Il primo dei programmi di questo mese è un esempio di come sia possibile dotare AUTOCAD, il più conosciuto pacchetto CAD per MS-DOS, di nuovi comandi utilizzando il linguaggio di programmazione incorporato e chiamato Autolisp.
Come si può immaginare il linguaggio è ovviamente molto simile al Lisp e in questo caso la scelta, apparentemente originale, è dovuta al fatto che un disegno AUTOCAD (quindi vettoriale) è una lista di oggetti e quindi facilmente gestibile da Lisp.
Il secondo programma è invece una utility in C per usare il mouse nei programmi Clipper. Il Clipper infatti è stato sviluppato in C e quindi consente facilmente di aggiungere procedure o funzioni personali linkandole al programma ".PRG"; è addirittura possibile riscrivere le procedure originali del Clipper semplicemente usando lo stesso nome per le proprie.
Infine un piccolo trucco per sproteggere i programmi in GW-Basic salvati erroneamente con ",P"

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

Tratteggio di poligonali in Autolisp

di Mauro Serra - Cagliari

È noto che il tratteggio in AutoCAD viene eseguito col comando HATCH che richiede, contrariamente ad un programma di grafica pittorica (ad ex. Dr. Halo), che venga puntato tutto il perimetro dell'area da campire, operazione che in certe circostanze è molto costosa in termini di tempo.

La routine **tratt** che vi presento pone rimedio, seppur con dei limiti, all'inconveniente suddetto facendo in modo che venga campita un'area indicando semplicemente un punto all'interno di essa. Per poterla usare occorre possedere la versione 10.0 in lingua inglese di AutoCAD; la routine certamente non funziona così com'è con una versione in lingua italiana; per versioni in lingua inglese precedenti alla 10.0 il funzionamento non è assicurato (provate!).

Per poterla lanciare caricare nella directory di AutoCAD il file **tratt.lsp**; dopo, dall'interno di AutoCAD digitate **(load "tratt")** <**Return**>.

A questo punto la routine è in memoria; digitate infine **tratt** <**Return**> per farla partire.

La prima domanda che viene posta è **«Vuoi selezionare la regione con una finestra? (S/N)** <**S**>**»**. Se rispondete con <S> o con <**Return**> AutoCAD cercherà la regione da campire dentro la finestra che gli specificherete; se battete <**N**> invece la cercherà in tutto il disegno compreso nei limiti specificati con il comando Limits.

Lo specificare la finestra è un'operazione non necessaria per la ricerca della regione da campire; essa fa però risparmiare molto tempo poiché consente ad AutoCAD di cercare le linee del contorno solo tra quelle completamente racchiuse nella finestra anziché in tutto il disegno. Una volta specificata la finestra puntando l'angolo inferiore sinistro e superiore destro vi viene chiesto: **«Punto interno alla regione»**; rispondete facendo click col puntatore in un punto interno qualsiasi della regione da campire. La domanda successiva è **«Tipo di tratteggio** <**Return**=**linee parallele**>**»**

A questo punto potete digitare o un tipo di tratteggio tra quelli disponibili in AutoCAD (dots, zigzag, stars ecc...) oppure **u** (o più rapidamente <**Return**>).

Se digitate un tipo di tratteggio vi viene poi chiesto in sequenza:

«Fattore di scala del tratteggio <**Return**=**1**>**»**

e

«Angolo del tratteggio<**Return**=**0**>**»**

a cui potete rispondere con un appropriato valore numerico o battendo semplicemente <**Return**> per impostare i valori di default indicati tra le parentesi <>.

Tratteggio di poligonali in Autolisp.

```
;questa routine differisce dalla tratt3sc perche' consente di
;fare la selezione delle linee anche tramite finestra, cosa questa
;che in genere abbrevia di molto i tempi di calcolo
(defun *error* (m)
(princ "\n ") (princ "\n ").
(prompt "\nSi e' verificato l'errore ")(princ m)
)
(defun orienta (linea)
        (setq alist (entget linea))
        (setq s1 (cdr (assoc 10 alist)))
        (setq s2 (cdr (assoc 11 alist)))
        (if (/= (cadr s1) (cadr s2))
            (progn
            (setq alist (subst (cons 10 s2) (assoc 10 alist) alist))
            (setq alist (subst (cons 11 s1) (assoc 11 alist) alist))
            (entmod alist)
            (setq s1 (cdr (assoc 10 alist)))
            (setq s2 (cdr (assoc 11 alist)))
        ))
)
(defun angolox ()
   (setq alfa (angle p1 p2))
   (setq beta (angle q1 q2))
         (if (<= alfa pi)
                (progn
                  (if (<= beta (+ alfa pi))
                     (progn
                      (setq gamma (- beta alfa))
                     )
                     (progn
                      (setq 2pi (* 2 pi))
                      (setq gamma (- (+ alfa (- 2pi beta))))
                     )
                  )
                )
                (progn
                   (if (<= beta (- alfa pi))
                       (progn
                         (setq gamma (+ (- (* 2 pi) alfa) beta))
                       )
                       (progn
                         (setq gamma (- beta alfa))
                       )
                   )
                )
         )
)

(defun base ()
(setq vis (getvar "limmin"))
```

```
(setq vsd (getvar "limmax"))
(if (= opz "S")
(progn
(setq selez1 (ssget "W" winf wsup))
(setq l (sslength selez1))
(setq k 0)
(setq selez (ssadd))
  (while (<= k (1- l))
  (progn
    (setq ename (ssname selez1 k))
    (setq alist (entget ename))
    (if (= "LINE" (cdr (assoc 0 alist)))
    (progn
    (if (= "0" (cdr (assoc 8 alist)))
    (ssadd ename selez)
    )
    ))

   (setq nlinee (sslength selez))
  (if (/= opz "S")
   (progn
     (setq selez (ssget "X" (list (cons 0 "LINE")(cons 8 "0"))))
   ))

   (if (= opz "S")
   (setq leftpoint (list (car winf) (cadr punto)))
   (setq leftpoint (list (car vis) (cadr punto)))
   )
   (setq distmin (distance leftpoint punto))
   (setq a 0)
   (setq nlinee (- nlinee 1))
   (while (<= a nlinee)
       (progn
           (setq e0 (ssname selez a))
           (setq alist (entget e0))
           (setq p1 (cdr (assoc 10 alist)))
           (setq p2 (cdr (assoc 11 alist)))
           (setq intersez (inters p1 p2 punto leftpoint))
           (if (/= intersez nil) ;inizio if n 2
               (progn
                    (setq dist (distance intersez punto))
                    (if (< dist distmin)   ;inizio if n 1
                        (progn
                        (setq distmin dist)
                        (setq closerline e0)

           ))
           ))
   (setq  a (1+ a))
   ))

   (setq contour(ssadd closerline))
   (ssdel closerline selez)
   )

   (defun minimo ()
   (angolox)
      (if (< gamma angolomin)
          (progn
             (setq angolomin gamma)
             (setq templine seg)
          )
      )
    )

   (defun addcontour ()
      (setq a 0)
      (setq angolomin (* 4 pi))
      (while (<= a nlinee)
      (progn
      (setq seg (ssname selez a))
      (if (/= seg nil)
      (progn
      (setq alist (entget seg))
      (setq q1 (cdr(assoc 10 alist)))
      (setq q2 (cdr(assoc 11 alist)))
      (setq dist00 (distance q1 p1))
      (setq dist0 (distance q2 p1))
      (setq dist1 (distance q1 p2))
      (setq dist2 (distance q2 p2))
      (if (= dist1 0)
          (progn
            (if (/= dist0 0) (minimo))
          )
      (if (= 0 dist2)
          (progn
              (if (/= dist00 0)
                (progn
                  (setq alist (subst (cons 10 q2) (assoc 10 alist) alist))
                  (setq alist (subst (cons 11 q1) (assoc 11 alist) alist))
                  (entmod alist)
                  (setq q1 (cdr(assoc 10 alist)))
                  (setq q2 (cdr(assoc 11 alist)))
                  (minimo)
                ))
      ))
      ))
      (setq a (1+ a))
      ));fine while di a
      (setq alist (entget templine))
      (setq q1 (cdr(assoc 10 alist)))
      (setq q2 (cdr(assoc 11 alist)))
      (setq p1 q1)
      (setq p2 q2)
      (ssadd templine contour)
      (ssdel templine selez)
      )
;-------------------------------------------------------------------------
;-------------------------------------------------------------------------
(defun c:tratt ()
(setq park (getvar "cmdecho"))
(setvar "cmdecho" 0)
(setq opz "S")
(setq opz (getstring
"\nVuoi selezionare la regione con una finestra?(S/N)<S> "))
(if (= opz "")(setq opz "S"))
(setq opz (strcase opz))
(if (= opz "S")
(progn
(setq winf (getpoint "\nAngolo inferiore sinistro "))
(setq wsup (getcorner winf "\nAngolo superiore destro "))
))
(setq motivo "u")(setq scala 1)(setq aaa 0)
(setq punto (getpoint "\nPunto interno alla regione: "))
(setq motivo (getstring "\nTipo di tratteggio<Return=linee parallele>: "))
(if (= motivo "") (setq motivo "u"))
(if (= motivo "u")
    (progn
    (setq angtratt (getreal "\nAngolo di tratteggio<Return=0>: "))
    (if (= angtratt nil)(setq angtratt 0))
    (setq interlinea (getreal "spazio fra le linee<Return=1>: "))
    (if (= interlinea nil)(setq interlinea 1))
    (setq dhatch (getstring "\nDoppio tratteggio?(S/N)<Return=N>: "))
    (if (= dhatch "") (setq dhatch "N")(setq dhatch "Y"))
    (setq dhatch (strcase dhatch))
    )
    (progn
    (setq scala (getreal"\nFattore di scala del tratteggio<Return=1>: "))
    (if (= scala nil) (setq scala 1))
    (setq aaa (getreal "\nAngolo del tratteggio<Return=0>: "))
    (if (= aaa nil) (setq aaa 0))
    )
 )

(princ "\n ")(princ "\n ")(princ "\n ")
(prompt "Attendere prego ...")
(base)
(orienta closerline)
(setq p1 s1)
(setq p2 s2)
(setq pfirst p1)
(setq q2 p2)
(while (/= (distance pfirst q2 ) 0)
(progn
(setq a 0)
(setq conc (ssadd))
(addcontour)
))
(if (= motivo "u")
(command "hatch" motivo angtratt interlinea dhatch contour"")
(command "hatch" motivo scala aaa contour"")
)
(setvar "cmdecho" park)
)
```

TRONCA.LSP

```
(defun c:tronca ()
(setq park (getvar "cmdecho"))
(setvar "cmdecho" 0)
(setq e0 (entsel "\nSeleziona la prima linea: ")
      e1 (entsel "\nSeleziona la seconda linea: "))
(setq e0 (car e0)
      e1 (car e1)
      alist0 (entget e0)
      alist1 (entget e1)
      p1 (cdr(assoc 10 alist0))
      p2 (cdr(assoc 11 alist0))
      q1 (cdr(assoc 10 alist1))
      q2 (cdr(assoc 11 alist1))
      intersez (inters p1 p2 q1 q2))
(if (/= intersez nil)
    (progn
    (command "break" e0 intersez intersez)
    (command "break" e1 intersez intersez)
    )
    (progn
    (princ "\n ")(princ "\n ")(princ "\n ")
    (prompt "\nLe linee non si intersecano")
    (princ "\n ")
    )
)
(setvar "cmdecho" park)
)
```

XPLODE.LSP

```
;questa routine consente di esplodere polilinee che si trovano entro i
;limiti del disegno o dentro una finestra specificata dall'utente
(defun c:xplode ()
(setq park (getvar "cmdecho"))
(setvar "cmdecho" 0)
(setq opz "S")
(setq opz (getstring
"\nVuoi selezionare la regione con una finestra?(S/N)<S> "))
(if (= opz "")(setq opz "S"))
(setq opz (strcase opz))
(if (= opz "S")
(progn
(setq winf (getpoint "\nAngolo inferiore sinistro "))
(setq wsup (getcorner winf "\nAngolo superiore destro "))
(setq selez (ssget "W" winf wsup))
)
(progn
(setq vis (getvar "limmin"))
(setq vsd (getvar "limmax"))
(setq selez (ssget "W" vis vsd))
)
);fine if opz s
(setq l (sslength selez))
(setq l (1- l))
(setq a 0)
(while (<= a l)
(progn
(setq ename (ssname selez a))
(setq alist (entget ename))
(if (= "POLYLINE" (cdr (assoc 0 alist)))
(command "explode" ename ))
(setq a (1+ a))
));fine while
(setvar "cmdecho" park)
)
```

Se invece digitate **u** o battete <**Return**> vi viene chiesto in sequenza:

«Angolo di tratteggio <Return=0>»
«Spazio fra le linee <Return=1>»
«Doppio tratteggio? (S/N) <Return=N>»

a cui potete rispondere col valore desiderato oppure battere semplicemente <**Return**> per impostare i valori di default indicati tra le parentesi <>.

Esaurita la fase di input dovete solo aspettare che l'algoritmo venga portato a compimento. Il tempo d'attesa è molto variabile in funzione del tipo di disegno e dell'hardware a disposizione: può andare da qualche secondo a diversi minuti.

Limiti della routine

La routine **tratt** funziona solo in determinate circostanze e cioè:
1) il contorno deve essere costituito solo da linee; non sono ammesse altre entità come polilinee o archi;
2) le linee devono trovarsi sul layer 0;
3) la figura deve essere perfettamente chiusa;
4) una qualsiasi linea del contorno deve appartenere totalmente a quel contorno; in altre parole non è possibile che solo una parte di una linea appartenga ad un contorno: o tutto o nulla. È invece possibile che una linea che appartiene ad un contorno appartenga anche ad un altro contorno;
5) non vengono riconosciuti fori all'interno di una regione.

Algoritmo

Una breve descrizione dell'algoritmo. Esso consiste nel cercare dapprima la linea più vicina in senso orizzontale al punto interno scelto per identificare la regione, dopodiché tra tutte le linee all'interno dei limiti del disegno (o all'interno della finestra di selezione se l'utente ha scelto quest'ultima opzione) e appartenenti al layer 0 vengono ricercate quelle collegate al suo punto finale; tra queste infine viene scelta quella che forma l'angolo di apertura più piccolo, intendendo con questo termine l'angolo antiorario che la linea in questione deve fare per sovrapporsi alla precedente. Il procedimento va ripetuto fino a quando il punto finale del segmento selezionato non coincide col punto iniziale del primo segmento; quando accade questo la figura si chiude ed il processo finisce.

Le routine TRONCA e XPLODE a corredo della routine TRATT

Come già detto per poter lanciare con

successo la routine TRATT non ci devono essere linee che appartengono solo parzialmente al contorno. Se questo accade si rivela utile la routine TRONCA che spezza due linee intersecantisi nel loro punto di intersezione trasformandole così in quattro linee e consentendo l'applicazione di TRATT. Se invece nel disegno ci sono delle polilinee queste possono essere esplose in blocco con la routine XPLODE per poter consentire l'applicazione di TRATT. Chiaramente queste routine per poter essere usate più agevolmente andrebbero inserite in un menu di schermo o di tavoletta.

CMOUSE

di Andrea Francescatti - Rovereto (TN)

CMOUSE è un driver scritto in Assembler per abilitare l'uso del mouse nei programmi scritti in Clipper versione Summer 87.

Il driver prevede il passaggio di un massimo di 4 parametri interi da parte del programma chiamante. Per il momento ne vengono sfruttati solo 3. Dopo aver ricevuto e memorizzato i 4 parametri, verifica il primo per decidere l'operazione da intraprendere. Infatti nel caso delle funzioni 0 e 3, il secondo parametro indica quale dei registri si vuole leggere.

Nel caso invece delle funzioni 4,7 e 8 il secondo e terzo parametro sono i dati da passare alla funzione richiamata tramite l'interrupt 33H.

Per utilizzare le routine nei vostri programmi Clipper, basterà LINKare il file CMOUSE.OBJ come si può vedere nell'esempio COMPILA.BAT o LINKER.LNK (presenti nel disco).

Le funzioni messe a disposizione sono le seguenti:

CMOUSE(0,1) Azione: Resetta il mouse. Da lanciare all'inizio dell'applicazione.
Ritorna: Viene ritornato un intero che vale −1 se il mouse è collegato e attivo.

CMOUSE(0,2) Azione: Come sopra.
Ritorna: Viene ritornato un intero che vale 2 se il mouse è Microsoft o compatibile, oppure 3 se il mouse è un Mouse System.

CMOUSE(1) Azione: Attiva la visualizzazione del cursore del mouse.

CMOUSE(2) Azione: Disattiva la visualizzazione del cursore del mouse.

CMOUSE(3,2) Azione: Richiede informazioni
Ritorna: viene ritornato un intero che

CMOUSE

```
;
;  Programma: CMOUSE.ASM  (CMOUSE.OBJ)
;  Autore...: Francescatti Andrea
;  Data.....: dicembre 1989
;  Note.....: Driver mouse per clipper
;  Sintassi : CMOUSE(num1,num2,num3,num4)    num1=AX,num2=BX,num3=CX,num4=DX
;  Ritorna  : stringa contenente i valori dei registri "xxxx,xxxx,xxxx,xxxx"
;-------------------------------------------------------------------------

    PUBLIC    CMOUSE

    EXTRN     __parni:far
    EXTRN     __retni:far

    DGROUP    GROUP DSEG

DSEG          SEGMENT
retval        dw 0000h
var1          dw 1234h
var2          dw 5678h
var3          dw 9abch
var4          dw 0defh
DSEG          ENDS

_PROG         SEGMENT 'CODE'

              ASSUME CS:_PROG,DS:DGROUP

CMOUSE        PROC FAR
              push bp
              mov bp,sp
              push ds
              push es
              push si
              push di
              mov ax,dgroup
              mov ds,ax

              mov dgroup:retval,0          ;azzera valore di ritorno

              mov ax,0001H                 ;parametro numero 1 = ax
              push ax
              call __parni                 ;intero in AX
              add sp,2
              mov dgroup:var1,ax

              mov ax,0002H                 ;parametro numero 2 = bx
              push ax
              call __parni                 ;intero in AX
              add sp,2
              mov dgroup:var2,ax

              mov ax,0003H                 ;parametro numero 3 = cx
              push ax
              call __parni                 ;intero in AX
              add sp,2
              mov dgroup:var3,ax

              mov ax,0004H                 ;parametro numero 4 = dx
              push ax
              call __parni                 ;intero in AX
              add sp,2
              mov dgroup:var4,ax

              mov bx,0
              mov cx,0
              mov dx,0
              mov ax,dgroup:var1           ;verifica parametri
              cmp ax,0
              je mousres
              cmp ax,1
              je showcurs
              cmp ax,2
              je hidecurs
              cmp ax,3
              je getstat
              cmp ax,4
              je setpos
              cmp ax,7
              je sethor
              cmp ax,8
              je setver
              jmp fine

mousres:      mov ax,0                     ;inizializza mouse
              int 33H
              cmp dgroup:var2,1            ;se parametro 2 = 1
              jne mr1
              mov dgroup:retval,ax         ;ritorna ax = mouse status
              jmp fine
mr1:          cmp dgroup:var2,2            ;se parametro 2 = 2
              jne mr2
              mov dgroup:retval,bx         ;ritorna bx = tipo mouse
mr2:          jmp fine
```

(continua a pag. 272)

(segue da pag. 271)

```
showcurs:   mov ax,1                ;mostra cursore
            int 33H
            jmp fine

hidecurs:   mov ax,2                ;nasconde cursore
            int 33H
            jmp fine

getstat:    mov ax,3                ;legge stato pulsanti + posizione
            int 33H
            cmp dgroup:var2,2       ;se parametro 2 = 2
            jne gs1
            mov dgroup:retval,bx    ;ritorna bx = stato pulsanti
            jmp fine
gs1:        cmp dgroup:var2,3       ;se parametro 2 = 3
            jne gs2
            mov dgroup:retval,cx    ;ritorna cx = coordinata X
gs2:        cmp dgroup:var2,4       ;se parametro 2 = 4
            jne gs3
            mov dgroup:retval,dx    ;ritorna dx = coordinata Y
gs3:        jmp fine

setpos:     mov ax,4                ;setta posizione cursore
            mov cx,dgroup:var2
            mov dx,dgroup:var3
            int 33H
            jmp fine

sethor:     mov ax,7                ;setta limiti orizzontali
            mov cx,dgroup:var2
            mov dx,dgroup:var3
            int 33H
            jmp fine

setver:     mov ax,8                ;setta limiti verticali
            mov cx,dgroup:var2
            mov dx,dgroup:var3
            int 33H
            jmp fine

fine:       mov ax,dgroup:retval    ;ritorna valore intero
            pop di
            pop si
            pop es
            pop ds
            pop bp
            push ax
            call __retni
            add sp,2
            ret
CMOUSE      ENDP

;-------
; fine
;-------
_PROG       ENDS
            END
```

contiene lo stato dei pulsanti del mouse secondo il seguente significato:
0=nessun pulsante
1=pulsante sinistro
2=pulsante destro
3=pulsante sinistro+destro
4=pulsante centrale (se presente)
5=pulsante centrale+sinistro
6=pulsante centrale+destro
7=pulsante centrale+sinistro+destro

CMOUSE(3,3) Azione: Richiede informazioni.
Ritorna: Viene ritornato un intero che contiene la posizione orizzontale della posizione del cursore. Il valore (per le schede CGA, Hercules e altre in emulazione delle precedenti) varia tra 0 e 639. Per avere la coordinata in modo testo usure la seguente formula
x=INT(cmouse(3,3)/8).

CMOUSE(3,4) Azione: Richiede informazioni
Ritorna: viene ritornato un intero che contiene la posizione verticale della posizione del cursore. Il valore varia tra 0 e 199. Valgono le stesse considerazioni fatte per la funzione (3,3).

CMOUSE(4,X,Y) Azione: Setta la posizione attuale del cursore del mouse alle coordinate X,Y.

CMOUSE(7,m,M) Azione: Limita lo scorrimento orizzontale del cursore del mouse fra m ed M. I valori m ed M devono essere compresi fra 0 e 639 (per gli stessi motivi visti in precedenza) e deve essere m<M.

CMOUSE(8,m,M) Azione: Limita lo scorrimento verticale del cursore del mouse fra m ed M. I valori m ed M devono essere compresi fra 0 e 199. Valgono le stesse considerazioni fatte per la funzione (7,m,M).

Il file MENU.PRG presente nel disco in vendita è un sorgente scritto in Clipper che illustra l'uso di alcune delle funzioni di CMOUSE. Per la realizzazione di questo driver mi sono ispirato agli articoli da voi pubblicati ed al manuale fornito con il PC MOUSE della Mouse System Corp.

SP-BUILD

di Marcello Amormino - S. Giov. Gemini (AG)

Smanettando un poco con il GWBasic mi sono accordo di una cosetta: se, dopo avere caricato un file di sorgente Basic protetto con l'opzione P, si carica un altro file composto da due byte (FF+09), la protezione scompare.

Non vi sembra un modo molto veloce per risolvere questo problema? MC

```
10 '
11 '  SP-BUILD.BAS   (C) Amormino Marcello 1989
12 '
13 '
14 '  Questo programma genera il file SPROTECT.BAS per la sprotezione
15 '  dei programmi BASIC salvati con l'opzione ",p".
16 '
17 '
18 '        1)  Caricare l'interprete BASIC
19 '
20 '        2)  digitare:    LOAD "programma da sproteggere"
21 '
22 '        3)  digitare:    LOAD "SPROTECT"
23 '
24 '        4)  digitare:    LIST
25 '
26 '        5)  digitare:    SAVE "programma sprotetto"
27 '
28 '
29 '
30 OPEN"SPROTECT.BAS" AS #1 LEN=1
31 FIELD #1,1 AS A$
32 LSET A$=CHR$(255):PUT #1:LSET A$=CHR$(9):PUT #1
33 CLOSE
34 PRINT "Il file SPROTECT.BAS è stato creato."
```

NEWEL S.R.L

VIDEON

per IBM PC COMPATIBILI

VIDEON digitalizzatore d'immagine, consente ad un costo estremamente contenuto di catturare immagini a colori che possono essere visualizzate su standard VGA.
Funzionante su computer della classe AT/XT, permette di digitalizzare immagini in risoluzioni di:

- 320x200 - 640x400 - 040x480 - in 256 colori
- 1034x768 in 16 colori

Il temp necessario alla cattura dell'immagine é di 5 secondi B/W e di 50 secondi in 256 colori.
Le immagini catturate col **VIDEON** possono essere elaborate con i più diffusi programmi: PC PAINT - PAGE MAKER - VENTURA - GEM ecc.
Le immagini possono essere salvate nei seguenti modi:
P.CX - GIF - PIC

Caratteristiche Tecniche:

Il **VIDEON** é provvisto di un BY PASS, per poter visualizzare l'immagine standard e regolarne luminosità, colore, contrasto, sullo stesso monitor. Funziona con configurazione a 640 K di base, (consigliamo 2 Mb) livelli di grigio B/W 64, colori 256.
Elaborazione 24 bit, 8 per ogni componente RGB.
Il **VIDEON** funziona in standard parallela.
Tecniche di filtraggio e Dithering fanno si che si ottengano immagini ad altissima qualità.
Immagini che sfruttando tecniche di Dithering avranno apparentemente 200.000 colori.

NEWEL s.r.l.
computers ed accessori
20155 MILANO - Via Mac Mahon, 75 - Tel. neg. 02/32.34.92 - 33.00.00.36 - Tel. BBS 02/32.70.226 - Fax 02.33.00.00.35