Programmare in C su Amiga (38)

di Dario de Judicibus
(MC2120 su MC-link)

I controlli proporzionali e la nuova graphics.library della versione 2.0 sono gli argomenti di questa puntata. Continua così l'analisi dei controlli base di Intuition e, nel contempo, entriamo nel vivo del nuovo sistema disponibile oramai anche per l'A500 e l'A2000

Introduzione

Il mondo dei computer è un mondo in continua evoluzione, ed è quindi importante essere sempre aggiornati, senza per questo dimenticarsi completamente di quel passato sul quale tale rinnovamento si basa. L'Amiga non è certo un'eccezione a riguardo. Anzi, la particolare situazione contingente vede gli sviluppatori di programmi su Amiga dividersi in due grandi categorie: quelli che cercano di coprire la parte più consistente dell'attuale parco macchine, sviluppando sulla versione 1.3 del sistema operativo, e quelli che, con un occhio ad un futuro oramai non troppo lontano, si rivolgono alla più professionale versione 2.0, destinata a diventare il sistema principe delle vecchie e nuove macchine (escluso forse l'oramai vecchio ma sempre in gamba A1000).

Data la premessa, questa rubrica non poteva essere da meno. Avuto dalla Commodore il permesso di iniziare a parlare della nuova versione, abbiamo deciso di iniziare nella scorsa puntata uno slalom parallelo tra le due versioni, che cerca di coprire questo periodo di transizione, almeno fino a quando la 1.3 non verrà definitivamente considerata obsoleta.

D'altra parte, molto di quello che si può dire sulla 1.3 continua a valere anche per la 2.0, anche se quest'ultima mette spesso a disposizione soluzioni più eleganti agli sviluppatori dei programmi. Per questo motivo, per il momento, mentre da una parte continueremo la nostra descrizione di Intuition 1.3, seguendo la falsa riga delle puntate passate, dall'altra utilizzeremo la *Scheda Tecnica* per incominciare ad introdurre le caratteristiche della nuova versione.

In questa puntata, quindi, parleremo dei controlli proporzionali (terzo ed ultimo controllo base di Intuition 1.3) nella prima parte dell'articolo, mentre incominceremo a vedere più in dettaglio le funzioni della nuova **graphics.library** nella seconda parte. Dato che nel momento in cui sto scrivendo la 2.0 non è stata ancora rilasciata nella versione definitiva per tutte le macchine, è possibile che qualche funzione subisca ancora qualche piccola modifica. In tal caso provvederò in seguito a riportarla in questa stessa rubrica. Non si dovrebbe trattare di modifiche di rilievo tuttavia, essendo la nuova versione decisamente stabile.

I controlli proporzionali

Un *controllo proporzionale* funziona grosso modo come un *potenziometro*, tanto che spesso useremo anche questo termine per riferirci a questo tipo di controllo, il più complesso dei tre controlli base. In pratica, esso permette di indicare la percentuale desiderata di un certo ammontare, od in generale un valore compreso fra un minimo ed un massimo. Ad esempio, può servire ad indicare quanto testo è visualizzato in una finestra ed a partire da quale posizione del documento, oppure i tre valori relativi alle componenti **R**, **B** e **G** che formano un determinato colore.

Un potenziometro è formato da un contenitore rettangolare [*container*] in cui scorre un cursore [*knob*]. Il contenitore può essere a guida orizzontale o verticale, ed in tal caso il cursore potrà scorrere solo in una direzione (vedi figura 1), oppure coprire un'area che permetta al cursore di muoversi contemporaneamente in entrambe le dimensioni. Un esempio di quest'ultimo caso è rappresentato dal controllo che permette il posizionamento dello schermo del monitor utilizzabile dal programma **Preferences**, fornito con il sistema operativo. Ovviamente il controllo può essere utilizzato sia per fornire al programma un determinato valore, facendo operare l'utente sul cursore con il mouse, sia per fornire all'utente una determinata informazione, assegnando da programma il valore corrispondente.

In genere i controlli proporzionali sono usati per due scopi. Uno è quello di far scorrere un testo ad un'immagine più grande della finestra in cui deve essere visualizzata, in modo da poterla osservare tutta, anche se in passi successivi. L'altro consiste nel regolare il livello di un certo valore, come il volume o la percentuale della componente di un colore.

La struttura PropInfo

Anche nel caso dei potenziometri, come in quello dei campi, è necessario utilizzare una struttura addizionale, che serve ad estendere le informazioni contenute nella struttura **Gadget**, che ha carattere più generale. Questa struttura si chiama **PropInfo**, ed è riportata in figura 2. Anch'essa come già la **StringInfo**, va legata alla struttura madre utilizzando il puntatore **SpecialInfo**. Vediamola in dettaglio.

Il primo campo, **Flags**, serve a definire le caratteristiche del controllo. I valori

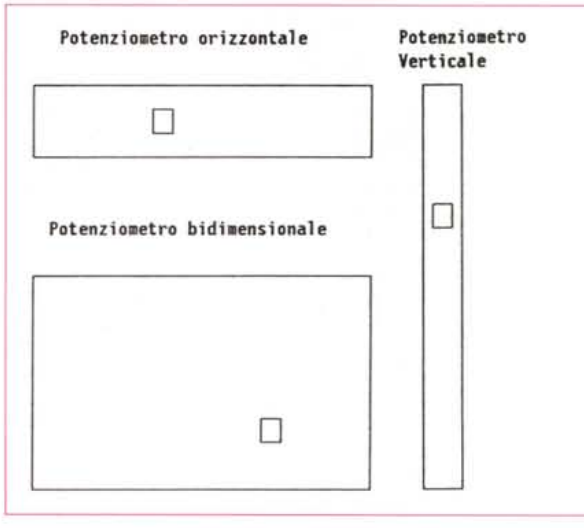

Figura 1 - Potenziometri.

```
/* ------------------------------------------------------------------------ **
**  struct PropInfo                                                         **
** ------------------------------------------------------------------------ **
** Questa è la struttura dati che descrive i parametri caratteristici dei   **
** controlli proporzionali, o potenziometri, ed il cui puntatore deve andrà **
** assegnato al campo SpecialInfo della struttura Gadget relativa.          **
** ------------------------------------------------------------------------ */
struct PropInfo
{
   /* --------------------------------------------------------------------- **
   ** I seguenti cinque campi vanno inizializzati PRIMA che il controllo    **
   ** sia aggiunto al sistema. Intuition si preoccuperà di mantenere questi **
   ** campi sempre aggiornati, anche mentre l'utente sta utilizzando il     **
   ** potenziometro. E' possibile modificare questi valori senza rimuovere  **
   ** il controllo dal sistema, utilizzando la funzione NewModifyProp(),    **
   ** che sostituisce la vecchia ModifyProp(), oramai obsoleta.             **
   ** --------------------------------------------------------------------- */
    USHORT Flags ;      /* Caratteristiche del controllo (vedi Fig.2)      */
    USHORT HorizPot ;   /* percentuale di scorrimento orizzontale (0-FFFF) */
    USHORT VertPot ;    /* percentuale di scorrimento verticale   (0-FFFF) */

   /* --------------------------------------------------------------------- **
   ** I seguenti due campi vengono utilizzati in congiunzione con AUTOKNOB. **
   ** Nel caso che il potenziometro venga utilizzato per visualizzare una   **
   ** certa parte di un insieme (un testo, un grafico, ecc.), questi campi  **
   ** indicano la percentuale visibile dell'insieme, e quindi, nel caso di  **
   ** "manopola a dimensione variabile", anche la dimensione della manopo-  **
   ** la del potenziometro. Lo stesso valore viene anche utilizzato per     **
   ** definire di quanto si sposterà la manopola quando l'utente seleziona  **
   ** il contenitore del controllo, secondo la formula                      **
   **                                                                       **
   **    spostamento = MAXPOT / (insieme / parte visibile)                  **
   **                                                                       **
   ** --------------------------------------------------------------------- */
    USHORT HorizBody ;  /* dimensioni orizzontali della manopola (in %)    */
    USHORT VertBody ;   /* dimensioni verticali della manopola (in %)      */

   /* --------------------------------------------------------------------- **
   ** Le seguenti variabili sono mantenute da Intuition                     **
   ** --------------------------------------------------------------------- */
    USHORT CWidth ;     /* Larghezza del contenitore (in valori assoluti)  */
    USHORT CHeight ;    /* Altezza del contenitore (in valori assoluti)    */
    USHORT HPotRes ;    /* Incremento orizzontale                          */
    USHORT VPotRes ;    /* Incremento verticale                            */
    USHORT LeftBorder ; /* Ascissa del contenitore                         */
    USHORT TopBorder ;  /* Ordinata del contenitore                        */
};
```

Figura 2 - Struttura PropInfo.

```
/* ------------------------------------------------------------------------ **
**  Valori utilizzabili nel campo Flags della struttura PropInfo            **
** ------------------------------------------------------------------------ */
#define AUTOKNOB        0x0001  /* Manopola a dimensione variabile          */
#define FREEHORIZ       0x0002  /* La manopola può scorrere orizzontalmente */
#define FREEVERT        0x0004  /* La manopola può scorrere verticalmente   */
#define PROPBORDERLESS  0x0008  /* Togli il bordo al potenziometro          */
#define KNOBHIT         0x0100  /* Manopola selezionata                     */
```

Figura 3 - Caratteristiche del controllo.

```
/* ------------------------------------------------------------------------ **
**  Altre costanti utilizzate nella gestione dei potenziometri              **
** ------------------------------------------------------------------------ */
#define KNOBHMIN   6       /* Dimensione orizzontale minima della manopola  */
#define KNOBVMIN   4       /* Dimensione verticale minima della manopola    */
#define MAXBODY    0xFFFF  /* Dimensione massima assoluta della manopola    */
#define MAXPOT     0xFFFF  /* Dimensione massima assoluta del potenziometro */
```

Figura 4 - Altre costanti.

prototipo

```
void ModifyProp // Non ritorna informazioni di alcun genere
(
  struct Gadget    *Gadget   , // Puntatore ad un potenziometro
  struct Window    *Window   , // Puntatore alla finestra (vedi sotto)
  struct Requester *Requester, // Puntatore al quadro (vedi sotto)
  USHORT            Flags    , // Nuovo valore di PropInfo.Flags
  USHORT            HorizPot , // Nuovo valore di PropInfo.HorizPot
  USHORT            VertPot  , // Nuovo valore di PropInfo.VertPot
  USHORT            HorizBody, // Nuovo valore di PropInfo.HorizBody
  USHORT            VertBody   // Nuovo valore di PropInfo.VertBody
) ;

// Il puntatore alla finestra, si riferisce alla finestra che contiene
// il potenziometro, od a quella che contiene il quadro che contiene il
// potenziometro. In quest'ultimo caso, il puntatore al quadro va asse-
// gnato al campo successivo, altrimenti nullo. Nota: non abbiamo usato
// il termine "contenitore" qui, per evitare di fare confusione con il
// contenitore vero e proprio del potenziometro.
```

Figura 5 - ModifyProp().

permessi per tale campo, sommabili logicamente, sono riportati in figura 3, ed hanno il seguente significato:

AUTOKNOB
che serve ad attivare il *cursore a dimensione variabile*, come vedremo più in dettaglio tra poco;

FREEHORIZ
che permette al cursore di scorrere orizzontalmente;

FREEVERT
che permette al cursore di scorrere verticalmente;

KNOBHIT
che indica se il cursore è stato selezionato o meno dall'utente, ed è impostato da Intuition;

PROPBORDERLESS
che serve a far si che Intuition non disegni un bordo rettangolare intorno al contenitore (default).

I successivi due campi, **HorizPot e VertPot**, servono a contenere il valore corrente del potenziometro. Ovviamente, nel caso di un potenziometro lineare, in grado di scorrere cioè in una sola direzione, il valore corrispondente all'altra direzione sarà zero, e comunque ignorato. Queste variabili possono assumere un valore compreso fra zero e **MAXPOT**, che al momento è **OxFFFF**. Inutile dire che si raccomanda vivamente di usare sempre la costante **MAXPOT** nel programma, evitando di codificare direttamente il valore numerico che in futuro potrebbe cambiare.

Il valore contenuto in queste variabili è continuamente aggiornato da Intuition, dato che esso varia con lo spostarsi del cursore comandato dall'utente via mouse. Esso tuttavia può anche essere impostato da programma, ed in genere viene inizializzato dal programma ad un valore di default prima che il controllo venga visualizzato sullo schermo.

I campi seguenti, **HorizBody e Vertody**, sono le variabili *di passo*, e servono ad indicare il passo minimo del cursore in entrambe le direzioni. Ad esempio, il potenziometro che indica la componente rossa di un colore viene di solito riportata nell'Amiga in sedicesimi, cioè il cursore si può muovere di un certo numero di sedicesimi della lunghezza complessiva del controllo proporzionale. Ci sono due modi in cui l'utente può muovere il cursore. O selezionandolo con il mouse e, tenendo premuto il tasto sinistro dello stesso, spostandolo all'interno del contenitore fino alla posizione voluta, per poi lasciarlo andare, oppure facendo click più volte nel contenitore, in una parte dello stesso non coperta dal cursore. In quest'ultimo caso, il cursore si muoverà di un passo (definito appunto dalle variabili **Body**) verso il punto in cui si trova il puntatore del mouse, per ogni click dell'utente. È importante notare, tuttavia, che se l'utente sposta invece il cursore direttamente, agganciandolo con il mouse, questo si può spostare anche di una frazione del passo stabilito. Se si vuole evitare questo, il programma deve aggiustare la posizione del cursore a quella più vicina ammessa, immediatamente dopo il rilascio dello stesso.

Le variabili *di passo* sono anche utilizzate nel caso si intenda utilizzare il cursore a *dimensione variabile*. Se infatti la posizione del cursore indica un valore compreso fra un minimo ed un massi-

```
prototipo

void NewModifyProp // Non ritorna informazioni di alcun genere
(
  struct Gadget    *Gadget   , // Puntatore ad un potenziometro
  struct Window    *Window   , // Puntatore alla finestra (= ModifyProp)
  struct Requester *Requester, // Puntatore al quadro (= ModifyProp)
  USHORT            Flags    , // Nuovo valore di PropInfo.Flags
  USHORT            HorizPot , // Nuovo valore di PropInfo.HorizPot
  USHORT            VertPot  , // Nuovo valore di PropInfo.VertPot
  USHORT            HorizBody, // Nuovo valore di PropInfo.HorizBody
  USHORT            VertBody , // Nuovo valore di PropInfo.VertBody
  int               NumGad     // Numero di controlli da restaurare
) ;

// NumGad = -1 restaura TUTTI i controlli in lista
//        =  1 restaura sblo la parte del controllo realmente cambiata
```

Figura 6
NewModifyProp().

```
/* ------------------------------------------------------------------------ **
** Attenzione:                                                              **
** APTR è stato ridefinito nella versione 2.0 come un puntatore assoluto **
** da 32 bit. Questo vuol dire che le classiche operazioni matematiche   **
** sui puntatori C non funzionano con APTR. In tal caso sarà necessario  **
** utilizzare "ULONG *".                                                    **
** ------------------------------------------------------------------------ */
#ifndef APTR_TYPEDEF
#define APTR_TYPEDEF
typedef void           *APTR;       // puntatore generale da 32 bit
#endif

/*
** Tipi correntemente validi -- Da usare in tutti i nuovi programmi
*/
typedef long           LONG;       // intero con segno da 32-bit
typedef unsigned long  ULONG;      // intero senza segno da 32-bit
typedef unsigned long  LONGBITS;   // 32 bit da manipolare individualmente
typedef short          WORD;       // intero con segno da 16-bit
typedef unsigned short UWORD;      // intero senza segno da 16-bit
typedef unsigned short WORDBITS;   // 16 bit da manipolare individualmente
typedef char           BYTE;       // intero con segno da 8-bit
typedef unsigned char  UBYTE;      // intero senza segno da 8-bit
typedef unsigned char  BYTEBITS;   // 8 bit da manipolare individualmente
typedef short          RPTR;       // puntatore (con segno) relativo
typedef unsigned char *STRPTR;     // puntatore a stringa che termina con \0

/*
** Vecchi tipi (obsoleti) -- Da non usare più nei nuovi programmi
*/
typedef short          SHORT;      // intero 16-bit con segno (usare WORD)
typedef unsigned short USHORT;     // intero 16-bit senza segno (usare UWORD)
typedef short          COUNT;      // contatore 16-bit con segno
typedef unsigned short UCOUNT;     // contatore 16-bit senza segno
typedef ULONG          CPTR;       //
```

Figura 7
Tipi obsoleti nella 2.0.

mo, le dimensioni dello stesso nel senso dello scorrimento possono essere utilizzate per indicare un intervallo appartenente allo stesso insieme. Facciamo un esempio. Supponiamo di visualizzare in una finestra una parte del testo di un documento, e di utilizzare un controllo proporzionale verticale per far scorrerne il contenuto. In tal caso il controllo si chiamerà *barra di scorrimento*. In ogni momento tale controllo fornirà anche due informazioni: la posizione della prima linea di testo visibile nella finestra all'interno del documento, indicata dalla posizione del cursore nel contenitore del potenziometro, e quante linee di testo tale finestra è in grado di mostrare. Quest'ultima informazione è appunto fornita dalle dimensioni verticali del cursore. A questo punto è facile capire che legame esiste tra questa informazione e lo spostamento verticale del cursore quando l'utente fa click nel contenitore: il valore è lo stesso perché così si fa scorrere il testo esattamente di una *pagina*, se con tale termine ci riferiamo all'insieme delle linee visualizzate in una finestra.

Il cursore a dimensione variabile viene attivato impostando il valore **AUTOKNOB** nel campo **Flags.** Il tutto verrà gestito automaticamente da Intuition. Per questo motivo chiameremo questo tipo di cursore *auto-cursore*. Un altro dei vantaggi dell'auto-cursore è che non è necessario per l'utente fornire alcuna immagine o bordo per lo stesso, e che esso varia di dimensioni anche in relazione a quelle del potenziometro. Se ad esempio avete creato una barra di scorrimento che si allarga e si restringe in accordo alle variazioni di dimensione della finestra che la contiene, allora anche l'auto-cursore varierà le sue dimensioni di conseguenza.

Una raccomandazione. Nel caso si usi il controllo come barra di scorrimento, se non c'è niente da visualizzare o comunque non c'è bisogno di far scorrere il testo o l'immagine in quanto essa è più piccola della finestra di visualizzazione, allora bisogna impostare il passo a **MAXBODY** (vedi figura 4), in modo da impedire lo scorrimento del cursore. In questo caso si consiglia vivamente l'utilizzo dell'auto-cursore. Per far questo, oltre ad aggiungere a **Flags** il valore **AUTOKNOB**, dovete assegnare a **GadgetEnder** il puntatore ad una struttura **Image** *non inizializzata*.

Nel caso non utilizziate l'auto-cursore, invece, dovete assegnare a tale variabile il puntatore ad una struttura **Image** o **Border** opportunamente riempita. Nel caso usiate anche l'immagine alternata (il cui puntatore va assegnato a **SelectRender**), assicuratevi che entrambe le immagini, o bordi, abbiano esattamente le stesse dimensioni.

I rimanenti sei campi della struttura **PropInfo** sono mantenuti da Intuition e servono a mantenere le seguenti informazioni:

CWidth, CHeight
Dimensioni assolute del contenitore;
HPotRes, VPotRes
Incrementi del potenziometro;
LeftBorder, TopBorder
Coordinate effettive dell'origine del potenziometro.

Le funzioni per i potenziometri

Per evitare di dover rimuovere il potenziometro ogni qual volta si desidera modificarne le caratteristiche, cosa che, come abbiamo visto in passato, si deve fare per qualsiasi controllo, Intuition fornisce due funzioni.

La **ModifyProp()** permette di variare tutte le caratteristiche fondamentali del potenziometro, e quindi restaura tutti i controlli nella lista a cui appartiene il potenziometro, a partire dal primo indicato (vedi figura 5).

La **NewModifyProp()** agisce come la precedente, ma permette di restaurare solo un sottoinsieme dell'intera lista dei controlli, con evidenti vantaggi in termini di prestazioni della funzione stessa (vedi figura 6). Se tuttavia il campo **NumGad** viene impostato a **−1**, la funzione si comporta come la **ModifyProp().**

Nella versione 1.3 del sistema operativo, la **NewModifyProp()** ridisegna sempre l'intero controllo. Nella versione 2.0, che stiamo analizzando in parallelo nella *Scheda Tecnica*, se al campo **NumGad** viene assegnato il valore **1**, allora Intuition restaura solo quella parte del controllo effettivamente cambiata, riducendo fortemente quel leggero effetto di intermittenza che a volte si ha durante lo scorrimento del cursore a causa dei continui restauri. Da notare che nella nuova versione, tutti i campi di tipo **USHORT** e **SHORT** vanno definiti rispettivamente come **UWORD** e **WORD** (vedi figura 7).

Conclusione

Nella prossima puntata vedremo le solite funzioni di creazione, visualizzazione e rimozione dei controlli, relativamente ai potenziometri. Continueremo inoltre con le funzioni grafiche della 2.0, e con le nuove strutture che esse utilizzano. Nel frattempo, inutile dirlo, esercitatevi, esercitatevi ed esercitatevi. A presto.

(segue a pag. 352 "La scheda tecnica")

La scheda tecnica: Inside 2.0

Questo mese inizieremo a vedere un po' più in dettaglio le nuove funzioni della versione 2.0. Piuttosto che andare in ordine alfabetico, ho preferito dare la precedenza alle librerie più usate, e quindi quelle di maggiore interesse. Partiremo quindi con la libreria grafica, che contiene ben venticinque nuove funzioni. Analogamente, non riporteremo tutte le funzioni della libreria (che sono oltre un centinaio), ma solo quelle nuove, e quelle che hanno subito modifiche di rilievo. Avremmo infatti voluto approfittare dell'occasione per riportare anche le funzioni rimaste invariate rispetto alla 1.3, per venire incontro a chi ancora programma su questa versione, ma evidenti motivi pratici lo sconsigliano. Non potendo infatti riportare più di una decina di funzioni a puntata, per ovvi motivi di spazi, ci sarebbero volute più di dieci puntate solo per descrivere le funzioni della libreria grafica. Nel frattempo la Commodore avrebbe fatto in tempo a rilasciare una eventuale versione 2.1!

Per quello che riguarda le singole funzioni, non riporteremo in dettaglio tutte le caratteristiche di ogni funzione, ma ne forniremo la sintassi e le informazioni di maggior rilievo, riservandoci in seguito di spiegare in dettaglio come si usa la 2.0, quando questa avrà preso piede anche per altri sistemi oltre all'A3000, nell'ottica di continuo rinnovamento ed attualità di questa rubrica dedicata alla programmazione in C su Amiga.

```
struct BitScaleArgs {
    UWORD          bsa_SrcX        ; // Origine del sorgente: ascisse
    UWORD          bsa_SrcY        ; // Origine del sorgente: ordinate
    UWORD          bsa_SrcWidth    ; // Dimensioni del sorgente: larghezza
    UWORD          bsa_SrcHeight   ; // Dimensioni del sorgente: altezza
    UWORD          bsa_XSrcFactor  ; // Fattori di scala: denominatori (X)
    UWORD          bsa_YSrcFactor  ; // Fattori di scala: denominatori (Y)
    UWORD          bsa_DestX       ; // Origine del bersaglio: ascisse
    UWORD          bsa_DestY       ; // Origine del bersaglio: ordinate
    UWORD          bsa_DestWidth   ; // Dimensioni del bersaglio: larghezza
    UWORD          bsa_DestHeight  ; // Dimensioni del bersaglio: altezza
    UWORD          bsa_XDestFactor ; // Fattori di scala: numeratori (X)
    UWORD          bsa_YDestFactor ; // Fattori di scala: numeratori (Y)
    struct BitMap *bsa_SrcBitMap   ; // Puntatore alla mappa di bit sorgente
    struct BitMap *bsa_DestBitMap  ; // Puntatore alla mappa di bit bersaglio
    ULONG          bsa_Flags       ; // Riservato. Deve essere zero (0).
    UWORD          bsa_XDDA        ; // Riservato.
    UWORD          bsa_YDDA        ; // Riservato.
    LONG           bsa_Reserved1   ; // Riservato.
    LONG           bsa_Reserved2   ; // Riservato.
};
```

Figura 8 - Struttura BitScaleArgs.

La libreria grafica

Ed ecco un primo blocco di funzioni nuove o modificate della libreria grafica dell'Amiga. In effetti molte altre funzioni sono state ritoccate ed ottimizzate, ma si tratta soprattutto di miglioramenti relativi alle prestazioni delle funzioni stesse, cambiamenti indubbiamente interessanti, ma che non impattano direttamente il modo in cui tali funzioni vanno utilizzate. In particolare, molte più funzioni ora sfruttano le grandi possibilità offerte dal *blitter*, oppure sono ora in grado di sfruttare appieno le caratteristiche dei nuovi *chip* speciali.

Questa è la *scheda riassuntiva* dei cambiamenti effettuati nella libreria grafica:

LIBRERIE DINAMICHE	1.3 --> 2.0	Aggiunte	Tolte	Stesse
graphics.library	Vecchia	25	--	108

Ed ora le schede delle singole funzioni.

BitMapScale

Permette di scalare una mappa di bit, utilizzando la nuova struttura **BitScaleArgs**.

prototipo

```
void BitMapScale // Non ritorna informazioni di alcun genere
(
  struct BitScaleArgs *bitScaleArgs // parametri di scala
) ;
```

Novità rispetto alla versione precedente.
Si tratta di una nuova funzione
La struttura **BitScaleArgs** è riportata in figura 8.

BltBitMap

Muove una regione rettangolare di bit in una mappa di bit.

prototipo

```
ULONG BltBitMap // Ritorna il numero di piani coinvolti effettivamente
(
  struct BitMap *SrcBitMap, // Mappa sorgente
  WORD           SrcX     , // Origine del rettangolo da copiare (X)
  WORD           SrcY     , // Origine del rettangolo da copiare (Y)
  struct BitMap *DstBitMap, // Mappa bersaglio
  WORD           DstX     , // Origine del rettangolo bersaglio  (X)
  WORD           DstY     , // Origine del rettangolo bersaglio  (Y)
  WORD           SizeX    , // Dimensione orizzontale del rettangolo
  WORD           SizeY    , // Dimensione verticale del rettangolo
  UBYTE          Minterm  , // Funzione logica di copiatura
  UBYTE          Mask     , // Maschera per i piani
  UWORD         *TempA      // Area di servizio (opzionale)
) ;
```

A differenza della versione precedente, i due puntatori **ScrBitMap** e **DstBipMap** possono assumere due valori speciali, che non rappresentano indirizzi a strutture **BitMap:** il valore nullo **0x00000000** rappresenta un piano di tutti **«0»**, mentre il valore **0xFFFFFFFF** rappresenta un piano di tutti **«1»**.

BltClear

Azzera un blocco di memoria formato da parole da quattro byte.

prototipo

```
void BltClear // Non ritorna informazioni di alcun genere
(
  void *memblock , // puntatore (pari) al blocco da azzerare
  ULONG bytecount, // numero di byte da azzerare (vedi NOTA 1)
  ULONG flags      // caratteristiche dell'operazione da fare
) ;

// --- NOTA 1 ------------------------------------------------------
// Se il bit 1 di flags è impostato, allora usa i 16 bit bassi come
// numero di byte per riga, e quelli alti come numero di righe,
// altrimenti rappresenta il numero di byte da azzerare.
// --- NOTA 2 ------------------------------------------------------
// Se il bit 0 di flags è impostato, allora la funzione aspetterà
// finchè il blitter non ha finito, se è impostato il bit 1, allora
// utilizza il modo (righe)/(bytes per riga), se infine è impostato il
// bit 2, allora i 16 bit alti di flags sono usati al posto di 0 per
// riempire il blocco di memoria.
```

Nella versione 2.0, il bit **2** della variabile **flags** può essere utilizzato per riempire il blocco di memoria con un valore diverso da zero. Nella versione precedente gli unici bit utilizzati erano lo **0** e l'**1**.

CINIT

Inizializza la lista utente del *copper* in modo da accettare istruzioni intermedie dell'utente.

```
prototipo

struct CopList *CINIT // Area che può contenere le "n" istruzioni
(
  struct UCopList *ucl, // puntatore ad una struttura non inizializzata
  UWORD            n    // numero massimo di istruzioni acquisibili
) ;
```

La macro (non si tratta di una funzione, infatti) è stata ridefinita perché nella versione precedente non ritornava il puntatore alla sottostruttura **(UCopList *)->FirstCopList.**

CloseMonitor

Apre un *monitor.*

```
prototipo

LONG CloseMonitor // se TRUE, il monitor non si è potuto chiudere
(
  struct MonitorSpec *msp // puntatore ritornato dalla OpenMonitor()
) ;
```

Si tratta di una nuova funzione.
Vedere la **OpenMonitor()** per maggiori informazioni sulla struttura **MonitorSpec**.

EraseRect

Riempie un'area rettangolare utilizzando il valore corrente di riempimento per il fondo.

```
prototipo

void EraseRect // Non ritorna informazioni di alcun genere
(
  struct RastPort *rp  , // puntatore al raster interessato
  SHORT            xmin, // Ascissa dell'origine del rettangolo
  SHORT            ymin, // Ordinata dell'origine del rettangolo
  SHORT            xmax, // Ascissa dell'angolo opposto all'origine
  SHORT            ymax  // Ordinata dell'angolo opposto all'origine
) ;
```

Si tratta di una nuova funzione.

ExtendFont

Estende le caratteristiche di un *font* grazie al campo **tf_Extension.**

```
prototipo

ULONG ExtendFont // Ritorna se la funzione ha avuto successo o meno
(
  struct TextFont *font    , // font da estendere
  struct TagItem  *fontTags  // attributi estesi
) ;
```

Si tratta di una nuova funzione.

FindDisplayInfo

Ritorna informazioni reperibili nella *base dati grafica* del sistema.

```
prototipo

DisplayInfoHandle FindDisplayInfo // Aggancio al record estratto
(
  ULONG ID // Chiave di ricerca del modo grafico interessato
) ;
```

Si tratta di una nuova funzione.

FontExtent

Questa funzione riempie la struttura **TextExtent** con una estensione minima (spaziatura) per i caratteri di quello specifico *font.*

```
prototipo

void FontExtent // Non ritorna alcuna informazione
(
  struct TextFont   *font      , // font su cui si effettua la ricerca
  struct TextExtent *fontExtent  // la struttura da riempire
) ;
```

Si tratta di una nuova funzione.

GetDisplayInfoData

Estrae dalle informazioni relative ad un certo modo grafico, ottenute dalla *base dati grafica* (vedi **FindDisplayInfo()**), alcune informazioni di dettaglio.

```
prototipo

ULONG GetDisplayInfoData // se non nullo, numero di byte trasferiti
(
  DisplayInfoHandle handle, // aggancio alle informazioni disponibili
  UBYTE            *buf   , // area da riempire con il risultato
  ULONG             size  , // dimensioni dell'area da riempire
  ULONG             tagID , // quali dati devono essere estratti
  ULONG             ID      // identificativo delle informazioni
) ;

// L'ultimo campo è opzionale, e va usato se il primo campo è nullo
```

Si tratta di una nuova funzione.
Fra i tipi di informazioni disponibili, ci sono:

DTAG_DISP:
(DisplayInfo) - proprietà e disponibilità delle informazioni
DTAG_DIMS:
(DimensionInfo) - dimensioni iniziali ed overscan
DTAG_MNTR:
(MonitorInfo) - tipo, posizione, frequenza di scansione e compatibilità
DTAG_NAME:
(NameInfo) - nomignolo associato a questo modo grafico

GetVPModeID

Ritorna il valore relativo all'identificativo **DisplayID** relativo ad una **ViewPort**.

```
prototipo

ULONG GetVPModeID // DisplayInfoRecord associato alla ViewPort
(
  struct ViewPort *vp // puntatore alla struttura interessata
) ;

// Questa funzione va verificata rispetto INVALID_ID, non NULL.
```

Si tratta di una nuova funzione. MS

EASYDATA

leader per l'informatica personale

Via A.Omodeo 21/29 - 00179 Roma

Tel 06/7858020

Fax 06/7806030

CONDIZIONI GENERALI DI VENDITA
TUTTI I PREZZI SI INTENDONO IVA ESCLUSA - LA GARANZIA HA LA DURATA DI UN ANNO-SI EFFETTUANO SPEDIZIONI TRAMITE CORRIERE ESPRESSO O POSTA-CHIEDICI IL NOSTRO LISTINO COMPLETO: DISPONIAMO DI OLTRE 400 ARTICOLI PER TUTTE LE ESIGENZE - VENDITE RATEALI DA 6 A 60 MESI PER I RESIDENTI NEL LAZIO-VIENICI A TROVARE-TI ASPETTIAMO.

MS/DOS COMPATIBILI

ATTENZIONE
Tutti i PC compatibili assemblati nei nostri laboratori dispongono di una garanzia "Completa" della durata di un anno, che prevede la sostituzione di qualsiasi componente guasto compresi drive e hard-disk.
Disponiamo di un attrezzato laboratorio in sede.

CONSEGNA IN 24 ORE

AMPIE DIMOSTRAZIONI IN SEDE -DISPONIBILITA' DI PACCHETTI SOFTWARE

EasyPower286/21
80286-ram 1024k-hd 44 mega
1 drive-scheda video vga 800x600
2 seriali-1 parallela-1 joystik
L. 880.000

EasyPower386/55
80386-ram 1024k-hd 44 mega
1 drive-scheda video vga 800x600
2 seriali - 1 parallela- 1 joystik
L.. 1.670.000

TITAN 486SX/95
82386-ram 1024k-cm 64k-hd 44mega
1 drive-scheda video vga 800x600
2 seriali -1 parallela- 1 joystik
L. 1.920.000

TITAN 486/150
80486-ram 1024k-cm 128k-hd 44 mega
1 drive-scheda video vga 800x600
2 seriali-1 parallela-1 joystik
L. 2.680.000

OFFERTE PORTATILI

LASER LT 321 (386SX)
3.2kg-1 drive 1.44M-hd 40M-ram 2MB
schermo lcd supertwist retroilluminato
in risoluzione vga con 64 livelli di grigio.
L. 2.825.000

COMMODORE LT286
16MHZ-1 FDD 1.44-1HD 20M
RAM 1M + W INDOWS 3.0
L. 2.940.000

NOVITA'!! CDTV COMMODORE

SISTEMA MULTIMEDIALE BASATO SU AMIGA + LETTORE CD AUDIO-VIDEO (DISPONIBILI TITOLI SU CD)

L. 1.050.000

IN OMAGGIO 2 CD AUDIO
SOLO PER I LETTORI DI MC

TELEFONIA

CELLULARI
PANASONIC L. 1.770.000
KIT AUTO L. 1.200.000
NEC P3 L. 1.700.000

FAX
PANASONIC CON SEGRETERIA
L. 1.330.000
PHILIPS 1615 CON MEMORIA
L. 799.000

STAMPANTE PORTATILE CANON BJ-10E

GETTO D`INCHIOSTRO-1.8 KG
142 CPS-RISOLUZIONE 360 DPI
EMULAZIONE IBM 24 AGHI
L.600.000

ATARI pcFolio

PC FOLIO ITALIANO - L. 335.000
INTERFACCIA PARALLELA - L. 68.000
INTERFACCIA SERIALE - L. 88.000
MEMORY CARD 32K - L. 100.000
MEMORY CARD 64K - L. 150.000
MEMORY CARD 128K - L. 240.000

MONITOR

PHILIPS 8833II 14" COLORE PER AMIGA L. 385.00	HANTAREX 14" COLORE VGA 640x480 L. 436.000
HANTAREX 14" COLORE VGA 1024x768 L. 579.000	NEC 3FG 14" COLORE MULTISYNC L. 1.130.000

STAMPANTI

PANASONIC P1123 80 COL-24 AGHI 190 CPS L. 465.000	PANASONIC P1624 136 COL-24 AGHI 190 CPS L. 899.000
STAR LC 20 80 COL-9 AGHI 150 CPS-4 FONT L. 335.000	STAR LC 200 80 COL-9 AGHI 225 CPS-COLORE L. 419.000
NEC P20 80 COL-24 AGHI 216 CPS-8 FONT L. 605.000	MANNESMAN MT82 80 COL-24 AGHII CARICATORE FOGLI SINGOLI L. 499.000

AMIGA 500 PLUS

V.2.0
RAM 1024K

L.629.000