Programmare in C su Amiga (39)

di Dario de Judicibus
(MC2120 su MC-link)

Seconda puntata dedicata ai potenziometri, o controlli proporzionali. Come di consueto vedremo in dettaglio le funzioni di creazione, visualizzazione e rimozione del controllo, mentre nella prossima puntata vedremo il loro utilizzo insieme ad alcune funzioni ausiliarie per l'acquisizione della posizione del cursore nel potenziometro. Continua infine la carrellata sulla libreria grafica della versione 2.0 del sistema operativo

Introduzione

Nella scorsa puntata abbiamo visto le strutture necessarie a definire i potenziometri, o *controlli proporzionali*. In questa puntata vedremo le solite tre funzioni per la creazione e la rimozione del controllo, e per la sua visualizzazione. Vedremo il tipo più semplice di potenziometro, ovvero quello *a barra*, formato da un contenitore rettangolare e da un cursore che può scorrere al suo interno. Dato che in genere questi controlli vengono usati facendo scorrere il cursore in una sola direzione, il contenitore è visualizzato come una barra rettangolare, da cui il nome. In generale, tuttavia, è possibile definire anche un contenitore «bidimensionale» sebbene le funzioni che vedremo tra poco permettano facilmente di costruire tanto un tipo quanto l'altro, per cui non si è ritenuto necessario scrivere delle funzioni *ad hoc* per il modello a scorrimento bidirezionale, chiameremo questo secondo tipo *potenziometro a piastra*, per distinguerlo dal primo.

A partire dai potenziometri semplici, è poi possibile definire dei controlli più complessi, come abbiamo già fatto in passato con i pulsanti. In quel caso tuttavia, tutti i controlli erano dello stesso tipo, per cui si trattava di fatto di controlli *multipli*. Nel caso dei potenziometri, invece si tratta di veri e propri controlli *compositi*, dato che sono formati da controlli di tipo differente.

Uno dei casi più conosciuti è la *barra di scorrimento*, formata da una barra e da due pulsanti con sopra disegnate delle frecce. I due pulsanti possono essere posti da entrambi i lati della barra, come per la maggior parte dei programmi per Amiga *(barre a frecce simmetriche)*, od entrambi dalla stessa parte, come nel caso del *WordPerfect per Amiga (barre a frecce asimmetriche)*. Io personalmente preferisco quest'ultima soluzione, dato che è migliore dal punto di vista dell'ergonomia del controllo, evitando di dover spostare da una parte all'altra della barra il mouse se si vuole cambiare direzione di scorrimento.

Nel caso di barre di scorrimento per la visualizzazione dei testi, è poi possibile estendere i pulsanti di controllo introducendo quelli di *paginazione*, che permettono lo scorrimento di un'intera pagina alla volta, o quelli di *scorrimento veloce*.

Controlli più complessi che contengono potenziometri, sono le liste di selezione *[list box]*, come quelle utilizzate nei quadri di apertura e salvataggio dei *file [file requester]*, o le barre multiple per la definizione dei componenti RGB di un colore. Anche per questi tipi di controlli, si può utilizzare una tecnica analoga a quella utilizzata nelle precedenti puntate. Tale tecnica, grazie all'allocazione e deallocazione contemporanea di tutte le strutture necessarie a definire il controllo composito, ed all'utilizzo della struttura di servizio per mantenere il vincolo tra le singole parti del controllo, permette di gestire il controllo composito come se fosse un singolo controllo di Intuition, per cui una volta scritte le tre funzioni **CreateXxxx()**, **DisplayXxxxs()** e **DeleteXxxx()**, e le eventuali funzioni o macro di gestione dello stesso, ci si può permettere il lusso di dimenticare i dettagli implementativi rendendo più semplice l'utilizzo di tali controlli nei vostri programmi, assicurandosi una maggiore standardizzazione dell'aspetto esterno delle vostre interfacce, e riducendo le dimensioni dell'eseguibile, dato che la maggior parte delle strutture viene ad essere allocata dinamicamente. Se poi modificate le varie funzioni in modo da poter far parte di una libreria dinamica, le dimensioni dei vostri programmi verrebbero ad essere ulteriormente ridotte.

Le funzioni base

Vediamo ora le tre funzioni base per i potenziometri a barra ed a piastra

```
/*************************************************************************
** CreateBar()                  FUNZIONE              Versione 1.00      **
**                                                                       **
** Funzione fornita:  crea dinamicamente una barra prporzionale          **
**                                                                       **
** Dati in ingresso:  id          identificativo della barra             **
**                    container   contenitore (finestra e/o quadro)      **
**                    txt         testo della barra                      **
**                    hinfo       dati per lo scorrimento orizzontale    **
**                    vinfo       dati per lo scorrimento verticale      **
**                    pr          dimensioni del controllo               **
**                    class       classe della barra (tipo barra)        **
**                                                                       **
** Dati in uscita:    Gdg         puntatore alla barra                   **
**                                                                       **
** Dati globali:      User        memorizza in Size la memoria utilizzata **
**                                                                       **
** Eventi emessi:     GADGETUP    sempre                                 **
**                                                                       **
*************************************************************************/
IGDG *CreateBar(id,container,txt,hinfo,vinfo,pr,class)
  USHORT id ;
  ICNT *container ;
  char *txt ;
  BINFO *hinfo, *vinfo ;
  PRECT *pr ;
  USHORT class ;
{
 /*
 **  Allocheremo quattro aree dati:
 **
 **  -- una struttura Gadget
 **  -- una struttura PropInfo
 **  -- una struttura IntuiText
 **  -- una struttura Image
 **
 **  più l'area di servizio
 */
  IGDG *Gdg ;
  PROP *Pro ;
  ITXT *Txt ;
  IIMG *Img ;
  UBAR *User ;
  USHORT GdgSize, ProSize, TxtSize, ImgSize, UsrSize, TotSize ;
  SHORT l , x, y ;

 /*
 **  Posizione fisica
 */
  x = pr->x ;
  y = pr->y ;

 if (class & HORIZONTALBAR)       /* Barra con movimento orizzontale     */
 {
   Pro->Flags      |= FREEHORIZ ; /*                                     */
   if (hinfo != NULL)
   {
     Pro->HorizPot    = hinfo->init * MAXPOT / (hinfo->max - hinfo->min) ;
     Pro->HorizBody   = hinfo->step * MAXBODY / (hinfo->max - hinfo->min) ;
   }
   else
   {
     Pro->HorizPot    = 0 ;
     Pro->HorizBody   = MAXBODY ;
   }
 }

 Gdg->Width   = (pr->w == 0)? MINBAR : pr->w ;
 Gdg->Height  = (pr->h == 0)? MINBAR : pr->h ;

 Gdg->UserData     = (APTR)User ; /* Questa è una struttura di servizio  */
 User->Number      = 1 ;          /* Numero di campi nel gruppo          */
 User->Size        = TotSize ;    /* Qui metto quanta memoria ho preso   */
 User->Type        = Pro->Flags ; /* Tipo della barra                    */
 User->hi          = *hinfo ;
 User->vi          = *vinfo ;

 /*
 **  Il campo fa parte di un quadro ?
 */
  if (container->r) Gdg->GadgetType |= REQGADGET ;

 /*
 **  Associa alla barra il suo testo
 */
  *Txt = textModel ;                /* Copia il prototipo del controllo */
  Txt->IText = (UBYTE *)txt ;       /* Copia il testo vero e proprio    */
  l = ITXTL(Txt) ;                  /* Lunghezza del testo              */

 /*
 **  Calcola le dimensioni delle varie aree
 */
  GdgSize = sizeof(IGDG) ;
  ProSize = sizeof(PROP) ;
  TxtSize = sizeof(ITXT) ;
  ImgSize = sizeof(IIMG) ;
  UsrSize = sizeof(UBAR) ;

  TotSize = GdgSize + ProSize + TxtSize + ImgSize + UsrSize ;

 /*
 **  Alloca memoria per il campo e le strutture collegate
 */
  Gdg = (IGDG *)AllocMem(TotSize, GDGMEM) ;
  if (Gdg == NULL) return(NULL) ;
  Pro       = POINTER( PROP, Gdg,    GdgSize) ;
  Txt       = POINTER( ITXT, Pro,    ProSize) ;
  Img       = POINTER( IIMG, Txt,    TxtSize) ;
  User      = POINTER( UBAR, Img,    ImgSize) ;

 /*
 **  Assegna i campi fissi della struttura Gadget
 */
  Gdg->NextGadget = NULL ;          /* Per ora è un controllo singolo      */
  Gdg->GadgetID   = id ;            /* Identificativo del controllo        */
  Gdg->GadgetType = PROPGADGET ;    /* Tipo di controllo                   */
  Gdg->Flags      = GADGHCOMP ;     /* Evidenziazione a complemento colore */
  Gdg->Activation = RELVERIFY ;     /* Mi interessa sapere se rilasciato   */
  Gdg->MutualExclude = NULL ;       /* Non utilizzato -- per sicurezza --  */
  Gdg->GadgetText   = Txt ;         /* Testo associato al campo            */
  Gdg->GadgetRender = (APTR)Img ;   /* Non utilizzato per la barra         */
  Gdg->SelectRender = NULL ;        /* Non utilizzato per la barra         */

  Gdg->SpecialInfo  = (APTR)Pro ;   /* Puntatore alla struttura PropInfo   */
  Pro->Flags        = AUTOKNOB ;    /* Usiamo il cursore automatico        */
  if (class & VERTICALBAR)          /* Barra con movimento verticale       */
  {
    Pro->Flags      |= FREEVERT ;   /*                                     */
    if (vinfo != NULL)
    {
      Pro->VertPot    = vinfo->init * MAXPOT / (vinfo->max - vinfo->min) ;
      Pro->VertBody   = vinfo->step * MAXBODY / (vinfo->max - vinfo->min) ;
    }
    else
    {
      Pro->VertPot    = 0 ;
      Pro->VertBody   = MAXBODY ;
    }
  }
  /*
  **  Per rendere più semplice la vita al programmatore, se la posizione è
  **  negativa, la consideriamo una coordinata rispetto al bordo del
  **  controllo PIU' VICINO a quello da cui si parte a misurare.
  */
  if (x > 0)                        /* Ascissa rispetto ad...              */
  {
    x += container->w->BorderLeft ;
    Gdg->LeftEdge = x ;                          /* ...il bordo sinistro */
  }
  else
  {
    x -= container->w->BorderRight ;
    Gdg->LeftEdge = x - Gdg->Width ;             /* ...il bordo destro   */
    Gdg->Flags |= GRELRIGHT ;
  }
  if (y > 0)                        /* Ordinata rispetto ad...             */
  {
    y += container->w->BorderTop ;
    Gdg->TopEdge  = y ;                          /* ...il bordo superiore */
  }
  else
  {
    y -= container->w->BorderBottom ;
    Gdg->TopEdge  = y - Gdg->Height ;            /* ...il bordo inferiore */
    Gdg->Flags |= GRELBOTTOM ;
  }

  /*
  **  Il testo di una barra è, per il momento, sempre sopra la barra, sia
  **  essa verticale od orizzontale, e centrato.
  */
  Txt->LeftEdge = (Gdg->Width - l)/2 ;
  Txt->TopEdge =  -20 ;

  /*
  **  Fatto. La barra è pronta.
  */
  return (Gdg) ;
}
```

Figura 1 - CreateBar().

Figura 2
Esempi di calcoli.

Calcoliamo 122 / (8 - (37 / 5)) ...	
...con parziali arrotondati	...con almeno quattro decimali
1. 37 / 5 -> 7	1. 37 / 5 -> 7.4
2. 8 - 7 -> 1	2. 8 - 7.4 -> 0.6
3. 122 / 1 -> 122	3. 122 / 0.6 -> $203.\overline{3}$

Calcoliamo (24 / 7) * 54 - 160 ...	
...con parziali arrotondati	...con almeno quattro decimali
1. 24 / 7 -> 3	1. 24 / 7 -> 3.4285
2. 3 * 54 -> 162	2. 3.4285 * 54 -> 185.1429
3. 162 - 160 -> 2	3. 185.1429 - 160 -> 25.1429

semplici, cioè senza pulsanti di estensione.

CreateBar()

La funzione più complessa è, come al solito, quella di creazione. Fate riferimento alla Figura 1.

La **CreateBar()** accetta in ingresso sette parametri.

id

Il primo campo, come al solito è l'identificativo del controllo.

container

Il secondo è l'elemento grafico che contiene il controllo, finestra o quadro che sia. Si tratta della solita struttura **Container** che abbiamo già utilizzato in passato. Ricordo che *non è* una struttura di Intuition, ma una struttura creata *ad hoc* per questo genere di funzioni.

txt

È un testo che può essere associato alla barra, come si fa per altri tipi di controlli (ad esempio, i pulsanti). Viene sempre posizionato centralmente sopra la barra, sia essa orizzontale, verticale, od un potenziometro a piastra. Ovviamente potete definire uno standard differente, se volete, od estendere la lista dei parametri in ingresso in modo da aggiungervi anche la posizione relativa del testo. Ricordatevi però che più parametri costringete a fornire, e più complesso risulterà l'utilizzo della funzione, perdendo così uno dei vantaggi principali di questo tipo di struttura modulare, e cioè la semplicità nella definizione e gestione dei controlli.

hinfo

Il quarto campo è il puntatore ad una struttura che fornisce i dati logici relativi allo scorrimento orizzontale del cursore nel potenziometro, e cioè: il valore minimo ed il valore massimo che può assumere il cursore, il valore iniziale dello stesso alla visualizzazione, ed il passo di scorrimento. Da notare che tutti questi valori sono relativi alla logica del programma, e non a quella di Intuition. Ci penserà la funzione a tradurre questi dati in quelli necessari ad Intuition per definire la barra. Ad esempio, se il potenziometro rappresenta un termometro per misurare la temperatura dell'aria, possiamo immaginare un valore minimo di meno 20 gradi Celsius, uno massimo di 50 gradi, un valore iniziale di 16 gradi, ed un passo di un grado Celsius. Questo modo di operare permette al programma di slegarsi dai valori minimi **(0)** e massimi **(MAXPOT)** fissati da Intuition, ed operare solo in termini logici.

vinfo

Come **hinfo,** solo che i dati sono relativi allo scorrimento verticale. Ovviamente uno dei due campi, od anche entrambi, possono essere nulli. In questo caso il cursore riempirà la barra ed, ovviamente, non potrà scorrere in quella direzione.

pr

Serve a fornire le dimensioni fisiche del controllo, e cioè la posizione nell'elemento grafico contenitore, l'altezza e la larghezza del rettangolo in cui si muo-

```
/*****************************************************************************
** DeleteBar()                  FUNZIONE            Versione 1.00          **
**                                                                         **
** Funzione fornita:  cancella dinamicamente una barra proporzionale       **
**                                                                         **
** Dati in ingresso:  bar         puntatore alla barra                     **
**                    container   contenitore (finestra e/o quadro)        **
**                                                                         **
** Dati in uscita:    nessuno                                              **
**                                                                         **
** Dati globali:      User        ricava da Size la memoria utilizzata     **
**                                e da Number il numero di barre           **
**                                                                         **
*****************************************************************************/
void DeleteBar(bar,container)
  IGDG *bar ;
  ICNT *container ;
{
 /*
 **  Rimuovi la barra dal contenitore
 */
  (void)RemoveGList(container->w,bar,
                    ((UBAR *)(bar->UserData))->Number);
  RefreshWindow(container->w) ;

 /*
 **  Cancella la memoria per la barra e le strutture collegate
 */
  FreeMem(bar,(((UBAR *)(bar->UserData))->Size)) ;
}
```

◀ *Figura 3 - DeleteBar().*

```
/*****************************************************************************
** DisplayBars()                FUNZIONE            Versione 1.00          **
**                                                                         **
** Funzione fornita:  aggiunge delle barre al contenitore e quindi le      **
**                    visualizza                                           **
**                                                                         **
** Dati in ingresso:  bar         puntatore alle barre  (uno o gruppo)     **
**                    container   contenitore (finestra e/o quadro)        **
**                                                                         **
** Dati globali:      User        ricava da Number il numero di barre      **
**                                                                         **
*****************************************************************************/
void DisplayBars(bar,container)
  IGDG *bar ;
  ICNT *container ;
{
  USHORT n ;

 /*
 **  Aggiungi la barra al contenitore
 */
  n = ((UBAR *)(bar->UserData))->Number ;
  (void)AddGList(container->w,bar,EOGL,n,container->r) ;
  RefreshGList(bar,container->w,container->r,n) ;
}
```

Figura 4 - DisplayBars().

La scheda tecnica: Inside 2.0

Continuiamo la nostra carrellata sulle novità della versione 2.0 del sistema operativo dell'Amiga. Nella scorsa puntata abbiamo visto undici tra funzioni e macro, di cui ben otto nuove delle venticinque che si vanno ad aggiungere alle oltre 100 funzioni della **graphics.library** 1.3. Vediamo questo mese altre sette schede, fino alla **OpenMonitor().**

GfxAssociate

Associa una struttura di estensione per i dati grafici ad un'altra struttura. La struttura di estensione è basata su una struttura **ExtendedNode**. Questa tecnica permette di estendere con nuove informazioni strutture che non potrebbero essere estese senza generare delle incompatibilità con la versione precedente del sistema operativo. Non tutte le strutture della 1.3, infatti, prevedono uno o più campi riservati, o non utilizzati, che possano essere riutilizzati per estenderne la struttura dati.

Una volta associato ad una struttura dati, il puntatore alla struttura estesa può essere ricavato da quello della struttura dati utilizzando la **GfxLookUp().**

prototipo

```
void GfxAssociate // Non ritorna alcuna informazione come funzione
(
  VOID              *pointer, // puntatore ad una struttura dati
  struct ExtendedNode *node     // struttura da associare al puntatore
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

La struttura **ExtendedNode** è riportata in Figura 7. Nella stessa figura, per comparazione, sono riportate anche le strutture **Node** e **MinNode.**

```
struct  ExtendedNode            // In GRAPHICS\EXTNODES.H
{
   struct Node *xln_Succ      ; // Puntatore al nodo successsivo
   struct Node *xln_Pred      ; // Puntatore al nodo precedente
   UBYTE        xln_Type      ; // Tipo di nodo
   BYTE         xln_Pri       ; // Priorità (per l'ordinamento)
   char         *xln_Name     ; // Nome (stringa)
   UBYTE        xln_Subsystem ; //
   UBYTE        xln_Subtype   ; //
   LONG         xln_Library   ; //
   LONG         (*xln_Init)() ; //
} ;

struct Node                     // In EXEC\NODES.H  (allineamento alla parola)
{
   struct Node *ln_Succ       ; // Puntatore al nodo successsivo
   struct Node *ln_Pred       ; // Puntatore al nodo precedente
   UBYTE        ln_Type       ; // Tipo di nodo
   BYTE         ln_Pri        ; // Priorità (per l'ordinamento)
   char         *ln_Name      ; // Nome (stringa)
} ;

struct MinNode                  // In EXEC\NODES.H - Nodo minimale
{                               //                   senza controllo del tipo
   struct MinNode *mln_Succ   ; // Puntatore al nodo minimale successsivo
   struct MinNode *mln_Pred   ; // Puntatore al nodo minimale precedente
};
```

Figura 7 - Strutture per i nodi.

GfxFree

Libera una struttura di estensione per i dati grafici. Il nodo è deallocato dalla memoria.

prototipo

```
void GfxFree // Non ritorna alcuna informazione come funzione
(
  struct ExtendedNode *node     // struttura estesa da liberare
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

GfxLookUp

Trova la struttura di estensione per i dati grafici associata alla struttura specificata, ritornando il puntatore al nodo esteso in testa a tale struttura di estensione.

prototipo

```
struct ExtendedNode *GfxLookUp // puntatore alla struttura estesa
(
  VOID *pointer // puntatore alla struttura associata a quella estesa
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

GfxNew

Alloca una nuova struttura di estensione per i dati grafici.

prototipo

```
struct ExtendedNode *GfxNew // puntatore alla struttura allocata
(
  ULONG node_type // tipo di struttura grafica estesa da allocare
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

ModeNotAvailable

Controlla se un determinato modo grafico è disponibile o meno, e se così, ritorna un codice di errore che ne spiega il probabile motivo.

prototipo

```
ULONG ModeNotAvailable // Codice di indisponibilità del modo, o NULL
(
  ULONG modeID // un identificativo da 32 bit del tipo DisplayInfoRecord
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

Nelle figure da Figura 8 a Figura 11 sono riportate le strutture che contengono le informazioni relative ai vari *modi grafici* reperibili via **FindDisplayInfo()** e **GetDisplayInfoData()** già viste nella scorsa puntata.

```
   struct DisplayInfo    // Proprietà e Disponibilità delle informazioni
   {
      struct QueryHeader Header          ; // Testata
      UWORD              NotAvailable    ; // Disponibilità del modo grafico
      ULONG              PropertyFlags   ; // Proprietà del modo grafico
      Point              Resolution      ; // Risoluzione
      UWORD              PixelSpeed      ; // Approssimazione in nanosecondi
      UWORD              NumStdSprites   ; // numero di Standard Sprites
      UWORD              PaletteRange    ; // Toni distinguibili disponibili
      Point              SpriteResolution; // Risoluzione per uno Standard Sprite
      UBYTE              pad[4]          ; // (riempitivo di allineamento)
      ULONG              reserved[2]     ; // (riservato)
   };
```

Figura 8 - Struttura DisplayInfo.

```
struct DimensionInfo  // Dimensioni iniziali ed Overscan
{
   struct QueryHeader Header         ; // Testata
   UWORD              MaxDepth       ; // log2( massimo numero di colori )
   UWORD              MinRasterWidth ; // Minima larghezza in pixel
   UWORD              MinRasterHeight; // Minima altezza in pixel
   UWORD              MaxRasterWidth ; // Massima larghezza in pixel
   UWORD              MaxRasterHeight; // Massima altezza in pixel
   struct Rectangle   Nominal        ; // Dimensioni "standard"
   struct Rectangle   MaxOScan       ; // Valore fisso dipendente dall'hardware
   struct Rectangle   VideoOScan     ; // Valore fisso dipendente dall'hardware
   struct Rectangle   TxtOScan       ; // Modificabile via Preferences
   struct Rectangle   StdOScan       ; // Modificabile via Preferences
   UBYTE              pad[14]        ; // (riempitivo di allineamento)
   ULONG              reserved[2]    ; // (riservato)
};
```

Figura 9 - Struttura DimensionInfo.

```
struct MonitorInfo // Tipo, posizione, frequenza di scansione e compatibilità
{
   struct QueryHeader  Header           ; // Testata
   struct MonitorSpec *Mspc             ; // Specifiche del monitor
   Point               ViewPosition     ; // Modificabile via Preferences
   Point               ViewResolution   ; // Risoluzione del monitor standard
   struct Rectangle    ViewPositionRange; // Valore fisso dipendente dall'HW
   UWORD               TotalRows        ; // Altezza in linee di scansione
   UWORD               TotalColorClocks ; // Largezza della scansione (/280ns)
   UWORD               MinRow           ; // Linea di scansione minima assoluta
   WORD                Compatibility    ; // Informazioni sulla compatibilità
   UBYTE               pad[36]          ; // (riempitivo di allineamento)
   ULONG               reserved[2]      ; // (riservato)
};
```

Figura 10 - Struttura MonitorInfo.

```
struct NameInfo
{
   struct QueryHeader Header             ; // Testata
   UBYTE              Name[DISPLAYNAMELEN]; // Nome del modo grafico
   ULONG              reserved[2]        ; // (riservato)
};
```

Figura 11 - Struttura NameInfo.

NextDisplayInfo

Questa funzione permette di navigare attraverso la *base dati grafica del sistema,* per estrarre uno dopo l'altro i vari identificativi che permettono di ottenere le informazioni relative ai vari modi grafici tramite la **FindDisplayInfo().**

prototipo

```
ULONG NextDisplayInfo // Identificativo successivo, o INVALID_ID
(
  ULONG last_ID // Identificativo precedente, od INVALID_ID se primo
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

OpenMonitor

Apre un *monitor* relativo ad uno specifico modo grafico. Se entrambi i parametri sono nulli, apre il monitor di *default,* se solo il primo parametro è nullo, verrà usato solo l'identificativo del modo grafico, altrimenti il monitor è aperto sulla base del nome fornito. Il secondo parametro è opzionale.

Il monitor può essere successivamente chiuso utilizzando la funzione **CloseMonitor().**

prototipo

```
struct MonitorSpec *OpenMonitor // Puntatore alla struttura MonitorSpec
(
  char * monitor_name , // Nome del monitor da aprire
  ULONG  display_id     // Identificativo del modo grafico
) ;
```

Novità rispetto alla versione precedente
Si tratta di una nuova funzione.

ve il cursore. Se una delle due dimensioni è nulla, il programma definisce una dimensione minima. Questo permette di costruire barre a scorrimento unidirezionale fornendo solo la lunghezza della barra.

class

La classe di una barra serve a definire il tipo di barra (semplice, a frecce simmetriche od asimmetriche), e lo scorrimento del cursore (verticale, orizzontale, od entrambi).

E veniamo al codice.

Il primo blocco serve a definire le strutture dell'area di memoria da allocare. In questo caso divideremo tale area in cinque parti. Una per la struttura base del controllo (**Gdg**), una per la sezione estesa con le informazioni tipiche dei potenziometri (**Pro**), una per il testo da associare alla barra (**Txt**), una per l'immagine del cursore (**Img**), ed una, come al solito, per la struttura di servizio (**User**). In questo blocco viene anche estratta la posizione richiesta della barra dalla struttura **pr**. Viene quindi allocata la memoria necessaria e conseguentemente ripartita nelle cinque sezioni.

Il secondo blocco serve ad assegnare i campi fissi, quelli cioè per i quali non bisogna fare alcun calcolo. Viene allora assegnato l'identificativo del controllo, specificato che si tratta di un potenziometro, che l'evidenziazione del contenitore dovrà essere effettuata con la tecnica del colore complementare, e vengono legate le varie strutture fra di loro. Per comodità, inoltre, useremo il cursore automatico, così che non sarà necessario inizializzare la struttura **Image**.

Il terzo blocco definisce le caratteristiche del potenziometro sulla base della classe, e delle informazioni logiche per lo scorrimento. È qui che avviene la trasformazione tra dati logici e dati fisici. Questa parte è molto più delicata di quanto sembra. Volutamente abbiamo riportato un calcolo di prima approssimazione. Matematicamente è corretto, ma quando si effettuano queste operazioni con un computer bisogna tenere conto di molti fattori, come l'ordine in cui viene eseguita l'operazione, eventuali *overflow,* troncamenti. Vi accorgerete che i calcoli daranno dei risultati più o meno accettabili a seconda dei valori logici specificati.

Il motivo è semplice. Un'operazione formata da una successione di operazioni elementari con interi, viene effettuata dal computer in vari passi. Ogni passo deve dare ancora un intero. È evidente che questo fa sì che ogni risultato intermedio potrà subire degli arrotondamenti nelle operazioni di divisione, o degli «straripamenti» in quelle di somma e moltiplicazione. Tali approssimazioni intermedie possono propagarsi in modo differente a seconda dell'ordine in cui vengono effettuate le varie operazioni. Ad esempio, è noto che dividere un numero grande per un numero piccolo fa sì che anche una piccola approssimazione nel valore del divisore comporta una grossa approssimazione in quello del risultato. Analogamente, se una certa operazione porta ad una approssimazione di una parte per mille, e tale numero viene poi ad essere moltiplicato per un valore elevato e sottratto per uno dello stesso ordine di grandezza, il risultato può risultare errato

anche del 100%. Non ci credete? Guardate in Figura 2 per un esempio. È chiaro a questo punto che fare dei *calcoli di scala* con interi è una cosa che può richiedere tecniche sofisticate, specialmente se si devono calcolare valori incrementali, per cui anche un piccolo errore, alla lunga, si vede (come nel caso del passo di un potenziometro).

L'ideale è allora usare i numeri reali, magari a doppia precisione, ed effettuare noi gli arrotondamenti solo sui risultati finali. Esistono comunque tecniche che si basano sulla matematica *diofantea* e che permettono di evitare di ricorrere ai **float** o ai **double**.

Da notare inoltre che abbiamo scelto per la struttura **BarInfo** campi interi da 16 bit con segno. Quindi, se pensate di aver bisogno di gestire numeri più grandi, potreste aver bisogno di ridefinire tale struttura in **gdgusrh.c** usando interi da 32 bit.

Il quarto blocco di codice serve a definire le caratteristiche fisiche del controllo ed a riempire la struttura di servizio. Quest'ultima, oltre alle solite informazioni relative al numero di controlli ed alla memoria allocata, conterrà anche le caratteristiche logiche del potenziometro, in modo da permettere di ricalcolare i valori logici a partire da quelli reali durante la gestione del controllo, come vedremo nella prossima puntata.

Il posizionamento della barra segue le stesse regole di posizionamento relativo già utilizzate per tutti i controlli precedenti. Si potrebbe a questo punto pensare di estrarre questo pezzo di codice e costruire una funzione di conversione di tipo generale, utilizzabile da tutte le funzioni di creazione.

DeleteBar()

Questa funzione è analoga a quelle similari già viste in precedenza per gli altri tipi di controllo. Potrebbe diventare tuttavia più complessa nel caso di controlli compositi, per cui si è ritenuto opportuno farne una funzione a parte, piuttosto che scrivere una macro basata sulle funzioni precedenti.

DisplayBars()

Un discorso analogo vale anche per la funzione di visualizzazione, su cui quindi, per il momento, non ci soffermeremo oltre. Inutile dire che, nel momento in cui la **CreateBar()** dovesse venire estesa per generare anche barre più complesse, anche questa funzione dovrebbe essere estesa per gestire la visualizzazione contemporanea dei vari controlli componenti.

```
/* ------------------------------------------------------------------ **
** Definizioni da precompilare per generare la tabella GDGUSRH.SYM    **
** ------------------------------------------------------------------ */
/*
** Costanti
*/
 #define VERTICALBAR         0x0001
 #define HORIZONTALBAR       0x0002
 #define NOBARARROWS         0x0010
 #define SYMBARARROWS        0x0020
 #define ASYMBARARROWS       0x0040

/*
** Tipi
*/
 typedef struct Image        IIMG;
 typedef struct PropInfo     PROP;

/*
** Strutture
*/
 typedef struct PhysRect
 {
   SHORT  x, y, w, h ;
 }
 PRECT ;

 typedef struct BarInfo
 {
   SHORT  min, max ;
   SHORT  init, step ;
 }
 BINFO ;

 typedef struct UsrBar
 {
   USHORT Number ;
   USHORT Type ;
   USHORT Size ;
   BINFO hi, vi ;
 }
 UBAR;

/*
** Prototipi delle funzioni di servizio
*/
 IGDG *CreateBar     ( USHORT, ICNT *, char *, BINFO *, BINFO *, PRECT *, USHORT);
 void  DeleteBar     ( IGDG *, ICNT * );
 void  DisplayBars   ( IGDG *, ICNT * );
 SHORT HBarValue     ( IGDG * );
 SHORT VBarValue     ( IGDG * );
```

▲ *Figura 5*
Definizione per le barre in gdgusrh.c.

```
// Creiamo un pulsante automatico con su il testo PUSH ME
// Dimensioni ed altri attributi sono presi di default o calcolati
  obj = AutoButton( ooMsg_NEW, "Push me" ) ;

// Decidiamo di fare il testo in BLU
  AutoButton( ooMsg_SET, obj, ooTag_TEXTCOLOR, ooAttr_BLUECOLOR ) ;

// Mettiamo l'oggetto in un certo punto della finestra, e visualiziamolo
  AutoButton( ooMsg_PLACE, obj, row, col ) ;
  AutoButton( ooMsg_SHOW, obj ) ;
  :

// Associamo al bottone la funzione che lo deve gestire se attivato
// Da notare il complesso "cast" del puntatore alla funzione
//   void       vuol dire che la funzione non ha dati di ritorno
//   (*)()      indica che si tratta di un puntatore ad una funzione senza
//              parametri in ingresso
// Notare la differenza tra "void *" e "void (*)", in questo caso.
  AutoButton( ooMsg_ASSOCIATE, obj, (void (*)())AutoButtonHandler ) ;
  :

// Qual'è la lunghezza del pulsante?
  AutoButton( ooMsg_GET, obj, ooTag_TEXTLENGTH, &txtl ) ;

// OK, raddoppiamola. Il testo deve rimanere centrato.
  AutoButton( ooMsg_SET, obj, ooTag_TEXTLENGTH, txtl*2,
                              ooTag_TEXTAALIGN, ooAttr_TEXTCENTERED) ;
  :

// Cancelliamo del tutto il pulsante
  AutoButton( ooMsg_HIDE, obj ) ;
  AutoButton( ooMsg_DELETE, obj ) ;
```

◀ *Figura 6*
Esempio di codice oo.

Conclusione

E chiudiamo qui anche per questo mese. Ancora poche puntate ed avrete un quadro complessivo di tutti i controlli di Intuition, più un certo numero di controlli compositi inventati appositamente per questa rubrica. A questo punto non vi resta che utilizzare quanto appreso per continuare per vostro conto a sviluppare nuove possibilità, od a migliorare quelle pubblicate in questa rubrica.

Per ragioni di chiarezza, infatti, il codice presentato non è stato ottimizzato ai fini di elevate prestazioni in velocità, o per evitare ridondanze tra una funzione ed un'altra. Esistono quindi enormi possibilità di migliorare detto codice, o magari di dargli una struttura diversa.

A questo scopo vi propongono un esercizio che potrebbe tenervi occupati per svariati mesi. Provate a riscrivere le funzioni presentate in una ottica orientata agli oggetti *[object-oriented]*. Non avete bisogno di un linguaggio OO per far questo. Potete farlo anche in C. Ad esempio, per ogni controllo potreste definire una singola funzione del tipo:

```
result = NomeControllo( int msg, ... ) ;
```

dove il messaggio può essere

- **NEW,** per creare un nuovo controllo,
- **PLACE,** per posizionarlo in una finestra e visualizzarlo,
- **REMOVE,** per toglierlo da una finestra, senza deallocarlo,
- **DELETE,** per deallocarlo completamente,
- **SET,** per impostarne alcune caratteristiche,
- **GET,** per ricavarne le caratteristiche, e
- **RESET,** per riportare le varie caratteristiche al valore di default.

La lista dei parametri verrebbe quindi ad essere differente per ogni messaggio (da qui l'utilizzo della *lista a lunghezza variabile* già presentata molte puntate fa).

Inoltre i messaggi **SET**, **GET** e **RESET**, potrebbero utilizzare la tecnica dei *tag* per i parametri, in modo che non sia necessario fornirli sempre tutti, come riportato nell'esempio in Figura 6.

Cercate di usare variabili costanti (**const**) invece delle costanti per il preprocessore definite via **#define**, in modo da avere anche per questi valori il controllo del tipo ed una ben definita locazione di memoria, cosa molto utile in fase di *debugging*. Inoltre, se avete una lista di costanti che corrispondono ad i possibili valori di un certo parametro, ad esempio, usate il formato **enum**.

Vi avverto fin d'ora che non si tratta di un esercizio molto impegnativo dal punto di vista dello sviluppo del codice, quanto da quello del disegno delle funzioni. Non buttatevi quindi subito a codificare, ma cercate di definire tutti i prototipi delle funzioni ed il loro comportamento *prima* di iniziare a scrivere anche una sola istruzione.

Per la logica delle funzioni potete usare i *diagrammi a blocchi*, la *pseudocodifica*, o qualunque altra tecnica formale conosciate per descrivere un flusso logico. Per quello che riguarda lo sviluppo del codice vero e proprio, penso che abbiate oramai tutti gli elementi necessari per scriverlo senza troppi problemi.

Buon lavoro, e non mancate la prossima puntata. MG

Il software MS-DOS, Amiga e Macintosh di Pubblico Dominio e Shareware distribuito da

in collaborazione con Microforum

Questo software non può essere venduto a scopo di lucro ma solo distribuito dietro pagamento delle spese vive di supporto, confezionamento, spedizione e gestione del servizio. I programmi classificati Shareware comportano da parte dell'utente l'obbligo morale di corrispondere all'autore un contributo indicato al lancio del programma.

MSDOS

COMUNICAZIONE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
COM/01	ONE TO ONE	mc104
COM/02	PROCOMM Noto programma di comunicazione	Hard disk
COM/03	OMEGA LINK	mc106
COM/04	BACKCOMM	mc103
COM/05	ZIP	mc110
COM/06	FOSSIL DRIVER & .TPU	mc110
COM/07	MAXIHOST	mc110

DATABASE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
DBS/01	EASY LABELS Per creare etichette	
DBS/02	VIDEO DATABASE	mc105 Hard disk
DBS/03	HOME MANAGER DataBase, calcolatrice e calendario	Hard disk
DBS/04	MAIL-MONSTER	mc103
DBS/05	MAKE MY DAY Per organizzare il lavoro	
DBS/06	PC-FILE+	mc106
DBS/07	TASK MASTER Projet Plaining	
DBS/08	RELIANCE MAILING LIST Mailing per associazioni culturali	
DBS/09	DMS	mc107
DBS/10	ARCHIVIO PARROCCHIALE	mc109

EDUCATIVO

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
EDU/01	ABC FUN KEYS	mc103
EDU/02	COMPUTER TUTOR Auto-apprendimento del computer	
EDU/03	PC-FASTYPE Imparare professionalmente ad usare la tastiera	CGA
EDU/04	GEOBASE ARCH. GEOGRAFICO	mc109

GIOCO

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
GIO/02	2BIT POKER Poker Canadian	EGA/VGA
GIO/03	ASTRO BLASTER Clone di Space Invaders	PC-AT/286
GIO/04	ALDO'S ADVENTURE	mc103 EGA/VGA
GIO/05	CAESAR Strategia	BASIC+EGA/VGA
GIO/07	CLONE INVADERS Clone di Space Invaders	
GIO/08	EGAINT	mc104 EGA/VGA
GIO/09	PC-JIGSAW Puzzle	
GIO/10	MAHJONG Solitario orientale	EGA/VGA
GIO/11	SUPER PINBALL Super Flipper	
GIO/12	ARK Clone di Arkanoid	EGA/VGA
GIO/13	BANYON WARS Strategia	EGA/VGA
GIO/14	CAPTAIN COSMIC Gioco grafico	EGA/VGA
GIO/16	EGA GOLF Gioco del Golf	EGA/VGA
GIO/17	EGA TREK Star Trek	EGA/VGA
GIO/18	JOUST VGA Gioco da bar	VGA
GIO/19	MINER VGA	mc104 VGA
GIO/21	MOSAIX Puzzle	VGA
GIO/22	OTHELLO EGA	mc103 EGA/VGA
GIO/23	POKER SOLITAIRE Poker da soli	EGA/VGA
GIO/24	QUATRIS Tetris con Bombe ecc.	EGA/VGA
GIO/25	SHARKS Giocate ai sommozzatori	EGA/VGA
GIO/26	SLOT EGA Slot Machine	EGA/VGA
GIO/27	BASSTOUR Pesca d'altura	EGA/VGA
GIO/28	BLACKJACK Gioco da Casinò	EGA/VGA
GIO/29	GALACTIC BATTLE Clone di Invaders con sonoro	EGA/VGA
GIO/30	HOUSE OF HORRORS Casa degli orrori	EGA/VGA
GIO/31	NOID Consegnate la pizza all'ultimo piano	EGA/VGA
GIO/32	PINBALL EGA Super Flipper	EGA/VGA
GIO/33	STARDEF Missili distruggono la terra...	
GIO/34	MAHJONG EGA Gioco di società orientale	EGA/VGA
GIO/35	MR.SPOCK	mc105 EGA/VGA
GIO/36	MONUMENTS OF MARS	mc106
GIO/37	PHARAOH'S TOMB	mc106
GIO/38	POKER	mc107EGA/VGA
GIO/39	NIM	mc108 CGA
GIO/40	TESORI	mc108 CGA
GIO/41	TOMBOLA	mc108 CGA
GIO/42	SMILE	mc109 VGA
GIO/43	CHINESE SOLITARIE	mc111 VGA
GIO/44	TRETRIX	mc111 VGA
GIO/45	SICHUAN	mc112 VGA
GIO/46	EGAWALLS	mc113 EGA
GIO/47	GRID HER	mc113 VGA

GRAFICA

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
GRF/01	FINGER PAINT Programma di disegno	
GRF/02	PC-KEY-DRAW	mc107 CGA
GRF/03	H&P CALENDAR	mc103
GRF/04	PC-DEMO SYSTEM	mc105
GRF/05	GRAPHICWORKSHOP	mc106
GRF/06	SOLAI & TRAVI	mc106
GRF/07	GOSTPAINT	mc112

SPREADSHEET

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
SPD/01	AS-EASY-AS	mc103
SPD/02	EXPRESS-CALC	mc104
SPD/03	EZ-SPREADSHEET Calcoli di budget	
SPD/04	INSTACALC	mc107
SPD/05	QUEBECALC Spreadsheet 3D	

UTILITY

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
UTI/01	PC-DESK-TEAM	mc107
UTI/02	HARD DISK UTILITIES Per gestire l'Hard Disk	Hard Disk
UTI/03	DOS HELP	mc104
UTI/04	DISK SPOOL II	mc103
UTI/05	LOCKTITE Protegge i file con password	
UTI/07	LHARC	mc105
UTI/08	ARJ	mc105
UTI/09	LZEXE	mc105
UTI/10	DIET	mc105
UTI/11	PKLITE	mc105
UTI/12	NEWSPACE	mc105
UTI/13	CATDISK	mc105
UTI/14	POINT&SHOOT	mc105
UTI/15	SHEZ	mc106
UTI/16	ZZAP	mc106
UTI/17	GUARDIAN ANGEL	mc107
UTI/18	STORE	mc107
UTI/19	TXT	mc107
UTI/20	xSET	mc108
UTI/21	ZAPDIR	mc108
UTI/22	UTILITY COLLECTION	mc109
UTI/23	DiR	mc109
UTI/24	CLEANUP	mc111
UTI/25	SAB DISKETTE UTILITY	mc111
UTI/26	TIF2GRAY	mc111
UTI/27	FILLDISK	mc111
UTI/28	ORASCO	mc111
UTI/29	XDIR	mc111
UTI/30	WINCOMMANDER	mc112
UTI/31	MOUSE FORMATTER	mc112
UTI/32	WINZIP	mc112
UTI/33	MOUSE EDITOR	mc113
UTI/34	DEPURA	mc113
UTI/35	DISK FATTER	mc113

VARIE

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE
VAR/01	COMPOSER Per suonare al computer e stampare lo spartito	
VAR/02	CHECK-MATE Controllo delle finanze personali	
VAR/03	PIANO-MAN	mc104
VAR/04	BARTENDER Tutti i cocktail	mc103

CODICE	TITOLO&DESCRIZIONE	REC. HARDWARE	
VAR/05	DIET DISK La dieta al computer		
VAR/06	ELEMENTARY C Per programmatori in C		
VAR/07	RECIPES	mc104	
VAR/08	PERSONAL C COMPILER	mc105	
VAR/09	MOUSE.TPU & NEWEXEC	mc106	
VAR/10	TSR, PRINT & GESTECC	mc106	
VAR/11	ARIANNA	mc106	
VAR/12	TOTOPROJET	mc108	CGA
VAR/13	COVER	mc108	
VAR/14	CODICE FISCALE	mc109	Hard disk
VAR/15	FLIGHT	mc109	
VAR/16	DIZIONARIO INFORMATICO	mc109	
VAR/17	ITALIA90	mc110	
VAR/18	TATA-BIGNOMIX UTILITY	mc110	
VAR/19	QUICK BASIC ROUTINES	mc110	
VAR/20	MICROGESTS	mc113	
VAR/21	CALCOLO INDICE ELO	mc113	
VAR/22	MENU	mc113	

WORDPROCESSOR

WPR/01	W.P.FOR CHILDREN Per insegnare ai bambini il WP	
WPR/02	FREEWORD	mc103
WPR/03	PC-WRITE	mc106
WPR/04	THESAURUS PLUS Sinonimi in inglese (TSR)	
WPR/05	GALAXY	mc104
WPR/06	EDITOR	mc110
WPR/07	NOTEBOOK	mc112
WPR/08	WORDY	mc113

AMIGA

COMUNICAZIONE

AMCO/01	AMIPAC	mc110
AMCO/02	FC FREE COMMUNICATION	mc113

DATABASE

AMDB/01	BADGER	mc113

GIOCO

AMGI/02	WELLTRIX	mc105
AMGI/03	SYS	mc105
AMGI/04	SCOPONE SCIENTIFICO	mc108
AMGI/05	LA FINE DI UN TIRANNO	mc109
AMGI/06	LA PANTERA SIAMO NOI	mc109
AMGI/07	MEGABALL	mc110

GRAFICA

AMGR/01	PRINTSTUDIO	mc104
AMGR/02	TEXTPAINT	mc105
AMGR/03	SCREENX	mc105
AMGR/04	SETPAL	mc105
AMGR/05	FREEPAINT	mc113

SPREADSHEET

AMSP/01	SPREAD	mc104
AMSP/02	EQUATIONWRITER	mc110

UTILITY

AMUT/01	MACH III	mc104
AMUT/02	RULER	mc104
AMUT/03	HEX	mc104
AMUT/04	MOM	mc104
AMUT/05	CB	mc104
AMUT/06	ZETAVIRUS	mc104
AMUT/07	DIRMASTER	mc105
AMUT/08	KDC	mc105
AMUT/09	XCOPYIII	mc105
AMUT/10	CD2TAPE	mc105
AMUT/11	BBS & LOG	mc106
AMUT/12	UTILITIES	mc106
AMUT/13	VIEW80 II	mc106
AMUT/14	MATCALC	mc106
AMUT/15	ICONMASTER	mc106
AMUT/16	HERMIT	mc106
AMUT/17	TURBO IMPLODER	mc106
AMUT/18	FONTS2PRINTER	mc107
AMUT/19	SVD	mc107
AMUT/20	MC-PROGRAMS	mc107
AMUT/21	CHP&SAVE-PREFS	mc107
AMUT/22	CIDITEIP	mc108
AMUT/23	DISKEDITOR	mc108
AMUT/24	5 UTILITY	mc108
AMUT/25	OROLOGIO PARLANTE	mc108
AMUT/26	LSLAB	mc110
AMUT/27	DIRWORK	mc111
AMUT/28	SCREENMOD	mc111
AMUT/29	SYSINFO	mc111
AMUT/30	SUPERDUPER	mc111
AMUT/31	PRFONT	mc113
AMUT/32	TG	mc113
AMUT/33	ICONS	mc113
AMUT/34	TURBOGIF	mc113

VARIE

AMVR/01	FRACTUS	mc108
AMVR/02	RUBRICA, DACIA & GESTFAT	mc109
AMVR/03	FUNZ3D	mc109
AMVR/04	PLAYSMUS	mc110
AMVR/05	MULTI PLAYER	mc111
AMVR/06	DRAWMAP	mc111
AMVR/07	TOTAMIGA	mc112
AMVR/08	AUTO	mc112
AMVR/09	SOUNDMASTER	mc112
AMVR/10	AMIGA L8	mc112
AMVR/11	FRACTAL	mc112

MACINTOSH

COMUNICAZIONE

MICO/01	RED RYDER	mc110

EDUCATIVO

MIED/01	KID PIX	mc107
MIED/02	NUMBER TALK	mc107
MIED/03	ALPHA TALK	mc107

GIOCO

MIGI/01	STELLA OBSCURA	mc106
MIGI/02	PARARENA	mc106
MIGI/03	VIDEO POKER FOR FUN	mc106
MIGI/04	SPACE STATION PHETA	mc106
MIGI/05	STRATEGO	mc106
MIGI/06	THE LAWNZAPPER	mc107
MIGI/07	MACTRIS	mc107
MIGI/08	CANFIELD	mc107
MIGI/09	YAHTZEE	mc108
MIGI/10	GLIDER	mc108
MIGI/11	MACNINJA	mc108
MIGI/12	GLIPHA	mc108
MIGI/13	MONOPOLY	mc109
MIGI/14	GOLF	mc109
MIGI/15	WHEEL	mc109
MIGI/16	GUNSHY	mc109
MIGI/17	MEGAROIDS	mc110
MIGI/18	SHUFFLEPUCK	mc110
MIGI/19	CRIMINALS	mc111
MIGI/20	SQUIX	mc112
MIGI/21	HOTEL CAPER	mc112

GRAFICA

MIGR/01	CALENDAR MAKER	mc106

SPREADSHEET

MISP/01	BIPLANE	mc112

STACK

MISK/01	FOOD 1	mc111
MISK/02	BUSINESS 1	mc111
MISK/03	SOUND 1	mc111

UTILITY

MIUT/01	OLIVER'S BUTTONS	mc107
MIUT/02	POPCHAR	mc107
MIUT/03	RAMDISK	mc108
MIUT/04	SCROLL2	mc109
MIUT/05	DECK EDITOR	mc109
MIUT/06	BANNER MAKER	mc110
MIUT/07	SPEEDOMETER	mc110
MIUT/08	LOODLE	mc112
MIUT/09	FAST FORMAT	mc112
MIUT/10	SOUND MASTER	mc112
MIUT/11	STUFFIT CLASSIC	mc112

VARIE

MIVR/01	RIDICOLO	mc108
MIVR/02	ELIZA	mc109
MIVR/03	HYPERSTAR	mc113

Compilare e spedire a: ***MCmicrocomputer***

Desidero acquistare il software di seguito elencato al prezzo di **L. 8.000 a titolo (ordine minimo: tre titoli).** Per l'ordinazione inviare l'importo (a mezzo assegno, c/c o vaglia postale) alla: Technimedia srl, Via Carlo Perrier 9, 00157 Roma.

dischetti da ☐ 3.5" ☐ 5.25"

Codici: _____

_____

Totale dischi ☐ x 8.000=Lire _____

Manuali in italiano:

☐ TSPD/01 AS EASY AS ☐ TUTI/01 HARD DISK UTILITIES

☐ TVAR/02 CHEKMATE ☐ TWPR/05 GALAXY

Totale manuali ☐ x 8.000=Lire _____

Nome e Cognome _____

Indirizzo _____

CAP/Città _____

Telefono _____

MCmicrocomputer non offre alcuna garanzia e non si assume alcuna responsabilità sugli eventuali danni diretti o indiretti derivanti dall'utilizzo del software distribuito

The Bits !

FEDERAL SOFTWARE LIBRARY
SANTACLARA, CA U.S.A.

Unlimited PD & Shareware MS-DOS

Ormai da tempo selezioniamo per Voi i migliori programmi. Ordinand presso di noi avrete la certezza di ricevere direttamente a casa Vostra l ultime versioni dei programmi immuni da Virus, ad un prezzo eccezional e con consegna immediata...

STOP **Prezzo Bloccato L. 8000/DISCO**

★ Ultima novità ✓ Best-seller
Blue Soft Software in Italiano

Giochi

✓ **PC-JIGSAW** (GA 102) Stupendo puzzle grafico; funziona su schede CGA, EGA, VGA o MCGA.

STRIKER (GA 124) Gioco tipo Defender o Top Gun, richiede scheda grafica CGA o compatibile.

✓ **FLY 2.0** (GA 142) Stupendo simulatore di volo con molte opzioni ed indicazioni in italiano. Richiede scheda CGA o compatibile.

✓ **CYRUS** (GA 160) Il programma di scacchi più apprezzato, in grafica 3D (tridimensionale). Richiede scheda grafica EGA o compatibile.

✓ **FORD SIMULATOR II** (GA 164, GA 165 - n. 2 dischi) La famosa corsa automobilistica in una nuova versione. Richiede scheda grafica CGA.

STARS DEFENSE (GA 176) Un gioco con voci digitalizzate ed effetti sonori, per difendere i missili Usa da attacchi nemici. Richiede EGA ed AT.

CRAZY SHUFFLE (GA 178) Un bel gioco di abilità, bisogna fissare delle immagini che rimangono sullo schermo per pochi secondi. Richiesti VGA e Mouse.

★ **COMMANDER KEEN** (GA 161) Simpatico arcade game, riconosce in automatico la scheda grafica utilizzata.

★ **PERESTROIKA** (GA 179) Eccezionale scacciapensieri nato nella Russia di Gorbaciov! Richiede grafica EGA.

Word Processing

✓ **EZ-FORMS EXECUTIVE** (WO 123) Questo pregevole programma per creare modulistica non ha davvero bisogno di presentazioni!

✓ **CHIWRITER** (WO 119) Famoso wordprocessor, stampa in grafica con i fonts multilingua a corredo. Richiede CGA e supporta numerose stampanti ad aghi.

✓ **PC-WRITE 3.02** (WO 101, WO 102, WO 103 - n. 3 dischi) Potente wp con mailmerge e spell-check; numerose le opzioni.

✓ **MULTILINGUAL WP** (Da WO 127 a WO 137 - n. 11 dischi) Finalmente un WP Multilingue: Lingue Europee, Arabo, Irlandese, Ebraico, Turco, Russo, Polacco, Yugoslavo, Urdu e Persiano. Bellissimo !!

Database

✓ **PC-FILE PLUS** (DA 101, DA 102, DA 103 - n. 3 dischi) Grande dBASE molto diffuso negli Stati Uniti - richiede Hard Disk.

PC-GRAPH PLUS (DA 104) Generatore grafico per PC-File Plus.

✓ **FILE EXPRESS** (DA 113, DA 114 - n. 2 dischi) Completo ma facile programma di archivio - si impara davvero velocemente.

AUDIO LIBRARY (DA 123) Un ottimo programma per la gestione della biblioteca musicale.

✓ **VIDEO PRO** (DA 125, DA 126 n.2 dischi) Professionale per gestire una raccolta di videocassette.

Spreadsheet

✓ **PC-CALC PLUS** (SP 101, SP 102, SP 103 - n. 3 dischi) Super clone di 1-2-3, interattivo con PC-File Plus.

EZ-SPREADSHEET (SP 104) Un foglio elettronico completo ma facile da usare, in linea con la qualità degli altri prodotti EZ.

✓ **AS-EASY-AS** (SP 105) Ottimo programma, best-seller negli Stati Uniti dove è uno degli spreadsheet Shareware più stimati.

✓ **PIVOT 1.01** (SP 106) Per stampare i fogli elaborati con AS-EASY-AS, con vari orientamenti e fonts. Diverse le stampanti supportate.

Grafica

PC KEY-DRAW (GR 101, GR 102, GR 103, GR 104 - n. 4 dischi) Programma di grafica multi-purpose. Richiede CGA.

✓ **PRINTSHOP GRAPHICS I** (GR 107) Libreria con oltre 200 disegni per il PrintShop (non incluso nel disco).

✓ **PRINTSHOP GRAPHICS II** (GR 136) Altri disegni per esaltare la propria creatività (PrintShop non incluso).

✓ **PRINTMASTER GRAPHICS I** (GR 108) Libreria con oltre 200 disegni per PrintMaster (non incluso).

✓ **PRINTMASTER GRAPHICS II** (GR 109/1, GR 109/2 - n. 2 dischi) Altri disegni per PrintMaster (non incluso).

✓ **IMAGE 3D 2.6** (GR 111) Pacchetto tipo CAD bidimensionale, molto ben fatto. E' configurabile per schede CGA, EGA, VGA.

✓ **GRASP** (GR 118) Per creare favolose animazioni in grafica, è un programma assolutamente da non perdere. Funziona sia in modalità CGA che EGA.

✓ **VGA CAD 1.60** (GR 156, GR 157 - n. 2 dischi) Il TOP dei programmi di Paint in VGA; carica e salva in .GIF ed in .BLD (ideale per programmatori QuickBASIC e BASIC 7.x); richiede un mouse Microsoft compatibile ed una scheda VGA compatibile.

✓ **EASY PRESENTATION GRAPHICS** (GR 172, GR 173, GR 174 - n. 3 dischi) Il miglior pacchetto di grafica commerciale in Shareware. Supporta molte schede grafiche, inclusa la VGA.

★ **FRACTINT 14.0R** (GR 220) Mai visto un programma di frattali così completo: visualizzazione dalla CGA fino a 1024x768x256 colori, salvataggio in .GIF, animazione della palette, etc. Bellissimo!

Windows 3

✓ **ICON UTILITIES** (WI 101) Due utility per disegnare e/o modificare icone colorate.

✓ **ICONS FOR WINDOWS** (WI 102) Oltre 600 icone a colori già pronte, modificabili (vedi WI 101) o da utilizzare direttamente nel Program Manager.

✓ **SCREEN SAVERS & BLANKERS** (WI 103) Una serie di utility salvaschermo: tutte offrono possibilità di disattivazione, ritardo programmabile, ed effetti grafici spettacolari.

✓ **WALLPAPER FOR WINDOWS** (WI 104) Una raccolta di bitmaps (formato .BMP) ideali come sfondi per Windows.

✓ **GAMES FOR WINDOWS 2** (WI 125, WI 126 n. 2 dischi) Nuova collezione di ottimi giochi per Windows 3.x. Ottimi, da non perdere. I dischi sono compressi, a corredo l'utilità per decomprimerli.

✓ **ICONS & ICON EDITOR FOR WINDOWS** (WI127, WI 128, WI 129, WI 130 n. 4 dischi) Una straordinaria raccolta di icone, oltre 2500 files .ICO. In più le ultime novità in fatto di Icon Editor.

★ **ATM FONTS COLLECTION VOLUME I** (WI 131, 132, 133, 134 - n. 4 dischi) Per chi possiede ATM (Adobe Type Manager), il gestore di fonts sotto Windows 3, ecco una raccolta di ben 129 fonts (sic!) suddivisi tra i volumi I, II, III e IV (che trovate di seguito); i 4 volumi possono essere ordinati anche separatamente. Per utilizzare i fonts bisogna avere Windows 3 e ATM.

★ **ATM FONTS COLLECTION VOLUME II** (WI 135, 136, 137, 138 - n. 4 dischi) Vedi sopra.

★ **ATM FONTS COLLECTION VOLUME III** (WI 139, 140, 141, 142 - n. 4 dischi) Vedi sopra.

★ **ATM FONTS COLLECTION VOLUME IV** (WI 143, 144, 145, 146 - n. 4 dischi) Vedi sopra.

Varie

✓ **THE WORLD** (ED 102) Visualizza dettagliate mappe del globo, richiede CGA o schede compatibili.

✓ **MICRO WORLD DATABANK** (ED 109, ED 110, ED 111, ED 112, ED 113 - n. 5 dischi) Mappe della terra molto dettagliate (fonte CIA), con possibilità di variare anche l'altitudine (nel caso di vista dal satellite). Include versioni per PC con e senza coprocessore matematico. Configurabile per PC XT ed AT; gestisce schede CGA ed EGA.

✓ **UNIVERSE** (ED 147) Rassegna di immagini digitalizzate (in formato .PIC IBM Storyboard) del sistema solare e dello spazio. Richiede una scheda grafica EGA o compatibile. Raccomandato.

✓ **CHEMICAL** (ED 121) Per creare i modelli 3D (tridimensionali) di molecole chimiche. Gestisce diverse schede grafiche, fino alla VGA.

✓ **CHEMVIEW** (ED 120) Ruota, ingrandisce, esamina i modelli 3D creati con Chemical (ED 121). Richiede EGA o compatibile.

✓ **ONLYONE** (BU 118, 119, 120) Programma per lo sviluppo di sistemi Totocalcio e Totip completo in tutto, ad eccezione della stampa schedine.

ADMINISTRATOR 4.36 (BU 121,122, 123 - n. 3 dischi) Dalla SIM di Napoli, la Gestione per Condomini. Completo in tutto, ad eccezione della stampa.

★ **I MINERALI** (ED 168) Foto digitalizzate (320x200x256 colori) di alcuni tra i più colorati minerali. Educativo. Richiede MCGA, VGA o SuperVGA.

★ **CAP 2.1** (BU 124) Ottimo programma di gestione dei CAP (codici di avviamento postale); questa versione Shareware contiene 500 codici, incluse tutte le province.

Utility

✓ **PC-SURVIVAL KIT** (UT 117) Preziosa raccolta di utilities per il vostro PC, assolutamente da non perdere: print to disk, data path, click della tastiera, etc.

BRADFORD 2.04 (UT 120) Stampa testi in alta qualità e con diversi fonts usando stampanti Epson compatibili (particolarmente indicato per le 9 aghi).

HARD DISK MENU IV (UT 124) Uno dei menu per hard disk più professionali, con passwords, salvaschermo ed ampia configurabilità.

✓ **THE SPACE MAKER** (UT 136) Collezione di compattatori-decompattatori (PKZIP, LHARC, PKXARC, PAK, ZOO), include ottimo ram-disk.

DISK COMMANDO (UT 177) Questo è stato unanimemente definito il miglior clone delle Norton Utilities 4.5: da non perdere, quindi.

✓ **VIRUSCAN 4.5** (UT 225) Scan, Vshield e Clean: la triade più agguerrita contro la maggior parte dei virus informatici.

✓ **LIST PLUS** (UT 179) Il più famoso e completo pacchetto per visionare files di testo di qualsiasi dimensione - raccomandato.

Programmazione

✓ **CXL** (PR 133) Una delle migliori librerie di windowin per il linguaggio C. Supporta Turbo C, Quick C, Zortec ed altri compilatori.

✓ **THE NEW TURBO PASCAL TUTOR** (PR 136, P 137 - n. 2 dischi) Nuova release per Turbo Pascal 5.0 e 5.

✓ **ADVBAS** (PR 147) Completa libreria per programmatori QuickBASIC 3.0 (80% compatibile QuickBASIC 4.x - inclusi sul disco anche tutti i moduli oggetto.

✓ **CLIPGRAF** (PR 148) Completa libreria grafica pe Clipper, con dimostrativo incluso.

Telecomunicazioni

✓ **PROCOMM PLUS** (TE 122) E' probabilmente uno de più famosi ed apprezzati pacchetti di comunicazione.

✓ **SAPPHIRE** (TE 128) Una BBS molto facile da usare ideale per chi inizia.

BIMODEM (TE 136) Questo protocollo di comunicazio ne via modem contente uploads e download contemporaneamente!

★ **ODISSEY 1.32** (TE 143) Nuovo pacchetto d telecomunicazioni, assolutamente da non perdere: supporta VIDEOTEL, emulazione software MNP5, ed altro ancora

Vietato ai Minori

✓ **XXXPRINTMASTER GRAPHICS** (AD 107) Oltr 60 disegni XXX per il PrintMaster (non incluso).

✓ **MANDY** (AD 133, AD 134, AD 135 - n. 3 dischi) I assoluto il best-seller tra le nostre animazioni made i USA, lo stato dell'arte tra le animazioni in VGA!!! (Richiede Hard Disk e scheda grafica VGA)

✓ **BEACHBALL** (AD 136, AD 137, AD 138 - n. 3 dischi Sulla scia di MANDY, un'altra realistica animazione a 256 colori (richiede HD e scheda grafica VGA)

✓ **VGA PINUPS - THE FINEST COLLECTION** (AD 141, AD 142 - n. 2 dischi) Diverse Pinups in immagini .GIF di altissima qualità (alta risoluzione 256 colori) - da non perdere. Richiesta VGA, SuperVGA o MCGA.

VGA X-RATED PIX II (AD 150, AD 151 - n. 2 dischi) Altre immagini .GIF XXX-Rated per schede EGA.

MOANA (AD176, AD 177 n.2 dischi) Storia italiana in VGA (Richiede Hard Disk e VGA).

★ **DL-VIEW IMAGES** (AD 180, AD 181, AD 182, AD 183 - n. 4 dischi) Un nuovo programma completo di svariate animazioni 320x200x256 colori (richiede MCGA, VGA o SuperVGA).

★ **XXX-GRAYSCALE IMAGES** (AD 184, AD 185, AD 186 - n. 3 dischi) Immagini mooolto ... XXX-Rated (autore D. Molina). Ultima novità. Le immagini (in formato .GIF) sono in formato 640x480x16 grigi; richiesta quindi MCGA, VGA o SuperVGA.

Come ordinare:

☎ (081) 579.91.52
(081) 579.91.51

✆ Fax: (081) 579.91.51

✉ Posta:
IDC - Via Falvo,20
80127 NAPOLI

☎ Servizio AssistenzaTelefonica *HOT-LINE* h 16.00 - 18.00

Richiedete *GRATIS* il catalogo completo

Ordinate oggi stesso i vostri programmi

Vogliate spedire i seguenti dischi (indicati dai codici) nel formato: ❑ 3" ❑ 5"

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

Totale dischi n. _____ x lire 8.000/cad. _____ = lire _____

Pagamento contrassegno (lire 10.000) _____ = lire _____

Spedizione corriere espresso 24 ore (lire 10.000) = lire _____

(Corriere **GRATUITO** per **contrassegno** oltre 10 dischi)

Spedizione contributo fisso _____ = lire 4.000 =

Totale del pagamento _____ = lire _____

NOME _____ TEL. _____

INDIRIZZO _____

C.A.P. _____ LOCALITA' _____ PV. _____

Per la fattura si specifica codice fiscale e partita IVA: _____

PAGAMENTO:

❑ ASSEGNO DI C/C NON TRASFERIBILE INCLUSO
❑ PAGAMENTO IN CONTRASSEGNO (CONTRIBUTO LIRE 10.000)
❑ VAGLIA POSTALE (RICEVUTA O FOTOCOPIA ALLEGATA)

MC1291

Info Data Communication
Via Rodolfo Falvo, 20 - 80127 NAPOLI