a cura di Valter Di Dio

WarBot Arena

di Andrea Nini - Modena

Il programma di questa volta, di cui non è possibile pubblicare il listato perché lungo quasi 70.000 caratteri, è sostanzialmente un gioco. Ma è un gioco piuttosto anomalo, infatti non si gioca in due, né contro il computer, non si usa il joystick e nemmeno la tastiera; per giocare si deve anche essere abili programmatori e, una volta fatte le operazioni preliminari, non resta che stare a guardare cosa accade. Ma allora che gioco è? La versione presentata dal lettore si chiama WarBot ed è una versione per MS-DOS di un gioco, Robot-War, che era uscito molti anni fa per Apple II. La filosofia è la stessa, si devono allestire e programmare dei robot da mandare poi in battaglia da soli. Se il programma è valido, o almeno è migliore di quello del nemico, si vincerà lo scontro, altrimenti si perde e converrà mettere mano al programma del robot per evitare altre brutte figure. Nella versione per Apple i robot erano molto meno sofisticati degli attuali, ma si potevano avere più robot in campo anche di tipi differenti; in questa versione i robot sono solo due e questo porta spesso a partite un po' monotone. Interessante è comunque la possibilità di scambiarsi i robot che sono dei semplici file di testo, magari per via telematica, e quindi di organizzare dei tornei di WarBot un po' come accade per Core Wars. Alzate gli scudi, dito sul grilletto e occhio al radar...

È disponibile, presso la redazione, il disco con il programma presentato in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 279.

Istruzioni

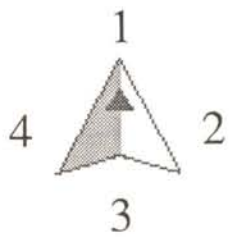
Nell'articolo di introduzione alle Core Wars nel numero 67 di MCmicrocomputer si è parlato del programma RobotWars, in cui si devono programmare ed equipaggiare robot e farli combattere tra loro: questo concetto di gioco mi ha interessato ed ho provato a scrivere io stesso un programma ispirato a questa struttura di gioco; da qui è nato WarBot Arena. Anche in questo programma lo scopo è quello di programmare e equipaggiare due robot, chiamati WarBot, e farli combattere in un'arena, il linguaggio utilizzato per questo scopo è il WBCode: un linguaggio assimilabile a quelli ad alto livello: infatti prende spunto dal Logo, dal Pascal e dal Basic, e per questo si differenzia in modo sostanziale dal Redcode, paragonabile ad un Assembler; anche la visualizzazione dello scontro è sostanzialmente diversa, in questo programma si possono vedere WarBot stilizzati che si muovono su di una scacchiera al posto delle astratte matrici di memoria di Core Wars. Il programma è stato scritto e compilato in Turbo Pascal 5.0, il listato è stato suddiviso in due parti: nel programma principale di nome wbarena.pas, e in una parte caricata dal primo programma come modulo include, di nome wbaren2.pas.

Caratteristiche di WarBot Arena

Lo scontro si svolge in una griglia di 256 caselle (16 x 16) in cui all'inizio vengono disposti casualmente 16 ostacoli, rappresentati da piramidi, le quali sono, come le pareti dell'arena, impassabili e indistruttibili.

I due WarBot vengono disposti all'inizio in posizioni casuali a più di cinque caselle di distanza e su righe e colonne differenti, dato che questa posizione di partenza è casuale e che tutte le istruzioni di movimento sono date in forma soggettiva, ai WarBot non è dato rendersi conto della loro posizione all'interno dell'arena. I WarBot vengono letti da disco, devono essere perciò file scritti in formato puro ASCII con estensione *.wbc, le prime venti righe del file verranno interpretate come i codici dei componenti, uno per riga, questi codici sono formati da tre lettere e verrano esposti nella sezione Componenti; le righe seguenti verranno lette e memorizzate come programma, tra le due sezioni non devono essere inseriti separaratori di nessun tipo, nelle righe non ci devono essere spazi iniziali, non sono cioè permesse le indentature. La lunghezza massima di un programma è di 100 righe, quelle in soprannumero non verranno considerate facendo terminare il programma alla centesima riga, il programma deve essere senza numeri di linea.

I numeri da uno a quattro rappresentano i codici delle direzioni adoperate con le funzioni di sensore e con WHIT, si fa notare che la direzione uno non indica il Nord, ma la punta del WarBot, il due la sua destra, il tre la direzione contraria alla punta e il quattro la sua sinistra, in modo da rimanere sempre relative e non dare un riferimento assoluto.



L'interprete WBCode esegue una istruzione da ogni programma in ogni ciclo di gioco, il parallelismo è simulato: in realtà si parte con una istruzione del White WarBot, una del Black WarBot e così via, per cui per poter stabilire un vincitore fra due combattenti, si consiglia di ripetere lo scontro almeno una seconda volta a colori invertiti; il parallelismo rimane anche con l'utilizzo dell'istruzione PARALLEL (vedi sezione Comandi), con esse l'esecuzione del programma normale viene alternata con quella delle istruzioni poste dopo le Parallel. I WarBot eseguono il programma fino a quando hanno energia, se essa si esaurisce il WarBot si blocca, ma rimangono attive le mine, le mine a tempo, le caselle grease, le centrali e le unità vampiro: può quindi accadere che con le centrali o le unità vampiro il WarBot riacquisti energia e si rimetta in funzione, con l'istruzione successiva all'ultima eseguita; durante l'esecuzione dello scontro si può controllare l'energia attraverso due barre sul fondo dello schermo. Quando l'energia arriva a zero, viene disattivato lo scudo corrente, e non verrà riattivato in nessun caso. Non ci sono limiti di tempo, lo scontro può terminare con la distruzione di uno o di tutti e due i WarBot, con l'esaurimento dell'energia di tutti e due, oppure alla pressione di un tasto qualsiasi, nel qual caso verrà rilevato se uno dei due ha esaurito l'energia, e quindi arrivato a una vittoria parziale. Un WarBot viene distrutto quando subisce un danno, non ha scudi attivi e ha esaurito tutte le placche protettive.

Uso del programma interprete

Dopo la presentazione verrà chiesto il path da seguire per cercare il White WarBot, dopodiché verranno mostrati sullo schermo tutti i programmi WarBot localizzati nella directory indicata e si dovrà selezionare quello desiderato con i tasti cursore più il tasto RETURN (premendo il tasto **p** si ritorna alla richiesta della directory), poi si eseguirà la stessa operazione per il Black WarBot; è possibile scegliere lo stesso WarBot per i due colori. Dopo il caricamento verranno visualizzate tre schermate per ogni parte, queste mostreranno l'equipaggiamento, le caratteristiche e il programma, poi verrà visualizzata l'arena e la pressione di qualsiasi

tasto farà partire il combattimento. Alla fine di ogni scontro sarà presentata una classifica in ordine di punteggio, assegnato nella misura di 4 per vittoria per distruzione, 3 per vittoria per energia, 2 per il pareggio, 1 per sconfitta per energia e 0 per sconfitta per distruzione. Le colonne mostrano, in ordine, nome del WarBot, punteggio, partite giocate, vinte per distruzione, vinte per energia, pareggiate, perse per energia e perse per distruzione; dopo la pressione del tasto SPAZIO si arriverà all'ultimo menu che offre sei scelte selezionabili attraverso la pressione dei seguenti tasti funzione:
F1: viene rigiocata la partita con gli stessi WarBot.
F2: viene rigiocata la partita con gli stessi WarBot a colori invertiti.
F3: vengono richiesti i nomi dei WarBot e viene giocata una nuova partita.
F4: viene azzerata la classifica (un beep indica l'avvenuta cancellazione).
F5: vengono attivate/disattivate le barre dell'energia (un beep indica l'operazione).
F6: uscita dal programma.

Procedure

F O R W A R D a

Questa istruzione fa avanzare il WB di **a** caselle, il massimo di caselle percorribili in un round è di 21 meno il peso del WB:
— nel caso in cui **a** sia maggiore di questo numero, la distanza percorsa viene ridotta al massimo possibile e ritorna il messaggio OVERMOVE ERROR;
— nel caso in cui il movimento porti il WB oltre i confini dell'arena, nella casella contenente un ostacolo, l'altro WB, una ragnatela, oppure in caso di paralisi (vedi componente Paraser) il movimento viene interrotto nell'ultima casella utile e viene emesso il messaggio IMPMOVE ERROR.

Il costo per il movimento è di 1 punto d'energia per ogni casella, gli errori nel movimento non diminuiscono l'energia. Se il WB si porta in una casella contenente una qualsiasi mina ne subisce l'esplosione, se cade dentro una casella perforata da una trivella viene automaticamente distrutto, se arriva in una casella che è stata resa scivolosa dal componente greaser esso scivola nella casella successiva, a meno che non sia a ridosso di un ostacolo come WarBot e piramidi, nel qual caso non scivola e si ferma in quella casella.

B A C K W A R D a

Questa istruzione è del tutto simile alla FORWARD, l'unica differenza è che il WB viene fatto indietreggiare invece che avanzare, sempre di **a** caselle.

R I G H T a / L E F T a

Queste istruzioni fanno ruotare il WB rispettivamente a destra e a sinistra di **a** angoli retti, il massimo della rotazione è anch'essa di 21 meno il peso del WB, nel caso di superamento ritorna il messaggio OVERTURN ERROR, se la rotazione viene tentata quando il Warbot è paralizzato, viene emesso l'errore IMPTURN. Il costo di rotazione per ogni 90 gradi è di 1 punto di energia.

Si prega il Sig. Giancarlo Del Sordo, autore del programma Shape Tool-Kit pubblicato sul n. 87, di mettersi in contatto con la redazione per comunicare i dati anagrafici.

```
Sorgente del Warbot IMP

ctr
ctr
ctr
ctr
ctr
ctr
ctr
ctr
ctr
ctr
pla
emp
emp
emp
emp
emp
emp
emp
emp
emp
right rnd(4)-1
forward rnd(21-wght)
jump -2

Fine del sorgente
```

T U R N a

Se l'istruzione TURN ha argomento positivo, viene trattata come RIGHT, se è negativo, viene trattata come LEFT, tutte le annotazioni viste per questi due comandi valgono anche per TURN.

I F a > < = @ b THEN comando ELSE comando

Questa istruzione simula abbastanza fedelmente il suo equivalente Basic, vengono valutate le espressioni **a** e **b** e, a seconda dell'operatore presente al loro interno, viene eseguita l'istruzione presente dopo il THEN o dopo l'ELSE, quest'ultimo è facoltativo. I simboli >, < e = sono trattati nella loro normale accezione, ma non è possibile combinarli. Il simbolo @ identifica il diverso al posto dell'usuale simbolo '<>'.

N.B.: Devono esserci degli spazi tra IF e **a**, tra **b** e THEN, tra THEN e la prima istruzione, tra questa e ELSE, tra ELSE e la seconda istruzione. Non garantisco che vengano correttamente interpretati IF..THEN..ELSE nidificati, se però sono indispensabili consiglio di sostituirli con delle chiamate a subroutine.

Esempi
CORRETTO: if rad(1)+[0]@0 then call 3 else jump 2
CORRETTO: if son(0)>2 then right 2
ERRORE: if [0]>=3 then loop (non è permesso il >=)
ERRORE: if wght=10 else activate 1 (non compare il THEN)

D O a : b / L O O P

Queste due istruzioni servono per creare dei cicli, la prima istruzione seguita da due espressioni inizia un ciclo ed inizializza un contatore al valore **a**, ciclo che viene ripetuto all'istruzione LOOP fino a quando il contatore, incrementato di 1 ad ogni passaggio, raggiunge il valore della seconda espressione; queste due espressioni possono mancare del tutto, e in questo caso il ciclo viene eseguito infinite volte. Il segmento di istruzioni all'interno del ciclo viene eseguito sempre almeno una volta, anche quando **a** è maggiore di **b**. Le due espressioni devono essere separate dai ':'.

I cicli possono essere nidificati uno nell'altro, fino ad un massimo di 10, e il contatore può essere letto da programma attraverso l'espressione {**f**} dove f rappresenta il grado di profondità, a partire da zero. Se viene aperto un ciclo quando ci sono già altri dieci cicli aperti ritorna DO ERROR.

N.B.: Non c'è limite al numero di cicli complessivi nel programma, purché non ve ne siano mai più di dieci aperti contemporaneamente.

Esempio
Questo programma scrive la sequenza 1 2 per dieci volte:
do 1:10
do 1:2
write {1}
loop
loop

G O T O a

Questa istruzione provoca un salto alla linea di programma numero **a**, nel caso in cui **a** è minore di 1 o maggiore della lunghezza del programma ritorna l'errore GOTO ERROR e non viene eseguito alcun salto.

J U M P a

Con questa istruzione l'esecuzione del programma salta, se **a** è positiva a linee in avanti, se **a** è negativa a linee all'indietro. Se si ottengono destinazioni al di fuori del programma ritorna un JUMP ERROR e non viene eseguito alcun salto.

N.B.: Con JUMP 0 il programma ripeterà all'infinito questa istruzione.

Esempio
I due programmi che seguono sono uguali:
1) goto 2 1) jump 1
2) goto 4 2) jump 2
3) goto 5 3) jump 2
4) goto 3 4) jump -1
5) goto 5 5) jump 0

A C T I V A T E a

Serve per adoperare il componente situato nel settore **a** del WarBot, se esso non è un componente attivabile ritorna ACTIVATE ERROR.

I componenti attivabili sono:
LASER 1-2-3
MISSILE 1-2-3
MINA
MINA TEMPO
TRIVELLA
CHAOTIZER
SHOCKER
PARASER
MORTAR 1-2-3
GREASER
STEALTHER
WEB

I componenti non attivabili sono:
SCUDO 1-2-3
RADAR
INFRARED
SONAR
BATTERIA
CENTRALE
VAMPIRO
PLACCA
COMPUTER

L E T a = b

Questa istruzione pone nella variabile numero **a** il valore ritornato dall'espressione **b**, se **a** esce dai limiti 0 e 99, viene riportata dentro questo intervallo.

S U B R a/E N D a/C A L L b

I primi due comandi servono ad indicare l'inizio e la fine della subroutine numero **a**, il comando CALL serve per richiamare la subroutine numero **b**; **a** e **b** devono andare da 1 a 10, una subroutine può richiamare un'altra subroutine ma non se stessa, non può cioè essere ricorsiva; nel caso questo avvenisse, ritornerebbe il messaggio CALL ERROR, RETURN ERROR ritorna invece se l'interprete incontra un END **a** senza aver prima eseguito il CALL **a**.

N.B.: Il punto di partenza di una subroutine viene individuato in pre-esecuzione, non durante l'esecuzione del programma. Se l'interprete incontra durante il programma il comando SUBR lo tratta come se fosse un WAIT e non emette alcun messaggio d'errore.

T I M E R a

TIMER serve per cambiare il ritardo dell'esplosione dopo la deposizione di una mina a tempo, di default esso è di 100 cicli, cioè la mina a tempo esplode dopo che l'interprete ha eseguito un centinaio di righe del programma.

N.B.: I cambiamenti hanno effetto a partire dalla prossima mina deposta, non su quella attuale.

W A I T

Fa passare un ciclo senza fare niente. Può essere usata per scrivere commenti dato che l'argomento viene ignorato dall'interprete.

TABELLA
CODICI COMPONENTI

0)	VUOTO	emp
1)	LASER 1	ls1
2)	LASER 2	ls2
3)	LASER 3	ls3
4)	MISSILE 1	ms1
5)	MISSILE 2	ms2
6)	MISSILE 3	ms3
7)	SCUDO 1	sh1
8)	SCUDO 2	sh2
9)	SCUDO 3	sh3
10)	RADAR	rad
11)	INFRA RED	inf
12)	SONAR	son
13)	BATTERIA	bat
14)	CENTRALE	ctr
15)	VAMPIRO	vmp
16)	PLACCA	pla
17)	MINA	min
18)	MINA TEMPO	mit
19)	TRIVELLA	tri
20)	CHAOTIZER	cht
21)	SHOCKER	shk
22)	PARASER	prs
23)	MORTAR 1	mr1
24)	MORTAR 2	mr2
25)	MORTAR 3	mr3
26)	GREASER	grs
27)	STEALTHER	stl
28)	WEB	web
29)	COMPUTER	cmp

P A R A L L E L comando

Se il programma contiene una o più istruzioni PARALLEL al normale flusso delle istruzioni verranno alternate le istruzioni che seguono sulla linea di questo comando; le istruzioni eseguite in parallelo non dovranno contenere istruzioni di salto o ciclo, che in ogni caso verrebbero intercettate e non altererebbero comunque il normale andamento del programma. Il tempo per le istruzioni parallele viene sottratto alla normale esecuzione; se c'è solo una PARALLEL allora il flusso normale del programma verrà rallentato della metà, dato che verranno eseguite alternativamente l'istruzione corrente e l'istruzione parallela, nel caso di due PARALLEL il flusso del programma verrà eseguito ad un terzo della velocità normale e così via. In ogni programma WarBot hanno effetto solo dieci PARALLEL, se ce ne sono di più verranno valutate le prime dieci, quelle con il numero di riga più basso. Questa valutazione avviene in pre-esecuzione: se una PARALLEL viene incontrata durante l'esecuzione del programma viene trattata come WAIT e non viene emesso alcun messaggio d'errore.

R A N G E a

Altera la gittata a cui deve essere lanciato un colpo di mortaio. Per default è 10 caselle.

A U T O a _ comando

La stringa posta dopo il simbolo _ viene posta alla riga **a**, cancellando qualsiasi cosa ci fosse stata prima, se **a** è minore di 1 o maggiore di 100, viene riportata a questi estremi. Se l'esecuzione di AUTO porta delle righe vuote nel programma, queste vengono riempite con WAIT. AUTO non altera i processi paralleli iniziati dalla PARALLEL, e non altera il punto iniziale delle subroutine creato con SUBR, dato che questi due fattori vengono valutati solo in fase di pre-esecuzione.

Esempio
Prima e dopo l'esecuzione della prima riga:

1) auto_3 forward 1	1) auto_3 forward 1
2)	2) wait
3)	3) forward 1

H I T a

Con questo comando il WarBot può colpire l'avversario con i suoi arti meccanici se questo si trova in una delle otto caselle ad esso circostanti, il parametro **a** indica quanta energia deve essere utilizzata nel colpo. A seconda di un esito casuale dipendente dalla quantità di energia, il WarBot subirà o meno un danno.

S C A N a , b

Questo comando attiva i computer del WarBot, il parametro **a** indica la direzione verso la quale deve essere fatta l'analisi, il parametro **b** indica la variabile in cui deve essere messo il risultato: i computer analizzano la casella adiacente il WarBot nella direzione indicata e riportano nella variabile **b** ciò che si trova sulla superficie secondo questa tabella:
0 Casella Vuota
1 Buca
2 Mina
3 Mina a Tempo
4 Casella Grease

I computer riportano nella variabile b+1 il contenuto della casella secondo questa tabella:
0 Casella Vuota
1 White WarBot
2 Black WarBot
3 Piramide
4 Ragnatela

Il costo per questa operazione è di 10 unità di energia.

Variabili pre-definite

Le variabili pre-definite ritornano dei valori rappresentanti dati. Sono funzioni che, al contrario di queste, non hanno bisogno di argomenti.

WGHT - Riporta il peso del WarBot, con valori possibili: da 0 a 20.

ENER - Riporta il valore attuale dell'energia.

MINE - Riporta il numero delle mine attualmente contenute in tutti i componenti mina del WarBot.

MINT - Riporta il numero delle mine a tempo attualmente contenute in tutti i componenti mina tempo del WarBot.

TIME - Riporta il tempo (in cicli) che rimane prima che esploda la mina a tempo rilasciata dal WarBot, se è a zero non c'è alcuna mina a tempo nell'arena.

DIST - Se l'altro WarBot è nel raggio dei sensori infrarossi (non importa in quale direzione) questa variabile riporta la distanza da esso, altrimenti ritorna zero.

WHIT - Se a zero, il WarBot non è stato colpito, se a 5 il WarBot è stato danneggiato da una mina, se a 6 è stato colpito da un mortaio, se a 7 è stato colpito da un attacco HURT, mentre un numero da 1 a 4 indica la direzione relativa da cui è provenuto un attacco (laser o missile).

Funzioni

Le funzioni hanno bisogno di un argomento racchiuso tra parentesi tonde, è permessa l'indentazione di funzioni, non ci sono limitazioni nei livelli. (Variabili pre-definite possono far parte dell'argomento di una funzione).

RAD(a) - Ritorna la distanza che c'è tra un ostacolo e il WarBot nella direzione **a**, se entro il raggio del radar non c'è niente allora ritorna zero; per ostacolo si intende un WarBot, una piramide oppure una ragnatela.

INF(a) - Ritorna la distanza a cui si incontra l'altro WarBot se questo è nella direzione **a**, se questo non c'è entro il raggio dell'infrarosso allora ritorna zero.

ANG(a) - Se la distanza tra i due WarBot è inferiore al raggio dell'infrarosso ritorna la differenza tra le due coordinate dei WarBot, trasversalmente alla direzione **a**; se l'altro WarBot è fuori dalla portata oppure si trova in una direzione diversa ritorna il valore 20, se l'altro WarBot si trova sulla stessa riga o colonna controllata naturalmente ritorna zero. Esempio: se il White WarBot è nella direzione 1, possiamo avere le seguenti risultati:
— ang(1) ritorna la differenza tra le coordinate X dei WarBot se il Black WarBot si trova in alto rispetto al White WarBot.
— ang(2) ritorna la differenza tra le coordinate Y dei WarBot se il Black WarBot si trova alla destra del White WarBot.
— ang(3) ritorna la differenza tra le coordinate X dei WarBot se il Black WarBot si trova in basso rispetto al White WarBot.
— ang(4) ritorna la differenza tra le coordinate Y dei WarBot se il Black WarBot si trova alla sinistra del White WarBot.

SON(a) - Ritorna la distanza che c'è tra una buca o una mina e il WarBot nella direzione **a**, se entro il raggio del sonar non c'è niente allora ritorna zero.

EQU(a) - Ritorna il codice del componente nel settore **a** del WarBot, se il settore è vuoto ritorna zero; **a** deve essere nell'intervallo 1-20, e viene riportato in esso se ne esce.

RND(a) - Ritorna un numero casuale estratto tra 1 e **a**, estremi compresi.

Variabili

Utilizzando l'interprete WBCode ogni WarBot può accedere a 100 variabili intere (da −32768 a +32767), attraverso l'espressione **[a]** si può leggere il contenuto di una variabile, **a** deve contenere un valore compreso tra 0 e 99, altrimenti viene riportato in questo intervallo; si può scrivere in una variabile attraverso il comando let a=b, in cui **a** è il numero della variabile (senza le parentesi quadre!) e **b** un'espressione qualsiasi. Si può mettere una variabile o un'espressione con variabili come argomento di una funzione, o come valore di un codice, e viceversa, senza alcuna limitazione nell'uso dei tre tipi di parentesi.

Componenti

LASER 1-2-3: Questi componenti quando attivati lanciano un raggio laser nella direzione in cui è puntato il WarBot e si arrestano contro l'altro WarBot o contro muri, piramidi, o ragnatele, distruggendo queste ultime; queste armi consumano rispettivamente 10-20-30 punti energia e causano 1-2-3 danni se il colpo va a segno, non c'è una gittata massima, ma maggiore è la distanza, minore è la probabilità di colpire il bersaglio, infine maggiore è il danno che il laser può provocare, minore è la sua precisione.

MISSILE 1-2-3: Quando attivati essi vengono lanciano nella direzione del WarBot e, come i laser, colpiscono WarBot, muri, piramidi, e ragnatele, non costano energia ma possono essere adoperati una volta sola, dopodiché il loro settore rimane vuoto e il peso del WarBot diminuisce di una unità. Causano rispettivamente 2-3-4 danni, maggiore è il danno che il missile può provocare, minore è la sua precisione, ma sono sempre meno precisi dei laser.

SCUDO 1-2-3: Non c'è bisogno di attivarli, l'interprete mette in funzione il primo che incontra durante la scansione iniziale dei settori del WarBot, questo rimane in funzione fino a che non viene distrutto, a questo punto se ce ne sono ancora, entreranno via via in funzione in ordine di settore. Consumano 1-2-3 punti d'energia a ciclo e proteggono il WarBot dai colpi di qualunque arma e dalle esplosioni delle mine: differiscono tra di loro per la resistenza. Se in uno scontro lo scudo attuale viene distrutto da un colpo, quel colpo viene comunque annullato. Al termine dell'energia, lo scudo attuale viene disinserito.

RADAR: Questo componente serve per

```
Equipaggiamento Warbot TRILLER      Caratteristiche Warbot TRILLER

Elemento 1>Radar                    Peso WarBot    :20
Elemento 2>Sonar                    Energia Totale:1800
Elemento 3>Trivella                 Raggio Radar   :4
Elemento 4>Batteria                 Raggio Infrar.:0
Elemento 5>Batteria                 Raggio Sonar   :4
Elemento 6>Batteria                 Numero Mine    :0
Elemento 7>Batteria                 Mine a Tempo   :0
Elemento 8>Batteria                 Centrali Ener.:0
Elemento 9>Batteria                 Unita' Vampiro:0
Elemento 10>Batteria                Computer       :0
Elemento 11>Batteria                Tipo di Scudo  :Assente
Elemento 12>Batteria
Elemento 13>Placca
Elemento 14>Placca
Elemento 15>Placca
Elemento 16>Placca
Elemento 17>Placca
Elemento 18>Placca
Elemento 19>Placca
Elemento 20>Chaotizer


Programma Warbot TRILLER (13 linee)

1 do
2 activate 20
3 activate 3
4 if rad(1)=1 then goto 10 else if son(1)=1 then goto 10
5 forward 1
6 activate 3
7 if rad(1)=1 then goto 12 else if son(1)=1 then goto 12
8 forward 1
9 loop
10 right 1
11 goto 4
12 right 1
13 goto 7
```

localizzare ostacoli come piramidi, muri, WarBot e ragnatele, ogni radar dà un raggio di quattro caselle in linea retta, non ha bisogno di essere attivato e non consuma energia.

INFRA-RED: Questo componente serve per localizzare l'altro WarBot, anche attraverso piramidi o ragnatele; ogni componente infra-red dà un raggio di quattro caselle in linea retta con l'uso di inf(a), oppure in qualunque direzione con ang(a) o con dist. Non ha bisogno di essere attivato e non consuma energia.

SONAR: Questo componente serve per localizzare le mine oppure un buco sulla superficie dell'arena; ogni sonar dà un raggio di quattro caselle in linea retta, non ha bisogno di essere attivato e non consuma energia.

BATTERIA: Per ogni batteria di cui il WarBot è dotato, gli vengono aggiunti 200 punti d'energia all'inizio dello scontro, questo componente non deve essere attivato e non viene rimosso quando si esaurisce.

CENTRALE: Ogni centrale aggiunge 1 punto d'energia per ciclo, non deve essere attivata, il processo è totalmente automatico.

VAMPIRO: Se l'altro WarBot è entro 5 caselle, per ogni unità vampiro gli viene tolto un punto d'energia a ogni ciclo, e aggiunto al proprio WarBot, l'unità vampiro non deve essere attivata, il processo è totalmente automatico.

PLACCA: Quando il WarBot viene colpito questo componente lo protegge da un danno, distruggendosi e alleggerendo il WarBot di una unità, non deve essere attivato.

MINA: Per ogni componente mina il WarBot ha a disposizione dieci mine normali, che provocano un danno al WarBot che entri nella loro casella. Ogni volta che questo componente viene attivato, una mina viene deposta nella casella dietro il WarBot, se non è occupata da un ostacolo o da un'altra mina; se c'è una buca, la mina va persa. Quando tutte le mine del WarBot vengono rilasciate vengono liberati i settori corrispondenti e diminuito il suo peso.

MINA TEMPO: Per ogni componente mina tempo il WarBot ha a disposizione dieci mine a tempo, che provocano cinque danni meno la distanza al WarBot che si trovi nel raggio dell'esplosione. Ogni volta che questo componente viene attivato e se non c'è un'altra mina a tempo dello stesso WarBot nell'arena, una mina viene deposta nella casella dietro il WarBot, se non è occupata da un ostacolo o da un'altra mina; se c'è una buca, la mina va persa. Quando tutte le mine tempo del WarBot vengono rilasciate vengono liberati i settori corrispondenti. La mina a tempo esplode dopo che è scaduto il tempo contenuto nella variabile TIME, all'inizio uguale a 100 cicli e alterabile con il comando TIMER a.

TRIVELLA: Questo componente quando attivato consuma 10 punti d'energia e scava una buca nella casella retrostante il WarBot, a meno che questa non sia occupata da un ostacolo, da una mina o dall'altro WarBot. Se un WarBot nel suo movimento cade in una buca viene automaticamente distrutto.

CHAOTIZER: Questo componente quando attivato consuma 10 punti d'energia e per dieci turni c'è una probabilità che i radar e i sonar dell'altro WarBot siano accecati riportando zero ai controlli, questa possibilità dipende dal turno di attivazione e dalla distanza.

SHOCKER: Questo componente quando attivato consuma 10 punti d'energia e crea una probabilità che l'interprete dell'altro WarBot salti una riga avanti o indietro, questa possibilità dipende dalla distanza.

```
Equipaggiamento Warbot BLOKADE

Elemento 1>Laser 1
Elemento 2>Infra Red
Elemento 3>Infra Red
Elemento 4>Infra Red
Elemento 5>Batteria
Elemento 6>Batteria
Elemento 7>Batteria
Elemento 8>Batteria
Elemento 9>Radar
Elemento 10>Placca
Elemento 11>Placca
Elemento 12>Placca
Elemento 13>Placca
Elemento 14>Placca
Elemento 15>Placca
Elemento 16>Placca
Elemento 17>Placca
Elemento 18>Placca
Elemento 19>Placca
Elemento 20>Paraser


Caratteristiche Warbot BLOKADE

Peso WarBot    :20
Energia Totale:800
Raggio Radar   :4
Raggio Infrar.:12
Raggio Sonar   :0
Numero Mine    :0
Mine a Tempo   :0
Centrali Ener.:0
Unita' Vampiro:0
Computer       :0
Tipo di Scudo :Assente


Programma Warbot BLOKADE (13 linee)

1 do 1:4
2 if inf(1)>0 then goto 8
3 right 1
4 loop
5 if rad(1)=1 then goto 12
6 forward 1
7 goto 1
8 activate 20
9 activate 1
10 if inf(1)>0 then jump -2
11 goto 2
12 right 1
13 goto 5
```

N.B.: Se l'altro WarBot si trova all'ultima linea e lo shocker lo fa saltare avanti di una riga, viene bloccato per il resto della partita.

PARASER: Questo componente funziona come un laser 1, non provoca danno ma se colpisce l'altro WarBot, ne paralizza i movimenti e le rotazioni per un certo numero di cicli, a seconda della distanza.

MORTAR 1-2-3: Questi componenti consumano 10-20-30 punti d'energia quando attivati e funzionano in modo simile al laser, tranne che per essi deve essere specificata la distanza alla quale sparare; possono però superare ostacoli come piramidi e ragnatela, la gittata di default è dieci caselle, e può essere cambiata con il comando RANGE a. Questa arma è meno precisa dei laser ma più precisa dei missili, e provoca 1-2-3 danni se colpisce un WarBot, con precisione decrescente.
N.B.: Nel caso in cui colpisca una mina la fa esplodere, sia normale e sia a tempo.

GREASER: Questo componente quando attivato consuma 10 punti d'energia e rende la casella retrostante il WarBot scivolosa; facendo slittare qualunque WarBot nella casella successiva a seconda della direzione d'entrata nella casella. Non c'è limitazione al numero di caselle grease nell'arena, queste possono essere localizzate soltanto dal componente COMPUTER.

STEALTHER: Questo componente quando attivato consuma 10 punti d'energia e per dieci turni c'è una probabilità che gli infrarossi dell'altro WarBot siano accecati riportando zero ai controlli, questa possibilità dipende dal turno di attivazione e dalla distanza.

WEB: Questo componente quando attivato consuma 10 punti d'energia e crea alle spalle del WarBot una zona d'energia impassabile. Questa zona viene visualizzata sullo schermo e può essere distrutta con una qualsiasi arma che colpendo provochi un danno (escluso il comando HURT).

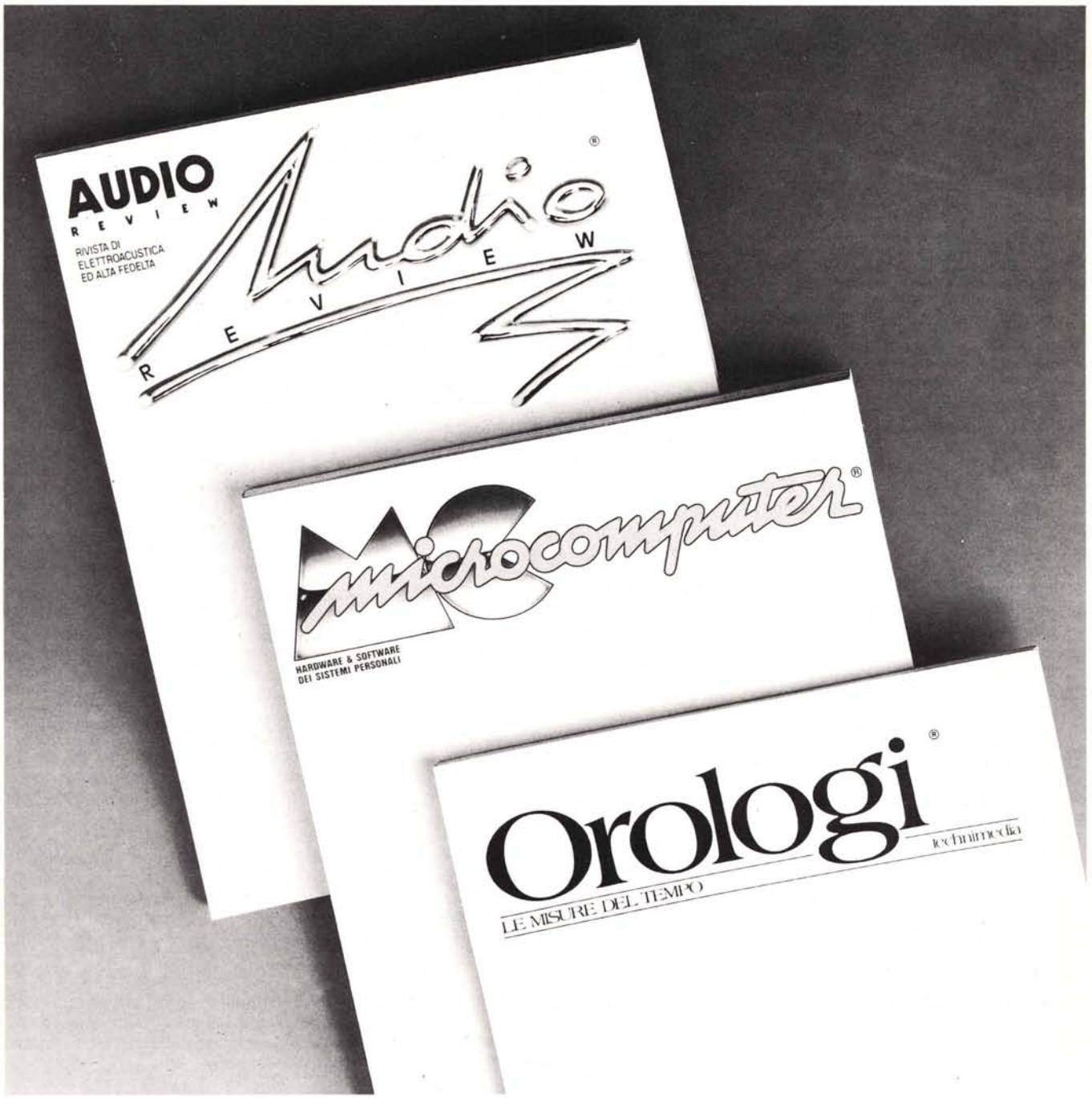
COMPUTER: Questo componente consuma 10 punti d'energia ad ogni utilizzo e permette l'analisi completa di una casella circostante WarBot, vedi comando SCAN.

Espressioni

Le classiche operazioni di +, −, * e / sono state mantenute, ma non esiste priorità e ogni espressione viene valutata da sinistra a destra, sono permesse le parentesi, ma per le espressioni possono essere utilizzate solo le tonde, poiché le quadre e le graffe hanno altri significati.

Questi che seguono sono nuovi operatori aritmetici:
a ! b : ritorna il valore maggiore tra a e b.
a . b : ritorna il valore minore tra a e b.
a # b : ritorna il modulo di a e b (resto della divisione tra a e b).
a % b : ritorna la media tra a e b.
a ; 0 : valore assoluto di a.
a ; 1 : segno di a (1 se positivo, −1 se negativo, 0 se zero). MC

Le pubblicazioni Technimedia



AUDIOREVIEW
La più qualificata rivista italiana di elettroacustica ed alta fedeltà

MCMICROCOMPUTER
La più diffusa e più autorevole rivista italiana di informatica

OROLOGILE MISURE DEL TEMPO
La prima rivista per tutti gli appassionati di orologi

Technimedia
Via Carlo Perrier, 9 - 00157 Roma - Tel. 06/4180300 (12 linee ric. aut.)