Programmare in C su Amiga (32)

di Dario de Judicibus (MC2120)

Eccoci finalmente a parlare di pulsanti a rilascio incrociato, un controllo composito che non fa parte di quelli standard dell'Amiga, ma che Intuition ci dà comunque la possibilità di definire, estendendo così l'interfaccia standard del sistema in modo elegante pur restando in linea con quelle che sono le caratteristiche del sistema

Nelle scorse due puntate abbiamo visto quelle che sono le due classi di pulsanti più comuni, e precisamente quelli a rilascio automatico e quelli a rilascio manuale. In questa e nella prossima puntata vedremo una tecnica per definire un gruppo di pulsanti mutualmente esclusivi, e cioè *a rilascio incrociato.*

Mentre per i pulsanti a rilascio automatico e per quelli a rilascio manuale esistono delle ben precise tecniche di sviluppo, la possibilità di costruire dei pulsanti a rilascio incrociato è solo accennata nei *ROM Kernel Manuals* della Commodore. Questi tuttavia riportano comunque una serie di regole da seguire nello sviluppare questa classe di pulsanti, al fine di evitare effetti collaterali e problemi anche alla luce di nuove versioni del sistema operativo.

La tecnica qui adoperata è quindi solo una delle tante che possono essere utilizzate per la definizione e la gestione di pulsanti mutualmente esclusivi, ed è stata da me sviluppata proprio per questa serie di articoli pubblicati da *MCmicrocomputer.* Inutile dire che essa segue tutte le raccomandazioni della Commodore relative a questa classe di pulsanti.

Qui, più che nei casi precedenti, si fa uso della possibilità di estendere la struttura **Gadget** per mezzo del puntatore **UserData**, a cui assoceremo come di consueto l'indirizzo ad una struttura **UBUT**, leggermente modificata rispetto a quella utilizzata per i pulsanti a rilascio automatico e manuale.

Rispetto alle scorse puntate c'è una novità: i *quiz.* Inframezzate al testo dell'articolo, ho introdotto un certo numero di domande, riportate in corsivo. Ogni domanda si riferisce al codice che in quel momento sto descrivendo, chiedendovi di identificare in esso errori, imprecisioni, o possibili miglioramenti. Nella prossima puntata pubblicherò le risposte a questi *quiz.*

Andiamo quindi a descrivere le due nuove funzioni **CreateXButtons()** e **DeleteXButtons()**. Nella prossima puntata vedremo la **SelectXButtons()** e la **DisplayButtons()**, che sostituisce la **DisplayGadget()** presentata in precedenza.

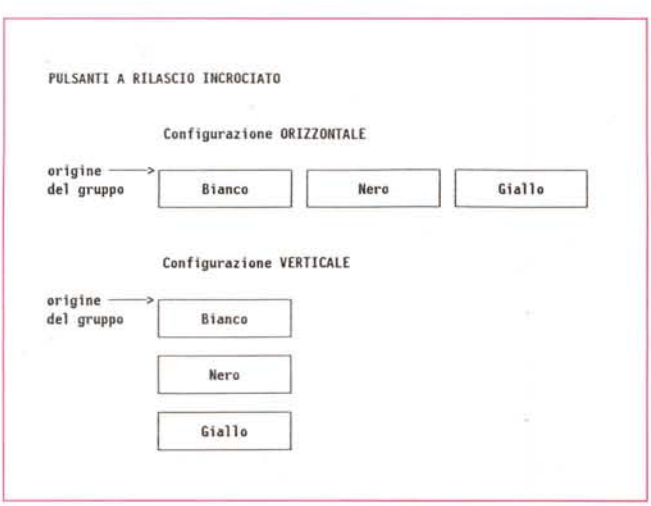

Figura 1
Configurazioni possibili.

Pulsanti a rilascio incrociato

Il criterio di base utilizzato è il seguente:

Considerare il gruppo dei pulsanti mutualmente esclusivi come un unico controllo, in cui ci sia sempre uno ed un solo pulsante selezionato, ed in cui la selezione di un altro pulsante comporti la deselezione automatica di quello selezionato fino a quel momento.

Per quello che riguarda la struttura di questo nuovo controllo, ho scelto due configurazioni fra le tante possibili: una struttura *orizzontale,* in cui tutti i pulsanti si trovano allineati lungo una retta disposta parallelamente al bordo superiore della finestra, ed una struttura *verticale,* in cui i vari pulsanti sono incolonnati uno sotto l'altro, leggermente distanziati e con i bordi laterali allineati, come si può vedere in figura 1.

In entrambi i casi i pulsanti hanno tutti le stesse dimensioni, e cioè hanno la stessa altezza, basata sulle dimensioni dei caratteri del testo, e la stessa larghezza, calcolata prendendo il testo più

Figura 2 Configurazioni orizzontali.

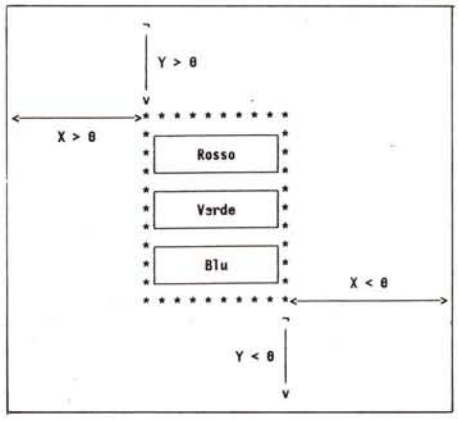

Figura 3 Posizionamento del gruppo.

lungo fra quelli associati ad i singoli pulsanti del gruppo. La scelta è stata fatta per motivi puramente estetici, e quindi soggettivi. D'altra parte, se per la configurazione verticale tale scelta porta a dare al gruppo una forma rettangolare, esteticamente più elegante, nel caso della configurazione orizzontale si potrebbe optare per una scelta tesa ad ottimizzare lo spazio disponibile, dato che comunque la forma dell'intero gruppo rimane quella di un rettangolo basso e molto lungo, come si può vedere in figura 2.

In quest'ultimo caso, tuttavia, sarà necessario modificare la funzione che crea il gruppo di pulsanti, differenziando i due casi. Lascio a voi questo semplice esercizio.

Per quello che riguarda il posizionamento del gruppo, valgono le stesse considerazioni già fatte nella 30ª puntata per gli altri pulsanti, con l'unica differenza che tale posizionamento non è relativo ad un singolo pulsante, ma al rettangolo che circoscrive il gruppo, preso come un tutt'uno.

Quindi, se le ascisse sono positive, si riferiscono alla distanza del bordo sinistro del *primo* pulsante dal bordo sinistro del contenitore. Se esse sono negative, si riferiscono alla distanza del bordo destro dell'*ultimo* pulsante dal bordo destro del contenitore. Analogamente, se le ordinate sono positive, allora si riferiscono alla distanza del bordo superiore del *primo* pulsante dal bordo superiore del contenitore, mentre se sono negative, si riferiscono alla distanza del bordo inferiore dell'*ultimo* pulsante dal bordo inferiore del contenitore (vedi figura 3).

L'unico controllo che non viene effettuato automaticamente è che il gruppo, così posizionato, rientri effettivamente tutto nella finestra.

```
/*****************************************************************************
**  Programmare in C su Amiga  (c) 1991  Dario de Judicibus - Roma (I)     **
** ------------------------------------------------------------------------ **
**  Definizioni da precompilare per generare la tabella GDGUSRH.SYM        **
*****************************************************************************/

/*
**  Costanti
*/
 #define TOGGLEON  1
 #define TOGGLEOFF 0
 #define AUTOBUTTONCLASS   0x0001
 #define TOGGLEBUTTONCLASS 0x0002
 #define ROWBUTTONGROUP    0x0003
 #define COLBUTTONGROUP    0x0004

/*
**  Tipi
*/
 typedef struct Gadget       IGDG;
 typedef struct Border       IBRD;

 typedef struct Container
 {
   struct Window *w;
   struct Requester *r;
 }
 ICNT;

 typedef struct UsrButton
 {
   USHORT Number;
   IGDG  *First, *Selected;
   USHORT Size;
 }
 UBUT;

/*
**  Macro di servizio
*/
 #define CreateAutoButton(id,txt,cont,x,y) \
    CreateButton((id),(txt),(cont),(x),(y),AUTOBUTTONCLASS,TOGGLEOFF)
 #define DeleteAutoButton(b,c) DeleteButton((b),(c))
 #define CreateToggleButton(id,txt,cont,x,y,status) \
    CreateButton((id),(txt),(cont),(x),(y),TOGGLEBUTTONCLASS,(status))
 #define DeleteToggleButton(b,c) DeleteButton((b),(c))
 #define DeleteXButtons(b,c) DeleteButton((b),(c))
 #define ToggleButtonStatus(b)   (BOOL)((b)->Flags & SELECTED)

/*
**  Prototipi delle funzioni di servizio
*/
 IGDG *CreateButton  ( USHORT, char *, ICNT *, SHORT, SHORT, USHORT, BOOL );
 void  DeleteButton  ( IGDG *, ICNT * );
 IGDG *CreateXButtons( USHORT, USHORT, char **, ICNT *, SHORT, SHORT, USHORT, USHORT );
 IGDG *SelectXButtons( IGDG *, ICNT * );
 void  DisplayButtons( IGDG *, ICNT * );
 void  RefreshWindow ( struct Window *);
```

Figura 4 - gdgwsrh.c.

Strutture e costanti

Vediamo quali nuove costanti e quali strutture è stato necessario definire per disegnare e gestire un gruppo di pulsanti a rilascio incrociato. Fate riferimento al listato riportato in figura 4.

Innanzi tutto abbiamo due nuove classi, e precisamente **ROWBUTTONGROUP** per la configurazione orizzontale, e **COLBUTTONGROUP** per quella verticale. Si è reso necessario inoltre modificare la struttura **UserButton**, già utilizzata in precedenza, aggiungendovi i puntatori a due strutture **Gadget**. **First** punta sempre al primo pulsante del gruppo, mentre **Selected** punta sempre al pulsante che al momento si trova nello stato *selezionato*. In questo modo, in pratica, ogni pulsante del gruppo sa come trovare questi due pulsanti. Il primo serve infatti quando si vuole operare sull'intero gruppo, dato che di fatto il puntatore a tale pulsante rappresenta anche il puntatore all'intero gruppo. Ad esempio, quando si deve rimuovere il gruppo per poter effettuare qualche cambiamento ai dati ad esso associati. Il secondo serve quando l'utente seleziona un altro pulsante, e quindi la procedura di gestione del gruppo deve deselezionare quello precedentemente selezionato. Una raccomandazione: *se dovete effettuare una qualunque modifica ad un campo in una delle strutture che definiscono un pulsante, dovete prima rimuovere il pulsante dalla lista dei controlli, effettuare la modifica in questione, e quindi raggiungere quel pulsante nella lista.*

Questo non vale solo per i pulsanti a rilascio incrociato, ma per qualsiasi altro pulsante gestito da Intuition.

CreateXButtons()

Questa è la funzione che definisce le varie strutture necessarie ad Intuition per disegnare il gruppo di pulsanti mutualmente esclusivi. La struttura è analoga a quella della **CreateButton()**, già utilizzata per i pulsanti a rilascio automatico e manuale. A differenza di quest'ultima, tuttavia, sono presenti una serie di cicli [*loop*] dato che in effetti dobbiamo definire più di un pulsante.

Vediamo ora in dettaglio la procedura.

Per prima cosa vediamo quali parametri vanno passati per definire il gruppo dei pulsanti. Sono otto, e precisamente:

id
l'identificativo del gruppo, dal quale vengono ricavati gli identificativi dei singoli pulsanti;
txt
il puntatore ai testi da visualizzare nei singoli pulsanti;
container
il puntatore alla solita struttura *contenitore*;
x,y
la posizione del contenitore del gruppo, secondo i criteri precedentemente riportati;
class
la classe relativa alla configurazione scelta per il gruppo, e cioè se a pulsanti allineati oppure incolonnati;
button
il numero d'ordine del pulsante che va inizialmente selezionato (e cioè se il primo, il secondo, il terzo, e così via).

QUIZ #1: nel codice riportato in figura 5 mancano alcuni controlli che vanno assolutamente effettuati per prevenire un utilizzo improprio della funzione, e garantire così che esso venga eseguito correttamente. Quali sono queste verifiche?

Un paio di considerazioni aggiuntive. Dato che il gruppo di pulsanti a rilascio incrociato è un controllo composito, è necessario assegnare *ad ogni singolo pulsante* del gruppo un proprio identificativo. Intuition infatti non riconosce il nuovo controllo come un singolo oggetto, ma vede ogni pulsante come una entità a sé stante, e quindi passa alla funzione che gestisce i messaggi relativi alla selezione di un pulsante del gruppo l'identificativo di quel messaggio, e non quello del gruppo. Quando definite l'identificativo del gruppo, quindi, riservatevi un certo numero di identificativi a partire da quello, pari almeno al numero di pulsanti nel gruppo. Meglio ancora, nel caso che in una versione successiva del vostro programma decidiate di aggiungere altri pulsanti al gruppo, lasciatevi almeno un 50% di identificativi in più. Non si sa mai.

A questo proposito, provate a scrivere una funzione che verifichi se un certo pulsante fa parte di un gruppo o meno. In figura 6 ho riportato le specifiche della funzione in questione.

La seconda considerazione riguarda i testi da associare ad i singoli pulsanti. Al contrario di quanto avveniva nella **CreateButton()**, qui viene passato il puntatore ad un vettore di stringhe di caratteri, piuttosto che quello ad una singola stringa. Attenzione quindi ad assicurarvi che tale vettore contenga un numero di elementi uguale (o superiore)

```
/*****************************************************************************
** CreateXButtons               FUNZIONE              Versione 1.00       **
**                                                                         **
** Funzione fornita:  crea dinamicamente un gruppo di pulsanti             **
**                    mutualmente esclusivi                                **
**                                                                         **
** Dati in ingresso:  id          identificativo del gruppo                **
**                    n           numero di pulsanti                       **
**                    txt         vettore di testi                         **
**                    container   contenitore (finestra e/o quadro)        **
**                    x, y        posizione del gruppo nel contenitore     **
**                    class       classe del pulsante (estensione)         **
**                    buton       pulsante inizialmente selezionato        **
**                                                                         **
** Dati in uscita:    Gdg         puntatore al gruppo                      **
**                                                                         **
** Dati globali:      User        memorizza in Size la memoria utilizzata  **
**                                                                         **
** Eventi emessi:     GADGETDOWN                                           **
**                                                                         **
*****************************************************************************/
IGDG *CreateXButtons(id,n,txt,container,x,y,class,buton)
  USHORT id ;
  USHORT n ;
  char *txt[] ;
  ICNT *container ;
  SHORT x, y ;
  USHORT class ;
  USHORT buton ;
{
 /*
 **  Allocheremo due aree dati per ogni pulsante:
 **
 **  -- una struttura Gadget
 **  -- una struttura IntuiText
 **
 **  e sei aree dati per l'intero gruppo:
 **
 **  -- quattro strutture Border
 **  -- due vettori da cinque coordinate
 **
 */
  IGDG *Gdg ;
  ITXT *Txt ;
  IBRD *InnBrdUns, *InnBrdSel, *OutBrdUns, *OutBrdSel ;
  UBUT *User ;
  SHORT *InnCoord, *OutCoord ;
  USHORT GdgSize, TxtSize, BrdSize, CrdSize, UsrSize, TotSize ;
  SHORT i, l, h;
  USHORT n1 ;

  n1 = n - 1;

5a
```

Figura 5 CreateXButtons().

```
/*
**  Calcola le dimensioni delle varie aree
*/
 GdgSize = sizeof(IGDG)*n ;
 TxtSize = sizeof(ITXT)*n ;
 BrdSize = sizeof(IBRD) ;
 CrdSize = sizeof(SHORT)*10 ;
 UsrSize = sizeof(UBUT)*n ;
 TotSize = GdgSize + TxtSize + 4*BrdSize + 2*CrdSize + UsrSize ;

/*
**  Alloca memoria per il pulsante e le strutture collegate
*/
 Gdg = (IGDG *)AllocMem(TotSize, GDGMEM) ;
 if (Gdg == NULL) return(NULL) ;
 Txt       = POINTER( ITXT, Gdg,       GdgSize) ;
 InnBrdUns = POINTER( IBRD, Txt,       TxtSize) ;
 InnBrdSel = POINTER( IBRD, InnBrdUns, BrdSize) ;
 OutBrdUns = POINTER( IBRD, InnBrdSel, BrdSize) ;
 OutBrdSel = POINTER( IBRD, OutBrdUns, BrdSize) ;
 InnCoord  = POINTER(SHORT, OutBrdSel, BrdSize) ;
 OutCoord  = POINTER(SHORT, InnCoord , CrdSize) ;
 User      = POINTER( UBUT, OutCoord , CrdSize) ;

/*
**  Per ogni pulsante del gruppo...
*/
 for (i = 0, l = 0, h = 0; i < n; i++)
 {
  /*
  **  Assegna i campi fissi della struttura Gadget
  */
   Gdg[i].NextGadget = &Gdg[i+1] ;     /* Pulsante successivo nel gruppo  */
   Gdg[i].GadgetID   = id + i ;        /* Identificativo del controllo    */
   Gdg[i].GadgetType = BOOLGADGET ;    /* Tipo di controllo               */
   Gdg[i].Flags      = GADGHIMAGE ;    /* Evidenziato da un bordo         */
   Gdg[i].Activation = GADGIMMEDIATE   /* Mi interessa sapere quando è    */
                     | TOGGLESELECT ;/* premuto. Va subito selezionato  */
   Gdg[i].MutualExclude = NULL ;       /* Non utilizzato -- per sicurezza */
   Gdg[i].SpecialInfo   = NULL ;       /* Non utilizzato -- per sicurezza */
   Gdg[i].GadgetText    = &Txt[i] ;
   Gdg[i].GadgetRender = (APTR)InnBrdUns ;
   Gdg[i].SelectRender = (APTR)InnBrdSel ;
   Gdg[i].UserData     = (APTR)&User[i] ;       /* Struttura di servizio */
   User[i].Size        = TotSize ;  /* Qui metto quanta memoria ho preso */
   User[i].Number      = n ;        /* Numero di pulsanti nel gruppo     */
   User[i].First       = &Gdg[0] ;     /* Primo pulsante del gruppo      */
   User[i].Selected    = &Gdg[buton] ;/* Pulsante selezionato nel gruppo */
```

5b

```
/*
**  Definiamo ora la posizione di OGNI pulsante se il gruppo è disposto
**  orizzontalmente.
*/
 if (class == ROWBUTTONGROUP)
 {
   for (i = 0; i < n; i++)
   {
     if (x > 0)
     {
       Gdg[i].LeftEdge = Gdg[0].LeftEdge + i*(Gdg[0].Width + ROWGAP);
       Gdg[i].Flags    = Gdg[0].Flags    ;
     }
     else
     {
       Gdg[i].LeftEdge = Gdg[n1].LeftEdge - (n1-i)*(Gdg[0].Width + ROWGAP);
       Gdg[i].Flags    = Gdg[n1].Flags   ;
     }
     Gdg[i].TopEdge = Gdg[((y>0)?0:n1)].TopEdge ;
   }
 }

/*
**  Definiamo ora la posizione di OGNI pulsante se il gruppo è disposto
**  verticalmente.
*/
 if (class == COLBUTTONGROUP)
 {
   for (i = 0; i < n; i++)
   {
     if (y > 0)
     {
       Gdg[i].TopEdge = Gdg[0].TopEdge + i*(Gdg[0].Height + COLGAP);
       Gdg[i].Flags   = Gdg[0].Flags   ;
     }
     else
     {
       Gdg[i].TopEdge = Gdg[n1].TopEdge - (n1-i)*(Gdg[0].Height + COLGAP);
       Gdg[i].Flags   = Gdg[n1].Flags   ;
     }
     Gdg[i].LeftEdge = Gdg[((x>0)?0:n1)].LeftEdge ;
   }
 }
```

5d

```
  /*
  **  Il pulsante fa parte di un quadro ?
  */
   if (container->r) Gdg[i].GadgetType |= REQGADGET ;

  /*
  **  Associa al pulsante il suo testo
  */
   Txt[i] = textModel ;              /* Copia il prototipo del controllo */
   Txt[i].IText = (UBYTE *)txt[i] ;  /* Copia il testo vero e proprio    */
   l = max(l,ITXTL(&Txt[i])) ;               /* Lunghezza massima finora */
   h = max(h,container->w->RPort->TxHeight) ; /* Altezza massima finora  */
 }
 Gdg[n1].NextGadget = NULL ;          /* Ultimo pulsante del gruppo      */

/*
**  Alloca i campi variabili della struttura Gadget
*/
 for (i = 0; i < n; i++)
 {
   Txt[i].LeftEdge = l/8 ;           /* Ascissa del testo nel controllo  */
   Txt[i].TopEdge  = h/3 ;           /* Ordinata del testo nel controllo */
   Gdg[i].Width    = l + 2*Txt[i].LeftEdge ;
   Gdg[i].Height   = h + 2*Txt[i].TopEdge ;
 }

/*
**  Definiamo ora la posizione del GRUPPO, cioè solo del PRIMO pulsante,
**  o dell'ULTIMO, a seconda da dove prendo le misure.
**  ------------------------------------------------------------------
**  Per rendere più semplice la vita al programmatore, se un campo è
**  negativo, lo consideriamo una coordinata rispetto al bordo del
**  controllo PIU' VICINO a quello da cui si parte a misurare.
*/
 if (x > 0)                        /* Ascissa rispetto al bordo...       */
 {
   Gdg[0].LeftEdge = x + container->w->BorderLeft ;      /* ...sinistro  */
 }
 else
 {
   Gdg[n1].LeftEdge = x - container->w->BorderRight - Gdg[n1].Width ;
   Gdg[n1].Flags |= GRELRIGHT ;                          /* ...destro    */
 }
 if (y > 0)                        /* Ordinata rispetto al bordo...      */
 {
   Gdg[0].TopEdge  = y + container->w->BorderTop ;       /* ...superiore */
 }
 else
 {
   Gdg[n1].TopEdge  = y - container->w->BorderBottom - Gdg[n1].Height ;
   Gdg[n1].Flags |= GRELBOTTOM ;                         /* ...inferiore */
 }
```

5c

```
/*
**  Definisci le strutture bordo (valgono per TUTTI i pulsanti)
*/
 *InnBrdUns = rectModel ;
 *InnBrdSel = rectModel ;
 *OutBrdUns = rectModel ;
 *OutBrdSel = rectModel ;

 InnBrdUns->FrontPen = 2; /* Se il pulsante non è selezionato usa un   */
 InnBrdSel->FrontPen = 2; /* colore diverso da quello dei pulsanti a   */
 OutBrdUns->FrontPen = 0; /* rilascio automatico. Se è selezionato usa */
 OutBrdSel->FrontPen = 3; /* un bordo a due colori per evidenziarlo.   */

 InnBrdUns->NextBorder = OutBrdUns ;
 InnBrdSel->NextBorder = OutBrdSel ;
 OutBrdUns->NextBorder = NULL ;
 OutBrdSel->NextBorder = NULL ;

 InnBrdUns->XY = InnCoord ;
 InnBrdSel->XY = InnCoord ;
 OutBrdUns->XY = OutCoord ;
 OutBrdSel->XY = OutCoord ;

/*
**  Definisci i vettori di coordinate
*/
 InnCoord[2] = Gdg[0].Width - 1 ;  InnCoord[4] = Gdg[0].Width - 1 ;
 InnCoord[5] = Gdg[0].Height - 1 ; InnCoord[7] = Gdg[0].Height - 1 ;

 for (i=0; i<10; i++)
   OutCoord[i] = InnCoord[i] + (InnCoord[i] ? 1 : -1) ;

/*
**  Selezioniamo in partenza il pulsante specificato
*/
 Gdg[buton].Flags |= SELECTED ;

/*
**  Fatto. Il pulsante è pronto.
*/
 return (Gdg) ;
}
```

5e

```
BOOL isButtonInGroup ( button, id )
     IGDG   *button ;
     USHORT id      ;
```

Scopo:	verifica se un certo pulsante appartiene o meno ad un gruppo di pulsanti
Ingresso:	puntatore al pulsante ed identificativo del gruppo
Uscita:	vero (TRUE) se il pulsante appartiene al gruppo, falso (FALSE) in caso contrario

Figura 6 - isButtonInGroup().

```
/*****************************************************************************
** DeleteXButtons()          MACRO                 Versione 1.00             **
**                                                                           **
** Funzione fornita:  cancella dinamicamente un gruppo di pulsanti           **
**                    mutualmente esclusivi.                                 **
**                                                                           **
** Dati in ingresso:  button      puntatore al pulsante                      **
**                    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 pulsanti          **
**                                                                           **
*****************************************************************************
**                                                                           **
** Definita in:  gdgusrh.c                                                   **
**                                                                           **
** #define DeleteXButtons(b,c) DeleteButton((b),(c))                        **
**                                                                           **
*****************************************************************************/
```

Figura 7 - DeleteXButtons().

a quello passato nel parametro **n**, altrimenti correte il rischio di associare al testo di un pulsante un'area di memoria a caso, magari lunga svariate centinaia di byte prima che venga incontrato un byte nullo che segnali la fine della falsa stringa.

E torniamo ora alla **CreateXButtons().**

La prima parte è analoga a quella della **CreateButton().** Serve cioè ad allocare la memoria necessaria ad allocare le singole strutture dati necessarie a definire il gruppo. Notate come la tecnica adottata rivela ora tutta la sua potenza permettendo di effettuare una sola allocazione di memoria a fronte dell'allocazione di un numero elevato di strutture. Si tratta di una tecnica da me sviluppata quando ho scritto la prima versione del programma **gadgets** e che oramai applico spesso nei miei programmi.

Il blocco di codice seguente contiene un ciclo che assegna ad ogni pulsante quei campi che sono comuni a tutti i pulsanti (come **Flags** o **GadgetRender**), l'identificativo del pulsante (calcolato a partire da quello di gruppo), ed il puntatore al testo associato a quel pulsante. Ogni pulsante, poi, viene agganciato a quello successivo nel gruppo, utilizzando il campo **NextGadget**.

Come si può vedere in figura, al campo **Activation** è stato assegnato il valore

GADGETIMMEDIATE|TOGGLESELECT

Ora, se ben ricordate quanto detto tre puntate fa, esistono delle precise raccomandazioni della Commodore a questo riguardo. Queste consigliano di utilizzare il valore **GADGETIMMEDIATE** ma *non* il valore **TOGGLESELECT**. Ora, è vero che seguire una raccomandazione da parte della *casa madre* è importante, ma è pur vero che è fondamentale per un programmatore capire il perché della stessa, in modo da utilizzarla nel modo migliore, ed avere la possibilità di valutare i rischi che si corrono od i problemi che si possono incontrare ignorandola.

Saper programmare non vuol dire infatti seguire pedissequamente un insieme di regole ed applicare alla lettera algoritmi definiti da qualcun altro, ma capire a fondo un problema in un *contesto ben definito*, ed individuare una soluzione che risolva tale problema in modo efficace, e con un buon rapporto tra efficienza e costi.

Vediamo quindi di capire il perché la Commodore consiglia l'utilizzo di certi valori, e perché ho deciso di seguire solo parzialmente tali raccomandazioni.

Per quello che riguarda il valore **GADGETIMMEDIATE** il motivo è sostanzialmente il seguente. Quando un pulsante del gruppo viene premuto, il programma che riceve da Intuition tale informazione, prima di effettuare qualunque modifica alle strutture relative al gruppo al fine di gestirne un corretto funzionamento, deve rimuovere tali strutture dalla lista dei controlli, per evitare seri problemi ad Intuition.

Dopo tutto non si può certo pretendere che uno continui ad usare la propria automobile mentre il meccanico ne sta riparando i freni, no? Ora, almeno sotto 1.3, rimuovere un controllo attivo può creare dei problemi. Supponiamo di aver specificato **RELVERIFY** invece di **GADGETIMMEDIATE**. Allora un utente può tenere premuto un certo pulsante per un periodo lungo senza che il programma lo sappia, dato che è il rilascio del pulsante ad emettere il messaggio. Supponiamo ora che l'utente faccia un *doppio click* sul pulsante, tenendolo poi premuto sul secondo *click*. Il primo *click* scatena l'evento **RELVERIFY**, la procedura di gestione del gruppo di pulsanti a rilascio incrociato viene attivata e, ovviamente, cerca di rimuovere tutti i pulsanti del gruppo, uno dei quali, però (proprio quello che ha fatto emettere il messaggio da Intuition) è tuttora attivo e premuto. E questo crea seri problemi ad Intuition. Usando viceversa solo **GADGETIMMEDIATE**, il controllo rimane attivo solo all'istante della selezione, e non per tutto il tempo che l'utente lo tiene premuto.

In quanto al valore **TOGGLESELECT**, c'è tuttora un dibattito in corso nel mondo dei programmatori Amiga. Il motivo iniziale che ha portato la Commodore a consigliare l'utilizzo del *rilascio automatico* per i pulsanti mutualmente esclusivi, consiste nel fatto che di tali pulsanti si sa sempre lo stato dopo che l'utente li ha selezionati, mentre nel caso dei pulsanti a *rilascio manuale* bisogna controllare il campo **Flags**, per verificarne effettivamente lo stato. In pratica è più semplice gestire i primi che i secondi. Non esiste tuttavia una vera controindicazione all'uso di **TOGGLESELECT** per i pulsanti a rilascio incrociato. Ed è proprio questo il motivo che mi ha spinto ad utilizzare questa tecnica di rilascio. Volevo vedere effettivamente a che tipo di problemi potevo andare incontro. Al momento non ne ho riscontrato alcuno. Se poi qualcuno dovesse invece trovarsi di fronte ad un comportamento anomalo imputabile alla **TOGGLESELECT**, gli sarò grato se vorrà informarmi a riguardo, e con me molti altri che stanno tuttora analizzando la cosa negli Stati Uniti ed in altri paesi.

QUIZ #2: esiste un modo per risparmiare memoria ed ottimizzare così l'insieme delle strutture che definiscono il gruppo

Il ciclo calcola anche la lunghezza e l'altezza massima dei singoli testi associati ai pulsanti del gruppo.

QUIZ #3: nel codice contenuto in questo ciclo, c'è un'istruzione che potrebbe essere tranquillamente portata fuori ciclo. Quale?

A questo punto c'è un secondo ciclo, che associa ad ogni pulsante la stessa

```
/*************************************************************************
** DeleteButton()              FUNZIONE          Versione 1.00          **
**                                                                      **
** Funzione fornita:  cancella dinamicamente un pulsante a rilascio     **
**                    automatico.                                       **
**                                                                      **
** Dati in ingresso:  button      puntatore al pulsante                 **
**                    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 pulsanti     **
**                                                                      **
*************************************************************************/
void DeleteButton(button,container)
  IGDG *button ;
  ICNT *container ;
{
 /*
 **  Rimuovi il pulsante dal contenitore
 */
  (void)RemoveGList(container->w,button,
                    ((UBUT *)(button->UserData))->Number);
  RefreshWindow(container->w) ;

 /*
 **  Cancella la memoria per il pulsante e le strutture collegate
 */
  FreeMem(button,(((UBUT *)(button->UserData))->Size)) ;
}
```

Figura 8 - DeleteButton().

```
/*************************************************************************
**  Prima possibilità: vettore definito in MyInput()                   **
**  ------------------------------------------------------------------ **
**  NOTA: al posto di 256, sarebbe consigliabile utilizzare la         **
**        costante BUFSIZ definita in stdio.h od il valore esterno     **
**        _bufsize nel caso si usi il Lattice 5.xx                     **
*************************************************************************/
 char *MyInput()
 {
     char stringa[256] ;
     return (gets(stringa)) ;
 }


/*************************************************************************
**  Seconda possibilità: allocazione dinamica                          **
**  ------------------------------------------------------------------ **
**  NOTA: al posto di 256, sarebbe consigliabile utilizzare la         **
**        costante BUFSIZ definita in stdio.h od il valore esterno     **
**        _bufsize nel caso si usi il Lattice 5.xx                     **
*************************************************************************/
 char *MyInput()
 {
     char *stringa ;
     stringa = (char *)calloc(1, 256) ;
     return (gets(stringa)) ;
 }


/*************************************************************************
**  Terza possibilità: vettore esterno                                 **
*************************************************************************/
 char *MyInput(stringa)
 char *stringa ;
 {
     return (gets(stringa)) ;
 }
```

Figura 9 - MyInput(). ▶

lunghezza ed altezza, e centra il testo.

QUIZ #4: nel codice contenuto in questo ciclo, c'è un errore che altera il risultato che si vorrebbe ottenere. Qual è?

Il blocco successivo calcola la posizione del primo e/o dell'ultimo pulsante, a seconda del segno delle coordinate passate per il posizionamento del gruppo. Da notare che il codice in questione non calcola necessariamente la posizione completa di uno dei due pulsanti estremi. Infatti, se ad esempio l'ascissa è positiva ma l'ordinata è negativa, viene calcolata l'ascissa del primo pulsante, e l'ordinata dell'ultimo.

Per questo, quando poi si calcola la posizione completa di tutti gli altri pulsanti del gruppo a partire da questi dati, vengono inclusi nel ciclo successivo anche i due pulsanti estremi. Tale calcolo è impostato in due blocchi di codice separati, a seconda della configurazione scelta.

Nell'ultimo blocco di codice si costruiscono le strutture bordo. La scelta dei colori in questo caso è stata la seguente:

il bordo interno ha sempre lo stesso colore (registro #2), sia che il pulsante sia stato selezionato, sia che sia nello stato non selezionato, mentre il bordo esterno è trasparente (registro #0) per i pulsanti rilasciati, mentre ha un colore differente (registro #3) per quello selezionato.

Il risultato finale è quello di avere tutti i pulsanti del gruppo rappresentati da una cornice di un certo colore che incornicia il testo, salvo quello selezionato che in più ha una seconda cornice di colore differente che circonda la prima. Selezionando un altro pulsante, la cornice di selezione si sposta su quest'ultimo, e lì rimane fino a che non viene effettuata una nuova selezione.

L'ultima istruzione serve appunto a selezionare inizialmente uno dei pulsanti del gruppo, dato che in questo ci deve essere sempre un pulsante selezionato.

QUIZ #5: notate come questa istruzione si trovi dopo i due blocchi di codice che posizionano i vari pulsanti, e non prima. Perché?

DeleteXButtons()

Questa macro (vedi figura 7) permette di chiamare la funzione **DeleteButton().** Quest'ultima è stata leggermente modificata rispetto alla versione precedente, come riportato in figura 8, in modo da poter gestire anche gruppi di pulsanti, e non più solo singoli pulsanti. In pratica ora la funzione **DeleteButton()** ricava da **UserData** anche il numero di pulsanti che costituiscono il gruppo, come già faceva in precedenza per le dimensioni dell'area di memoria utilizzata per definire i pulsanti. Ovviamente, nel caso di singoli pulsanti, tale numero sarà uguale ad uno. I pulsanti di un gruppo, infatti, sono aggiunti in sequenza alla lista dei controlli appartenenti ad un certo contenitore. *Per questo, se utilizzate questo nuovo tipo di controllo, aggiungete sempre i vostri altri controlli in fondo alla lista, per evitare di inframezzare fra i pulsanti di un gruppo controlli che non hanno niente a che vedere con quest'ultimo. Se non fate così, correte il rischio di cancellare anche quest'ultimi quando cancellate il gruppo in questione.*

Conclusione

Nella prossima puntata vedremo altre due funzioni relative ai pulsanti a rilascio incrociato. La prima serve a gestire gli eventi emessi a fronte della selezione di un pulsante del gruppo da parte dell'utente.

La seconda è una nuova versione della vecchia **DisplayGadget(),** già presentata in precedenza in questa rubrica, modificata per supportare sia pulsanti singoli, che gruppi di pulsanti. Vedremo inoltre come gestire questo nuovo controllo composito in **gdgmain.c** ed, ovviamente, daremo la risposta ai vari *quiz* presentati in questa puntata. Come sempre, buon lavoro! MC

PARTNERS DECISI & CONCRETI

che garantiscono certezze di lavoro nel mondo ATARI.

L' evoluzione continua:

AT-SPEED C16, 80286 a 16Mhz, zoccolo per coprocessore *matematico, norton 8.2, CGA, HGC, EGAmono, VGAmono, OLIVETTI, anche con DOS DR5.*

AT-SPEED 80286 a 8 Mhz, *norton 6.7, CGA, HGC, EGAmono, VGAmono, OLIVETTI.*
PC-SPEED 8086 a 8 Mhz, *norton 4.2, CGA, HGC, OLIVETTI, ATT.*

Rivoluzione nel Desktop Video: C H I L I VISION

Digitalizzatore video a colori in tempo reale, scheda grafica da 512x512 a 1024x512 con 65535 colori contemporanei, Genlock professionale, software per titolazione, animazione, effetti speciali video real time, supporto scanner e stampanti a colori, uscita RGB e (opz.) CVBS e SVHS.

Problemi di risoluzione?

Schede grafiche ***MATRIX,*** *da 640x400 con 256 colori contemporanei fino a 1024x1024 con 16 Milioni di colori contemporanei, anche collegabile a Genlock.*
OVERSCAN, *eliminate il fastidioso bordo nero del vostro monitor.*
MEGASCREEN, 864x632 *con solo 199.000 lire!!*
TITAN REFLEX CARD, *la scheda rivoluzionaria, 1024x800 senza cambiare il vostro SM124!*

HANDY SCANNER *Logitech 400dpi, 32grigi, 105mm completo di soft di scansione e software OCR.*

SCANNER PANASONIC
Tutti i modelli, con interfacce e soft per ST/TT

SCANNER EPSON a colori
Completi di interfacce e soft per ST/TT

***** Espansioni interne per ATARI PC-FOLIO, da 256Kb a 512 Kb *****
E per il vostro ***PC-FOLIO,*** *moduli eprom esterni fino a 2Mb, interfaccia per disk drive e tanto software.*
Linguaggi ? TURBO C, PASCAL, FORTRAN, GFA BASIC per ST, AMIGA, DOS, XENIX.

Desktop Publishing

CRANACH STUDIO, CRANACH VEKTOR, *la massima espressione nel software di elaborazione immagini a colori raster e vettoriali, separazione colore, uscita anche Postscript.*
ARABESQUE PROFESSIONAL, *grafica bitmapped e vettoriale CVG, curve di Bezier, manipolazione immagini avanzata.*
GMA-PLOT, *software di gestione per plotter da taglio, ottimizzazione piano di lavoro, fino a 300 fonts in linea, import grafica vettoriale.*
Publishing Partner Master, *il software professionale per l'impaginazione elettronica, separazione colori, uscita Postscript e Linotronic, compatibilità fonts ADOBE type 1 e Compugraphics.*
MEGAPAINT II, *grafica bitmapped, lavoro in risoluzione reale, biblioteca oggetti, compatibile norme DIN.*
CONVECTOR, *programma di vettorizzazione automatica, velocissimo e preciso, ideale per tutte le applicazioni DTP.*
AUGUR, *il software OCR professionale, presto anche in versione DOS.*

EuroSoft
Hardware & Software
via del Romito 1D-r
50134 FIRENZE
tel. 055-49.64.55 Fax 055-47.49.59

Espansioni di memoria per tutti i modelli ST da 2Mbytes a 16 Mbytes.

Hard Disk SCSI per ATARI ST interni/esterni da 40 a 650 Mbytes.