Programmare in C su Amiga (29)

di Dario de Judicibus (MC2120)

Introdotte le strutture e le costanti necessarie a definire un qualunque tipo di controllo, affrontiamo in questa puntata l'analisi in dettaglio dei pulsanti. Vedremo le tre tipologie più comuni di pulsanti, e quali possibilità ci offre Intuition di inventarne di nuove. Anche la Scheda Tecnica di questo mese riporta cinque comandi della versione 1.3 dell'AmigaDos

Abbiamo visto nelle ultime quattro puntate le strutture che servono per definire di fatto qualunque tipo di controllo, e cioè: **IntuiText, Border, Image** ed ovviamente **Gadget**. Vediamo adesso il primo dei tre tipi base di controlli, e cioè i pulsanti. Vedremo come anche questi possono suddividersi in tre sottotipi, e cioè *pulsanti a rilascio automatico, pulsanti a rilascio manuale* e *pulsanti a rilascio incrociato*. Non dimentichiamoci tuttavia che, se da un lato Intuition non ci mette a disposizione un pacchetto di controlli preconfezionati, come avviene con il *Mac* o con il *Next*, dall'altra esso ci dà la possibilità di creare controlli molto vari e personalizzati. I tipi di pulsanti presentati in questa puntata quindi, non intendono rappresentare una descrizione esaustiva delle possibilità offerte dal sistema in questo campo, ma solo tracciare una strada che può offrire al programmatore ancora molte svariate opportunità.

I pulsanti

Un pulsante non è altro che un controllo che può essere selezionato con il cursore del mouse, ed è generalmente rappresentato come un rettangolino che contiene un testo che identifica lo scopo del pulsante (ad esempio, **Cancella** oppure **Va bene!**).

In genere i pulsanti sono del tipo a *rilascio automatico [hit-select]*, cioè rimangono selezionati (e quindi evidenziati secondo la tecnica che si è deciso di usare) solo fintanto che il bottone sinistro del mouse è premuto con il cursore posizionato nell'area di selezione del controllo.

Specificando TOGGLESELECT nel campo **Activation** della struttura **Gadget**, tuttavia, si ottiene un pulsante *a rilascio manuale*, per il quale è necessario fare di nuovo click con il mouse sul controllo per deselezionarlo. La differenza è la stessa che esiste tra quegli interruttori per lampada che ogni volta che vengono premuti ritornano fuori da soli, tanto che non è possibile capire se si è accesa o spenta la luce solo a partire dalla posizione dell'interruttore, e quelli in cui la pressione fa abbassare il pulsante, per cui è necessario premere di nuovo per farlo uscire fuori ed, ovviamente, spegnere la lampadina. Questo esempio è importante, perché fa capire come lo stato *selezionato/non selezionato* del pulsante non ha necessariamente a che vedere con quello che il programma fa a fronte dell'evento *il pulsante è stato premuto*.

Quali messaggi vengono notificati da Intuition al programma dipende da due fattori, e precisamente:

- quali eventi è stato detto ad Intuition di trasmettere, e
- quali eventi la porta IDCMP della finestra relativa al controllo selezionato è pronta ad accettare.

Consideriamo innanzi tutto il secondo punto e limitiamoci ai due eventi **GADGETDOWN** e **GADGETUP**. Vedremo in seguito un terzo evento: **FOLLOWMOUSE**. Come certamente ricorderete, è possibile specificare per ogni finestra, o meglio, per ogni porta IDCMP relativa ad una finestra, un filtro di selezione che lascia passare solo alcuni degli eventi fra tutti quelli che Intuition può trasmettere. Tale filtro si definisce tramite il campo **IDCMPFlags** quando si definisce la struttura **NewWindow** relativa alla finestra in questione, e può essere successivamente modificato tramite la funzione **ModifyIDCMP()**. Quindi, se vogliamo ricevere l'evento **GADGETDOWN** quando l'utente seleziona il pulsante con il mouse, dobbiamo aggiungere questo evento a quelli speci-

```
struct Gadget
{
  struct Gadget    *NextGadget    ; /* Controllo successivo nella lista     */
  SHORT             LeftEdge      ; /* [ Dimensioni e posizione dell'area ] */
  SHORT             TopEdge       ; /* [ di selezione del controllo, in   ] */
  SHORT             Width         ; /* [ valori assoluti o relativi come  ] */
  SHORT             Height        ; /* [ specificato nel campo Flags      ] */
  USHORT            Flags         ; /* Attributi del controllo - vedi testo */
  USHORT            Activation    ; /* Modi di attivazione e selezione      */
  USHORT            GadgetType    ; /* Tipo di controllo                    */
  APTR              GadgetRender  ; /* Puntatore alla grafica del controllo */
  APTR              SelectRender  ; /* Come sopra, ma in caso di selezione  */
  struct IntuiText *GadgetText    ; /* Eventuale testo associato            */
  LONG              MutualExclude ; /* < Riservato per usi futuri           */
  APTR              SpecialInfo   ; /* Estensione della struttura (tipi)    */
  USHORT            GadgetID      ; /* Per identificare il controllo        */
  APTR              UserData      ; /* Eventuali dati utente                */
};
```

Figura 1
La struttura Gadget.

ficati nel campo **IDCMPFlags**. Lo stesso dicasi per **GADGETUP**, emesso quando il pulsante è rilasciato dall'utente. Naturalmente è possibile ricevere entrambi gli eventi, specificando entrambe le costanti.

Tali eventi, tuttavia, non sono emessi da Intuition sempre e comunque a fronte della selezione e del rilascio di un pulsante. In effetti è possibile associare ad ogni pulsante un filtro *in uscita*, che definisce quali eventi il pulsante deve trasmettere e quali no. Tale filtro viene definito con il campo **Flags** della struttura **Gadget**. Se desideriamo che il pulsante trasmetta l'evento **GADGETDOWN**, è necessario impostare il valore **GADGETIMMEDIATE**, cioè: *avvisami immediatamente se qualcuno ti seleziona, indipendentemente da quello che succede dopo*. Se viceversa vogliamo sapere quando il pulsante è rilasciato, il valore da impostare è **RELVERIFY**, cioè: *avvisami solo quando il pulsante è rilasciato dall'utente mentre il puntatore del mouse è ancora sopra all'area di selezione*. In questo caso l'evento emesso è **GADGETUP**. Vedremo l'importanza di questo attributo nel caso di pulsanti a rilascio automatico.

Si dice *stato del pulsante* il fatto che quest'ultimo sia premuto ò meno. Un programma può *preselezionare* un pulsante, in modo che si trovi già nello stato *selezionato* quando il pulsante è visualizzato nella finestra.

I pulsanti a rilascio automatico e quelli a rilascio manuale, possono essere paragonati per analogia alle voci dei menu, rispettivamente con le voci normali e quelle provviste di marcatore [*CheckMark*]. E proprio come nel caso delle voci marcabili, è possibile definire una tecnica di mutua esclusione anche per i pulsanti. In effetti il campo **MutualExclude** è stato previsto nella struttura **Gadget**, ma purtroppo non viene utilizzato. È comunque possibile definire lo stesso un meccanismo di mutua esclusione, anche se purtroppo dovremo fare a meno degli automatismi che Intuition fornisce nel caso delle voci dei menu. Dovremo cavarcela da soli, insomma, ma stando attenti a seguire alcune regole per garantirci il corretto funzionamento del programma con le versioni future di Intuition.

Chiameremo il gruppo di pulsanti mutualmente esclusivi, pulsanti *a rilascio incrociato*, in quanto un solo pulsante alla volta può trovarsi nello stato *selezionato*, e il rilascio avviene solo a fronte della selezione di un altro pulsante del gruppo. Vedremo che tale tecnica va implementata solo con i pulsanti a rilascio automatico, anche se nell'analogia con le voci dei menu, erano le voci marcabili a poter essere mutualmente esclusive.

Facciamo ora un esempio pratico. Supponiamo che stiate cercando di simulare in un vostro programma il pannello di controllo di un videoregistratore. Abbiamo vari pulsanti, ovviamente. Innanzitutto quello di accensione e spegnimento dell'apparato. Dato che abbiamo deciso di simulare anche il led rosso di accensione, possiamo pensare questo pulsante del tipo a rilascio automatico. Un'alternativa avrebbe potuto essere farlo a rilascio manuale, ed evidenziare lo stato **SELECTED** proprio «accendendo» un puntino rosso nel mezzo del pulsante. Veniamo ora ad i pulsanti di scorrimento: avanti, avanti veloce, riavvolgimento, stop. Essi hanno due

Figura 3
Pulsante a rilascio automatico.

```
#define AUTO_BUTTON  1

SHORT ABvector[] =
{
  0, 0,
  101, 0,
  101, 71,
  0, 71,
  0, 0
};
struct Border ABborder =
{
  -1, -1,                       /* Posizione dell'origine del bordo    */
  1, 0, JAM1,                   /* Penne e modo grafico                */
  5,                            /* Numero dei punti nel vettore XY     */
  ABvector,                     /* Vettore dei punti del bordo         */
  NULL                          /* Singolo bordo                       */
};

struct IntuiText ABtext =
{
  2, 0, JAM1,                   /* Penne e modo grafico                */
  30, 1,                        /* Posizione del testo                 */
  NULL,                         /* Caratteri base (TOPAZ 8)            */
  "OK",                         /* Testo vero e proprio                */
  NULL                          /* Singolo testo                       */
};

struct Gadget AutoButton =
{
  NULL,                         /* Singolo pulsante                    */
  5, 1, 100, 70,                /* Dimensioni del pulsante e posizione */
  GADGHCOMP,                    /* Inversione di colore                */
  RELVERIFY,                    /* Emetti solo l'evento GADGETUP       */
  BOOLGADGET,                   /* Pulsante                            */
  (APTR)&ABborder,              /* Bordo                               */
  NULL,                         /* Nessun bordo alternato              */
  &ABtext,                      /* Testo del pulsante                  */
  NULL, NULL,                   /* Campi non utilizzati                */
  AUTO_BUTTON,                  /* Identificativo del pulsante         */
  NULL                          /* Nessun dato utente                  */
};
```

Figura 2
La struttura BoolInfo.
▼

```
struct BoolInfo
{
  USHORT          Flags       ; /* Per ora va impostato a BOOLMASK     */
  UWORD          *Mask        ; /* Mascherina                          */
  ULONG           Reserved    ; /* Per ora va impostato a NULL         */
};
```

```
#define HAND_BUTTON  2

SHORT HBvector[] =
{
  0, 0,
  101, 0,
  101, 71,
  0, 71,
  0, 0
};
struct Border HBborder =
{
  -1, -1,                       /* Posizione dell'origine del bordo     */
  1, 0, JAM1,                   /* Penne e modo grafico                 */
  5,                            /* Numero dei punti nel vettore XY      */
  HBvector,                     /* Vettore dei punti del bordo          */
  NULL                          /* Singolo bordo                        */
};

struct IntuiText HBtext =
{
  2, 0, JAM1,                   /* Penne e modo grafico                 */
  20, 1,                        /* Posizione del testo                  */
  NULL,                         /* Caratteri base (TOPAZ 8)             */
  "Hit me!",                    /* Testo vero e proprio                 */
  NULL                          /* Singolo testo                        */
};

struct Gadget HandButton =
{
  NULL,                         /* Singolo pulsante                     */
  5, 1, 100, 70,                /* Dimensioni del pulsante e posizione  */
  GADGHBOX,                     /* Cornice automatica                   */
  GADGETIMMEDIATE,              /* Emetti solo l'evento GADGETDOWN      */
  BOOLGADGET,                   /* Pulsante                             */
  (APTR)&HBborder,              /* Bordo                                */
  NULL,                         /* Nessun bordo alternato               */
  &HBtext,                      /* Testo del pulsante                   */
  NULL, NULL,                   /* Campi non utilizzati                 */
  HAND_BUTTON,                  /* Identificativo del pulsante          */
  NULL                          /* Nessun dato utente                   */
};
```

Figura 4 - Pulsante a rilascio manuale.

La scheda tecnica

Ecco i cinque comandi dell'AmigaDos 1.3 di questo mese, a partire da JOIN.

LEGENDA	
<parametro>	parametro da specificare
[<opzione>]	parametro opzionale
{<opz-rip>}	parametro opzionale che può essere ripetuto n volte
...	serie che può essere continuata
|	separatore per una lista di opzioni di cui una almeno VA specificata
/A	indica che il parametro DEVE essere specificato
/K	indica che quella determinata parola chiave VA specificata se si vuole usare l'opzione ad essa associata
/S	indica una parola chiave da specificare per attivare l'operazione ad essa associata

Comando:	REMRAD
Formato:	REMRAD
Sintassi:	REMRAD
Scopo:	Rimuove l'unità di lavoro RAD:
Specifiche:	Serve a rimuovere il disco virtuale recuperabile, cioè quella area di memoria che, grazie al ramdrive.device, viene accedu-ta come se fosse un vero e proprio disco, e che sopravvive ad una ripartenza "a caldo", cioè in seguito alla pressione con-temporanea dei tasti Control, Amiga sinistro, ed Amiga destro. In realtà non si tratta di una rimozione completa. REMRAD fa sì che tutti i file contenuti in RAD: vengano cancellati, ridu-ce al minimo le dimensioni del disco virtuale, e gli disabilita l'"invulnerabilità" che lo distingue dal disco virtuale RAM:, così che, alla successiva ripartenza a caldo, esso venga del tutto cancellato dal sistema.

Comando:	RUN
Formato:	RUN <comando> [+ <commento>]
Sintassi:	RUN "COMMAND"
Scopo:	Esegue un comando in "background"
Specifiche:	A differenza dal comando della versione 1.2, adesso è possibile lanciare processi "figli" in modo tale da permettere comunque la chiusura del processo "genitore", indipendentemente dal fatto che i primi siano terminati o meno. Per far questo basta reindirizzare l'uscita di RUN verso NIL:. La cosa non funziona qualora il programma in esecuzione nel processo figlio vada a bloccare con un lucchetto la finestra del processo genitore, cioè "*". Anche un comando residente può essere eseguito in questo modo, anzi, la lista residente è letta prima di andare a cercare il comando nel cammino di ricerca [path]. RUN lancia il nuovo processo utilizzando il PROFILO di partenza S:Shell-Startup se la SHELL è attiva, altrimenti S:CLI-Startup. Il commento....
Esempio:	RUN pippo Un'altro dei miei programmi

Comando:	SEARCH
Formato:	SEARCH FROM <nome>|<modello> [SEARCH] <stringa> [ALL] [NONUM] [QUIET] [QUICK] [FILE]
Sintassi:	SEARCH "FROM,SEARCH/A,ALL/S,NONUM/S,QUIET/S,QUICK/S,FILE/S"
Scopo:	Cerca una stringa di testo in uno o più file, oppure cerca un file in uno o più indirizzari.
Specifiche:	A differenza che nella versione 1.2, ora SEARCH ritorna un codice di errore 5, cioè WARN, se la ricerca non ha avuto un esito positivo, sia che si tratti di stringhe o di file. Inoltre la ricerca può essere fatta abortire con Ctrl-C. Le opzioni sono: NONUM Non visualizzare il numero di linea dove è stata tro-vata la stringa QUIET Non visualizzare niente (utile nelle macro di sistema) QUICK Visualizza in un formato più sintetico FILE Cerca un file, non una stringa in un file La restrizione che non si potevano avere più di 31 caratteri per il nome del file, nel caso venisse usato un modello, non è più valida. ALL ha lo stesso significato che nella versione 1.2, cioè di ricerca ricorsiva nei vari sottoindirizzari.
Esempio:	SEARCH #?.out FILE produce RAM DISK:res.out RAM DISK:ressys.out SEARCH FROM test SEARCH < produce 2 SEARCH <file> FILE 6 ECO "\2<file>\d found!"

Comando:	RESIDENT
Formato:	RESIDENT <nome virtuale> <file> [REMOVE] [ADD] [REPLACE] [PURE] [SYSTEM]
Sintassi:	RESIDENT "NAME,FILE,REMOVE/S,ADD/S,REPLACE/S,PURE/S,SYSTEM/S"
Scopo:	Carica un comando e lo aggiunge alla lista di quelli residenti

Specifiche: Questo comando permette di caricare in memoria in modo residente comandi potenzialmente "puri", cioè riusabili e rientranti, e di aggiungere tali comandi alla lista mantenuta dallo SHELL. Rimandiamo alla prossima puntata la descrizione dettagliata di questi termini e come si scrivono questo tipo di programmi. L'opzione di default è REPLACE. La sintassi è la seguente:

Opzioni	Parametri	
nessuna	nessuno (*)	lista i comandi attualmente residenti
REMOVE	nessuno (*)	lista i comandi attualmente residenti
ADD	nessuno (*)	lista i comandi attualmente residenti
REPLACE	nessuno (*)	lista i comandi attualmente residenti
nessuna	nome virtuale	errore
REMOVE	nome virtuale	rimuove il comando dalla lista
ADD	nome virtuale	errore
REPLACE	nome virtuale	errore
nessuna	file	errore
REMOVE	file	rimuove il comando dalla lista
ADD	file	se già esiste dà errore altrimenti carica il comando (**)
REPLACE	file	errore
nessuna	nome & file	se già esiste ricarica il comando altrimenti dà errore
REMOVE	nome & file	rimuove il comando dalla lista
ADD	nome & file	se già esiste dà errore altrimenti carica il comando
REPLACE	nome & file	se già esiste ricarica il comando altrimenti dà errore

(*) Se viene specificata l'opzione SYSTEM, viene aggiunta alla lista anche la lista dei residenti di sistema.
(**) Viene usato come nome virtuale il nome del file senza cammino ed estensione.

Un comando viene rimpiazzato nella lista solo se non è in esecuzione. Il nome del file deve essere completo di cammino. Oltre alla lista base, esiste una seconda lista, detta di SISTEMA. Questa lista è caratterizzata dal fatto che non è possibile rimuovere i comandi residenti dalla lista. Questa ultima può essere ottenuta lanciando "RESIDENT SYSTEM". L'opzione PURE serve a caricare come residenti comandi che non abbiano l'attributo PURE impostato (bit P). Non utilizzare mai con comandi che non siano ALMENO riusabili. Se un programma non è rientrante, infatti, può sempre essere caricato residente con l'opzione PURE purchè si abbia l'accortezza di non lanciarlo in più di un processo contemporaneamente.

Esempio: RESIDENT ECHO SYS:TOOLS/ECO SYSTEM
Il programma ECO viene caricato dall'indirizzario TOOLS del disco di sistema, ed aggiunto alla lista dei comandi di sistema come ECHO, sostituendo così il comando AmigaDOS ECHO. Da notare che il programma in questione deve essere già di tipo PURE, e che, essendo stato caricato con l'opzione SYSTEM, non può più essere rimosso.

Comando:	SETCLOCK
Formato:	SETCLOCK LOAD|SAVE|RESET
Sintassi:	SETCLOCK "LOAD/S,SAVE/S,RESET/S"
Scopo:	Opera sull'orologio di sistema

Specifiche: SETCLOCK permette di

SAVE	impostare la data e l'ora dell'orologio di sistema a partire da quella corrente (impostata con DATE)
LOAD	impostare la data e l'ora corrente a partire da quella dell'orologio di sistema
RESET	riazzerare completamente l'orologio di sistema

Esempio: DATE 15:12:00
SETCLOCK SAVE
Ora l'orologio di sistema è stato rimesso alle 15:12 circa

	Pulsanti a rilascio		
Flags	**Automatico**	**Manuale**	**Incrociato**
GADGHIGHBITS	non usato	non usato	non usato
GADGHCOMP	sì	sì	limitazioni
GADGHBOX	sì	sì	no
GADGHIMAGE	sì	sì	limitazioni
GADGHNONE	sì	sì	sì
GADGIMAGE	sì	sì	limitazioni
GRELBOTTOM	sì	sì	sì
GRELRIGHT	sì	sì	sì
GRELWIDTH	sì	sì	sì
GRELHEIGHT	sì	sì	sì
SELECTED	non usato	sì	sì
GADGDISABLED	sì	sì	sì

Figura 5 - Flags. Attributi che si applicano ai pulsanti.

	Pulsanti a rilascio		
Flags	**Automatico**	**Manuale**	**Incrociato**
RELVERIFY	sì	raro	sì
GADGIMMEDIATE	raro	sì	no
FOLLOWMOUSE	raro	sì	sì
RIGHTBORDER	sì	sì	sì
LEFTBORDER	sì	sì	sì
TOPBORDER	sì	sì	sì
BOTTOMBORDER	sì	sì	sì
ENDGADGET	sì	raro	raro
TOGGLESELECT	no	sì	no
STRINGCENTER	no	no	no
STRINGRIGHT	no	no	no
LONGINT	no	no	no
ALTKEYMAP	no	no	no
BOOLEXTEND	sì	sì	sì

Figura 6 - Activation. Attributi che si applicano ai pulsanti.

caratteristiche: la prima è che deve essere chiaro a chi guarda il quadro quale dei quattro pulsanti è premuto, la seconda è che non è possibile avere più di un pulsante premuto allo stesso tempo. In effetti, per rilasciare un pulsante di questo gruppo, è necessario premerne un altro qualsiasi dello stesso gruppo. Va bene. Lo so che ci sono VCR in cui *Play* ed *FF* possono essere premuti contemporaneamente, ma non è il caso di fare i pignoli, no? La scelta quindi è chiara: dobbiamo usare quattro pulsanti mutualmente esclusivi a rilascio incrociato. C'è poi il tasto di pausa, che pur essendo anch'esso del tipo a rilascio manuale, è indipendente dai precedenti.

E se volessimo un pulsante tondo, invece che quadrato? Qualcuno potrebbe pensare che basta definire un bordo od una immagine rotonda, ma così facendo

	Pulsanti a rilascio		
Flags	**Automatico**	**Manuale**	**Incrociato**
BOOLGADGET	sì	sì	sì
GADGET0002	??	??	??
PROPGADGET	no	no	no
STRGADGET	no	no	no
GZZGADGET	sì	sì	sì
REQGADGET	sì	sì	sì

Figura 7 GadgetType. Attributi che si applicano ai pulsanti.

il pulsante potrebbe essere selezionato anche facendo click fuori dal bordo che lo delimita, perché comunque l'area di selezione, invisibile all'utente, è rimasta rettangolare. Un sistema c'è: è possibile applicare al pulsante una mascherina che serve a modificare l'area di selezione in modo da poter disporre di pulsanti con aree di forma differente da quella rettangolare. Vedremo tuttavia che tale tecnica ha dei limiti.

Vediamo ora le tre tipologie più comuni per i pulsanti.

Pulsanti a rilascio automatico

Il campo **GadgetType** deve contenere **BOOLGADGET**, oltre che, eventualmente, le costanti **GZZGADGET** nel caso che esso debba essere posizionato nel bordo di una finestra GZZ, o **REQGADGET**, nel caso che esso vada aggiunto ad un quadro (vedi figura 3).

Il campo **Flags** può contenere qualunque valore in funzione delle caratteristiche del pulsante, come spiegato nella scorsa puntata, salvo il valore **SELECTED** che non ha molto senso per i pulsanti a rilascio automatico, a meno che non si stia implementando la tecnica a rilascio incrociato.

Il campo **Activation** non deve contenere **TOGGLESELECT**. Per quello che riguarda gli eventi che il pulsante deve trasmettere al programma, consiglio di utilizzare **RELVERIFY**, in modo da essere sicuri che l'utente ha voluto selezionare effettivamente proprio quel pulsante, dandogli così la possibilità di cambiare idea spostando il cursore fuori dall'area di selezione *prima* di rilasciare il bottone del mouse. Viceversa **GADGETIMMEDIATE** non è necessario, a meno che non si desideri effettuare una qualche operazione indipendentemente dal fatto che l'utente voglia realmente selezionare quel pulsante. Se non è questo il caso, l'unico evento che vi arriverà da questo tipo di pulsante è **GADGETUP**.

Pulsanti a rilascio manuale

Il campo **GadgetType** deve contenere **BOOLGADGET**, oltre che, eventualmente, le costanti **GZZGADGET** nel caso che esso debba essere posizionato nel bordo di una finestra GZZ, o **REQGADGET**, nel caso che esso vada aggiunto ad un quadro (vedi figura 4).

Il campo **Flags** può contenere qualunque valore in funzione delle caratteristiche del pulsante, come spiegato nella scorsa puntata, compreso il valore **SELECTED**, nel caso che si desideri che il pulsante sia nello stato *selezionato* fin dall'inizio. Ovviamente, non appena l'utente posizionerà il cursore del mouse sull'area di selezione pulsante e premerà il bottone sinistro, il pulsante verrà rilasciato.

Il campo **Activation** deve contenere **TOGGLESELECT**, che attiva appunto il blocco del rilascio automatico, e **GADGETIMMEDIATE**, dato che l'evento che ci interessa è appunto **GADGETDOWN**. Non va impostato invece **RELVERIFY**, in quanto non significativo per questo tipo di pulsante. Non che sia proibito. È solo inutile.

Pulsanti a rilascio incrociato

Dato che non si tratta di un vero e proprio pulsante «ufficiale», ma di una tecnica che sopperisce ad una carenza della versione 1.3 di Intuition, esiste una serie di raccomandazioni da seguire se non si vuole avere problemi di incompatibilità con le versioni successive del sistema. Chi vuole può anche divertirsi a vedere cosa succede a non seguirle, sempre tenendo presente però che, anche se il risultato può essere interessante, è bene non utilizzare tecniche non conformi agli standard Amiga per programmi di una certa importanza.

Vediamo allora come vanno definiti i vari pulsanti di un gruppo mutualmente esclusivo.

Il campo **GadgetType** deve contenere **BOOLGADGET**, oltre che, eventualmente, le costanti **GZZGADGET** nel caso che esso debba essere posizionato nel bordo di una finestra GZZ, o **REQGADGET**, nel caso che esso vada aggiunto ad un quadro.

Per quello che riguarda il campo **Flags**, esso può contenere qualunque valore a eccezione della costante **SELECTED**. Questo in quanto valgono le seguenti regole:

- solo un pulsante alla volta può trovarsi nello stato selezionato;
- ci deve essere sempre un pulsante selezionato nel gruppo, non possono cioè essere tutti nello stato *rilasciato*.

Questo vuol dire che bisogna definire sempre un pulsante di default, che si trova nello stato **SELECTED** quando il gruppo è visualizzato sullo schermo. Quando l'utente va a selezionare un altro pulsante, starà al programma rilasciare il pulsante precedente ed impostare il campo **Flags** del pulsante selezionato a **SELECTED**.

Per quello che riguarda la tecnica di evidenziazione, valgono le seguenti regole.

- Se si usa la tecnica del colore complementare (**GADGHCOMP**) è necessario che l'immagine accessibile via **GadgetRender** abbia almeno le dimensioni dell'area di selezione (non sia cioè più piccola), oppure che sia utilizzata una mascherina per far coincidere quest'ultima con l'immagine fornita. Non possono essere usati bordi, in questo caso.
- Se si usa la tecnica dell'immagine alternata (**GADGHIMAGE|GADGHIMAGE**), le due immagini devono avere le stesse dimensioni e sovrapporsi perfettamente.
- Se si usa la tecnica dei bordi alternati (**GADGHIMAGE**), i due bordi devono differire solo per il colore.
- Non si può usare la tecnica della cornice automatica (**GADGHBOX**).

Tutte le altre combinazioni vanno assolutamente evitate. Questo non vuol dire che non possano funzionare, ma solo che il loro utilizzo può creare problemi, e comunque che non sono garantite nelle versioni successive di Intuition.

I pulsanti devono essere tutti del tipo *a rilascio automatico*, cioè il campo **Activation** non deve contenere **TOGGLESELECT**. In effetti, quello a rilascio incrociato, è più un meccanismo automatico che manuale, in cui il rilascio di un pulsante è attivato dalla selezione di un altro pulsante del gruppo.

Il campo **Activation** dovrà inoltre contenere la costante **GADGETIMMEDIATE**, in quanto è l'evento **GADGETDOWN** che va intercettato per questo tipo di pulsante. Non va invece impo-

Valori	Azioni dell'utente	Eventi emessi	Coord
FOLLOWMOUSE GADGIMMEDIATE RELVERIFY	L'utente ha selezionato il pulsante	GADGETDOWN	no
	L'utente muove il mouse	MOUSEMOVE	sì
	L'utente rilascia il mouse sopra il pulsante	GADGETUP	no
	L'utente rilascia il mouse ma non sopra il pulsante	nessuno	no
FOLLOWMOUSE GADGIMMEDIATE	L'utente ha selezionato il pulsante	GADGETDOWN	no
	L'utente muove il mouse	MOUSEMOVE	sì
	L'utente rilascia il mouse il pulsante	nessuno	no
FOLLOWMOUSE RELVERIFY	L'utente ha selezionato il pulsante	nessuno	no
	L'utente muove il mouse	MOUSEMOVE	sì
	L'utente rilascia il mouse sopra il pulsante	GADGETUP	no
	L'utente rilascia il mouse ma non sopra il pulsante	nessuno	no
FOLLOWMOUSE GADGIMMEDIATE	L'utente ha selezionato il pulsante	nessuno	no
	L'utente muove il mouse	MOUSEMOVE	sì
	L'utente rilascia il mouse il pulsante	nessuno	no

Figura 8
Tracciamento degli spostamenti di mouse.

stato **RELVERIFY**, che serve appunto a far sì che Intuition faccia arrivare al programma l'evento di selezione *se e solo se* il pulsante è stato rilasciato quando il cursore del mouse era posizionato nell'area di selezione del controllo.

Pulsanti non rettangolari

Esiste un attributo di attivazione, chiamato **BOOLEXTEND** che, se selezionato, permette di specificare una apposita struttura, chiamata **BoolInfo** (vedi figura 2), una maschera da applicare all'area di selezione, per modificarne la forma. Tale tecnica permette così di ottenere, ad esempio, aree di selezione ovali, ma ha alcuni difetti.

- Il primo è che l'immagine del pulsante non è interessata dall'applicazione di tale maschera, per cui continua ad essere rettangolare anche se il disegno può a sua volta essere scelto in modo da apparire come un ovale. Questo fa sì che controlli mascherati non possono essere avvicinati più di tanto, in quanto la sovrapposizione delle zone mascherate può comunque comportare effetti esteticamente poco desiderabili.
- Il secondo è che, se si disabilita il pulsante, l'immagine *fantasma* che appare non segue la maschera ma continua ad essere rettangolare, per cui, in funzione anche dei colori interessanti, gli angoli di tale immagine possono essere visibili al di fuori dei limiti della maschera.

La struttura **BoolInfo** ha tre campi:

Flags: che per ora può assumere solo il valore **BOOLMASK** ad indicare che la struttura si riferisce ad una mascherina;
Mask: punta ad una matrice di **UWORD** che contiene la mascherina definita come una immagine su un solo piano. Come al solito questi dati devono essere nella memoria di tipo *CHIP*. Le dimensioni della maschera non vanno fornite, ma vengono ricavate da Intuition a partire da quelle dell'area di selezione del pulsante a cui la maschera si riferisce. Ricordatevi quindi di modificare la maschera se modificate le dimensioni dell'area di selezione.
Reserved: come dice il nome, questo campo è riservato per usi futuri. Va quindi impostato a NULL.

FOLLOWMOUSE

Se imposta nel campo **Activation** la costante **FOLLOWMOUSE**, il vostro programma riceverà eventi di tipo **MOUSEMOVE** fintanto che il controllo è nello stato *selezionato*. Questo vuol dire ricevere per un certo periodo di tempo una quantità notevole di dati relativamente alla posizione del mouse nella finestra. Un possibile utilizzo di questa costante è in un programma per disegnare. Se il controllo è del tipo a rilascio manuale, esso rimarrà selezionato fino a che l'utente non lo selezionerà di nuovo. Per tutto questo tempo il programma riceverà la posizione del mouse nella finestra. Se questo si trova sul piano di disegno, il programma può tracciare una serie di punti man mano che il puntatore si muove. Un altro utilizzo può essere il seguente. Supponiamo che il nostro pulsante faccia parte di un elemento grafico e che noi vogliamo spostare tale elemento in una finestra, un po' come si fa con la barra del titolo dei quadri. Basta utilizzare un pulsante a rilascio automatico ed impostare **FOLLOWMOUSE|GADGETIMMEDIATE|RELVERIFY**. Quando l'utente seleziona il pulsante il programma si pone in attesa delle coordinate del puntatore del mouse. Queste iniziano ad arrivare non appena l'utente inizia a spostare il mouse, sempre tenendo il bottone sinistro del mouse premuto. Non appena i primi dati arrivano, il programma provvede a spostare l'elemento grafico nella stessa direzione, mantenendo così sempre il pulsante sotto il puntatore del mouse, in modo da dare l'impressione che il mouse abbia «agganciato» l'elemento grafico. Non appena l'utente rilascia il bottone del mouse, gli eventi **MOUSEMOVE** cessano ed un evento **GADGETUP** viene emesso. A questo punto il programma smette di spostare l'elemento grafico.

Lo schema riportato in figura 8 mostra come il fatto di avere o meno impostato le costanti **GADGETIMMEDIATE** e/o **RELVERIFY**, permette di gestire nel modo più appropriato gli eventi che Intuition emette mentre il mouse si muove.

Conclusione

Bene. Abbiamo ora tutti gli elementi necessari per lavorare con i comandi. Nella prossima puntata vedremo un primo tentativo di costruire una gamma di funzioni di alto livello per rendere l'utilizzo dei controlli semplice ed immediato, cioè una *ToolBox*. Ovviamente ognuno potrà partire dalle funzioni che presenterò per crearne di nuove o per dare ai vari controlli un proprio stile. Si tratta di un progetto che sto portando avanti man mano che scrivo questi articoli, anzi, appositamente per *MCmicrocomputer*, quindi non so ancora come si svilupperà. Se poi alcuni lettori vorranno contribuire con loro idee, sarò ben lieto di utilizzarle per costruire nuove funzioni ed ampliare così la libreria di quelle disponibili.

Nella prossima puntata parleremo anche di programmi *puri*, il significato dei termini *riusabile, rientrante* e *residente*, in modo da poter sfruttare efficacemente anche per i nostri programmi il comando **resident** presentato nella Scheda Tecnica di questa puntata. Ed anche per variare un po' argomenti, giusto?

MC