alto e vengono "distrutte", battendo i tasti corrispondenti. Le lettere eliminate tornano a calare dall'alto e possono abbattere la linea difensiva del giocatore, invadendo il suo territorio. Il gioco ha dieci livelli di difficoltà, che vanno dalla comparsa di semplici lettere maiuscole alla caduta di parole di cinque caratteri composte da lettere maiuscole, minuscole e numeri. Ci sono 3 velocità di caduta: lenta, veloce e accelerata, ma anche la più lenta richiede un discreta tecnica dattilografica. I principianti possono acquistare il "Typeasy", un pacchetto prodotto dalla stessa casa

Sprintyper

Editori: Micro Software International
Formato: su cassetta
Versione: Commodore 64

È un programma di dattilografia in formato testuale, sia per principianti che per esperti: punta soprattutto a sviluppare la velocità e la precisione della battitura. Possiede una "biblioteca" di ben 356.635 frasi (ahimè, in inglese!), per migliorare la tecnica manuale. All'inizio compare sullo schermo una frase che deve essere copiata nel minor tempo possibile. Un segnale sonoro evidenzia gli errori e il programma si blocca finché non vengono corretti. A questo punto compare sul video un messaggio che comunica il tempo impiegato, il tempo "record" e il numero degli errori commessi. Lo Sprintyper è essenzialmente un test di velocità e precisione e non è di grande aiuto ai principianti

Typing Tutor II

Editori: Microsoft
Formato: su disco
Versioni: Apple IIe, Apple IIe+

Per utilizzare questo pacchetto si deve possedere l'Applesoft residente in ROM, una memoria di 48 K, un'unità a dischetti ed il sistema operativo DOS 3.3. Il programma, in formato testuale, è pilotato tramite menu ed è composto da una serie di lezioni teoriche, prove pratiche e test di velocità. La sua caratteristica più significativa è il sistema "Time Response Monitoring", che controlla i tempi di battitura 100 volte al secondo ed individua anche il più lieve rallentamento, che si verifica quando lo sguardo viene spostato dallo schermo alla tastiera. I principianti cominciano ad esercitarsi con un limitato numero di lettere. Quando hanno fatto una certa pratica e la loro velocità di battitura ha raggiunto le 30 parole al minuto, il primo gruppo di lettere viene trasferito sulla colonna FAST (veloce) ed il sistema seleziona una nuova serie di lettere con le quali si ripetono gli esercizi iniziali. Per i dattilografi esperti, il Typing Tutor II stila un giudizio che contiene il numero degli errori commessi, i tasti maggiormente coinvolti negli errori, la velocità e la precisione di battitura. Questo pacchetto è adatto tanto ai principianti quanto ai dattilografi di ogni livello

Spruzzi di colore

Qualche aggiunta grafica d'effetto al gioco del campo minato per i computer Acorn

La prima routine da aggiungere simula la presenza di un cecchino, che spara attraverso il campo minato nel tentativo di colpire il rivelatore di mine o l'assistente. La traiettoria dei proiettili è rappresentata da una linea in alta risoluzione che va da sinistra a destra. Per variare in modo casuale la traiettoria dei proiettili, le coordinate iniziali e finali della linea vengono definite ricorrendo alla funzione RND. I valori delle variabili xstart e xfinish vengono selezionati all'interno della procedura inizializzavariabili. La differenza tra questi due valori è di 1024 unità grafiche, pari alla larghezza del campo minato.
Per verificare se il proiettile colpisce l'assistente o il rivelatore, il programma traccia un breve segmento della linea di sparo e verifica (tramite l'istruzione POINT) se l'area di schermo raggiunta presenta il colore logico 1. In caso negativo, traccia il segmento successivo e ripete la verifica, e così via fino a raggiungere il lato opposto dello schermo o uno dei due caratteri bersaglio.
Adesso occorre fissare il "passo" (ossia la lunghezza) del segmento. Se la sua misura è molto piccola, aumenta il tempo necessario per tracciare un'intera linea. D'altra parte, se il segmento è troppo lungo, c'è il rischio di incrociare un bersaglio senza individuarlo. Poiché ogni cella di carattere ha una larghezza di 64 unità grafiche, il giusto "passo" per il segmento potrebbe essere pari a mezza cella (ossia 32 unità grafiche). Optando per questa lunghezza (dx nella direzione x), l'intera linea di sparo viene tracciata in 32 passi (1024/32=32). Calcolando in modo casuale le coordinate y iniziali e finali della linea di sparo, l'esatta lunghezza del segmento dy in direzione y si ottiene dividendo per 32 la differenza tra i valori delle due coordinate. Vediamo, infine, come procedere per cancellare dallo schermo la linea che abbiamo appena tracciata. La soluzione a questo problema ricorre al concetto di "colore logico", tipico del BASIC BBC, ed alla possibilità di eseguire su di esso alcune operazioni logiche. Quando viene abilitato, il MODE 5 prevede i seguenti 4 colori logici:

Colore logico	0	1	2	3
Equivalente binario	00	01	10	11
Colore effettivo	nero	rosso	giallo	bianco

Il comando GCOL consente di eseguire alcune operazioni logiche tra il colore usato per tracciare la linea di sparo ed i colori presenti sullo schermo. Il comando richiede due parametri: il primo seleziona il metodo di tracciamento, mentre il secondo indica il colore logico da usare:

GCOL0	Traccia nel colore specificato
GCOL1	Esegue una OR logica
GCOL2	Esegue una AND logica
GCOL3	Esegue una OR Esclusiva logica
GCOL4	Esegue una NOT logica sul colore preesistente

Cerchiamo di chiarire con un breve esempio il funzionamento di questo comando. Se vogliamo tracciare una linea rossa (colore logico 1) su un'area dello schermo colorata in bianco (colore logico 3), le varie opzioni del comando GCOL daranno i seguenti risultati:

GCOL0,1	Cancella il bianco e traccia in colore rosso		
GCOL1,1	Pone in OR rosso e bianco, per ottenere il colore bianco	rosso bianco bianco	01 OR 11 11
GCOL2,1	Pone in AND rosso e bianco per ottenere il colore rosso	rosso bianco rosso	01 AND 11 01
GCOL3,1	Pone in OR Esclusivo rosso e bianco per ottenere il colore giallo	rosso bianco giallo	01 EOR 11 10
GCOL4,1	Pone il bianco in NOT per ottenere il colore nero	bianco nero	11 00

Possiamo quindi tracciare una linea in colore bianco e, successivamente, rintracciarla in nero per cancellarla. Però, se la linea interseca un oggetto, ad esempio una mina, questo metodo vi lascerà sopra una striscia nera. Col comando GCOL possiamo però porre il colore rosso in OR Esclusivo rispetto a tutti i colori che vengono a trovarsi sotto la linea. Quando essa incrocia un'area bianca, la linea lascia su di essa una banda di colore giallo (vedere tabella precedente). Ritracciando sulla stessa area, con il colore rosso posto in OR Esclusivo, abbiamo come risultato finale:

rosso	01
giallo	10
EOR	—
bianco	11

cioè, l'area bianca torna al colore originale. Pertanto, eseguendo due operazioni logiche OR Esclusive, si riottiene sempre il colore originale. Questa tecnica può essere adoperata per cancellare la nostra linea di sparo. Infatti, se tracciamo la linea usando il metodo logico OR Esclusivo e poi la ritracciamo, ap-

plicandolo nuovamente, la linea si cancella, lasciando inalterati i colori dello sfondo.
Listato completo della procedura "cecchino":

```
3110 DEF PROCcecchino
3120 ystart=RND(750)+220
3130 yfinish=RND(750)+220
3140 dx=32: dy=(yfinish-ystart)/32
3150 GCOL3,3
3160 PROClinea
3170 IF POINT(x,y)=1 THEN PROCesplosione(x,y) ELSE PROClinea
3180 ENDPROC
```

Questo è il listato della procedura "linea":

```
3450 DEF PROClinea
3460 SOUND 0,-8,4,5
3470 x=xstart: y=ystart
3480 MOVE x,y
3490 REPEAT
3500 DRAW x,y
3510 x=x+dx: y=y+dy
3520 UNTIL x>xfinish OR POINT(x,y)=1
3530 ENDPROC
```

Tre nuove caratteristiche

Come già accennato in precedenza, il Micro BBC può generare suoni compositi. La seguente procedura aggiunge al programma un breve accordo, usando un solo canale dei tre presenti sull'apparecchio. Semplicemente definendo la frequenza e la durata delle note, si può suonare un motivetto:

```
4090 DEF PROCmusica
4100 REM **PRIMA BATTUTA**
4110 SOUND 1,-8,213,5
4120 SOUND 1,-8,209,5
4130 SOUND 1,-8,213,5
4140 SOUND 1,-8,209,5
4150 SOUND 1,-8,213,5
4160 SOUND 1,-8,193,5
4170 SOUND 1,-8,205,5
4180 SOUND 1,-8,197,5
4190 REM **SECONDA BATTUTA**
4200 SOUND 1,-8,185,20
4210 SOUND 1,-8,165,5
4220 SOUND 1,-8,185,5
4230 SOUND 1,-8,193,20
4240 REM **TERZA BATTUTA**
4250 SOUND 1,-8,165,5
4260 SOUND 1,-8,193,5
4270 SOUND 1,-8,197,20
4280 ENDPROC
```

Pagina dei titoli: la tecnica di tracciamento OR Esclusivo può essere impiegata anche per creare la prima "videata" o "pagina di presentazione" del gioco. La seguente procedura visualizza infatti il titolo del gioco MINE usando la grafica in alta risoluzione. La parola viene realizzata con una serie di linee, ciascuna delle quali viene tracciata facendo riferimento alla linea precedente, cosicché è possibile visualizzare la parola in qualsiasi zona dello schermo, semplicemente indicando il punto di partenza della prima linea. Se tracciamo più volte la parola, con il metodo OR Esclusivo, e via via spostiamo verso l'alto il punto di inizio della prima linea, otteniamo un'immagine nella quale la parola MINE si muove verso il margine superiore dello schermo. L'istruzione GCOL0,129 seleziona il colore rosso per lo sfondo ed il successivo comando CLG colora di rosso l'intero schermo. Contemporaneamente, possiamo far suonare il motivetto musicale, richiamando la procedura "musica". Poiché quest'ultima viene elaborata più velocemente della procedura grafica, è necessario immagazzinare l'informazione SOUND in un "buffer" per suonarla alla comparsa dell'immagine. In questo modo, durante l'emissione del suono, il processore sarà libero di svolgere altre operazioni.
Livello di difficoltà: per rendere il gioco maggiormente impegnativo, possiamo introdurre una scelta tra più livelli di difficoltà. Dopo la comparsa del titolo, il programma chiederà al giocatore di inserire un numero compreso tra 0 e 9. Il numero, immagazzinato nella variabile "livello", accresce la rapidità di fuoco del cecchino ed il numero di mine del campo minato. Per questa aggiunta è necessaria la seguente modifica alle linee 1930 e 1940 della procedura "preparazione" (vedere a pag. 885):

```
1930 fattore=livello*3+30
1940 PROCmettimine (fattore)
```

Una seconda modifica va introdotta nella linea 3950 della procedura "ripristino", che calcola le mine rimaste e le reinserisce nel campo minato:

```
3950 minerimaste=fattore-punti/150
```

Segue listato completo della procedura "titolo":

```
1300 DEF PROCtitolo
1310 GCOL 0,129
1320 CLG
1330 GCOL 3,3
1340 PROCmusica
1350 Y=100: X=0
1360 REPEAT
1370 X=X+20: Y=Y+50
1380 FOR I=1 TO 2
1390 PROCmine
1400 NEXT I
1410 UNTIL Y>700
1420 :
1430 PROCmine
1440 PRINTTAB (0,20)"Livello di difficolta' (0-9)?"
1450 PROCmusica
1460 REPEAT
1470 livello=GET-48
1480 UNTIL livello>-1 AND livello<10
1490 ENDPROC
1500 :
1510 DEF PROCmine
1520 PLOT 4,X,Y
1530 REM **LETTERA M**
1540 PLOT 1,0,200
1550 PLOT 1,80,-100
1560 PLOT 1,80,100
1570 PLOT 1,0,-200
1580 REM **LETTERA I**
1590 PLOT 0,40,0
1600 PLOT 1,80,0
1610 PLOT 0,-40,0
1620 PLOT 1,0,200
1630 PLOT 0,-40,0
1640 PLOT 1,80,0
1650 REM **LETTERA N**
1660 PLOT 0,40,-200
1670 PLOT 1,0,200
1680 PLOT 1,120,-200
1690 PLOT 1,0,200
1700 REM **LETTERA E**
1710 PLOT 0,160,0
1720 PLOT 1,-120,0
1730 PLOT 1,0,-200
1740 PLOT 1,120,0
1750 PLOT 0,-40,100
1760 PLOT 1,-80,0
1770 ENDPROC
```

Finora ci siamo serviti di un programma di chiamata "temporaneo", per collegare tra loro le varie procedure (vedere a pag. 874). Ma ora abbiamo sviluppato tutte le procedure che compongono il ciclo principale del programma: possiamo quindi cancellare il programma temporaneo e sostituirlo con la versione definitiva:

```
2020 DEF PROCciclo
2030 REPEAT
2040 PROCaggiornatempo
2050 PROCcontrollotastiera
2060 rand=RND(50-livello)
2070 IF rand=1 THEN PROCcecchino
2080 UNTIL TIME>12099 OR endflag=1
2090 ENDPROC
```

Infine, dobbiamo inserire nel listato del programma le seguenti linee:

```
1060 maxpunti$="00000"
1110 MODE5
1120 REM **CANCELLA IL CURSORE**
1130 VDU23;8202;0;0;0;
1140 PROCtitolo
1150 CLS
1160 PROCpreparazione
1170 :
1180 PROCciclo
```

L'ultima parte del progetto riguarda la rifinitura dello scenario per il gioco, assieme al listato completo dell'intero programma.

Software in vetrina

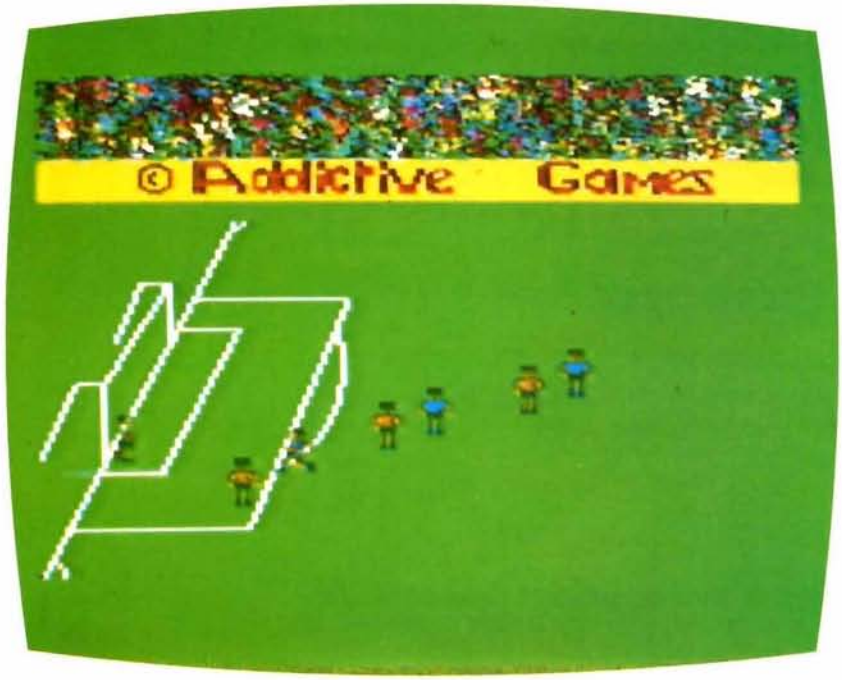
Una selezione tra i numerosi programmi disponibili per i computer più diffusi

Assieme ad un gioco del genere "avventura/labirinto", viene stavolta presentato un programma che non mancherà di attirare una folta schiera di entusiasti: il gioco del calcio.



Cavelon per lo Spectrum

Nella più classica tradizione dei giochi di questo tipo, il giocatore si avventura in un castello misterioso, dove, ad ogni passo, si corrono rischi imprevedibili. Sul percorso sono disseminati oggetti più o meno utili per passare all'ambiente successivo. L'intero svolgimento dell'azione è accompagnato da motivetti musicali. Per gli appassionati di questo genere, il gioco fornisce qualche ora di spensierato divertimento.



Football Manager per il Commodore 64

Ecco un programma che certamente farà drizzare le orecchie a più di un possessore di Commodore: una partita di calcio sul proprio computer!
Tuttavia, non si tratta di un semplice gioco d'azione, che altrimenti poco si discosterebbe dagli ormai antiquati videogiochi del tipo "ping pong": qui vengono messe alla prova le capacità organizzative del giocatore, che impersona il "manager" di una squadra di calcio. La partita in sé non è che il risultato finale di tutta una serie di decisioni, dalle quali dipende il successo (e quindi la conquista dello "scudetto") della propria squadra.

Il Macintosh

Considerato come una versione ridotta del Lisa, ne segue la stessa filosofia operativa

Il Macintosh è notevolmente diverso da tutti i microcomputer precedentemente descritti e non c'è niente di simile in commercio. Sebbene, fondamentalmente, il Macintosh sia un computer gestionale, la Apple ha sviluppato una propria linea di progettazione, rifiutandosi di seguire gli standard adottati dalla IBM e dalla maggior parte dei fabbricanti di computer. Pur correndo grossi rischi, la Apple ha proseguito per la propria strada e viene oggi considerata da tutti un'innovatrice in un settore popolato essenzialmente da "imitatori".

L'aspetto esterno del Macintosh è abbastanza insolito: l'unità centrale ha una linea snella e luminosa e sembra piuttosto piccola, per un apparecchio così potente. Lo schermo ad alta risoluzione è da 9 pollici e il "drive" usa dischetti da 3 pollici e mezzo. Sul retro dell'unità centrale c'è una maniglia sagomata, che facilita il trasporto del Macintosh da un luogo all'altro. Il peso totale, compresa la tastiera, il "mouse" e la valigetta opzionale per il trasporto, è di 11,6 kg. La valigetta possiede una serie di alloggiamenti per tutti i "pezzi" dell'apparecchio.

Il Macintosh ha una tastiera tipo macchina da scrivere, ben dimensionata e con una buona sensibilità. È dotata di un proprio processore che controlla le funzioni speciali ed il set di caratteri internazionali.

L'altro elemento del sistema è il mouse, così chiamato per la "coda" che lo collega all'unità centrale.

Questo strumento, delle dimensioni di un pacchetto di sigarette, viene spostato manualmente sulla superficie del tavolo di lavoro, guidando così i movimenti del cursore sullo schermo. Ciò permette di selezionare le differenti attività, previste dal sistema operativo dell'apparecchio.

Grazie al mouse, l'utente può accedere a tutte le aree di lavoro del Macintosh, senza dover per questo imparare a memoria una lunga serie di comandi operativi, come accade invece nella maggior parte degli altri computer. Se, ad esempio, si desidera "aprire" un file per archiviare un documento, tramite il mouse si sposta il cursore sul piccolo simbolo (chiamato "icona"), che rappresenta un foglio di carta e, premendo il tasto del mouse, si abilita sul video la funzione richiesta.

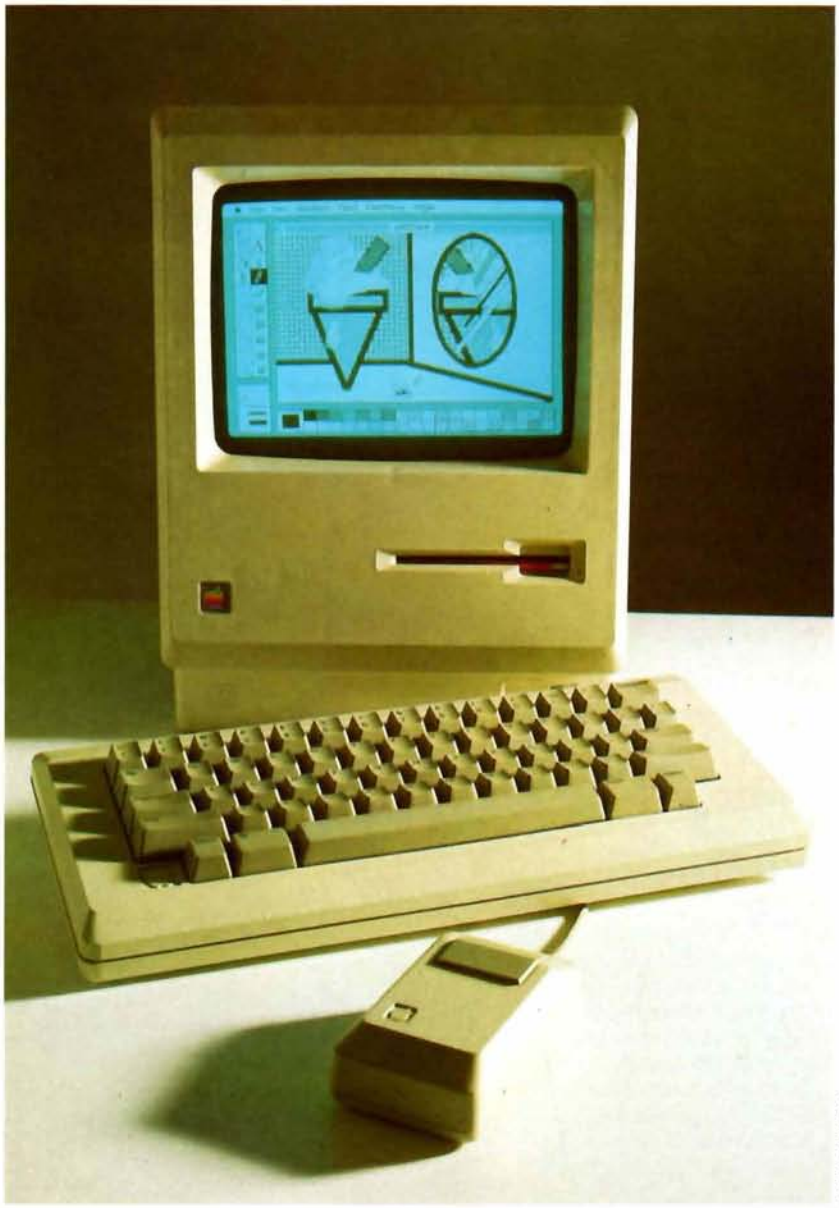
A fine lavoro, si può nuovamente usare il mouse, per tornare al menù principale composto da tutta una serie di icone e per salvare il file portando il cursore sul simbolo che rappresenta un dischetto.

Il Macintosh viene venduto con una memoria utente di 128 Kbyte, espandibile fino a 512 Kbyte, sostituendo la RAM di serie con 2 gruppi di chip da 256 Kbyte. Nella ROM da 64 Kbyte è contenuto il software operativo, che controlla tutte le operazioni di sistema ed alcune funzioni speciali.

L'unità a dischetti (della Sony) usa dischi da 3 pollici e 1/2 a singola faccia, capaci di contenere fino a 400 Kbyte. Per giunta, questi sono più affidabili dei dischetti da 5 pollici e 1/4.

Lo schermo dell'apparecchio ha una risoluzione di 512 × 342 pixel, per un totale di 175.000 punti, tutti indirizzabili individualmente. Queste caratteristiche rendono possibili alcune applicazioni grafiche veramente sbalorditive, particolarmente apprezzate da progettisti, architetti, fotografi ed altri professionisti legati al mondo della grafica e della pubblicità.

Il Macintosh è stato progettato per lavorare



IAN McKINNELL

Il sistema Macintosh
Il Macintosh occupa uno spazio estremamente ridotto sul piano di lavoro sul quale poggia. L'altissima risoluzione dello schermo consente di ottenere risultati fino a ieri possibili soltanto su apparecchi 10 volte più costosi



Un computer "firmato"

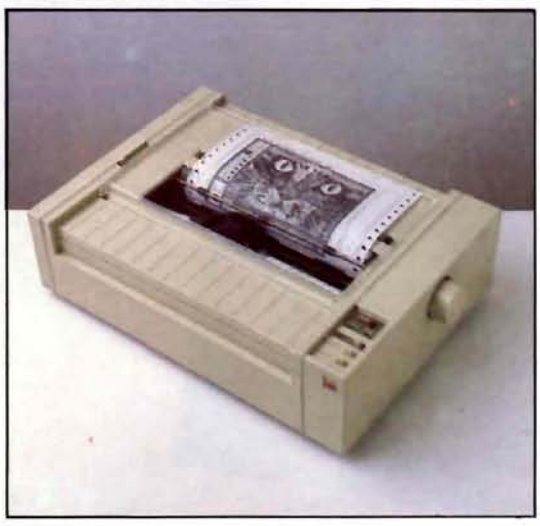
Su un pannello interno del Macintosh, la Apple riproduce le firme autografe del gruppo di lavoro che ha progettato l'apparecchio. Ci sono anche quelle di Steve Jobs e Steve Wozniak, fondatori della Apple e "padri" dei progetti Lisa e Macintosh

L'UNITÀ A DISCHETTI SONY

La seconda unità a dischetti "esterna", prodotta per la Apple dalla Sony, è un accessorio di notevole utilità, che evita perdite di tempo dovute alla sostituzione dei dischetti. Questa periferica non richiede l'acquisto di un'interfaccia extra



IAN McKINNELL



DALLO SCHERMO... ALLA CARTA

La ImageWriter è una stampante seriale con una velocità massima di stampa di 120 caratteri per secondo. Questa rapidità è ancor più significativa nel modo grafico, che consente di riprodurre direttamente le immagini video

Scheda analogica

Questa scheda controlla il monitor e l'alimentazione dell'apparecchio. Il Macintosh è privo di ventilatore perché l'eccesso di calore viene dissipato attraverso alcune piastrine metalliche, poste in prossimità delle griglie di ventilazione aperte nella custodia

Altoparlante interno

Testina di lettura/scrittura dell'unità a dischetti

Comando per il contrasto dello schermo

Unità a dischetti Sony da 3 pollici e 1/4

Costruita appositamente per la Apple, usa dischetti a singola faccia da 400 Kbyte. Nella versione per dischetti a doppia faccia, questi immagazzinano fino ad 800 Kbyte

RAM del video

Alcuni dei 22 Kbyte necessari al video sono contenuti in questi chip di memoria ad accesso diretto (Direct Access Memory, o DAM)

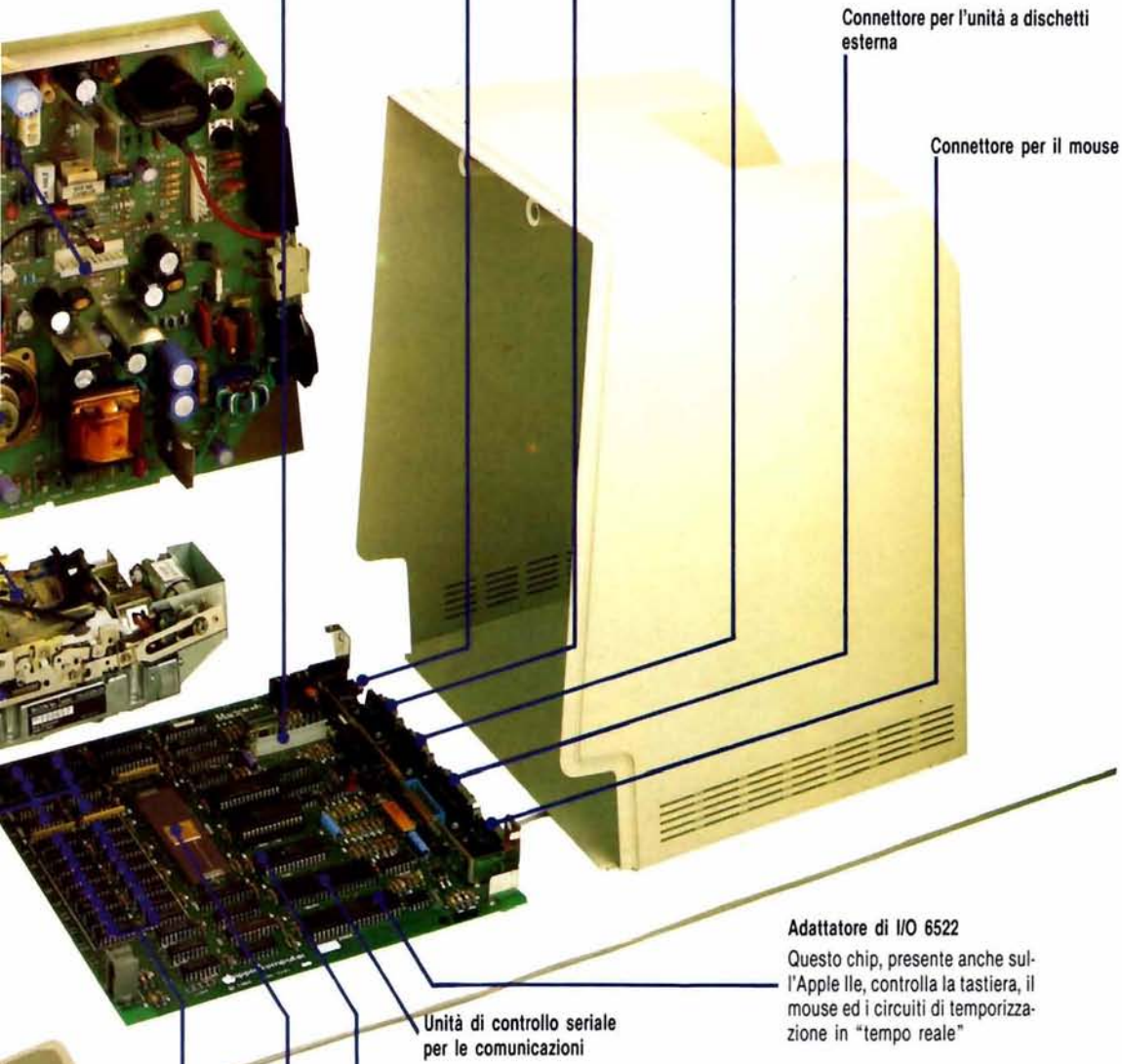
Connettore per la tastiera

Tastiera

La tastiera del Macintosh, separata dall'unità centrale, è dotata di un proprio microprocessore, che controlla il set di caratteri ed alcune funzioni speciali. Non sono previsti tasti per il cursore, perché i movimenti di quest'ultimo sono pilotati col mouse

Mouse

Controlla i movimenti del cursore e viene usato per selezionare le "icone" del menu



Connessione A/D
Questo cavetto a nastro collega la scheda analogica a quella digitale

Uscita audio

"Bus" seriale
Viene chiamato anche "slot virtuale" e permette di collegare al Macintosh diverse periferiche

Porta per stampante

Connettore per l'unità a dischetti esterna

Connettore per il mouse

Adattatore di I/O 6522
Questo chip, presente anche sull'Apple IIe, controlla la tastiera, il mouse ed i circuiti di temporizzazione in "tempo reale"

Unità di controllo seriale per le comunicazioni

Chip di controllo per le unità a dischetti
Questo chip controlla tanto il drive interno Sony quanto quello accessorio

CPU
Il chip Motorola 68000 esegue le operazioni interne usando 32 bit alla volta, ma trasmette e riceve informazioni nel formato a 16 bit

La memoria RAM
La RAM "utente", da 128 Kbyte, è contenuta in 16 chip, sostituibili da due serie di chip RAM da 256 Kbyte per un totale di 512 Kbyte. La RAM da 128 Kbyte è tuttavia sufficiente per il software standard dell'apparecchio

con una stampante ad alta velocità: la ImageWriter, prodotta dalla stesssa Apple. La stampante è in grado di riprodurre fedelmente tutto ciò che compare sullo schermo del computer.

L'hardware del Macintosh è di alta qualità ed estremamente affidabile, ma ciò che colpisce di più nell'apparecchio è l'originalità del software. Grazie alla perfetta fusione tra hardware e software ed alla completa gamma di comandi residenti in ROM, il trasferimento sul Macintosh di programmi scritti per altri apparecchi crea poche difficoltà. Il funzionamento del "Mac" è così semplice che anche l'utente più sprovveduto può imparare in pochi minuti ad eseguire operazioni che, sugli altri computer, richiedono ore di studio e numerosi tentativi.

Dal punto di vista della qualità e dello stile, il Macintosh è indiscutibilmente un computer che già "vive" nel futuro.

MACINTOSH

DIMENSIONI

mm. 343 x 254 x 254 (involucro dello schermo/unità a dischetti)

CPU

68000 della Motorola, funzionante a 7,83 MHz

MEMORIA

128 Kbyte di RAM, 64 Kbyte di ROM

VIDEO

Video monocromo incorporato, con risoluzione di 512 x 342 pixel. Possibilità di aprire "finestre", largo impiego di "icone"

INTERFACCE

Per il "Mouse", per la stampante, per unità a dischetti esterna, per impianto hi-fi, "bus" seriale

LINGUAGGI DISPONIBILI

BASIC, COBOL e PASCAL

TASTIERA

Tipo macchina da scrivere, con 59 tasti (tastierina numerica opzionale)

DOCUMENTAZIONE

Il manuale operativo è corredato di un dischetto dimostrativo molto utile a comprendere il funzionamento del sistema. Ci sono poi i manuali per il MacWrite (una specie di word processor) e per il MacPaint (un programma grafico). Tutti i manuali sono ben curati e l'esposizione è chiara

PRO

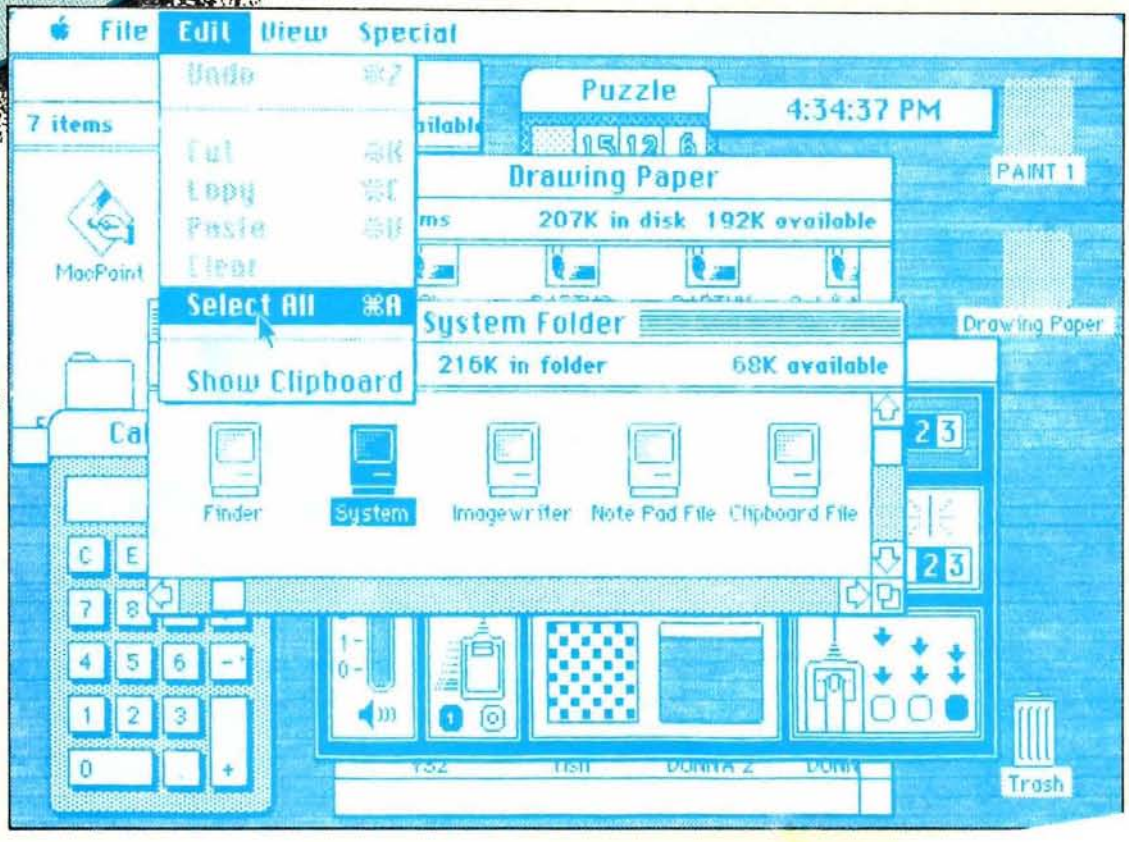
Il "Mac" è un computer molto facile da usare, ha un'ampia memoria ed è dotato di buon software. Il "Mouse" facilita notevolmente la maggior parte delle operazioni. Internamente, la razionale disposizione delle schede rende le operazioni di manutenzione molto semplici

CONTRO

Il prezzo del "Mac" è nettamente superiore a quello dei computer appartenenti più o meno alla stessa fascia di mercato. La presenza di un solo dischetto ostacola l'uso di archivi molto ampi. Attualmente, c'è carenza di software alternativo a quello offerto in dotazione

Tastiera
Il Macintosh possiede una tastiera QWERTY, di tipo americano, ma offre un set di caratteri di livello internazionale. Sono infatti presenti molti dei caratteri usati nelle lingue dei paesi europei (tedesco, francese, spagnolo, greco, ecc.)

Una "video... scrivania"
Il software del Macintosh è simile a quello del Lisa e si serve di "finestre" per visualizzare documenti le cui dimensioni non rientrano per intero sullo schermo. Le finestre possono sovrapporsi l'una all'altra, come mostrato in figura, ed il lavoro si svolge simulando quello su una comune scrivania



Il River Rescue

Inizialmente concepito per la consolle Atari, ne esistono ora diverse versioni per altri computer

Il *River Rescue* è un tipico gioco stile "arcade" realizzato dalla Creative Sparck, la divisione software della EMI Thorn. Lo stile di questa grande casa è molto evidente durante lo sviluppo dell'azione. Il programma è prodotto in quattro versioni adatte ad altrettanti home computer: lo Spectrum 48 K, il Commodore 64, il Vic-20 non espanso ed apparecchi Atari. La cassetta è contenuta in un originale involucro sferico, anziché nella consueta custodia portacassette.

Ogni versione è corredata da un breve ed utile opuscolo d'istruzioni (c'è anche un modulo per iscriversi al Creative Sparks Software Club: l'iscrizione è gratuita ed i membri possono beneficiare di offerte speciali e partecipare a competizioni).

La struttura del gioco è molto semplice ma, il suo sviluppo è così ben condotto da soddisfare anche gli appassionati più esigenti. Il giocatore pilota un battello del Soccorso Fluviale e deve cercare di salvare un gruppo di scienziati bloccati (per motivi ignoti!) in un anfratto del fiume. Tuttavia, le istruzioni avvertono il giocatore che gli scienziati vanno portati all'ospedale, quindi è ragionevole supporre che siano stati vittime di un incidente.

Durante i tentativi di salvataggio, il giocatore deve pilotare il battello, che si muove a notevole velocità, evitare secche e tronchi galleggianti, sfuggire anche agli improvvisi attacchi di numerosi alligatori. Nella versione per il Vic-20, sono stati aggiunti altri pericoli, rappresentati da enormi anaconda e da piroghe di selvaggi. Periodicamente, lungo le rive del fiume appaiono dei piccoli moli a cui il battello deve attraccare per mettere in salvo gli scienziati. Se il giocatore riesce a trasportare uno scienziato sull'altra sponda del fiume, incrementa il proprio punteggio, ma può guadagnare punti anche uccidendo gli alligatori che infestano il fiume.

Per ottenere un punteggio extra si dovranno trasportare gli scienziati in gruppi: il battello può infatti accogliere fino a di nove scienziati. Ovviamente, questo tentativo comporta maggiori difficoltà ed è bene ricordare che tutti i punti accumulati vengono perduti, se il battello urta un ostacolo. Dunque, il giocatore può decidere se rischiare il tutto per tutto, imbarcando un gruppo di scienziati per ottenere un alto punteggio, oppure se giocare con cautela, limitandosi a trasportare un passeggero alla volta. La situazione diviene ancor più drammatica quando compare l'elicottero (un aereo nella versione Spectrum) che lancia mine nel fiume.

La versione Vic-20 è in formato cartuccia (esclude quindi la noiosa procedura di caricamento da cassetta) prevede sei "vite" e consente di decidere se durante il gioco si desiderano salvare tre o sei scienziati. Inoltre, i punti accumulati in ciascuna vita vengono trasferiti alla "reincarnazione" successiva; ciò rende il gioco assai più semplice. Infine, questa versione possiede tre diversi fiumi navigabili.

Il *River Rescue* è un gioco tutto d'azione, e le difficoltà che propone hanno il chiaro intento di catturare per intero l'attenzione del giocatore. Tuttavia, risente di una certa mancanza di fantasia e questo limite gli ha impedito di divenire un vero classico nella gamma degli "arcade".

River Rescue: Per lo Spectrum 48 K
Per l'Atari
Per il Commodore 64
Per il Vic-20 (non espanso)
Editori: Creative Sparks (GB)
Autore: Kevin Buckner
Joystick: Kempston/Interfaccia 1 (Spectrum)
Joystick Commodore-compatibili (Commodore e Vic-20)
Joystick Atari-compatibili (Atari)
Formato: Cassetta



Chiamata di salvataggio
Queste due immagini del River Rescue sono state tratte dalla versione Spectrum. La prima riproduce la "prima pagina" del gioco ed offre un buon esempio della grafica impiegata. La seconda rappresenta la fase del gioco in cui il giocatore, dopo aver attraccato al molo, porta in salvo uno scienziato. A destra è visibile l'aereo che sta per lanciare le mine

LIZ HEANEY

Percorsi circolari

Una tecnica disponibile nei linguaggi di programmazione più evoluti

L'argomento trattato in questa lezione viene ben riassunto dal seguente gioco di parole:

Recursione: *vedi* Recursione

Questa definizione "circolare" indica un procedimento in cui ogni fase si serve dei risultati ottenuti dalle fasi che l'hanno preceduta. Tuttavia, il gioco di parole tralascia una caratteristica fondamentale: per essere funzionanti, i programmi di recursione devono possedere un'"uscita"! Il rompicapo usato per illustrare il problema è classico: *Le torri di Hanoi*. Il gioco inizia con una serie di dischi, di varie dimensioni, disposti l'uno sull'altro in ordine decrescente. Per risolvere il rompicapo, si deve creare una seconda pila di dischi attenendosi alle seguenti regole:

1) I dischi devono essere spostati uno alla volta;
2) Un disco non può essere appoggiato su un disco più piccolo;
3) Durante il gioco non si possono creare più di tre pile di dischi

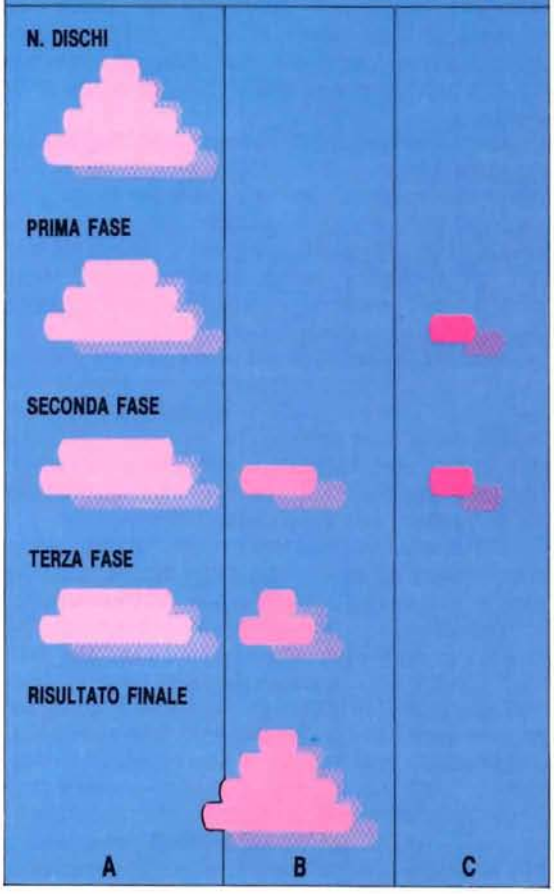
La figura a sinistra illustra come sfruttare il principio della recursione per risolvere il problema. Cominciamo con una pila formata da quattro dischi. Assegnando alla variabile N il valore 4, indichiamo il numero complessivo dei dischi da spostare. Poiché le regole permettono di muovere un solo disco alla volta, ci serviremo di una formula "recursiva" per ridurre di 1 il valore di N, proseguendo col calcolo, finché N non diventa uguale ad 1. Quando N=1 il programma interrompe l'operazione di calcolo, per spostare il disco appropriato.
Se la versione di BASIC con la quale si lavora prevede la recursione, è facile redigere un programma che esegua con precisione la procedura appena descritta. Nel programma, scritto in BASIC BBC, tutta la procedura di calcolo è contenuta nelle linee comprese tra 1000 e 1050. Il resto del programma si limita a visualizzare i vari movimenti dei dischi da una pila all'altra!

La versione Spectrum

Per trasformare il programma *Le Torri di Hanoi* in una versione eseguibile sullo Spectrum, occorre sostituire la procedura recursiva con una "subroutine recursiva", che inizia alla linea 1000 del listato. Tutte le volte che viene fatta una chiamata recursiva alle matrici M, A, B o C, la subroutine incrementa la variabile J che rappresenta il "puntatore" ed inserisce i nuovi valori in M(J), A(J), B(J) e C(J). Quindi i nuovi valori possono essere usati nella successiva chiamata di subroutine, senza modificare in alcun modo i valori precedenti. Questa tecnica può essere sempre usata per scrivere subroutine recursive in BASIC, indipendentemente dalla complessità della recursione.
La parte del programma che crea le immagini è piuttosto semplice: visualizza un disco nella nuova posizione e lo cancella da quella precedente, inserendo al suo posto dei caratteri "vuoti". Entrambi i programmi danno una visione laterale della pila di dischi. Per creare l'ombreggiatura, che aggiunge un senso di profondità, sono stati usati caratteri per metà colorati e per metà vuoti.

COME FUNZIONA IL GIOCO

Il programma chiede al giocatore di immettere il numero (N) di dischi che compongono la torre. I dischi vengono disposti in modo da formare la pila A. In seguito, l'algoritmo recursivo sposta i dischi, uno alla volta, attraverso le tre pile A, B e C, come mostrato nelle prime tre fasi in figura. Alla fine, la pila A scomparirà (quindi non ci saranno più dischi da muovere) e tutti i dischi si troveranno disposti in ordine decrescente nella pila B



LIZ DIXON

MICRO BBC

```
 10 DIM M(10): DIM A(10): DIM B(10): DIM C(10)
 20 DIM D$(10,10): DIM H(3): DIM P(3,10)
 30 GO SUB 3000
 90 DIM M(100): DIM A(100): DIM B(100): DIM C(100)
100 INPUT "QUANTI DISCHI (1-12)?";N
110 IF N<1 OR N>10 THEN GO TO 100
120 GO SUB 3100
130 LET J=1: LET M(J)=N: LET A(J)=1: LET B(J)=2:
    LET C(J)=3
140 GO SUB 1000
200 STOP
1000 IF M(J)=1 THEN GO SUB 1500: RETURN
1010 LET J=J+1
1020 LET M(J)=M(J-1)-1
1030 LET A(J)=A(J-1)
1040 LET B(J)=C(J-1)
1050 LET C(J)=B(J-1)
1060 GO SUB 1000
1100 LET M(J)=1
1110 LET A(J)=A(J-1)
1120 LET B(J)=B(J-1)
1130 LET C(J)=C(J-1)
1140 GO SUB 1000
1200 LET M(J)=M(J-1)-1
1210 LET A(J)=C(J-1)
1220 LET B(J)=B(J-1)
1230 LET C(J)=A(J-1)
1240 GO SUB 1000
1300 LET J=J-1
1310 RETURN
1500 LET PA=A(J): LET PB=B(J)
1510 LET M$=D$(P(PA,N+1-H(PA)))
1520 FOR I=22-H(PA) TO 7 STEP -1
1530 PRINT AT I-1,10*(PA-1);M$;
1540 PRINT AT I,10*(PA-1);B$;
1550 NEXT I
1560 FOR I=10*(PA-1) TO 10*(PB-1) STEP SGN (PB-PA)
1570 PRINT AT 6,I;M$;
1575 PRINT AT 6,I;B$;
1580 NEXT I
1590 FOR I=6 TO 20-H(PB)
1600 PRINT AT I,10*(PB-1);B$;
1610 PRINT AT I+1,10*(PB-1);M$;
1620 NEXT I
1640 LET H(PB)=H(PB)+1: LET P(PB,N+1-H(PB))=P
    (PA,N+1-H(PA))
1650 LET P(PA,N+1-H(PA))=0: LET H(PA)=H(PA)-1
1660 RETURN
3000 LET  B$="              ":  LET C$=CHR$
    143+CHR$  143+CHR$  143+CHR$  143
3010 LET C$="": FOR I=1 TO 10: LET C$=C$+CHR$
    143: NEXT I
3020 FOR I=1 TO 9 STEP 2
3030 LET  D$(I)=B$( TO 4-INT (I/2))+CHR$ 133+C$
    (  TO  2*INT (I/2))+CHR$ 138+B$( TO 4-
    INT (I/2))
3040 LET D$(I+1)=B$( TO 4-INT (I/2))+C$( TO
    I+1)+B$( TO 4-INT (I/2))
3050 NEXT I
3060 RETURN
3100 INK 3: PAPER 6: BORDER 6: CLS
3120 FOR I=1 TO N
3130 PRINT AT 21-N+I,0;D$(I);
3135 LET P(1,I)=I: LET P(2,I)=0: LET P(3,I)=0
3140 NEXT I
3150 LET H(1)=N: LET H(2)=0: LET H(3)=0
3160 RETURN
```

SPECTRUM

```
 10 DIM D$(12),H(3),P(3,12)
 20 PROCINIT
100 INPUT "QUANTI DISCHI?";N
110 IF N<1 OR N>12 THEN 100
120 PROCDISPLAY(N)
130 PROCHANOI(N,1,2,3)
200 END
1000 DEF PROCHANOI(M,PA,PB,PC)
1010 IF M=1 THEN PROCMOVE(PA,PB):
    ENDPROC
1020 PROCHANOI(M-1,PA,PC,PB)
1030 PROCHANOI(1,PA,PB,PC)
1040 PROCHANOI(M-1,PC,PB,PA)
1050 ENDPROC
1100 DEF PROCMOVE(PA,PB)
1110 D$=D$(P(PA,N+1-H(PA)))
1120 FOR I=24-H(PA) TO 10 STEP -1
1130 PRINT TAB(13*(PA-1),I);B$;
1140 PRINT TAB(13*(PA-1),I-1);D$;
1150 NEXT I
1160 FORI=13*(PA-1) TO 13*(PB-1)
    (PB-PA)
1170 PRINT TAB(I,9);D$
1180 NEXT I
1190 FOR I=9 TO 22-H(PB)
1200 PRINT TAB(13*(PB-1),I);B$
1210 PRINT TAB(13*(PB-1),I+1);D$;
1220 NEXT I
1240 H(PB)=H(PB)+1:P(PB,N+1-H(PB))
    =P(PA,N+1-H(PA))
1250 P(PA,N+1-H(PA))=0:H(PA)=H(PA)-1
1260 ENDPROC
3000 DEF PROCINIT
3020 FOR I%=1 TO 11 STEP 2
3030 D$(I%)=CHR$150+STRIG$(5-I%
    DIV2,"·")+CHR$234+STRING$(2*
    (I%DIV2),CHR$255)+CHR$53+
    STRING$(5-I%DIV2," ")
3040 D$(I%+1)=CHR$150+STRING$(5-I%
    DIV2," ")+STRING$(I%+1,CHR$255)
    +STRING$(5-I%DIV2," ")
3050 NEXT I%
3060 B$=CHR$150+STRING$(12," ")
3070 VDU 23,1,0;0;0;0;
3080 ENDPROC
3100 DEF PROCDISPLAY(N)
3110 CLS
3120 FOR I=1 TO N
3130 PRINT TAB(0,23-N+I);D$(I);
3135 P(1,I)=I:P(2,1)=0:P(3,I)=0
3140 NEXT I
3150 H(1)=N:H(2)=0:H(3)=0
3160 ENDPROC
```

Problemi... ricorrenti

Questa è un'immagine del programma "Le Torri di Hanoi" per lo Spectrum. Il colore dei dischi può essere facilmente modificato. Per capire come il computer riesca a risolvere il problema si dovrà prestare molta attenzione, perché l'azione si sviluppa rapidamente!



IAN McKINNELL

Dai concetti alla pratica

Come combinare assieme i moduli, sviluppati indipendentemente, in un programma completo

Quando si crea un programma conviene sviluppare fin dall'inizio uno schema globale che preveda un livello di base composto da routine generiche, utilizzate da altre routine di crescente specializzazione situate a livelli superiori. Entrambe dovrebbero essere poi pilotate da un unico modulo di controllo, posto al livello più alto. Questa struttura "piramidale" permette di adottare un metodo di progettazione detto "rifinitura del programma" o programmazione "top-down" (alto-basso).
Come il nome stesso suggerisce, la programmazione top-down comporta innanzi tutto, la creazione di un programma di controllo, col compito di chiamare le routine di livello inferiore (di cui ci occuperemo più avanti). Sviluppato questo programma, si scende di un livello, per descrivere il funzionamento delle varie routine richiamate dal modulo di controllo. Queste vengono descritte a loro volta, facendo riferimento alle routine di livello più basso da esse richiamate. Questa tecnica viene applicata, fino a raggiungere la base della piramide. Qui le funzioni elaborate da ciascuna routine sono talmente semplici, che vengono definite con sufficiente chiarezza anche dal solo codice di programmazione.
Come esempio, esaminiamo la struttura del gioco *L'impiccato*. Una versione "classica" di questo programma viene data a pagina 984, ma nel caso qui esaminato, non è il giocatore che deve indovinare una parola scelta dal computer, ma viceversa. Per raggiungere questo risultato, senza esser costretti a dotare il programma di un vasto vocabolario, è necessario passare al sistema una serie d'informazioni riguardanti la probabilità che ricorrano particolari sequenze di lettere.

```
100 REM Inizializza variabili e matrici

500 REM ********Routine di controllo********
510 REM
520 GOSUB 1000: REM Titolo & Messaggi informativi
530 GOSUB 2000: REM Prepara il tabellone di gioco
540 GOSUB 4000: REM Trova la lunghezza della parola
    pensata dal giocatore
550 GOSUB 8000: REM Seleziona un gruppo di dati
    e lo legge
560 GOSUB 3000: REM Cerca d'indovinare una lettera
570 GOSUB 4500: REM Chiede l'esattezza della lettera
    al giocatore
580 GOSUB 5000: REM Aggiorna il tabellone
590 IF GIOCO-NON-FINITO THEN 560: REM Cerca d'indovinare
    finche' il gioco non finisce
600 IF  VITTORIA  THEN GOSUB 10000 ELSE  GOSUB  11000:
    REM Sceglie la fine appropriata per vittoria o sconfitta
610 GOSUB 6000: REM Chiede al giocatore se vuole giocare
    ancora
620 IF ANCORA THEN 530: REM Da' il via ad una nuova partita
630 GOSUB 7000: REM Saluta e conclude il gioco
640 END
```

Il programma, come sempre, deve eseguire alcune operazioni prima di avviare il gioco: inizializzare le variabili, dimensionare le matrici, creare l'immagine di un "tabellone" e aggiornarlo, se necessario. Inoltre, devono esserci le routine per controllare il punteggio, per indovinare le lettere e per chiudere il gioco. Nel listato precedente, la prima linea è una semplice istruzione REM e rappresenta il primo nucleo della routine di controllo: si riferisce infatti alla inizializzazione delle variabili e delle matrici (da completare in un secondo momento). La routine di controllo vera e propria è formata da due cicli. Il ciclo esterno (linea 620) verifica se l'utente ha inviato o meno il segnale di fine gioco, mentre il ciclo interno (linea 590) controlla se il gioco è giunto al termine. Volendo sottoporre a verifica anche la routine di controllo, occorre creare delle false subroutine, da abbinare alle istruzioni GOSUB. Ciascuna GOSUB della routine di controllo deve essere seguita da un'istruzione REM che ne spieghi il funzionamento e la preceda di un particolare numero di linea, preferibilmente un numero facile da ricordare, come 1000 o 5000. È buona abitudine assegnare numeri di linea standardizzati alle routine che svolgono funzioni simili: è più facile individuarle e trasferirle da un programma all'altro. Ad esempio, le informazioni sul gioco potrebbero essere contenute in una subroutine che inizia alla linea 1000 mentre la linea di programma GOSUB 7000 potrebbe sempre chiamare la routine standard di fine gioco.
Nel nostro caso, abbiamo usato una routine di controllo breve e semplice perché può essere visualizzata per intero in un'unica "pagina" di schermo: ciò facilita la sua comprensione e le operazioni per la correzione di eventuali errori (debug). Le tre variabili GIOCO-NON-FINITO, VITTORIA ed ANCORA sono "flag", selezionati nelle varie subroutine chiamate dalla routine di controllo e qui usati per verificare se tale programma funziona nel modo previsto. Considerando la semplicità della routine di controllo, l'individuazione di eventuali errori logici non dovrebbe presentare eccessive difficoltà.
A questo punto, conviene riesaminare con occhio critico tutta la struttura del programma, per verificare "logicamente" se funziona correttamente in qualsiasi circostanza. In questa fase possiamo anche cercare di migliorare il progetto iniziale; ad esempio, facendo in modo che le informazioni sul programma siano disponibili in qualsiasi momento del gioco, oppure tenendo il conto delle partite vinte dal computer e dal giocatore oltre, magari, a un elenco delle parole che il programma non è riuscito a indovinare.
Il passo successivo prevede lo sviluppo di tutte le subroutine chiamate dal programma di controllo. I seguenti listati mostrano due di

queste subroutine. La prima (che inizia alla linea 4000) chiede semplicemente al giocatore di inserire un numero compreso tra 1 e 20 (la lunghezza della parola). Viene richiamata un'ipotetica subroutine "generica", con numero di linea 51000, che visualizza una stringa adatta alla richiesta, contenuta nella variabile PROMPT$, ed accetta quindi il numero digitato sulla tastiera dal giocatore. Se tale valore non è un intero compreso tra i valori indicati da MIN% e MAX%, sullo schermo compare un messaggio d'errore ed il sistema chiede di battere un nuovo numero. Questa subroutine può essere proficuamente utilizzata in molti altri programmi. In questa ottica, è consigliabile creare una "libreria" di routine generiche, dalla quale estrarre quelle che servono in particolari progetti.

```
4000 REM Chiede la lunghezza della parola pensata dal
     giocatore
4010 REM
4020 PROMPT$="Quante sono le lettere della parola ?"
4030 MIN%=1
4040 MAX%=20
4050 GOSUB 51000: REM Accetta in input un numero compreso
     tra MIN% & MAX%
4060 LPAROLA%=RISP%: REM RISP% viene utilizzata dalla
     subroutine alla linea 51000 per "passare" la
     risposta del giocatore
4070 RETURN

8000 REM Sceglie il gruppo di dati adatto e lo legge
8010 REM
8020 IF LPAROLA%>7 THEN FILEL%=8 ELSE FILEL%=LPAROLA%
8030 FILENOL$=STR$(FILEL%)
8040 FILENAME$="TAVOLA"+FILENOL$
8050 GOSUB 9000: REM Apre, legge e chiude il file contenente
     le informazioni di probabilita' per le parole di
     lunghezza pari a quella specificata
8060 RETURN
```

La seconda routine (che inizia alla linea 8000) usa le variabili locali FILE L% e FILENO L$. Abbiamo supposto che le informazioni necessarie per indovinare una lettera siano contenute in 8 tabelle, nelle quali viene indicata la probabilità che una data lettera segua immediatamente una qualsiasi altra. Poiché vogliamo avere in RAM una sola serie di informazioni alla volta, dobbiamo prima creare una stringa in FILENAME$ che contenga il nome del file dati, quindi chiamare la subroutine alla linea 9000 per leggere il file.

Spesso può darsi che sia necessario saltare direttamente da una routine all'altra. In questi casi conviene creare una routine aggiuntiva, che chiami ciascuna delle due routine in successione. Può sembrare una complicazione inutile, ma permette di tenere sotto stretto controllo il flusso del programma, pur mantenendo separati i vari moduli. Inoltre, ciò semplifica il loro trasferimento ad altri programmi. L'uso di subroutine "trasportabili" da un programma ad un altro implica un maggiore lavoro di programmazione. La struttura di queste routine deve infatti consentire un loro impiego in una vasta gamma di situazioni. Un tale risultato si raggiunge sostituendo le costanti con delle variabili. È ovvio che tutte le subroutine devono essere documentate in modo esauriente. La documentazione deve indicare l'esatto compito della routine, assieme a tutte le informazioni sulle variabili usate, sui valori in ingresso e in uscita e sui loro effetti collaterali (spostamenti del cursore, cambiamenti della mappa di memoria, chiusura dei file e così via).

Quando si scrive un programma, è buona norma seguire un'impostazione omogenea: la numerazione delle linee deve essere costante (ad esempio: 10, 20, 30... e così via), i titoli ed i commenti devono trovarsi all'inizio delle routine di appartenenza e l'istruzione RETURN deve occupare l'ultima linea. È sempre bene annotarsi i numeri di linea iniziale e finale di ciascuna routine. Nel prelevare una routine da una libreria, ci si accerti che il programma cui sono destinate preveda un adeguato intervallo tra i suoi numeri di linea, prima di eseguire la fusione (MERGE). Se il micro in uso non possiede il comando MERGE, si può usare un programma di text editor per combinare i due listati, memorizzandoli prima in formato ASCII anziché nel formato regolare (cioè con i "token"). Se anche questo sistema non è realizzabile, occorrerà ritrascrivere le routine ogni volta che si intende usarle. In ogni caso, non sarà mai necessario riprogettare la routine, se questa è stata ben impostata.



LIZ DIXON

Programmazione top-down

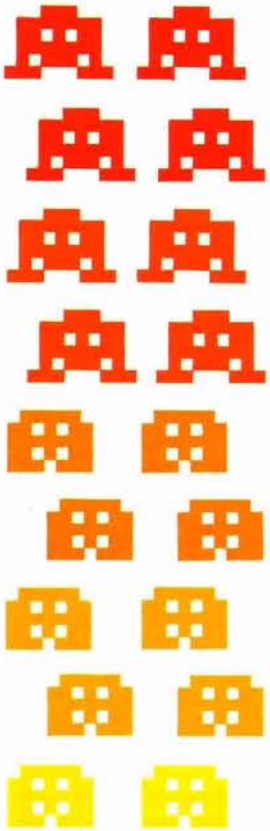
L'immagine a fianco ripropone il gioco "Le Torri di Hanoi" (vedere a pag. 954) per illustrare il metodo di programmazione top-down. I numeri di linea si riferiscono al listato BBC. L'apice della piramide rappresenta il programma d'inizializzazione, che è il primo ad essere eseguito. Il PROGRAMMA DI CONTROLLO rappresneta l'algoritmo recursivo, che esegue i calcoli e chiama le altre routine, al momento opportuno. Le SUBROUTINE SPECIFICHE (linee 1120-1200) vengono usate per spostare i dischi da una pila all'altra. Le due sezioni alla base della piramide rappresentano le SUBROUTINE GENERICHE, usate per disegnare i dischi e l'immagine iniziale del gioco sullo schermo. Confrontando questa struttura con il listato, si vedrà che il programma segue esattamente la sequenza qui descritta

Un attimo di tregua

Talvolta occorre creare una pausa nei programmi in codice macchina

Nell'Assembly del 6502, un ciclo di ritardo si può realizzare in più modi. Quello più ovvio e semplice consiste nel caricare un particolare valore in uno dei registri d'indice per poi decrementarlo, finché non giunge a zero, all'interno di un ciclo:

CICLI DI RITARDO	TEMPO IMPIEGATO DA OGNI OPERAZIONE
LDY #$07	2 cicli
DEY	2 cicli
BNE LOOP	2 cicli (3 cicli se "salta" alla stessa pagina di memoria, 4 cicli se "salta" ad un'altra pagina)



L'esecuzione di ogni istruzione in codice macchina impiega un certo numero di cicli di clock (detti anche "cicli macchina"), il cui valore è di solito indicato nei manuali del chip o del linguaggio Assembly. Per esempio, ciascuna delle istruzioni DEY e LDY (quest'ultima nel modo di indirizzamento immediato) impiega due cicli. Poiché un ciclo dura circa un microsecondo (un milionesimo di secondo), possiamo calcolare quanto impiega in "tempo reale" l'esecuzione di un ciclo iterativo di ritardo. Il numero totale di cicli può essere conteggiato nel seguente modo:

1) L'istruzione LDY impiega 2 cicli.
2) Il programma "rinvia" o salta indietro 7 volte e, per ogni salto, l'operazione BNE impiega 3 cicli. Quindi le due istruzioni assieme impiegano complessivamente $(2+3)\times7=35$ cicli.
3) L'ultima istruzione BNE non prevede il salto all'indietro, quindi dura solo due cicli (va tolto 1 dal precedente conteggio).

Il ritardo totale, contando ovviamente anche la LDY, è allora di $2+35-1=36$, ovvero 36 microsecondi.

L'uso di cicli di ritardo in codice macchina per ottenere ritardi in "tempo reale" (cioè ritardi che si possano misurare in secondi o microsecondi) talvolta causa alcuni problemi. Primo tra essi è che il processore, durante l'esecuzione di un programma in codice macchina, interrompe periodicamente la propria attività per effettuare controlli su altre parti del sistema (per esaminare la tastiera, aggiornare l'orologio interno, ecc.). Queste momentanee sospensioni, nell'esecuzione del programma, vengono dette "interrupt". Nel chip 6502 si possono verificare due tipi di interrupt: NMI (Interrupt Non Mascherabili) ed IRQ (Interrupt Di Richiesta). La definizione assegnata alle prime indica che non se ne può impedire l'esecuzione (sono indispensabili per il funzionamento di alcune parti del sistema), mentre è invece possibile bloccare gli interrupt IRQ che non sono "vitali" per il buon funzionamento del processore.

Gli interrupt IRQ si possono mascherare, (cioè filtrare) col comando SEI che pone ad 1 un particolare bit del registro di stato del processore. Questo bit viene nuovamente azzerato col comando CLI. Mascherando gli interrupt IRQ, prima di eseguire il ciclo di ritardo, possiamo verificare in pratica tale meccanismo. Tuttavia, si rammenti che l'eventuale comparsa di un interrupt non mascherabile, durante l'esecuzione del ciclo, provoca una variazione nella durata complessiva. Per mascherare gli interrupt, il precedente listato del ciclo di ritardo va così modificato:

ISTRUZIONE	FUNZIONE	TEMPO IMPIEGATO
SEI	Disabilita IRQ	2 cicli
LDI #$07		36 cicli
DEY		
VNE LOOP		
CLI	Riabilita IRQ	2 cicli

La mascheratura degli interrupt IRQ aggiunge un quarto ciclo di clock alla routine, che adesso durerà 40 microsecondi, purché non si verifichino interrupt del tipo NMI.

Un secondo aspetto dei cicli di ritardo è quello della "risoluzione". Questa si riferisce alla variazione del tempo impiegato, in ciascuna iterazione, tra un valore del contatore e quello successivo. Nella nostra routine abbiamo caricato nel registro Y il valore 7: se avessimo usato il valore 6, allora la durata del ritardo sarebbe stata di 35 microsecondi $(2+2+(2+3)\times6-1+2)$, mentre con un valore 5 la durata sarebbe di 30 microsecondi. Il nostro ciclo ha dunque una risoluzione minima di 5 microsecondi.

L'introduzione di istruzioni NOP esternamente al ciclo iterativo, permette di "calibrare" con precisione il programma, in modo da avere tempi di ritardo non multipli di 5. Un'istruzione NOP (Nessuna OPerazione) impiega 2 cicli di clock, senza alcun altro effetto. Pertanto, volendo ottenere un ritardo di 44 microsecondi, dobbiamo far precedere il ciclo iterativo da due istruzioni NOP:

SEI	2 cicli
LDY #$07	2 cicli
NOP	2 cicli
NOP	2 cicli
DEY BNE LOOP	34 cicli
CLI	2 cicli

La durata massima dei ritardi così programmati dipende dal massimo valore attribuibile a Y. Poiché Y è un registro d'indice ad 8 bit, il massimo valore che esso può contenere è 255,

corrispondente ad un ritardo massimo di 1280 microsecondi, vale a dire circa un centesimo di secondo. Questo è un tempo molto lungo per un microprocessore, ma estremamente breve in termini umani. Un metodo per ottenere intervalli di tempo più lunghi può essere quello di introdurre istruzioni NOP all'interno del ciclo. Ad esempio, aggiungendo un'unica istruzione NOP, si ha un ritardo massimo di 1790 microsecondi (2+2+(2+2+3)×255—1+2).
Ma per pause molto più lunghe occorre un'altra tecnica. I due sistemi più comuni prevedono l'uso di un secondo ciclo, "nidificato" nel primo, oppure decrementano un numero di maggiori dimensioni, conservato in una "parola" a 16 bit (ossia due byte), contenuta in memoria. In entrambi i sistemi è possibile calcolare la risoluzione che si vuol ottenere.

Contatore a ciclo "nidificato"

```
RITAR   SEI
        LDX     #$04    ; usa il registro x come contatore esterno del ciclo
CICL1   LDY     #$FF    ; usa il registro y come contatore interno del ciclo
CICL2   DEY
        BNE     CICL2   ; fine del ciclo interno
        DEX
        BNE     CICL1   ; fine del ciclo esterno
        CLI
```

L'esecuzione del ciclo interno del programma precedente dura 1276 microsecondi (2+(2+3)×255—1). Il ciclo esterno controlla l'esecuzione del ciclo interno ed esegue per quattro volte le istruzioni DEX e BNE.
La durata di questo ritardo sarà quindi di: 2+2+(1276+2+3)×4—1+2 = 5129 microsecondi.

Tempi di pausa per lo Z80

Ogni istruzione in codice macchina dello Z80 ha un proprio tempo d'esecuzione (misurato in unità chiamate "stati T"). Siccome lo Z80 viene fatto funzionare, a seconda della marca del computer, a differenti frequenze di clock, per calcolare il tempo reale impiegato per l'esecuzione di un'istruzione il numero di stati T va diviso per la frequenza di clock del micro. Ad esempio, un'istruzione che impiega quattro stati T per essere elaborata da un microprocessore con frequenza di clock di 2 MHz, avrà una durata in tempo reale di 2 microsecondi. Il volume "Programming the Z80" di Rodnay Zak offre un completo elenco dei tempi d'esecuzione dei comandi Z80 e delle frequenze di clock dei più popolari micro basati su processore Z80: ZX81 (3,25 MHz), Spectrum (3,5 MHz), TRS80/Video Genie della Tandy (1,7 MHz), Amstrad (4 MHz).
Per ottenere piccoli intervalli, si può usare l'istruzione NOP che, su di un micro a 2 MHz, produce un ritardo di 2 microsecondi. Questa istruzione può essere impiegata più volte ma, per ottenere ritardi leggermente più lunghi, si può ricorrere ad una o più chiamate a routine "fittizie". Quella che segue, ad esempio, produce un ritardo di 27 stati T:

```
        CALL   PAUSA
PAUSA RET
```

In questo esempio, l'istruzione CALL dura 17 stati T e l'istruzione RET 10. Quindi, con un processore che gira a 2 MHz, il ritardo è di 13,5 microsecondi. Per aumentarlo leggermente, si potrebbero inserire alcune NOP all'inizio della routine.
Per generare ritardi maggiori è necessario servirsi di un ciclo. Nel seguente esempio, un valore caricato in un registro viene successivamente decrementato all'interno di un ciclo. La routine produce un ritardo di 99 stati T (ovvero 49,5 microsecondi a 2MHz).

ISTRUZIONI	TEMPO IMPIEGATO (stati T)
CALL ATTESA	17
(CICLO DI ATTESA)	
LD B,5	7
DEC B	4
JR NZ,LOOP	12 (o 7 se "vero")

Le tre istruzioni che iniziano con LD B,5 rappresentano il vero e proprio ciclo di ritardo. Come nell'esempio fatto per la routine in codice macchina del 6502, il tempo totale di ritardo dipende dal valore caricato nel registro contatore. Il numero complessivo di cicli di clock necessari per eseguire la routine può essere espresso come:

C = 24+(N×16)—5

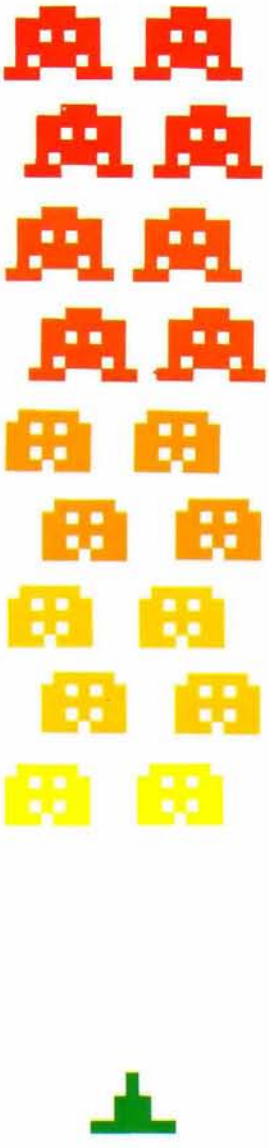
dove N indica il valore caricato nel registro B. Si può anche usare un ciclo "nidificato" ma, in questo caso, è necessario fare alcune ulteriori considerazioni. Innanzi tutto, si devono salvare (appoggiare) sullo stack i valori contenuti in qualsiasi registro impiegato nella routine. Secondariamente, in alcuni apparecchi le interrupt prodotte dall'hardware modificano inevitabilmente il ritardo programmato. Le interrupt "mascherabili" si possono invece disattivare e ripristinare con le istruzioni DI e EI. La seguente routine adopera un ciclo nidificato:

ISTRUZIONE	FUNZIONE	TEMPO (stati T)
DI	Disattiva le interrupt	4
PUSH DE	Salva i contenuti del registro DE	11
LD D,nn	Valore del contatore interno	7
LD E,nn	Valore del contatore esterno	7
CALL CICES	Richiama ciclo esterno	17
POP DE	Recupera i contenuti di DE	10
EI	Riabilita le interrupt	4
:		
(CICES)		
DEC E	Decrementa il ciclo esterno	4
RET Z	Chiude il ciclo se E=0	11 o 5
(CICIN)	Decrementa il ciclo interno	4
JP Z,CICES	Salta se D=0	10
JP CICIN	Continua con il ciclo interno	10

In questa routine, il ritardo aumenta al crescere del valore contenuto nel registro E. La routine termina quando il valore del registro E decresce fino a 0. Notare che, se il ciclo interno raggiunge lo 0 quando il ciclo esterno presenta ancora un valore maggiore di uno, il ciclo interno va inizializzato a 255, decrescendo poi fino a 0, prima che il controllo torni al ciclo esterno.

Invasione sincronizzata

I ritardi programmati in codice macchina sono molto usati nei programmi di gioco, soprattutto quando l'oggetto che si muove sullo schermo interagisce col giocatore. Un tipico esempio lo si trova in giochi del tipo "Space Invaders": l'introduzione di cicli di ritardo consente di rallentare la discesa degli alieni e permette di regolare in modo adeguato lo svolgimento del gioco



La Prism

Inizialmente distributrice dei prodotti Sinclair, la Prism è oggi una forte presenza commerciale nel settore informatico



Il presidente della Prism
Richard Hearth ha fondato la Prism come consociata della ECC Publications

La Prism adotta una politica industriale tutta proiettata nel futuro, contrariamente a molte altre ditte del settore, ed ha giocato un ruolo di primo piano nello sviluppo del Micronet, il primo database britannico di grosse dimensioni dedicato agli utenti di home computer. La Prism è stata fondata nel 1982, dalla ECC Publications con lo scopo di guidare la nascita del Micronet. A dirigerla sono stati chiamati Richard Heath e Bob Denton. Il Micronet è collegato al Prestel, il più sviluppato sistema pubblico viewdata della Gran Bretagna (vedere a pag. 581). La ECC Publications aveva già iniziato la pubblicazione della rivista "Sinclair user", quando ancora la Sinclair commerciava i propri prodotti soltanto tramite ordinazioni postali o attraverso la catena di negozi W.H. Smith. La pubblicazione ottenne un immenso successo, nonostante l'iniziale scetticismo di Terry Cartwright, oggi direttore commerciale della Prism.
Solo in seguito, la Sinclair decise di adottare una nuova politica commerciale, ampliando la propria presenza sul mercato e sottoscrivendo un contratto con la Prism per il lancio dello Spectrum. In effetti, il nome Prism deriva in parte da questa coincidenza con il lancio dello Spectrum: infatti, facendo passare un raggio luminoso attraverso un "prisma" si ottiene uno "spettro" di colori! La Prism ha recentemente annunciato di aver venduto 500.000 apparecchi della Sinclair, vale a dire il 25% di tutte le vendite di home computer avvenute in Gran Bretagna.
La Prism, comunque, non si è limitata allo Spectrum, ma commercializza anche i computer Oric, l'Atmos e il Wren oltre ad una serie di modem (fabbricati dalla OE Ltd e dalla Thorn EMI), adatti ai più diffusi home computer. Uno dei modem di maggior successo è il VTX5000, per lo Spectrum, ma ne esiste anche uno progettato appositamente per il Commodore 64.
Infine, la Prism si occupa anche della vendita di robot domestici, relativamente economici. Questo è un mercato in forte espansione e la Prism vi partecipa distribuendo il "Topo", un prodotto importato dagli Stati Uniti, ed una serie di altri robot economici in "scatola di montaggio", venduti sotto la sigla "Movits". La Prism, che continuerà ancora a distribuire i prodotti Sinclair, ha dichiarato che in futuro punterà su una sempre maggior diversificazione dei prodotti di listino.



Un computer portatile da ufficio
Questo è il Wren, un microcomputer commercializzato dalla Prism, basato sul chip Z80, dotato di due unità a dischetti e di un modem incorporato

	Errata	Corrige
pag. 598 **PROGRAMMA MONITOR**		
versione Spectrum	1100 LET X$"0123456789ABCDEF"	1100 LET X$ = "0123456789ABCDEF"
	3110 IF NM<256 THEN LET N=NM* GOSUB 3000:LET H$ = "00"+B$:RETURN	3110 IF NM<256 THEN LET N=NM: GOSUB 3000: LET H$="00"+B$:RETURN
	5280 IF DN>=0 THEN LET P=2:LET N=DN	5280 IF DN>0 THEN LET P=2:LET NM=DN
versione BBC	1050 LT=4::DIM C$(LT),O$(LT)	1050 LT=4:DIM C$(LT),O$(LT)
aggiungere la linea		2200 GET A$:IF A$ = " " THEN GOTO 2200
versione Commodore	1050 LT=4::DIM C$(LT),O$(LT)	1050 LT=4:DIM C$(LT),O$(LT)
aggiungere la linea		2200 GET A$:IF A$ = " " THEN GOTO 2200
	5300 A$=A$+ " ":IF LL<4 THEN A$=LEFT$ ("00 00",4—LL)+A$	5300 A$=A$+" ":IF LL<4 THEN A$=LEFT$ ("0000",4—LL) +A$
pag. 734 (2ª colonna, riga 4)	POKE 53256, PEEK (53256) OR 32	POKE 53265, PEEK (P3265) OR 32

pag. 724 nella tabella **FILE A CONFRONTO** cambiare **FILE SEQUENZIALI** con **FILE AD ACCESSO CASUALE** e viceversa.