di Raffaello De Masi

IL TURBO PASCAL

La Borland è ormai una realtà anche in Italia, con una presenza sul mercato estesa ed articolata, non solo nel campo dei linguaggi. Dei suoi Data Base ci siamo in diverse occasioni interessati, e dei suoi programmi di utility, primo tra tutti i Sidekick, avremo probabilmente modo di parlare presto. Ciononostante (anche se non solo) Borland resta sinonimo di linguaggi di programmazione, con presenza sul mercato articolata e complessa (i lettori ricorderanno la prova, nel gennaio scorso, dell'efficiente e potente Turbo Basic e, ancora prima, del superlativo «C») anche in campi non proprio tradizionali (leggasi Prolog); merito di chi, in un momento di fortuna, peraltro meritata, ha saputo intraprendere vie nuove, senza restare a cullarsi sugli allori

Sono così nati pacchetti di supporto, con utility grafiche, editor più raffinati, toolbox avanzati; ma questa è storia ormai! Il risultato è che Borland sta eclissando rapidamente gli altri piccoli produttori di idiomi, e sta minando alla base in questo campo, il colosso Microsoft.

Il Turbo Pascal per Mac, esiste sul mercato dal 1966, e che da allora ha subito modifiche sostanziali fino a giungere all'odierna versione 3.0. Si tratta di un linguaggio nato dopo le esperienze del suo gemello in MS-DOS, e che ha potuto pertanto beneficiare di tutta l'esperienza di messa a punto maturata dal fratello.

Vediamone oggi le caratteristiche, dopo che il linguaggio ha già acquisito, nel mondo Mac, una consolidata notorietà e fama di affidabilità ed efficienza.

Il Turbo Pascal

Il pacchetto si presenta costituito da due dischetti e da un pesante manuale d'istruzioni (circa 1 kg di peso e oltre 500 pagine) che, come d'uso con la Borland, rappresenta anche un efficace tutorial del linguaggio (pur non raggiungendo l'efficienza e la completezza del pacchetto Turbo Tutor dell'ambiente MS-DOS). Dopo le raccomandazioni d'uso circa il backup dei dischetti (che, come di consuetudine della Borland, sono senza protezione), si passa, come altrettanto d'uso, all'esame dell'editor di schermo, che, peraltro abbastanza intui-

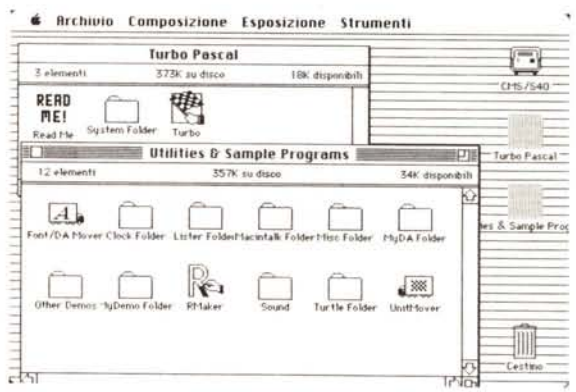

Figura A
Il contenuto dei due dischetti forniti nel package.

tivo, essendo interamente integrato nella filosofia MAC e del tutto analogo all'EDIT della Apple (ci sorge il dubbio che sia proprio quello, customizzato), abbisogna di pochi chiarimenti; è possibile accedere, anche tramite tastiera, alle diverse finestre, ed è ammesso il sistema del doppio click sulla barra della testata per ampliare al massimo lo schermo. Il resto è usuale, come dicevamo, e, d'altro canto non vediamo per quale motivo la Borland avrebbe dovuto modificare un ambiente versatile ed efficace come quello già esistente (non si dimentichi che è, in ogni caso, possibile redigere un programma con qualsiasi wp capace di salvare il documento in codice ASCII, ma anche in questo caso non si vede perché uno debba complicarsi la vita e rinunciare alle utility ed alle semplificazioni che l'ambiente Borland offre).

Altre utility dell'ambiente Mac sono ben note a chi abbia utilizzato almeno una volta il Write.

Ciononostante, secondo lo stile Borland, nulla viene dato per scontato o noto, e successivi paragrafi illustrano le opzioni di formattazione, taglio ed incollaggio di parti del testo, scelta del carat-

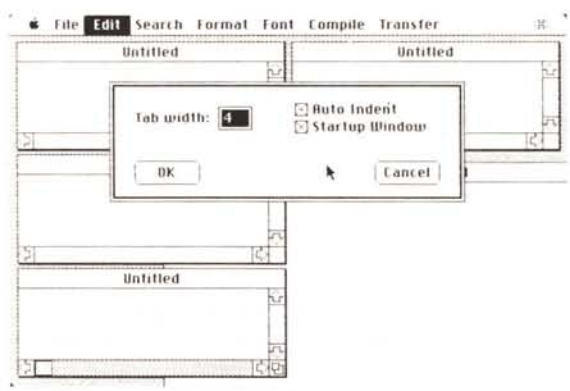

Figura B - Lo schermo di editing, con finestre accatastate e le opzioni relative alle funzioni di indentazione.

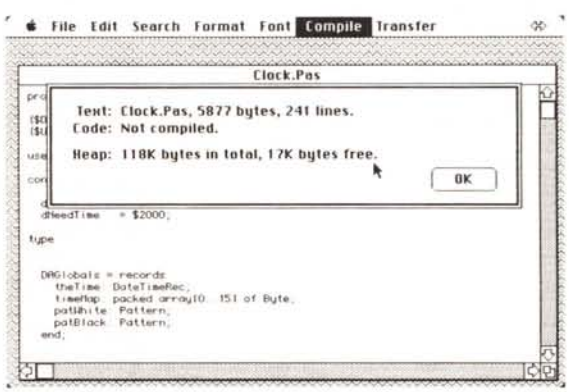

Figura C
Esempio di chiamata alle informazioni di compilazione, con evidenziati gli ingombri dell'heap.

Turbo Pascal

For the Mac
The ultimate Pascal
Development Environment
Produttore
Borland International inc.
4585 Scotts Valley Drive
Scotts Valley
CA 95066 (USA)
Distributore
Edia Borland s.r.l.
Viale Cirene, 11
20135 Milano
Tel. 02-588523

tere, salvataggio e recupero del file, ecc.

Il menu «Compile» ci fa entrare nell'ambiente di compilazione che, ancora una volta secondo una moda imperante in casa Borland, è immediatamente in linea, in modo che il linguaggio presenta il meglio dei due mondi della interpretazione e della compilazione. Il comando presenta un subset di 7 scelte: [Run], che esegue il programma presente in memoria in maniera del tutto analoga allo stesso comando del Basic; ad onor del vero il linguaggio compila il codice sorgente presente in memoria e visibile nell'editor, eseguendone un debug ed

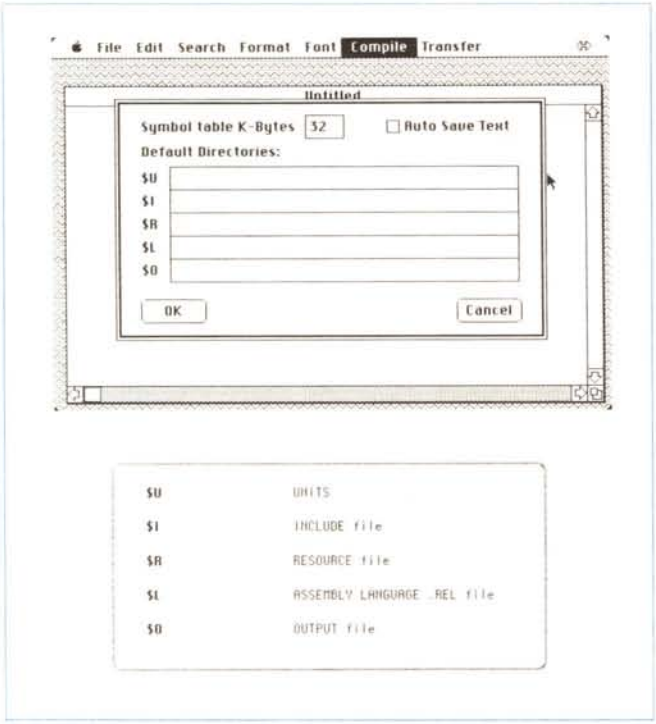

Figura D1 e D2
Directory di default di compilatore e finestre di editing per la loro modifica.

evidenziando, in maniera anche abbastanza pittoresca gli errori in cui incappa. Altri comandi consentono di eseguire una compilazione in memoria, di compilare e salvare su disco come programma stand-alone, di eseguire una verifica preventiva della sintassi, di ricevere informazioni circa lo status del programma (sorgente ed eventualmente oggetto, se ne è stata eseguita una compilazione) presente in memoria (v. fig. C).

Circa la individuazione degli errori, Turbo Pascal utilizza simbologie diverse; oltre lo scarafaggio, che individua errori o defaillance di tipo generale, esiste la bomba (!) che, però individua errori in runtime, come, ad esempio, divisioni per zero, errori di I/O, overflow, ecc. A ciò supplisce, ancora, una opzione del menu compile, il «Find Error» che consente di localizzare il primo errore reperibile nel programma.

L'ultimo elemento del menu «Compile», [Option], consente di settare alcune informazioni di default per l'uso del compilatore, vere e proprie direttive (che comunque possono essere settate anche direttamente dall'interno del programma col comando [$]; si tratta qui, per così dire, di una scorciatoia, ed infatti vengono maneggiate solo alcune delle direttive, globali, difficilmente poi modificabili da un programma. Tanto per intenderci le cinque opzioni modificabili, tutte riferentesi alla manipolazione di file, sono illustrate in figura D1 ed illustrate, nel loro significato, in figura D2. La figura D1 mostra, inoltre, al di sopra della griglia delle directory di default, due altre possibili opzioni, la prima riguardante l'ampiezza da riservare alla tabella dei simboli (32 k come valore massimo, ma che, in macchine come il 128, è opportuno tenere più bassi per lasciare uno spazio sufficiente alle operazioni di compilazione), la seconda, di estrema utilità, che consente l'autosalvataggio del testo del sorgente al lancio del [run], e questo per tutte le finestre attivate nell'editor.

È giunto il momento di parlare, almeno per sommi capi, delle caratteristiche dell'ambiente di Runtime. Come è già noto agli utenti di altri linguaggi compilati Borland, anche questa release del Turbo esime l'utente da qualsiasi preoccupazione di linking ed altre diavolerie. Turbo, per suo conto, «linka» nel programma al momento della compilazione, in maniera assolutamente non visibile all'utente, una serie di routine che consentono di far girare senza problemi il programma sul Mac (ovviamente se non esistono errori). Tutte le operazioni di editing vengono regolate da queste routine, che, sotto il nome comune di Standard Pascal Environment sono rappresentate da quattro unità: PasSystem, PasInOut, PasConsole e PasPrinter; semplificando la cosa è possibile affermare che una unità è una collezione di routine, procedure e dichiarazioni già pronte, destinate a risparmiare tempo e fatica al programmatore.

Una di queste routine, la prima, PasSystem, è sempre usata e caricata immediatamente. Le successive due, PasIn-Out e PasConsole, sono caricate automaticamente ed utilizzate alla bisogna, tranne quando espressamente escluse con la direttiva ($U). L'ultima, infine, è presente solo se esplicitamente richiesta. Molti compilatori Pascal consentono od impongono una serie di diretive al compilatore. La forma generale utilizzata da Turbo per fornire al compilatore queste direttive, vere e proprie istruzioni per la compilazione, è del tipo:

```
{$<lettera> <+ o ->}
o
{$<nome_del_file> <+ o ->}
```

dove la seconda opzione è di gran lunga la più utilizzata ed interessante in quanto consente di imporre al compilatore l'inclusione di un file esterno al sorgente stesso (un vero e proprio merging di librerie eseguito in runtime).

Un interessante esempio di utilizzo di questa istruzione ci viene fornito dalla lettura del manuale. Immaginiamo di aver scritto un programma (ci si perdoni la banalità) in cui viene chiesto di introdurre, dalla tastiera, due numeri, per poi eseguirne il prodotto; supponiamo, ancora, di aver definito le variabili destinate alla manipolazione dei valori come [Integer]. Se, per errore, introducessimo, per uno di questi valori, un numero in virgola mobile, avremmo un errore in rutime ed il programma, dopo una segnalazione d'errore, si bloccherebbe, e l'unica sarebbe di rilanciarlo. Nessun problema, ma cosa succederebbe se lo stesso programma chiedesse in input qualche decina di valori ed il guaio succedesse proprio verso la fine? Potrebbe essere preferibile che il programma, pur rilevando l'errore, non si blocchi e magari, alla fine dell'input risolvesse il tutto chiedendo, tramite una semplice routine all'uopo costruita, quale valore occorre cambiare. Turbo consente di inserire una direttiva (escludente il bloccaggio per errore verificatosi nelle operazioni di I/O), la {$I-}, che consente di bypassare l'errore (ovviamente alla relativa variabile verrà assegnato il valore zero), senza bloccare il tutto. Un'altra delle direttive utilizzabili è {$R+/−}, mediante cui è possibile tener conto od ignorare gli errori di supero di capacità delle array e dei vettori.

Ma, come dicevamo precedentemente, la più efficiente direttiva è quella rappresentata dall'[Include], codificata

dalla istruzione {$I(file)} dove «file» è il nome di un file testo presente sulla memoria di massa. All'incontro della metaistruzione, il compilatore sospende le operazioni sul sorgente principale, apre il file specificato, ne legge il contenuto e lo compila come parte integrante del programma principale, richiudendolo alla fine e rispettando tutte le possibili gerarchie di variabili locali e globali. Esiste, comunque, un metodo migliore di utilizzo di parti esterne complementari; trasformare le stesse in «unit», unità di compilazione, e chiamare le stesse col comando [uses]. Tanto per continuare ancora con gli esempi, un'altra opzione riguarda la nomenclatura del programma oggetto. In altri termini, dopo la compilazione, il linguaggio produce l'oggetto, come applicazione, con lo stesso nome del sorgente. Tramite l'opzione {$O file} dove «file» è, come sempre, il nome del programma desiderato, è possibile ridirigere la compilazione su un nome diverso da quello del listato. In effetti, ad onor del vero, si tratta più di una «chicca» che altro, ma sapere che esiste non guasta!

Ad un certo punto il manuale abbandona (già fin dalla pagina 47) la classica descrizione del Pascal per passare all'ambiente Macintosh. È una scelta obbligata, in quanto nessun utente si sentirebbe di rinunciare al mondo sommerso del toolbox per utilizzare il semplice (si fa per dire) Pascal così come è su questa macchina. In effetti l'utilizzo del toolbox e delle utility di sistema operativo è così semplice ed efficace che, ripetiamo le parole stesse del manuale, è molto difficile utilizzarli in maniera sbagliata, visto anche che essi sono redatti e rispettano l'ambiente Pascal (in particolare, il Lisa Pascal). Le routine di S.O. e di TB sono organizzate in gruppi logici, e nomi come Font Manager, Text-Edit, Dialog Manager, Segment Loader, Window Manager, QuickDraw, e tanti altri, sono troppo noti anche ai più smaliziati utenti Mac per aver qui bisogno di chiarimenti. Tutti comunque sono descritti, anche se sommariamente, nel manuale, che, comunque, riporta anche un piccola bibliografia dove cercare ulteriori chiarimenti circa gli argomenti trattati (non manca, ovviamente, il monumentale «Inside Macintosh» e gli eccellenti «Macintosh Revealed», pubblicati dalla Hayden Book).

I capitoli 7, 8 e 9 sono dedicati alle tecniche di costruzione di un programma, un piccolo tutorial ed una guida all'utilizzo delle risorse del Mac in ambiente Pascal. Tutto è visto in ottica di Unit (il capitolo 7, addirittura, si intitola «Units and others Mysteries») ed i continui riferimenti all'interfaccia Mac consentono di accedere in maniera estesa alla tecnica di costruzione di unità proprie, vera pietra miliare di questo linguaggio. Un paragrafo, in particolare, è dedicato ad uno stadio avanzato di questi tool, l'UNITMOVER, una utility prestigiosa che consente di includere nelle unità standard, caricabili automaticamente [{$U}], unità ben testate scritte dall'utente. In altri termini, è come se uno potesse crearsi una libreria di base, sempre utile, pronta all'uso, self-tailored a seconda dei desideri dell'utente.

Un lungo capitolo è rappresentato da un vero e proprio tutorial alla scrittura corretta di un programma, tallone d'Achille di questo linguaggio sussiegoso e bacchettone. Sebbene TB consenta elasticità non ammesse dal più rigido UCSD, esiste (e sarebbe innaturale se ciò non fosse) ancora una sintassi piuttosto contorta da utilizzare. Mi perdoni pertanto il lettore pascalista se non sono d'accordo con lui sulla necessità di «imporre» un rigido formalismo. È solo una mia opinione (peraltro condivisa da fior di menti informatiche ben più capaci di me nell'esprimere giudizi), ma, oggi, col fior fiore di linguaggi a disposizione, certi vincoli che il pascalista si autoimpone non hanno più motivo di esistere (tenendo anche conto che è possibile scrivere programmi illeggibili sia in Pascal, sia in Fortran, come è altrettanto vero che è facile redigere un programma in Basic da far invidia a qualunque linguaggio strutturato). Bene ha fatto Borland ad allargare i legami piuttosto stretti imposti dallo Svizzero, e se si deciderà ad andare avanti su questa strada sono sicuro che venderà ben più del milione di copie che si dice siano oggi in circolazione. Questione di punti di vista, è vero, ma, guarda caso, è un punto di vista condiviso da molti utenti; a meno che non faccia comodo, ai pascalisti, ritirarsi sull'Aventino in una sorta di disturbata e condiscendente

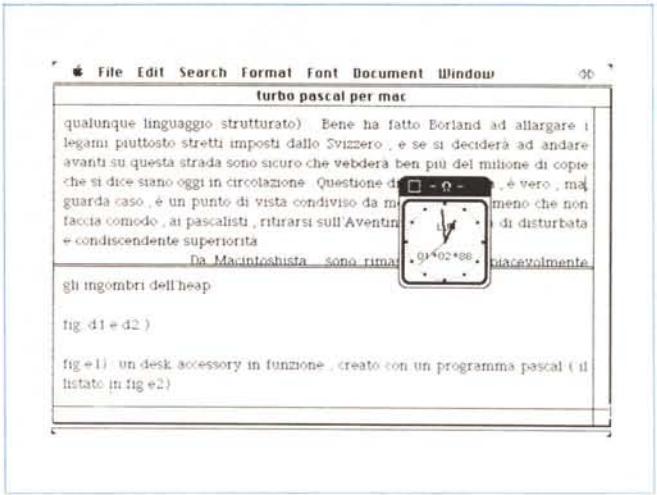

Figura E
Un desk accessory in funzione, creato con un programma Pascal.

Catasta operativa e suddivisione della memoria in runtime.

superiorità. Da Macintoshista, sono rimasto sorpreso piacevolmente dal contenuto del capitolo 10. Finalmente una via breve e sufficientemente facile per scrivere un desk accessory (ne vedete un esempio nella figura E). Anche qui nulla viene dato per sottinteso, e viene eseguita, passo passo, tutta la trafila per giungere al risultato finale, ivi compreso un esauriente discorso sulle Risorse, ed addirittura prevedendo ipotesi trascurabili, come la possibilità di lanciare per errore due volte lo stesso DA; il tutto utilizzando un esempio esauriente ed esteso di programma per la creazione di un DA, MYDA.

Un lungo capitolo è destinato alle operazioni di Debug, anche attraverso il debugger on line presente nel pacchetto, MACSBUG: si tratta di un debugger cui è possibile accedere in diversi modi: se presente nel system folder del disco di bootstrap, esso viene caricato automaticamente; dopo di ciò premendo il tasto di interrupt laterale al Mac, è possibile, tramite l'opzione Resume, entrare in ambiente di Debug; ciononostante, se installato, MACSBUG entra immediatamente in azione se si verifica un errore un runtime; infine è possibile chiamarlo direttamente da programma, con una opzione del tipo «se 'evento' allora MACSBUG». In caso di interrupt in runtime è possibile eseguire, in molti casi, un «resume», tramite una serie di comandi che vengono elencati ed esemplificati nello stesso capitolo. La cosa più interessante è che MACSBUG funziona anche al di fuori di Turbo Pascal, fornendo una diagnostica ogni volta che si verifica un errore, soprattutto con una forma ben più esauriente del solito ID = nn.

La seconda parte del volume assume l'aspetto più familiare agli utenti Mac. Oltre ad un blocco di riferimento di tutti i comandi disponibili in menu (si tratta, come prevedibile, di operazioni dedicate all'editing), i capitoli successivi sono dediti alla esplicitazione formale dei token, alle formalità di dichiarazione, alle istruzioni specifiche del dialetto, all'uso delle procedure, delle funzioni e, ancora meglio, dei sottoprogrammi e delle unità esterne. Un capitolo è accuratamente dedicato alle operazioni I/O; molto spazio è dedito alle funzioni matematiche, a segno dell'attenzione che è stata data a questo settore, talora trascurato da altre implementazioni.

Una decina di pagine è dedicata al SANE, l'ambiente numerico integrato di Apple, ed alle routine in esso contenute; citando a caso troviamo certe funzioni curiose del SANE Engine (sic!), come numeri denormalizzati, arrotondamento finalizzato ad altri tipi di precisione, procedure di conversione da basi numeriche diverse, scalature (logaritmi) in base 2, ricerca automatica di potenze di 2 che non superano il valore di X, copiatura di segno da una variabile ad un'altra, esponenziali diversi; e ancora, udite udite, funzioni già implementate di tipo finanziario, come calcolo dell'interesse composto, annualità su depositi e prestiti, valori di relazione tra due variabili. Il tutto tenendo sempre d'occhio lo standard IEEE.

Le appendici sono il pezzo forte del manuale; rappresentano un manuale nel manuale. La prima (evviva la modestia) compara il TP con altri linguaggi presenti sul mercato, primo tra tutti, ovviamente, il LISA. E poi, messaggi d'errore, un riassunto delle direttive al compilatore, un set ASCII-Macintosh, un esauriente riferimento al Turtlegraphic (vedi nota a fianco), un QuickReference ai comandi. È tutto!

Conclusioni

Turbo Pascal è il fratello siamese dell'analogo pacchetto già visto per l'ambiente MS-DOS. Potente, versatile, senza alcune delle pedanterie e delle idiosincrasie da zitella volute da Wirth, si presenta con un linguaggio destinato ai molti usi (altrimenti avrebbe potuto raggiungere una vetta di vendite così alta?). Non può competere, come popolarità, ovviamente, con alcun Basic, ma ha tutte le carte per rinverdire, in Italia, la fama di un linguaggio forse un po' facilonamente relegato a ruolo di docente per allievi ignoranti di programmazione.

Il fatto che abbia riscosso, anche in Italia, questo successo, in un momento di crisi di programmazione autonoma, conferma la bontà del prodotto, specie nell'area Mac, un po' debole nel settore dei linguaggi. La sig.na Cerrina della Borland, mi comunica che, oggi, sono disponibili per TB alcuni pacchetti di supporto sul tipo di quelli presenti su PC; non mancheremo di riferire appena ne saremo in possesso. Ci dispiace solo (ma speriamo che sia solo questione di tempo) che non sia presente, tra le novità annunciate, un pacchetto di grafica (leggi GRAPHICS), così versatile in MS-DOS; ma non disperiamo, suvvia! Ricordate che solo cinque anni fa dovevamo arrabattarci con l'Applesoft o, al massimo, col Basic 80?

Turtlegraphics, una alternativa alle routine Quickdraw

Quickdraw, il magic box grafico di Mac, efficientissimo nei programmi stand-alone, è spesso seccante e talora impegnativo da utilizzare.
Turbo Pascal include Turtle, un programma efficiente che rende più facile programmare grafica sul Mac.

Turtle è basato su un concetto sviluppato da S. Papert al Massachusetts Institute of Technology. Per superare il concetto di coordinate cartesiane, sempre ostico quando si è costretti a ragionare in termini di pixel, Papert ed i suoi colleghi acquisirono all'informatica l'idea, peraltro vecchia, già presente nella geodesia e nella topografia, dell'azimut e della distanza. Tanto per intenderci, se si desidera collegare un punto con un altro, è sufficiente specificare l'angolo tra i due punti e la distanza tra essi intercorrente per aver individuato, univocamente, la linea da tracciare. Sebbene il concetto non sia dei più utilizzati, i risultati grafici ottenuti con questo sistema sono validi e sofisticati altrettanto di quelli ottenuti utilizzando il più classico sistema delle coordinate cartesiane.

Come tutte le altre routine del Mac, anche Turtlegraphics lavora nella finestra attiva e corrente. Al contrario di esse, come dicevamo, Turtle opera in coordinate «a torta». Il centro dello schermo (finestra attiva) è considerato con coordinate (0,0); secondo la più classica notazione, X ed Y sono positivi in alto a destra, e negativi in basso a sinistra.

Il gran vantaggio è dato dal valore assumibile dalle coordinate, che è praticamente raddoppiato; si va per X fino a valori di 511 e per la Y si giunge a 341, ovviamente, sempre nei limiti della finestra attiva; valori superiori sono ammessi, ma il disegno, ovviamente, va fuori campo.

Turtlegraphics ammette 16 procedure (non si dimentichi la possibilità di combinarle tra loro) che, nella loro sintassi (e anche nella operatività) ricordano (ovviamente) da vicino la sintassi di analoghi statement del LOGO. Gli angoli (meno male, lo dico senza tema di smentite, e senza paura dell'aria di sufficienza di cui mi degneranno certi neosantoni della matematica) sono in gradi sessagesimali.