CADEL

Spediz. in abbonamento postale GR II/70 L. 2.200
(...)

62 CORSO PRATICO COL COMPUTER

422147

è una iniziativa
FABBRI EDITORI
in collaborazione con
BANCO DI ROMA
e **OLIVETTI**

F4 F5 F6 F7 F8

BATTERY LOW

diretto da **GIANNI DEGLI ANTONI**

FABBRI
EDITORI

Direttore dell'opera
GIANNI DEGLI ANTONI

Comitato Scientifico
GIANNI DEGLI ANTONI
Docente di Teoria dell'Informazione, Direttore dell'Istituto di Cibernetica dell'Università degli Studi di Milano

UMBERTO ECO
Ordinario di Semiotica presso l'Università di Bologna

MARIO ITALIANI
Ordinario di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

MARCO MAIOCCHI
Professore incaricato di Teoria e Applicazione delle Macchine Calcolatrici presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

DANIELE MARINI
Ricercatore universitario presso l'Istituto di Cibernetica dell'Università degli Studi di Milano

Curatori di rubriche
MARCO ANELLI, DIEGO BIASI, ANDREA GRANELLI, ALDO GRASSO, MARCO MAIOCCHI, DANIELE MARINI, GIANCARLO MAURI, CLAUDIO PARMELLI

Testi
CLAUDIO PARMELLI, DANIELE MARINI, Eidos (TIZIANO BRUGNETTI), ANDREA GRANELLI, ALDO GRASSO, Etnoteam (ADRIANA BICEGO)

Tavole
Logical Studio Communication
Il Corso di Programmazione e BASIC è stato realizzato da Etnoteam S.p.A., Milano
Computergrafica è stato realizzato da Eidos, S.c.r.l., Milano
Usare il Computer è stato realizzato in collaborazione con PARSEC S.N.C. - Milano

Direttore Editoriale
ORSOLA FENGHI

Redazione
CARLA VERGANI
LOGICAL STUDIO COMMUNICATION

Art Director
CESARE BARONI

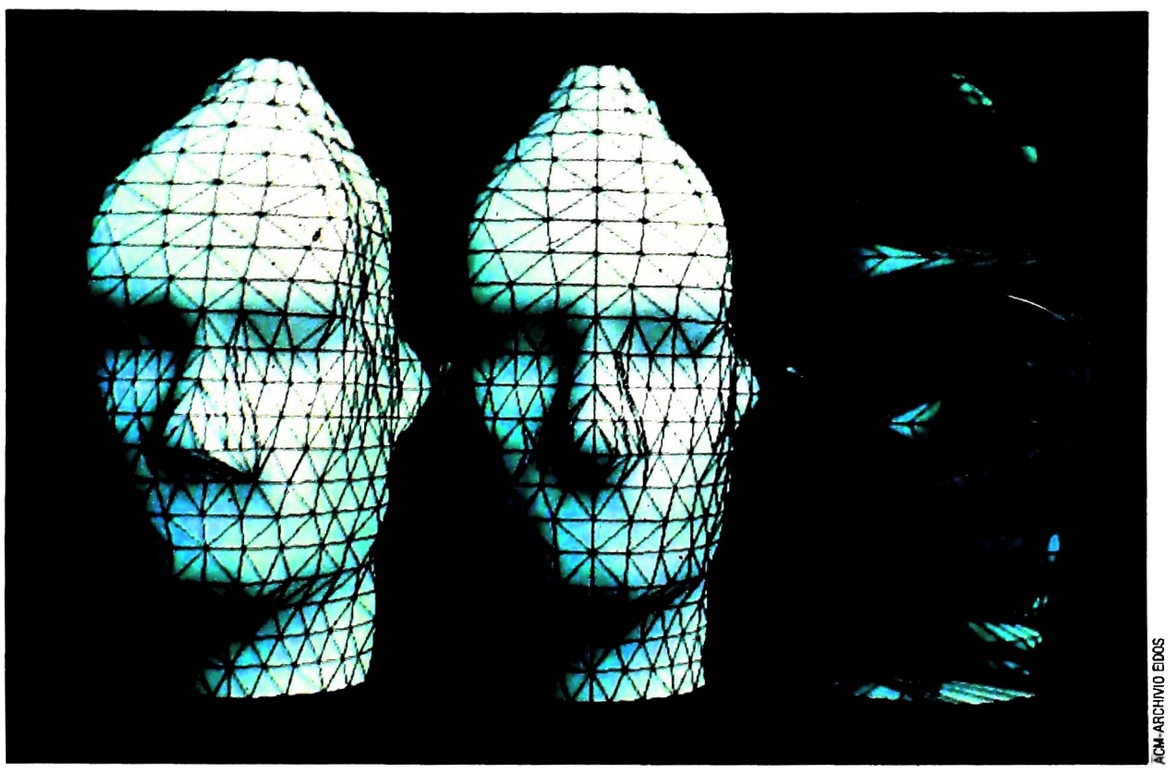
Impaginazione
BRUNO DE CHECCHI
PAOLA ROZZA

Programmazione Editoriale
ROSANNA ZERBARINI
GIOVANNA BREGGÉ

Segretarie di Redazione
RENATA FRIGOLI
LUCIA MONTANARI

L'ANIMAZIONE TRIDIMENSIONALE

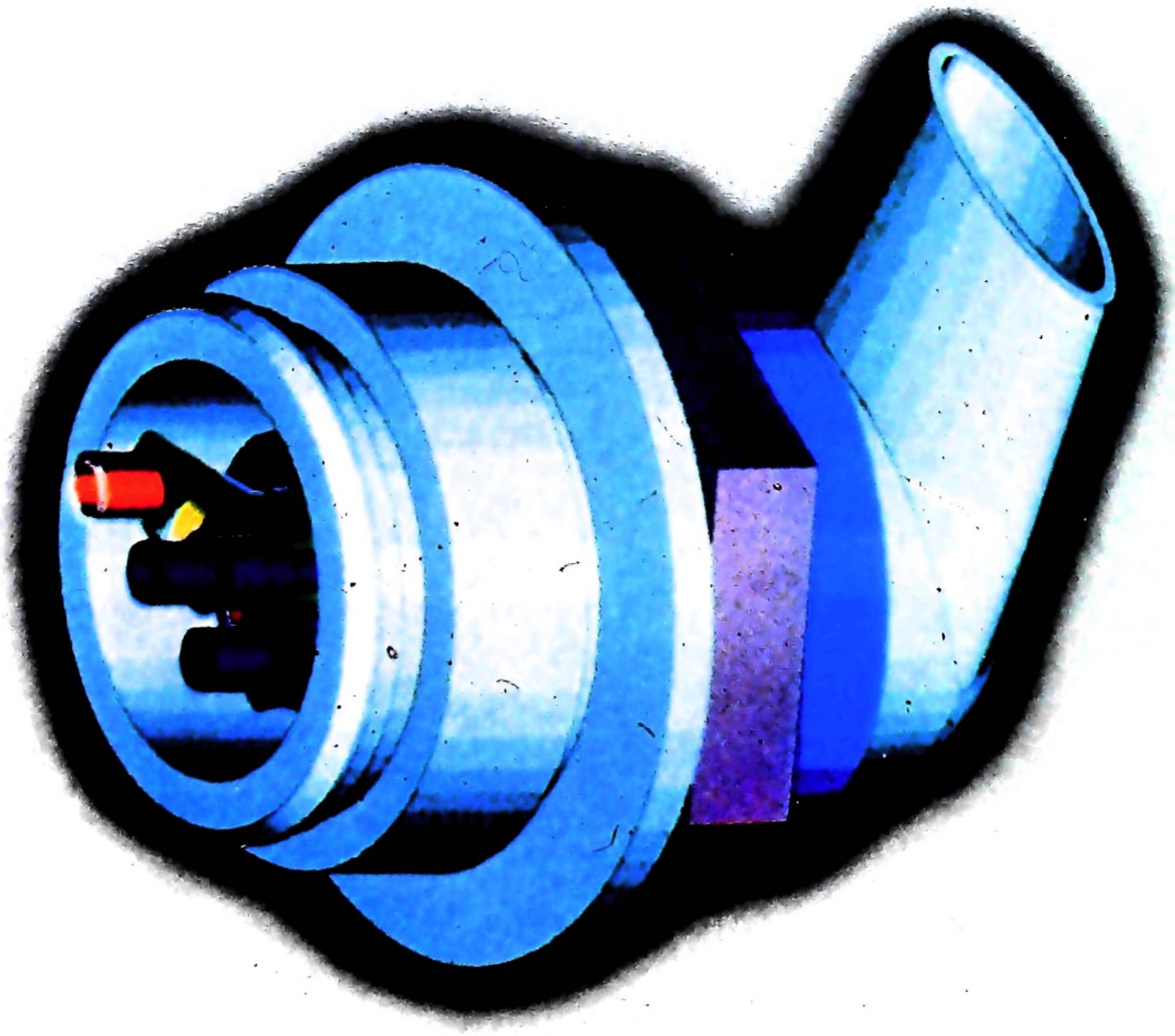
Le problematiche associate all'animazione 3D.



Nell'animazione tridimensionale, in primo luogo, occorre sottolineare che l'immagine — anziché essere costituita dalla codifica digitale del valore di luminosità e di colore dei punti che la costituiscono — è descritta in termini geometrici. Per esempio, la figura di un cubetto viene realizzata mediante dati che descrivono la geometria del cubetto stesso in un sistema di riferimento cartesiano. Ogni vertice del cubetto è codificato nell'elaboratore mediante le sue coordinate geometriche mentre gli spigoli e le facce sono descritte mediante relazioni che legano le coordinate ai vertici. La visualizzazione del cubetto è ottenuta applicando automaticamente le regole della visualizzazione prospettica.
La descrizione di una figura come è stata sopra delineata consiste in un vero e proprio modello dell'oggetto: quindi, alla base di un sistema per la realizzazione di visualizzazioni tridimensionali vi deve essere un sistema per la costruzione di modelli geometrici.
Una volta creato il modello, l'animazione consiste nel simulare sia le possibili azioni di una macchina da presa, sia i possibili movimenti che le componenti della scena devono compiere. Nel primo caso, l'azione del computer è relativamente semplice, mentre nel secondo si tratta di disporre di strumenti potenti per descrivere le azioni degli "attori" e quindi l'intera sceneggiatura.

Modellazione geometrica

La problematica della costruzione dei modelli geometrici è stata ereditata nella computer animation dal mondo della



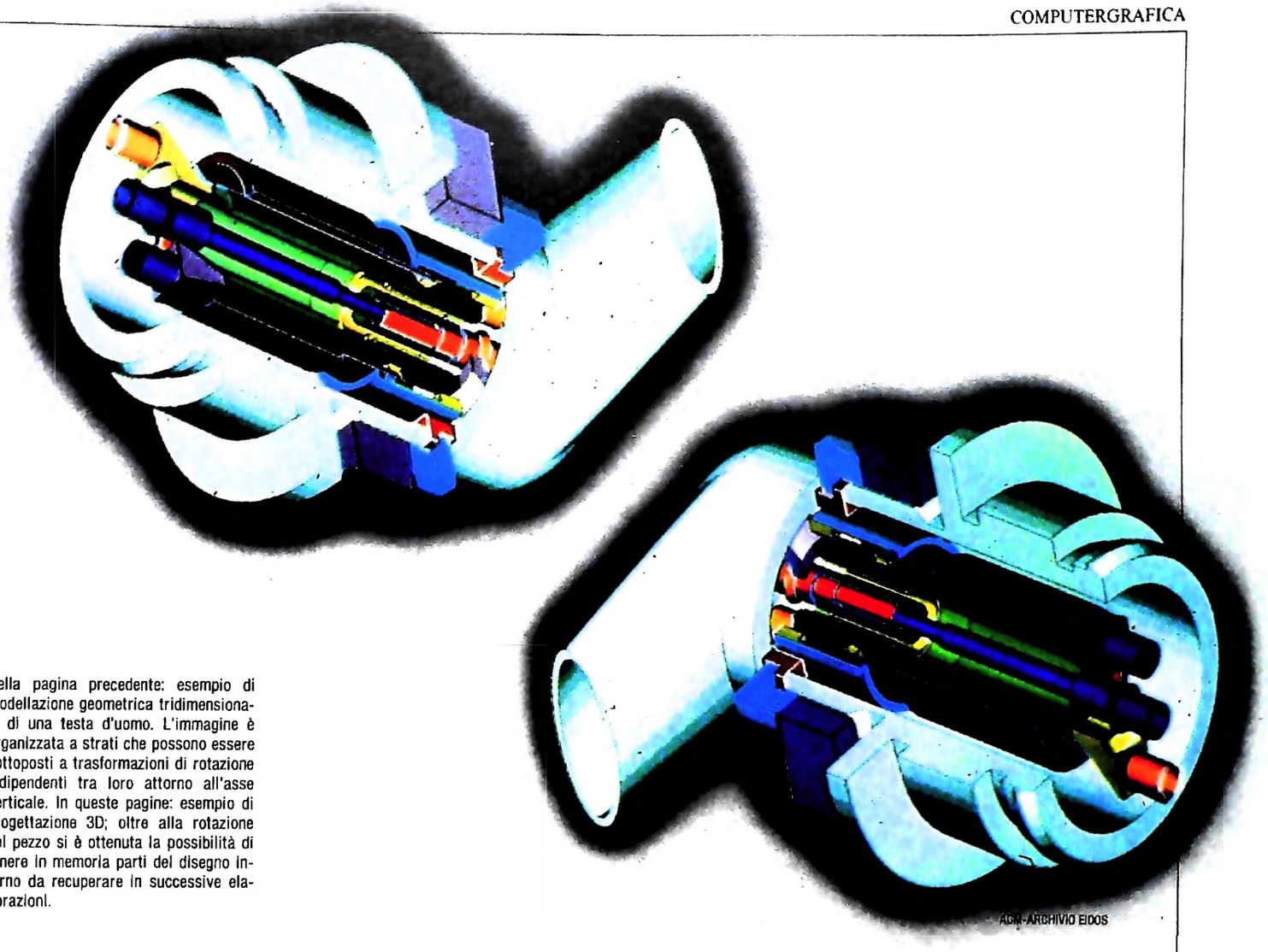
progettazione ingegneristica. Per modello geometrico si deve qui intendere sia il modello per esempio di un autoveicolo (come il soggetto di uno spot pubblicitario), sia quello di un personaggio schematizzato, come un burattino. Mentre il primo è relativamente semplice (non si tratta infatti di creare modelli tecnicamente completi, quanto modelli che comunichino il messaggio desiderato e quindi graficamente essenziali), il secondo tipo di modello è estremamente complesso, ed è oggetto di ricerca in svariati centri universitari.
La modellazione geometrica che è necessaria all'animazione deve quindi essere supportata da sistemi che siano soprattutto di semplice uso, e che non siano appesantiti da prestazioni più tipiche dell'ingegneria.
La costruzione del modello può essere effettuata trascurando gli effetti di realismo che deve possedere l'immagine finale, in quanto — in questa fase — ciò che conta è soprattutto la possibilità di verificare velocemente se l'oggetto è stato correttamente definito.
Inoltre, su un modello semplificato (per esempio visualizzato a "fil di ferro" con le sole linee di contorno) è possibile effettuare con elevata velocità verifiche di simulazioni di movimenti che la scena finale deve compiere.

Movimenti di camera

Il primo tipo di movimenti (e, come si è detto, il più semplice) è quello che permette di simulare i movimenti della macchina da presa. Carrellate, rotazioni, modifiche di focale dell'ottica di ripresa possono venire agevolmente attuate facendo eseguire all'elaboratore i calcoli su cui si basa l'ottica geometrica. Questo tipo di animazione sostituisce la più nota "computer controlled animation" consistente nell'eseguire i movimenti di una camera di ripresa sotto il diretto controllo dell'elaboratore (tecnica usata per esempio nei noti film della serie "Guerre stellari"), eseguendo la ripresa ravvicinata dei modelli in scala.

Descrizione delle traiettorie degli oggetti

Più complessa è l'animazione che si basa sul movimento separato degli elementi che compongono la scena. Per semplicità, possiamo chiamare tali oggetti "attori". Il loro movimento può venire descritto con equazioni matematiche, oppure richiede descrizioni molto più complesse. Nel primo ca-



Nella pagina precedente: esempio di modellazione geometrica tridimensionale di una testa d'uomo. L'immagine è organizzata a strati che possono essere sottoposti a trasformazioni di rotazione indipendenti tra loro attorno all'asse verticale. In queste pagine: esempio di progettazione 3D; oltre alla rotazione del pezzo si è ottenuta la possibilità di tenere in memoria parti del disegno interno da recuperare in successive elaborazioni.

so, è relativamente semplice insegnare all'elaboratore a calcolare le traiettorie degli attori; un classico esempio è la situazione del volo della sonda spaziale Voyager, in cui il movimento della sonda e dei pianeti è descritto sulla base delle equazioni del moto dei corpi celesti.

Per descrivere movimenti più complessi, come per esempio quello degli arti di un manichino, occorrono strumenti concettualmente più potenti di quelli ereditati dalla cultura fisico-matematica. In questo caso, il problema della definizione del comportamento degli attori in una sceneggiatura presenta grandi somiglianze con problematiche tipiche dei sistemi di elaborazione dei dati, quali la sincronizzazione di processi concorrenti.

Una sceneggiatura può essere considerata (dal punto di vista organizzativo) come la descrizione delle relazioni che differenti oggetti devono assumere, con particolare enfasi al tempo, ovvero agli istanti in cui tali relazioni devono compiersi. Dal punto di vista informatico, ciò corrisponde alla descrizione del comportamento di sistemi complessi che richiedono sincronizzazione di eventi.

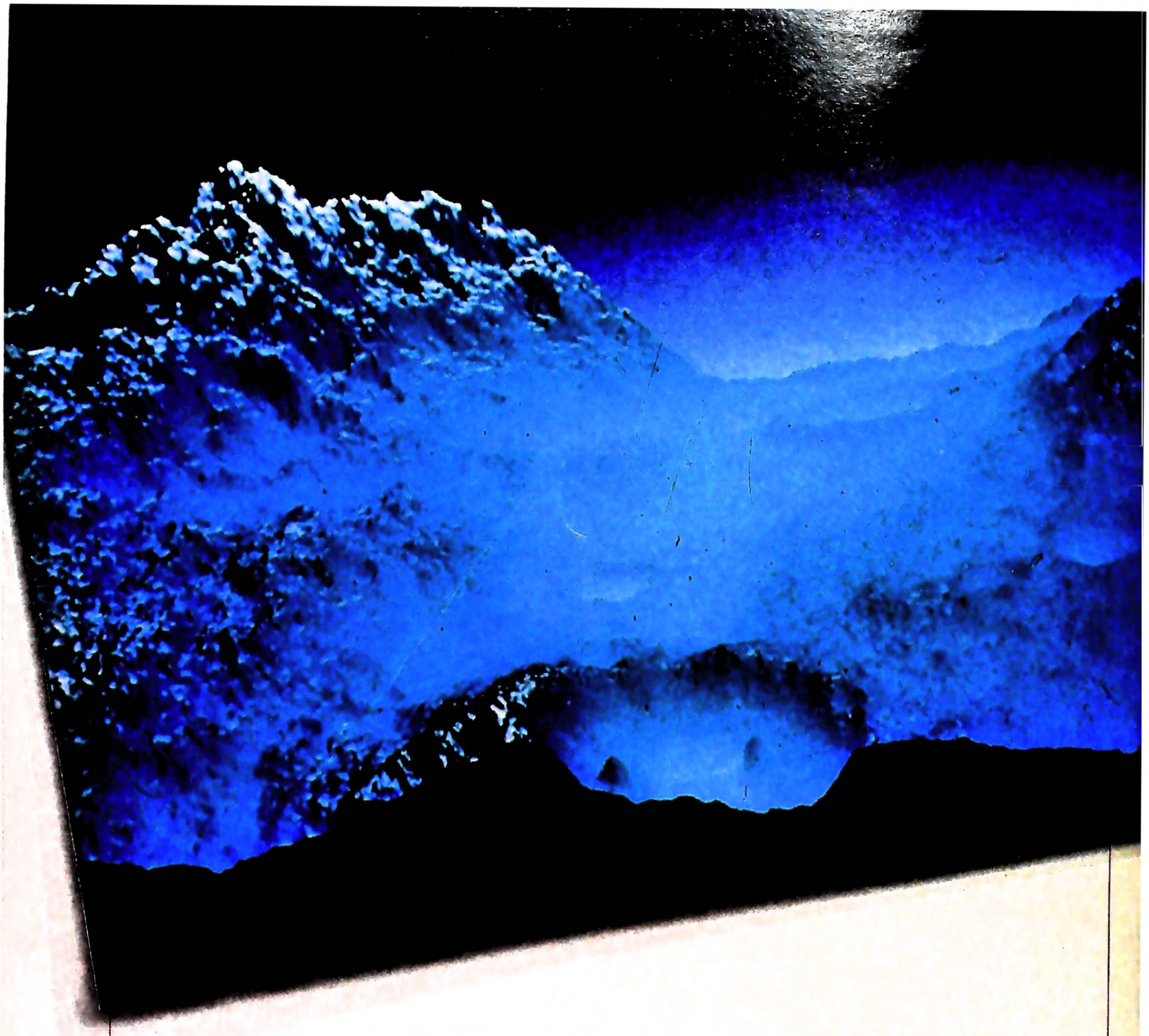
L'entrata in scena di un attore non può che accadere nel momento imposto dall'autore o dal regista, pena il decadere della scena complessiva, che assumerebbe significati imprevedibili (dai comici ai drammatici).

Ovviamente, si è qui concentrata l'attenzione sugli aspetti sintattici di una sceneggiatura, lasciando completamente al di fuori i significati e le forme, che, nel caso del cinema, assumono sempre ruoli dominanti.

Purtuttavia, lasciando a registi e creativi questi problemi, anche solo la descrizione di sequenze animate con azioni indipendenti dei vari "attori" costituisce un compito estremamente arduo per l'elaboratore, che richiede sempre descrizioni accuratissime e complete dei compiti da svolgere.

Una breve riflessione porta immediatamente a cogliere nel linguaggio attualmente disponibile le difficoltà maggiori. Se potessimo dire con semplicità che gli attori A e B devono compiere l'azione C a partire dall'istante in cui l'attore D ha completato l'azione E e che l'evento F ha inizio proprio nell'istante in cui A e B iniziano l'azione C... allora tutti i problemi sintattici sarebbero risolti.

Apparentemente, il problema non appare in tutta la sua complessità, ma basta pensare alla difficoltà con cui può essere colta la visione d'insieme della situazione sopra descritta per rendersi conto delle difficoltà date.



Linguaggi di animazione

La problematica delineata in precedenza può trovare spunti di soluzione adottando linguaggi ad altissimo livello concettuale, quali le Reti di Petri e i linguaggi funzionali. Con i primi, tutte le problematiche di sincronizzazione, concorrenza, parallelismo (tipiche tra l'altro dei processi naturali) possono venire trattate anche in modo automatico. Poiché inoltre le Reti di Petri permettono una rappresentazione grafica (seppur schematica) dei processi in esame, esse consentono un'immediata percezione della situazione complessiva.
Con l'aiuto di linguaggi funzionali diventa viceversa relativamente semplice programmare il movimento per esempio di un braccio (posto che se ne disponga un modello geometrico); infatti, le operazioni di movimento non sono altro che funzioni applicate a oggetti che vengono agevolmente organizzati in forma gerarchica.
Non è impossibile ipotizzare, in un futuro molto prossimo, la comparsa di sistemi integrati che permettano di definire una sceneggiatura con gli strumenti linguistici più potenti, e di tradurla in istruzioni da inviare a sistemi di rappresentazione di dati grafici per generare singole immagini.

Image rendering

Il penultimo passo nella creazione delle sequenze animate con l'aiuto del computer consiste nell'avere a disposizione sistemi che aiutino nella creazione di immagini realistiche a partire dai modelli geometrici rappresentati a "fil di ferro" e le cui sequenze di movimento sono state accuratamente predisposte. Per image rendering si intende solitamente quell'in-

ACM-ARCHIVIO EIDOS

Immagini di paesaggi naturali ottenute mediante la tecnica del frattale: sono il risultato di un processo matematico che genera strutture casuali.

sieme di tecniche di programmazione che permettono di creare scene realistiche (e a volte iperrealistiche), simulando effetti di illuminazione, di riflesso, d'ombra, trasparenza e addirittura la tessitura tipica dei supporti materiali su cui si presume nella realtà debba essere realizzato l'oggetto proposto dall'elaboratore.

Generazione di paesaggi

Per concludere questa panoramica, citiamo un ultimo promettente supporto prodotto dalla cultura informatico-matematica. Si tratta dei frattali, ovvero di quelle strutture matematiche che permettono di creare e rappresentare forme naturali quali per esempio il profilo di montagne o più semplicemente la struttura simmetrica dei fiocchi di neve ecc.
Le potenzialità ancora da esplorare di queste tecniche risiede in gran parte nella creazione di scene e paesaggi realistici senza la necessità di ricorrere a riprese dal vero e alla loro registrazione nel computer.

POSTA ELETTRONICA (IV)

Comandi e relative funzioni.

READ
Si può abbreviare con le iniziali REA.
Il comando in oggetto permette la lettura di uno o più messaggi che sono stati ricevuti, composti o archiviati in un file.
Aperta la sessione, Infomark visualizza la presenza e la quantità dei messaggi in attesa di lettura. I messaggi inviati sono archiviati nel file di sistema UNREAD.
Una volta letti, vanno nel file CHRONO. Dopo il comando READ, il sistema visualizza il primo messaggio contenuto nell'archivio UNREAD e l'identificazione del messaggio, in alto a destra, contenente la data e l'ora di invio del messaggio e un codice unico attribuito dal sistema a ogni messaggio inviato, composto da quattro caratteri numerici.
Quindi, ogni volta che viene premuto il tasto return, Infomark visualizza gli altri messaggi sino alla visualizzazione di READING COMPLETED.
Giunti a quest'ultima fase, il file UNREAD rivela di non contenere più messaggi da mostrare. Terminato di leggere il messaggio, si possono usare i comandi ANSWER, FORWARD, COMPOSE, FILE e DELETE secondo l'occorrenza.
Per gestire una buona corrispondenza è consigliabile operare, leggendo e rispondendo, ovvero archiviando singolarmente ogni messaggio. Dopo aver composto (COMPOSE) o modificato (EDIT) un messaggio, è sufficiente dare il comando READ per visualizzarlo. Alcuni esempi di uso del comando READ sono i seguenti:
1) READ
2) READ 2
3) READ 5, 9
4) READ 5, REST
5) READ 2-7
6) READ ALL
7) READ TRANSITO, TEMP
Il primo comando viene impartito in apertura di sessione o dopo un comando di COMPOSE o EDIT.
Negli esempi 2, 3, 4, 5, 6, il sistema, dopo un comando di SCAN o SCAN più nome file, visualizza il o i messaggi appena analizzati e più precisamente:
2) visualizza il secondo
3) visualizza il quinto e il nono
4) visualizza prima il quinto e poi gli altri
5) visualizza dal secondo al settimo
6) visualizza tutti i messaggi
Il comando READ più nomi file visualizza, se codificato come nell'esempio 7, tutti i messaggi contenuti nel file TRANSITO e poi quelli contenuti nel file TEMP.
Se in apertura di sessione si batte il comando READ ed il file di sistema UNREAD non contiene messaggi, Infomak visualizzerà:

A SCAN NUMBER OR FILENAME IS REQUIRED

volendo significare che attende che sia battuto dall'utente:
a) il comando SCAN più nome file, e in seguito READ con accanto il numero del messaggio da leggere
b) il nome dell'archivio accanto al comando READ.

RETRIEVE
Si può abbreviare con il comando RET.
Tale comando permette di ricercare attraverso uno o più file i messaggi desiderati mediante l'indicazione di uno o più parametri contenuti nella testata.
La ricerca dei messaggi, conoscendo tutto o in parte il contenuto dei campi, si può eseguire sui campi:
— TO,CC e FROM
— SUBJECT, conoscendo parole o frasi in esso contenute
— MESSAGE ID, unico per messaggio
— DATE
Per quest'ultimo campo si può specificare la data di invio del messaggio se nota, oppure gli estremi di un periodo.
Dopo aver battuto il comando RETRIEVE vengono richiesti i seguenti parametri per la ricerca del messaggio:
1) FILENAME (S)
2) MESSAGE ID
3) TO
4) CC
5) FROM
6) DATE
7) SUBJECT
Parametro FILENAME: la ricerca si può fare su un solo file, su più file o su tutti i file dell'utente, battendo ALL. Si può anche ricercare un messaggio tra gli ultimi messaggi ricevuti battendo, accanto al nome file o ALL, la quantità.
Parametro MESSAGE ID: è unico per ogni messaggio inviato, ragion per cui, conoscendolo, Infomark non chiederà gli altri parametri cosicché la ricerca e la visualizzazione risulteranno immediati.
Esso ha la seguente formula:
GG-MMM-AA HH:MM:SS NNN N

GG-MMM-AA	: identifica il giorno, le prime tre lettere del mese e le ultime due cifre dell'anno dell'invio del messaggio;
HH:MM:SS	: identifica l'ora, i minuti ed i secondi dell'invio del messaggio;
NNN N	: è un codice unico, attribuito dal sistema ad ogni messaggio.

Parametri TO, CC, FROM: inserendo il nome di un utente in questi campi, verranno ricercati e visualizzati tutti i messaggi di quell'utente. Se si inseriscono più nomi, verranno ricercati e visualizzati tutti i messaggi che hanno uno solo di questi nomi nell'appropriato campo.
Parametro DATE: la ricerca del o dei messaggi sulla data può essere fatta in diversi modi. La data è introdotta in un'unica forma:

GG-MMM-AA

Se vi è necessità, i parametri da digitare prima di dare la data dell'invio del messaggio sono:
a) BEFORE
b) AFTER
c) BETWEEN
d) UNSENT
Si usa il formato BEFORE GG-MMM-AA per fare una ricerca tra i messaggi inviati prima di una certa data VV.
Si usa invece il formato AFTER GG-MMM-AA per fare una ricerca tra i messaggi inviati dopo una certa data VV.
Il formato BETWEEN GG-MMM-AA, GG-MMM-AA viene utilizzato per fare una ricerca tra i messaggi inviati fra una data e l'altra VV.
Infine si usa il formato UNSENT per fare una ricerca tra i messaggi che si trovano nel file di sistema UNSENT che non hanno il campo data poiché non sono stati inviati.
Parametro SUBJECT: questo campo si può riempire sia con una sola sia con più parole uguali a quelle contenute nel SUBJECT del messaggio da ricercare. È evidente che più sono le parole o le frasi, meno sono i messaggi visualizzati.
Specificando diverse parole o frasi, separate da una virgola, sono visualizzati tutti i messaggi che contengono una ed entrambe le parole o frasi specificate nel SUBJECT del comando RETRIEVE del messaggio da ricercare.

SCAN

Abbreviato con le iniziali SC, il comando elenca un sommario dei messaggi contenuti nel file UNREAD, visualizzando i dati più significativi, quali mittente, destinatario, oggetto e data.
Per analizzare gli altri file, occorre che il comando SCAN sia seguito dal nome del o dei file che si desidera visualizzare.
Se si volessero analizzare solo gli ultimi messaggi di uno, più o tutti i file, occorre battere anche la quantità dei messaggi che si desidera visualizzare.
Il comando SCAN, in caso di messaggio inviato a più utenti, visualizza solo il nome del primo destinatario del messaggio, facendolo precedere da TO; per i messaggi ricevuti, Infomark visualizza solo il nome dell'ultimo mittente.
I messaggi non inviati sono evidenziati dal display UNSENT nel campo della data; il prefisso "RE": è incluso prima del SUBJECT, se si tratta di un messaggio che è stato inviato per rispondere ad un altro/altri ricevuti.
Ecco alcuni esempi:
1) SCAN
2) SCAN TRANSITO
3) SCAN TRANSITO, TEMP, 8
4) SCAN ALL
5) SCAN ALL 5
Il comando del primo esempio analizza il file di sistema UNREAD.
Il comando SCAN TRANSITO analizza il file il cui nome è digitato accanto al comando (TRANSITO).
Il comando SCAN TRANSITO, TEMP, 8 permette con un unico comando di SCAN di analizzare più file. Nel nostro esempio verrà visualizzato prima il file TRANSITO, ed indi, gli ultimi otto messaggi contenuti nel file TEMP.
Il comando SCAN ALL visualizza tutti i file, analizzando tutti i messaggi in essi contenuti.
Il comando SCAN ALL 5 visualizza gli ultimi cinque messaggi di ogni archivio o file.
Quando è dato il comando SCAN privo del nome di un file, e il file UNREAD è privo di messaggi, Infomark visualizza

"YOUR UNREAD FILE IS EMPTY"

SEND

Il comando SEND trasmette uno o più messaggi a tutti i relativi destinatari. Infomark invia una sola copia del messaggio, anche nel caso in cui il nominativo del destinatario sia nei campi TO che CC o nelle liste di distribuzione.
Dando il comando SEND, Infomark riempie il campo FROM che si identifica con il nome del mittente, il campo DATE che si identifica con la data e l'ora di invio del messaggio, ed il campo "MESSAGE ID" che è composto dallo stesso contenuto del campo DATE, e dal codice unico per ogni messaggio inviato.
Quando si invia un messaggio, questo termina nel file UNREAD del destinatario e è posto, dal file UN SENT, in quello CHRONO del mittente.
Se si invia un messaggio precedentemente archiviato in un file, che non sia lo stesso file UNSENT, Infomark lo copia nel file CHRONO e non lo cancella come accade coi messaggi che sono archiviati solo in UNSENT, e questo perché ogni messaggio archiviato in un file utente e non ancora inviato è automaticamente contenuto nel file di sistema UNSENT.
È bene ricordare che dopo i comandi COMPOSE, ANSWER e FORWARD, Infomark invia i messaggi soltanto dopo aver ricevuto il comando SEND, altrimenti li archivia nel file di sistema UNSENT.
1) SEND
2) SEND 4
3) SEND 2-5
4) SEND 2, 5
5) SEND 5, REST
6) SEND ALL
Il comando SEND è da utilizzarsi come spiegato sopra.
I comandi 2, 3, 4, 5 devono essere usati dopo un comando SCAN più nome file.
Negli esempi sono inviati nell'ordine:
- il quarto messaggio
- dal secondo al quinto messaggio
- il secondo ed il quinto
- prima il quinto poi gli altri

Il comando SEND ALL è utilizzato per inviare tutti i messaggi contenuti nel file UNSENT.

I GIOCHI DI SIMULAZIONE

Il game riproduce la sua essenza, la trama celata si fa evidente, l'inganno è palese.

La digitalità elettronica si esalta in questa pratica del "come se": in un'ora si vola da Milano a New York ("Flight Simulator"), ma nello stesso tempo, per inesauribile contagio, si compie un viaggio al centro delle viscere umane ("Microsurgeon") o si completa un Gran Premio automobilistico ("Pole Position").
Simulare — con l'elaboratore, in questo caso — significa da tempo per l'uomo riprodurre artificialmente (in laboratorio, o in luoghi comunque attrezzati) condizioni identiche a quelle reali ma più "manovrabili", e perciò sotto controllo, ripetibili e verificabili.

Lezione 61

Ancora sui formati di stampa

Nell'ultima lezione abbiamo visto un programma che stampava il calendario di un mese, che leggeva i nomi dei giorni della settimana in forma estesa, e che poi li stampava usando un formato che ne visualizzava solo il carattere iniziale.
Perché allora leggere i nomi per esteso, se poi stampiamo solo il carattere iniziale?
Per avere programmi modificabili. Infatti, in seguito potremmo desiderare di stampare i nomi per intero, oppure un numero diverso di caratteri dall'inizio: il fatto di avere il nome per esteso e di poterlo visualizzare solo parzialmente ridurrà lo sforzo che dovremo fare in seguito per modificare il programma.
In particolare vediamo qui come è possibile stampare un certo insieme di caratteri iniziali di una stringa, con il comando di formato

"ç ç"

ove il numero di posizioni compreso tra i due caratteri "ç", questi ultimi inclusi, indica il numero di caratteri che vogliamo visualizzare dall'inizio della stringa.
Prendendo per esempio il programma precedente sulla stampa di un calendario mensile, osserviamo come è semplice modificare il programma per ottenere una stampa che visualizzi le prime tre lettere di ciascun nome del giorno:

```
5 DIM A$(7)
10 FOR I=1 TO 7
20 READ A$(I)
30 NEXT I
35 INPUT "Quale mese (1=gen,12=dic)";M
40 INPUT "Qual e' il giorno della settimana del
primo del mese (1=lun,7=dom)";N
50 G=31
60 IF M=4 OR M=6 OR M=9 OR M=11 THEN G=30
70 IF M=2 THEN G=28
90 LPRINT "Calendario del mese"
100 LPRINT
110 FOR I=1 TO G
120 IF N>7 THEN N=1
130 LPRINT USING "ç ç";A$(N);
140 LPRINT USING "£££";I
145 LET N=N+1
150 NEXT I
1000 DATA LUNEDI',MARTEDI',MERCOLEDI',GIOVEDI',
VENERDI',SABATO,DOMENICA
```

A fronte del nuovo formato inserito alla riga 130, al momento dell'esecuzione cui forniamo i seguenti dati:

```
run
Quale mese (1=gen,12=dic)? 2
Qual e' il giorno della settimana del pr
imo del mese (1=lun,7=dom)? 6
```

il programma fornisce il seguente risultato:

```
Calendario del mese

SAB  1
DOM  2
LUN  3
     •
     •
```

Approfondiamo ora il comportamento dei formati numerici.
Innanzitutto, cosa succede quando usiamo male un formato numerico? L'interruzione del programma sarebbe un po' punitiva: la stampa avviene visualizzando correttamente i valori, anche se il formato non può essere rispettato, e viene stampato il simbolo "%" prima del valore incriminato, evidenziando l'errore incontrato.
Per esempio, l'istruzione

```
10 LPRINT USING "£££";100*100
```

produce un risultato che non può essere ottenuto in tre posizioni e fornisce:

```
%10000
```

Ancora sul formato numerico: i valori vengono arrotondati! Così un'istruzione come la seguente:

```
10 LPRINT USING "£££";129.6
```

produce il risultato arrotondato al più vicino intero

```
130
```

e un'istruzione come

```
10 LPRINT USING "£££";165.2
```

produce il risultato, ancora arrotondato all'intero più vicino.

```
165
```

Di più, se l'arrotondamento produce un numero troppo grande rispetto al formato descritto, verrà fornito il valore arrotondato, con il messaggio d'errore, come è evidenziato dall'istruzione

```
10 LPRINT USING "£££";999.6
```

che produce il risultato

```
%1000
```

Vediamo altri formati. Il comando di formato

"+"

chiede che il segno venga evidenziato; in particolare, messo all'inizio del formato chiede che esso venga stampato all'inizio.
Per esempio, il seguente programma

```
10 REM Somma di N numeri
20 READ N
30 LPRINT "Somma di ";N;" valori"
40 LPRINT
45 LET S=0
50 FOR I=1 TO N
60 READ X
70 LET S=S+X
75 LPRINT TAB(10);
80 LPRINT USING "+£££";X
90 NEXT I
100 LPRINT
110 LPRINT "Somma:";
115 LPRINT TAB(9);
120 LPRINT USING "+£££££";S
200 DATA 10
210 DATA 12,-45,76,22,-44,-53,4,-9,10,9
```

richiede che venga stampata una serie di numeri con segno e la relativa somma. Si noti che il formato di ciascun numero occupa 5 posizioni globalmente con quella del segno; analogamente, la somma viene stampata su 6 posizioni. Il risultato è:

```
Somma di  10  valori

          +12
          -45
          +76
          +22
          -44
          -53
           +4
           -9
          +10
           +9

Somma:    -18
```

Se il segno viene inserito alla fine del formato, cambiando le linee 80 e 120:

```
80 LPRINT USING "+£££";X
120 LPRINT USING "+£££££";S
```

otteniamo il seguente risultato:

```
Somma di  10  valori

          12+
          45-
           •
           •
```

Mettere il segno alla fine non è poi così insolito: molti rapporti bancari lo fanno, anzi, spesso individuano solo con un segno "-" finale i valori negativi, quelli cosiddetti "in rosso". Ciò è possibile se si usa il comando di formato

"-"

al posto del "+" precedentemente adottato; cambiando ancora le istruzioni 80 e 120 nel programma precedente come segue:

```
80 LPRINT USING "+£££";X
120 LPRINT USING "+£££££";S
```

otteniamo proprio il risultato richiesto:

```
Somma di  10  valori

          12
          45-
          76
           •
           •
```

Nella prossima lezione completeremo il discorso sui formati di stampa.

Cosa abbiamo imparato

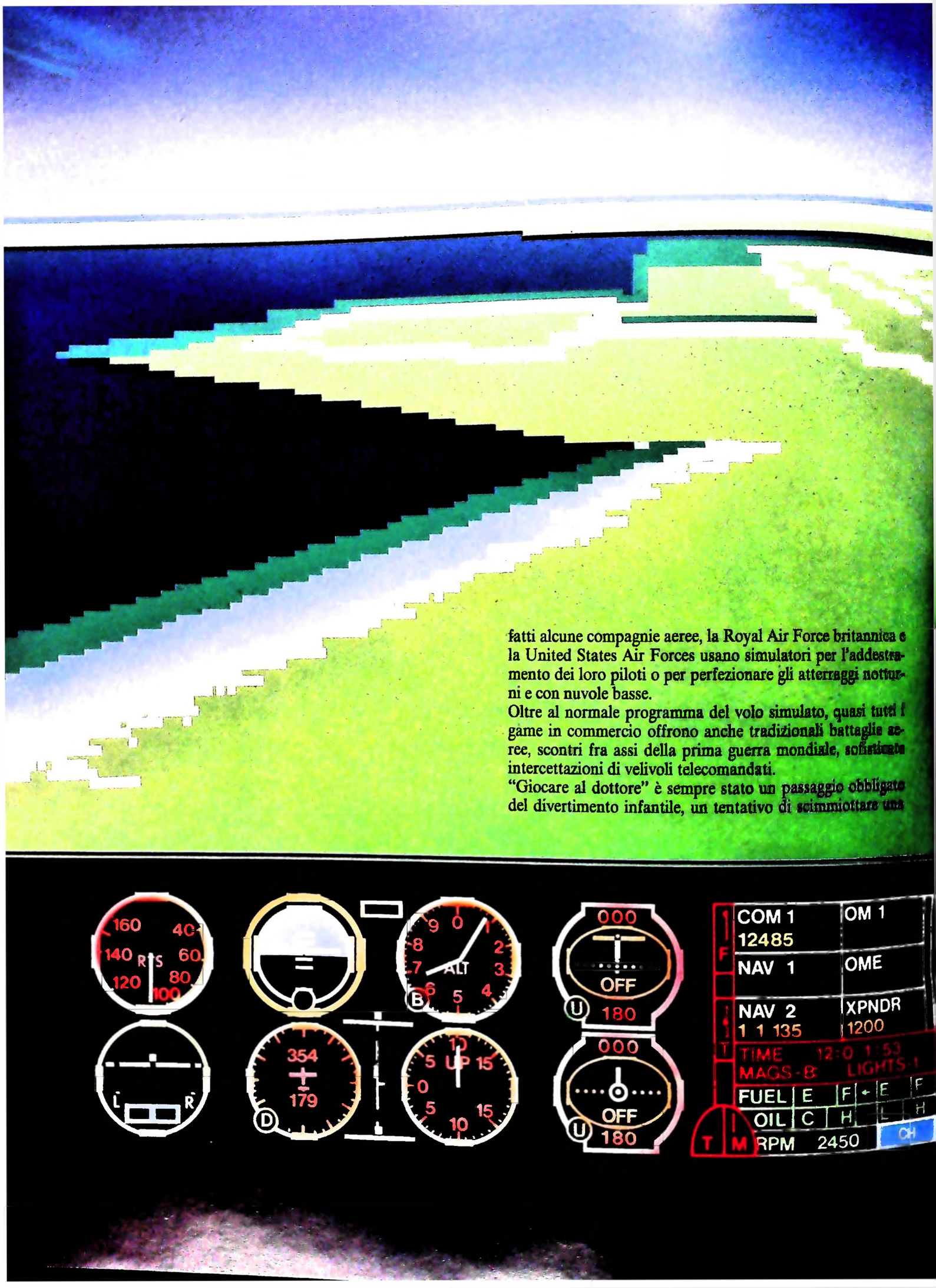
In questa lezione abbiamo visto:

- il comportamento del formato numerico in caso di errore di formato;
- il comportamento del formato numerico rispetto all'arrotondamento dei risultati;
- il comando di formato "ç ç", per la visualizzazione di una parte di stringa;
- il comando di formato "+" per visualizzare il segno all'inizio o alla fine del valore stampato;
- il comando di formato "-" per visualizzare esclusivamente il segno meno nei valori negativi, lasciando senza segno i positivi.

È per questo che i giochi di simulazione sono, in realtà, cose assai serie. Nati forse soltanto come game, offrono ora a specialisti — e le possibilità di applicazione saranno sempre più estese — condizioni sperimentali "asettiche", perciò ottimali.

Gioco e realtà

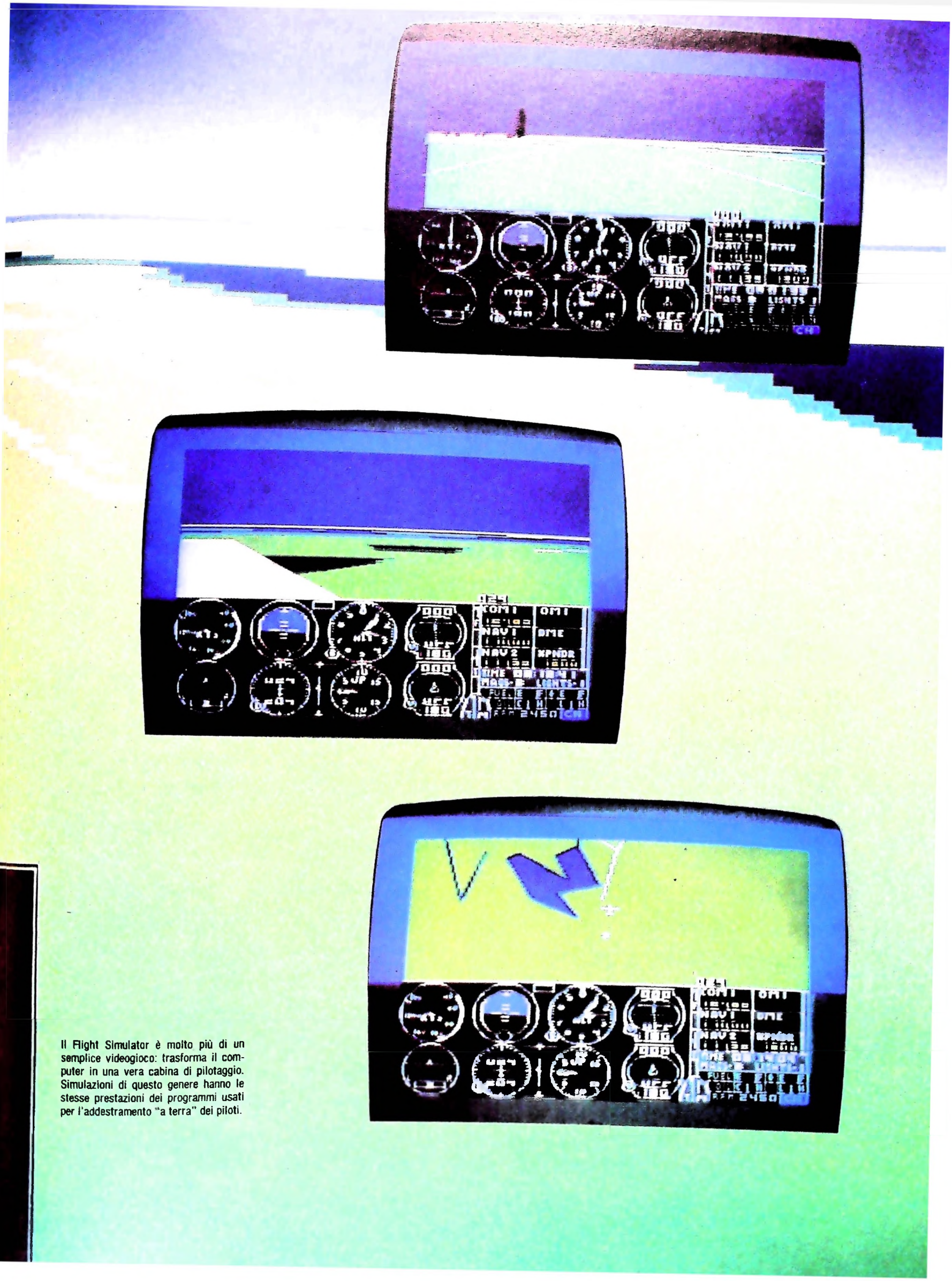
Con il "Flight Simulator" ci si trova ai comandi di un piccolo aereo da turismo, il Piper 181 Cherokee Archer. Sulla parte superiore dello schermo compare il paesaggio così come lo si vedrebbe dalla cabina di comando, mentre nella parte inferiore c'è tutta la strumentazione di un vero Piper: altimetro, orizzonte artificiale, una specie di radiogoniometro che fornisce la posizione dell'aereo rispetto ai quadranti di volo e una radio per le comunicazioni con la torre di controllo.
È tutto molto realistico: i comandi per azionare i flap, l'accensione delle luci di posizione, la discesa del carrello. Si hanno a disposizione quattro mappe del territorio degli Stati Uniti: una del Nord Est, una del Midwest, una del Nord Ovest e una del Sud Ovest. Una volta scelta la mappa, si può decollare da uno qualsiasi degli aeroporti della zona e dirigere verso l'aeroporto in cui si desidera atterrare. Il viaggio avviene in tempo reale con tanto di scenario che riproduce l'ora del giorno, stagione, situazione atmosferica (normale o particolarmente avversa, con nebbia, vento forte in quota).
Negli Stati Uniti i programmi di volo simulato stanno spiazzando i videogiochi classici, soprattutto perché "girano" sui personal e si rivolgono a un pubblico di professionisti. E in-



fatti alcune compagnie aeree, la Royal Air Force britannica e la United States Air Forces usano simulatori per l'addestramento dei loro piloti o per perfezionare gli atterraggi notturni e con nuvole basse.

Oltre al normale programma del volo simulato, quasi tutti i game in commercio offrono anche tradizionali battaglie aeree, scontri fra assi della prima guerra mondiale, sofisticate intercettazioni di velivoli telecomandati.

"Giocare al dottore" è sempre stato un passaggio obbligato del divertimento infantile, un tentativo di scimmiottare una



Il Flight Simulator è molto più di un semplice videogioco: trasforma il computer in una vera cabina di pilotaggio. Simulazioni di questo genere hanno le stesse prestazioni dei programmi usati per l'addestramento "a terra" dei piloti.

professione prestigiosa ma anche, come ci avrebbero poi spiegato gli psicologi, una prima, maldestra occasione per conoscere il proprio corpo.
Ebbene "Microsurgeon" va oltre, nel senso che il suo realismo è tale che potrebbe benissimo essere usato come test per accedere alla facoltà di Medicina. È comunque facile prevedere una sua applicazione a questo tipo di studi. Bisogna operare con una sonda che esplora minuziosamente il corpo del paziente: lo schermo segnala le aree di intervento immediato: tumori, ulcerazioni, calcoli, virus, infezioni batteriche, persino la tenia! Nella memoria del gioco ci sono ben 197 pazienti, compresi quelli che vengono direttamente dal Pronto Soccorso. Da far provare ai malati immaginari: o guariscono o si ammalano davvero.
Un vecchio sogno di Bernie Ecclestone, gran patron delle corse automobilistiche, per rendere più emozionante la ripresa televisiva delle gare di Formula 1, era quello di montare delle telecamere nell'abitacolo dei piloti. "Pole Position" (come quasi tutti gli altri giochi di simulazione delle corse) tenta di riprodurre questo effetto di immedesimazione, con in più una bella trovata: non è la solita gara di Gran Premio, ma la conquista dei posti per partecipare al Gran Premio. Uno sbaglio e si finisce fuori strada con la carcassa fumante della macchina. Per partecipare alla gara è dunque necessario stabilire prima un tempo al di sotto dei 73 secondi, corrispondente all'ottavo posto della griglia di partenza. Fissati i tempi, ha inizio la seconda parte del gioco; le invenzioni sceniche sono pura televisione. Si tratta di un game dotato di un realismo davvero eccezionale.
Simulazione è realtà liberata dalla necessità di essere vera.

Il programma

In questo programma viene rappresentato un semplice autodromo, che occupa tutto lo schermo ed è definito dalla matrice AU(x,y), che vale 0 se (x,y) rappresenta un tratto di pista percorribile, e 1 se invece è il bordo o il fuori pista, e in cui il giocatore controlla una macchina, stando attento a non farla uscire di strada e a non farla scontrare con il guard-rail. La macchina è dotata di velocità propria per cui il conducente controlla solamente la sua direzione (con i tasti "A" e "D" per girare rispettivamente a destra e a sinistra). La macchina viene rappresentata da 4 caratteri differenti ("<" ">" "^" "!"), a seconda della sua direzione corrente, per aiutare il conducente a prendere le decisioni corrette.

```
10  DIM AU(40,8)
12  mx=20 : my=6 : dd=1
20  FOR x=0 TO 39
25   FOR y=0 TO 7
28    AU(x,y)=0
30  NEXT y,x
32  FOR x=1 TO 38
35   AU(x,0)=1
37   AU(x,7)=1
40  NEXT x
42  FOR x=3 TO 36
45   AU(x,3)=1
47   AU(x,4)=1
50  NEXT x
52  FOR y=1 TO 6
55   AU( 0,y)=1
57   AU(39,y)=1
60  NEXT y
99  CLS
100 FOR y=0 TO 7
110  FOR x=0 TO 39
112  IF (x=39) AND (y=7) THEN GOTO 140
115   IF AU(x,y)=1 THEN 125
120    PRINT " "; : GOTO 130
125   PRINT "+";
130  NEXT x
140 NEXT y
150 x=mx : y=my : GOSUB 800
200 GOTO 900
400 REM legge i tasti
405 IF ky$="S" THEN 410
406 IF ky$="D" THEN 440
409 GOTO 490
410 dd=dd+1
415 IF dd>4 THEN dd=1
435 GOTO 490
440 dd=dd-1
445 IF dd<1 THEN dd=4
490 RETURN
700 PRINT CHR$(27)+"Y"+CHR$(y+32)CHR$(x+
32);
710 RETURN
750 REM cancella la macchina
755 IF x<0 OR x>39 THEN 770
756 IF y<0 OR y> 7 THEN 770
758 GOSUB 700
760 PRINT " ";
770 RETURN
800 REM scrive la macchina
801 IF x<0 OR x>39 THEN 830
802 IF y<0 OR y>7  THEN 830
805 GOSUB 700
810 ON dd GOTO 815,816,817,818
815 PRINT "<"; : GOTO 830
816 PRINT "^"; : GOTO 830
817 PRINT ">"; : GOTO 830
818 PRINT "!";
830 RETURN
900 REM programma principale
910 ky$=INKEY$
915 IF ky$<>"" THEN GOSUB 400
916 GOSUB 750 : REM cancella la macchina
920 ON dd GOTO 921,925,931,935
921 dx=-1 : dy=0  : GOTO 938
925 dx=0  : dy=-1 : GOTO 938
931 dx=1  : dy=0  : GOTO 938
935 dx=0  : dy=1  : GOTO 938
938 mx=mx+dx : my=my+dy
939 IF (mx <=1) OR (mx>=39) OR (my<=0) OR
(my>=7) THEN 990
950 x=mx : y=my : GOSUB 800
980 GOTO 900
990 BEEP
999 END
```

UNIX (III)

Analizziamo la "shell", la componente che funge da interfaccia tra il sistema e l'utente.

Che cos'è la shell

La shell è l'interprete dei comandi; funge da interfaccia tra il sistema e l'utente. La più conosciuta è la "Bourne shell" che prende il nome dal suo autore, o *sh*. Un'altra shell molto diffusa è la "C shell" o *csh*, sviluppata all'Università di Berkeley. Esiste un linguaggio di shell che prevede strutture di controllo, variabili e istruzioni di assegnamento, con cui si possono scrivere delle "procedure di shell", cioè un file contenente un insieme articolato di comandi che sono posti in esecuzione semplicemente richiamando il nome del file stesso. È inoltre la shell che si prende cura di gestire le redirezioni che possono essere operate al momento dell'esecuzione dei comandi, e di realizzare le pipeline richieste.

Come ci si collega

Seguiamo le operazioni che un utente compie volendosi collegare con il sistema. Il terminale dal quale si vuole effettuare il collegamento visualizzerà una richiesta di "login":

```
login:
```

L'utente deve digitare il proprio identificatore, che gli sarà stato attribuito dall'amministratore del sistema, e, nel caso fosse stata già definita una "password" segreta, questa gli sarà successivamente richiesta:

```
login: mario
Password:
```

Nel caso di identificatore e password corretti, comparirà l'indicatore "$" (detto *prompt*) della shell, che segnala che la shell è in attesa di un comando:

```
login: mario
Password:...
$
```

A questo punto, possono essere eseguiti i comandi che si desiderano. In generale, i comandi sono composti da una o più parole, separate da spazi: la prima è il nome del comando, le eventuali altre sono i relativi parametri (per esempio, il nome di uno o più file, oppure un insieme di opzioni costituite da "flag", cioè singoli caratteri alfabetici che specificano qualche richiesta particolare circa l'esecuzione del comando stesso). Per esempio:

```
$ fortune
Of all forms of caution, caution in love is the most fatal.
$
```

è l'esecuzione del comando fortune, senza alcun parametro. Tale comando fornisce una massima famosa, scelta a caso. Oppure:

```
$ who am I
mario  tty7  Maj 23 16:23
$
```

è l'esecuzione del comando who che, con i due parametri "am I", fornisce il nome dell'utente collegato, il suo terminale, la data e ora di collegamento.
Per uscire dalla shell, e interrompere così la sessione di lavoro, è sufficiente digitare il carattere di fine file, o CTRL D.
Quando l'utente si collega per la prima volta, al proprio identificatore non è associata alcuna password. È l'utente stesso, infatti, che può scegliere di introdurre o cambiare la propria password, mediante l'apposito comando passwd:

```
$ passwd
Changing password for mario
Old password: ...
New password: ...
Retype new password:...
$
```

Questa è l'intera sequenza che comporta la modifica della propria password.

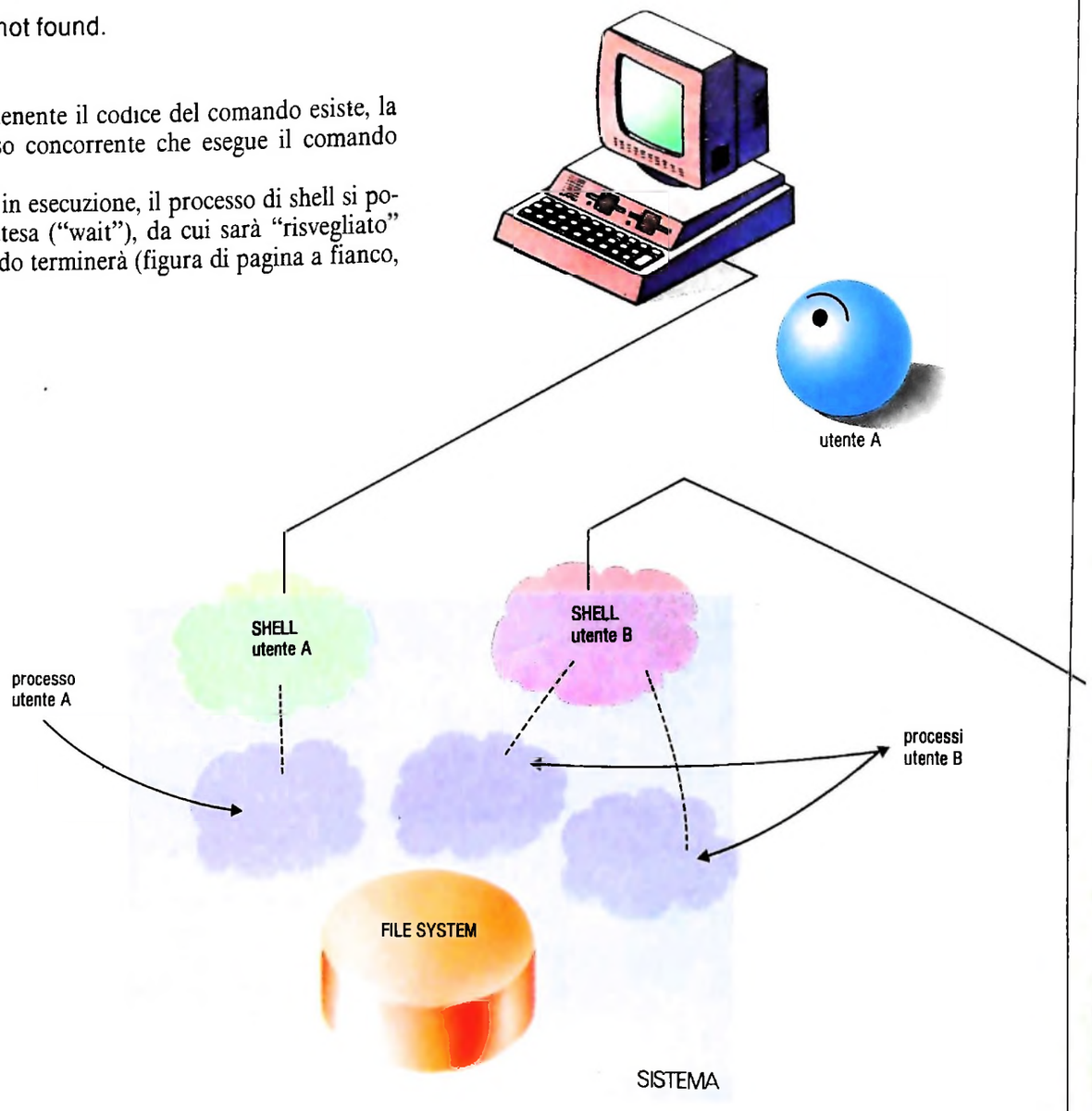
L'esecuzione dei comandi: i processi

Consideriamo, in sintesi, come la shell è in grado di porre in esecuzione un programma richiesto dall'utente (figura di pagina seguente). A ciascun comando corrisponde, in una direttrice del file-system, un file avente lo stesso nome del comando, contenente il programma che lo realizza. La shell è in grado di ricercare questo file automaticamente, senza cioè che l'utente debba preoccuparsi di specificare l'intero, e talvolta complesso, pathname del programma relativo. Se que-

sto file non esiste, viene segnalato l'errore:

```
$ passwk
passwk: Command not found.
$
```

Se, invece, il file contenente il codice del comando esiste, la shell crea un processo concorrente che esegue il comando stesso.
Mentre il comando è in esecuzione, il processo di shell si pone in uno stato di attesa ("wait"), da cui sarà "risvegliato" non appena il comando terminerà (figura di pagina a fianco, in alto). Questa modalità di esecuzione è detta *foreground*.



Questo modo di eseguire un comando non è, però, sempre conveniente. Supponiamo, per esempio, di voler stampare il contenuto di un file con il comando pr, dirottandone lo standard output sulla stampante identificata dal 'device' /dev/1p. Il comando:

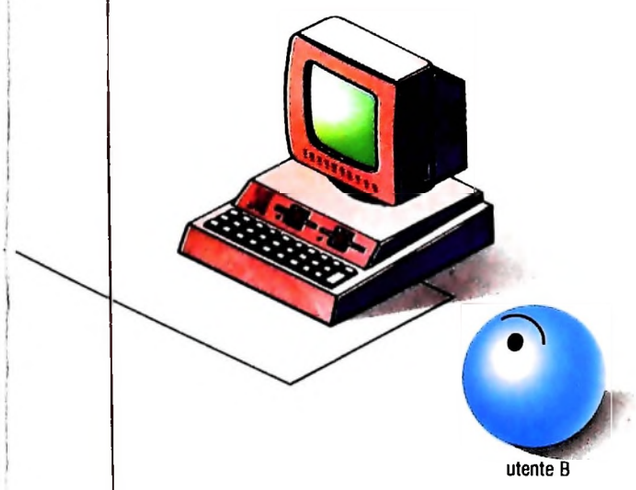
```
$ pr file > /dev/1p
$
```

richiederà un certo tempo per la sua esecuzione, che può essere notevole se il file richiesto è molto grande. Mentre la stampante è in funzione, la shell è in attesa che il comando termini, per cui l'utente non è in grado di svolgere dal terminale alcuna altra operazione.
Volendo, invece, svolgere altre attività la shell consente l'esecuzione dei programmi in modalità *background* (figura della pagina a fianco, in centro): la shell, dopo aver generato il processo figlio è in grado di interpretare altri nuovi comandi, senza attendere la terminazione del figlio. In questo modo l'utente eseguirà contemporaneamente due processi (figura della pagina a fianco, in basso): quello della shell e quello del comando.
La shell riconosce la volontà dell'utente di voler eseguire un

comando background dal carattere "$" posto in coda al comando; così l'esempio precedente diventa:

```
$ pr file > /dev/1p$
3247
$
```

Il numero visualizzato dalla shell (nell'esempio, 3247) è un identificatore numerico, con il quale il sistema distingue i processi.
L'utente è in grado in ogni momento di conoscere lo stato dei processi tramite il comando ps (process status) che forni-

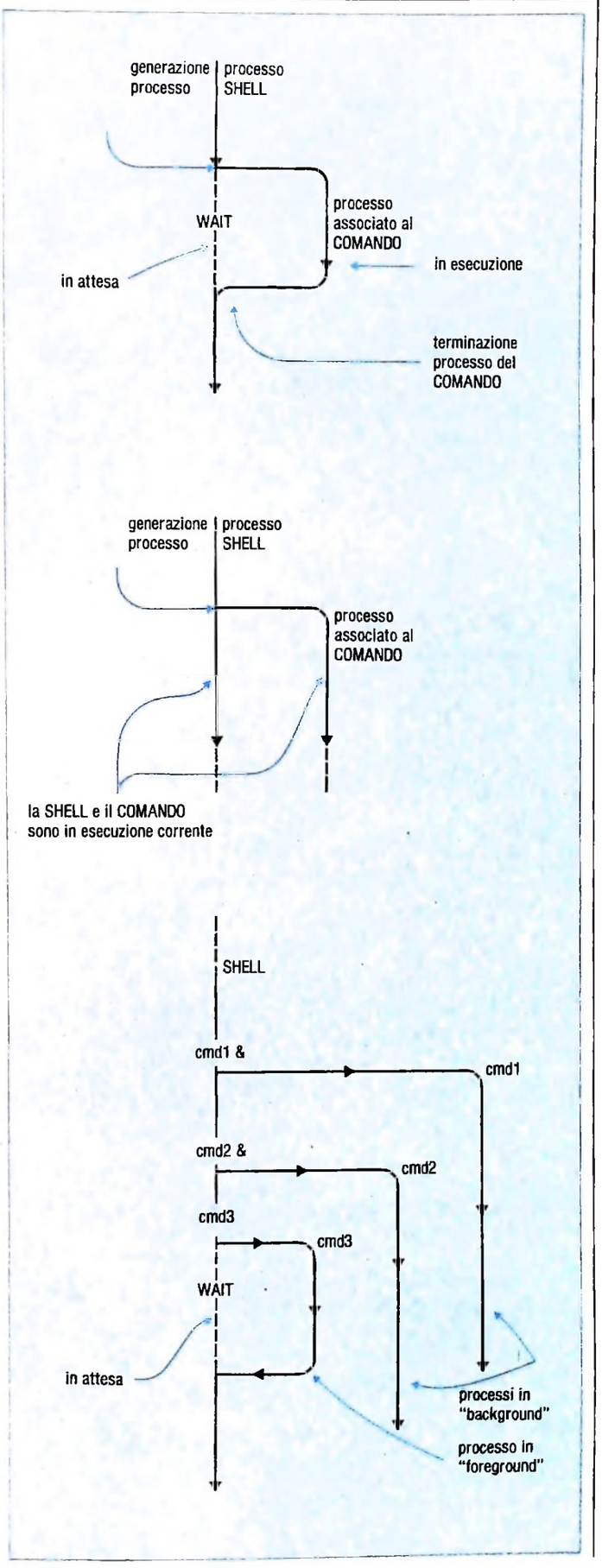


Nell'illustrazione centrale è schematizzato il ruolo della shell nell'interazione fra l'utente e il sistema. A destra, in alto: esecuzione di un comando in "foreground". Al centro: esecuzione di un comando in "background". In basso: un esempio della dinamica dei processi "foreground" e "background".

sce l'elenco dei processi richiesti; se è in corso un processo di stampa eseguito in background si può ottenere:

```
$ ps
PID TTY      TIME CMD
3218 7       2:37 - sh
3417 7       0:12 pr file > /dev/1p$
3441 7       0:03 ps
$
```

dove nella prima colonna appaiono gli identificatori numerici dei processi (PID sta per "Process Identifier"), nella secon-



da (TTY) il numero del terminale dal quale sono stati mandati in esecuzione, nella terza (TIME) il tempo di esecuzione e in ultima (CMD) il nome del comando che il processo sta eseguendo. Così, nell'esempio, si vedono tre processi in esecuzione: il processo di shell sh, il processo di stampa pr, e il processo che esegue il comando ps stesso.

Le pipeline

La shell, interpretando il carattere di "pipe" ":", riconosce la volontà dell'utente di eseguire concorrentemente due programmi e di voler redirigere lo standard output del primo nello standard input del secondo.
Consideriamo, per esempio, il comando who, che produce un elenco di tutti gli utenti collegati, con il nome del terminale che stanno utilizzando e la data e ora di collegamento:

```
$ who
lucia    console Maj 23 13:39
mario    tty7    Maj 23 16:23
franco   tty6    Maj 23 12:52
```

Se vogliamo contare quante linee, parole e caratteri sono prodotti possiamo realizzare la seguente "pipeline":

```
$ who : wc
3     15          84
$
```

Il comando who ha prodotto l'elenco di tutti gli utenti collegati al sistema, che è stato rediretto sullo standard input del comando wc (word count), che ha contato 3 linee, 15 parole e 84 caratteri. La shell, in questo caso, ha eseguito due processi concorrenti distinti: il primo associato a "who", il secondo a "wc" e li ha posti in comunicazione, in modo da consentire a "wc" di ricevere l'output prodotto da "who".

La C shell

Tra le numerose shell che un utente può decidere di utilizzare, ve ne è una che raccoglie sempre più le simpatie degli utenti di Unix: la C shell. Il motivo di tale successo risiede nel tipo di interfaccia offerta, caratterizzata da una sintassi simile a quella del linguaggio di programmazione "C" (da cui il nome), e da un insieme di meccanismi offerti per facilitare e semplificare il lavoro dell'utente. In particolare, la C shell conserva la storia dei comandi eseguiti dall'utente, consentendone la riesecuzione (eventualmente con delle modifiche) senza necessità di ribatterli. È pure consentito all'utente di ridefinire i nomi dei comandi con nomi sintetici scelti appositamente; ovvero è possibile definire degli "alias" relativamente a quei comandi che l'utente è solito utilizzare con maggiore frequenza. Anche nella C shell, come nella shell sh, sono utilizzati i metacaratteri che danno luogo alla generazione dei nomi dei file, permettendo di specificare con un identificatore unico un insieme di file i cui nomi seguono un "pattern" comune.

Glossario

CP/M - *sigla per Control Program for Microprocessors (programma di controllo per microprocessori), indica il più famoso tra i sistemi operativi realizzato dalla Digital Research statunitense nel 1974 per calcolatori personali. La versione originale era per sistemi a otto bit basati sul microprocessore Z80, uno dei migliori presente sul mercato, ma ne è stata realizzata una versione (CP/M 86) anche per sistemi a 16 bit.*

FORTH - *linguaggio di programmazione formulato negli anni Sessanta da Charles Moore, con caratteristiche di linguaggio intermedio, cioè relativamente vicino all'hardware della macchina, inteso a rendere massima la produttività dei programmatori, e orientato in particolare alla programmazione di sistemi di controllo. Il suo concetto centrale è quello di stack (catasta o pila): il linguaggio, molto flessibile nelle applicazioni, configura un calcolatore virtuale di cui lo stack è l'elemento fondamentale.*

MP/M - *sigla per "Multiprogramming control Program for Microprocessors" (programma di controllo di multiprogrammazione per microprocessori), indica un sistema operativo realizzato dalla Digital Research, derivato dal CP/M, ma indirizzato alla gestione di sistemi multiutente, in cui cioè al sistema di calcolo sono collegati più terminali, attivi simultaneamente.*

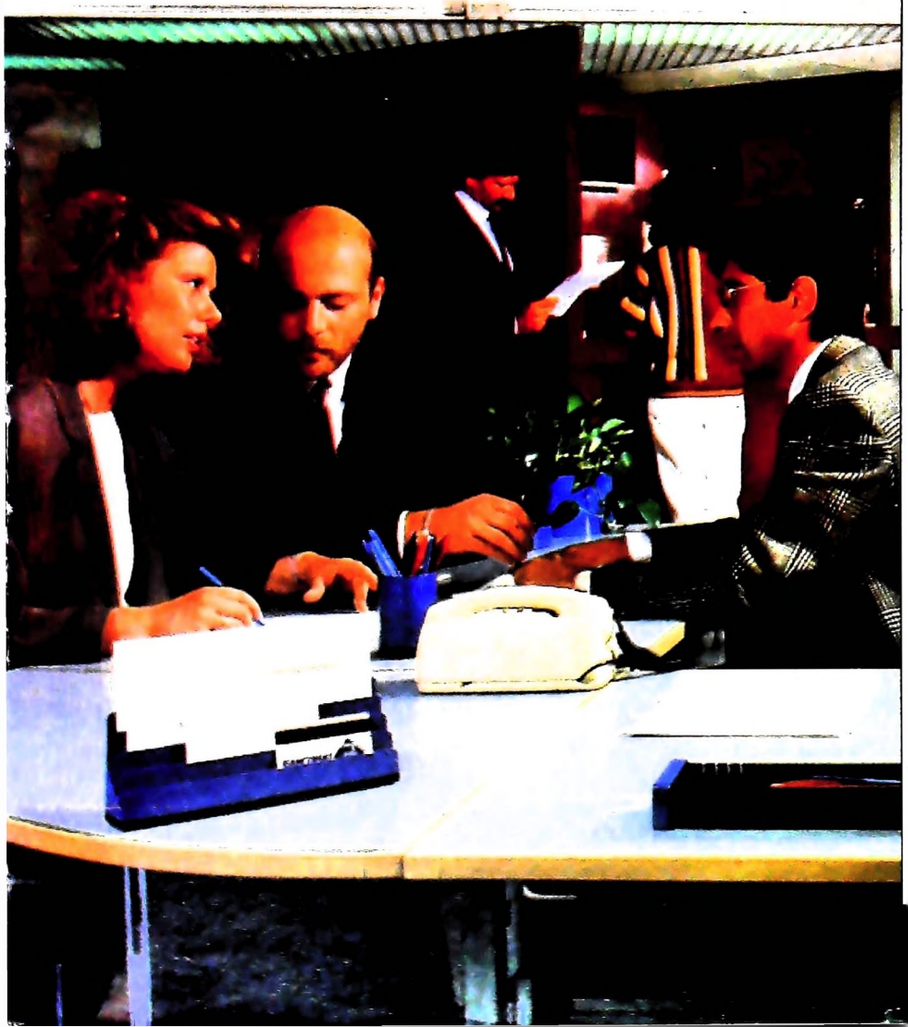
Semantica - *quella parte dello studio di un linguaggio che si occupa del significato degli elementi, semplici e complessi, del linguaggio stesso, ovvero della relazione che i segni linguistici hanno con il mondo.*

Sintassi - *quella parte dello studio di un linguaggio che si occupa esclusivamente dei segni fondamentali del linguaggio stesso e del modo in cui si combinano a formare elementi più complessi (parole, frasi, periodi o, nel caso di linguaggi artificiali come quelli di programmazione, istruzioni, comandi, programmi), prescindendo dal loro significato.*

Subroutine - *una serie di istruzioni che dice all'elaboratore come svolgere un determinato compito, considerata come un insieme a se stante, utilizzabile all'interno di altri programmi. Il termine può essere tradotto come "sottoprogramma" e identifica un programma (non necessariamente breve) che compare all'interno di altri programmi, e si distingue dal programma principale perché deve possedere un'istruzione che restituisce il controllo al programma principale, una volta esaurito il suo compito.*

UN NUOVO MODO DI USARE LA BANCA.

Conto corrente più



TANTI PENSIERI IN MENO CON IL CONTO CORRENTE "PIU'" DEL BANCO DI ROMA.

Essere cliente del Banco di Roma vuol dire anche essere titolari del conto corrente "più". Un conto corrente più rapido: perché già nella maggior parte delle nostre filiali trovate gli operatori di sportello che vi evitano le doppie file.

Più comodo, perché potete delegare a noi tutti i vostri pagamenti ricorrenti: dai mutui all'affitto, dalle utenze alle imposte.

Più pratico, perché consente l'utilizzo del sistema di prelievo automatico Bancomat e l'ottenimento della carta di credito.

Inoltre un servizio utilissimo, soprattutto per imprenditori e commercianti denominato "esito incassi", consente di avere comunicazione dell'eventuale insolvenza entro solo cinque giorni dalla scadenza. Una opportunità veramente speciale.

Più sicuro, perché con una minima spesa potrete assicurarvi contro furti e scippi mentre vi recate in banca o ne uscite.

Veniteci a trovare, ci conosceremo meglio.

BANCO DI ROMA
CONOSCIAMOCI MEGLIO.

Olivetti M10 vuol dire disporre del proprio ufficio in una ventiquattrore. Perché M10 non solo produce, elabora, stampa e memorizza dati, testi e disegni, ma è anche capace di comunicare via telefono per spedire e ricevere informazioni. In grado di funzionare a batteria oppure collegato all'impianto elettrico, M10 mette ovunque a disposizione la sua potenza di memoria, il suo display orientabile a cristalli liquidi capace anche di elaborazioni grafiche, la sua tastiera professionale arricchita da 16 tasti funzione.

Ma M10 può utilizzare piccole periferiche portatili che ne ampliano ancora le capacità, come il microplotter per scrivere e disegnare a 4 colori, o il registratore a cassette per registrare dati e testi, o il lettore di codici a barre. E in ufficio può essere collegato con macchine per scrivere elettroniche, con computer, con stampanti. Qualunque professione sia la vostra, M10 è in grado, dovunque vi troviate, di offrirvi delle capacità di soluzione che sono davvero molto grandi. M10: il più piccolo di una grande famiglia di personal.

PERSONAL COMPUTER OLIVETTI M10

L'UFFICIO DA VIAGGIO

Anche in leasing con Olivetti Leasing.

olivetti

Per informazioni rivolgersi a negozi contrassegnati da Olivetti M10 Punto di Vendita o inviare il coupon a Olivetti Divisione Personal Computer, Via Meravigli 12, 20123 Milano

NOME-COGNOME

VIA/N

CAP CITTÀ

TELEFONO