coordinamento di Andrea de Prisco

Programmare in C su Amiga

di Dario de Judicibus

sesta puntata

Con questa puntata inizia la seconda parte del corso sulla programmazione in C per Amiga. Ci occuperemo di Intuition, l'interfaccia grafica dell'Amiga, delle funzioni grafiche di base e di animazione. Oggi parleremo di... finestre

Introduzione

Intuition è l'interfaccia grafica dell'Amiga, una interfaccia basata su una simbologia *ad oggetti [object-oriented interface]*. Una interfaccia utente è un meccanismo con il quale l'utente è in grado di dialogare con il sistema: effettuare richieste, ricevere informazioni, iniziare o terminare vari tipi di attività. Una interfaccia «classica» è quella CLI, basata su una interazione di tipo *terminale*, nella quale ogni richiesta viene effettuata digitando sulla tastiera uno o più comandi, o premendo speciali tasti funzionali ai quali è associata una transizione ben definita. Il risultato, sotto forma di informazioni ritornate dal sistema, viene presentato sullo stesso schermo nel quale era stato lanciato il comando. Una interfaccia *object-oriented* si basa invece su una rappresentazione del sistema di tipo grafico, in cui l'utente, agendo attraverso un apposito strumento che governa un cursore sullo schermo, detto mouse, manovra una serie di oggetti in modo intuitivo (da qui il nome Intuition), come farebbe nella realtà. Ad esempio, per cancellare un file, basta agganciarlo con il mouse e «buttarlo» nel cestino della carta straccia, proprio come faremmo con un pezzo di carta che non ci serve più.

Non insisto sull'argomento anche perché chi sta leggendo queste righe, molto probabilmente ha usato più di una volta il Work Bench, che si basa appunto su Intuition per fornire all'utente un'interfaccia semplice ed intuitiva all'Amiga DOS. Quello che però non tutti sanno è che lo stesso tipo di interfaccia può essere utilizzato da qualunque programma, anche i vostri. Non solo, ma le possibilità sono ancora di più di quelle che vengono sfruttate dal Work Bench. Intuition infatti, non è solo un componente del sistema che ci permette di costruire e gestire finestre od altri oggetti grafici, ma un vero e proprio *manager* che ci tiene aggiornati su quello che sta facendo l'utente e ci permette di rispondere puntualmente alle richieste corrispondenti alle operazioni effettuate sullo schermo. Ecco allora che possiamo costruire menu [*menu*] come nel WorkBench, ma in più possiamo associare ad ogni elemento di un menu [*item*] una combinazione di tasti che agisce come scorciatoria [*shortcut*]. E ancora possiamo far sì che alcuni menu contengono figure, o complessi pannelli di controllo [*gadgets and requesters*] per modificare i colori dello schermo o caricare un file. Possiamo creare dei *pop-up* menu che, invece di scendere dalla parte superiore dello schermo (come nel 90% del software Amiga), appaiono all'improvviso al posto del cursore quando premiamo due volte velocemente il bottone destro del mouse. Le possibilità sono veramente tante, e molte di nuove potreste essere proprio voi a scoprirle. Uno strumento come Intuition infatti, se ben compreso, può fornire davvero molti spunti alla vostra creatività.

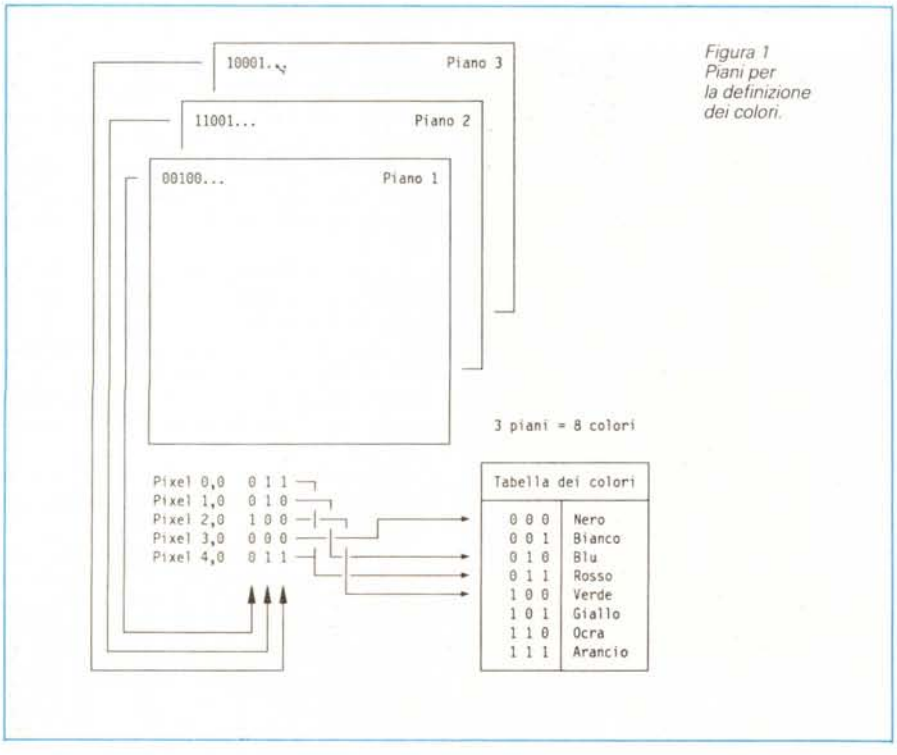


Figura 1 Piani per la definizione dei colori.

```
struct NewWindow
{
  SHORT LeftEdge, TopEdge;    /* Angolo superiore sinistro della finestra   */
  SHORT Width, Height;        /* Larghezza ed altezza della finestra        */

  UBYTE DetailPen, BlockPen;  /* Colori usati per i vari elementi           */

  ULONG IDCMPFlags;           /* IDCMP flags: vedi il testo dell'articolo   */
  ULONG Flags;                /* Altri flags: vedi il testo dell'articolo   */

  struct Gadget *FirstGadget; /* User Gadgets: vedi il testo dell'articolo. */
  struct Image *CheckMark;    /* CheckMark   : vedi il testo dell'articolo. */

  UBYTE *Title;               /* Il titolo della finestra                   */

  struct Screen *Screen;      /* Schermo al quale la finestra appartiene    */
                              /*   NULL se è lo schermo del WorkBench       */
  struct BitMap *BitMap;      /* Puntatore ad una struttura BitMap          */
                              /*   Solo per finestre SUPERBITMAP            */

  SHORT MinWidth, MinHeight;  /* Dimensioni minime per la finestra          */
  USHORT MaxWidth, MaxHeight; /* Dimensioni massime per la finestra         */

  USHORT Type;                /* Tipo di schermo in cui si apre la finestra */
};
```

Figura 2
Struttura NewWindow.

Vedremo in seguito qualche esempio un po' originale dell'uso di Intuition. Per il momento ci occuperemo di schermi e finestre. In questa puntata impareremo ad:

— aprire e chiudere una finistra,
— definire le caratteristiche di una finestra,
— impostare i segnalatori per la comunicazione tra Intuition ed il vostro programma.

Non analizzeremo subito tutti i possibili tipi di schermi e finestre che possono essere creati con Intuition, anche perché in alcuni casi sarebbero necessari concetti e conoscenze che introdurremo più in là. Vedremo tuttavia come, anche con pochi elementi, sia possibile ottenere una interfaccia valida per la maggior parte dei programmi che presenteremo nelle prossime puntate. Un passo alla volta arriveremo infine ad interfacce più sofisticate e spettacolari. Una cosa tuttavia bisogna dirla subito. Chiunque di voi abbia una certa esperienza di AmigaBasic, leggendo questa seconda parte del corso non potrà fare a meno di constatare come alcune cose che in Basic si fanno con estrema facilità (in particolar modo i menu), richiedono uno sforzo molto maggiore in C. Questo è vero, tuttavia non bisogna dimenticare che, attraverso Intuition, si possono fare cose che mai vi sareste sognati in Basic nativo, senza cioè aprire le librerie del sistema (ed in tal caso vi sareste trovati di fronte alle stesse difficoltà che avreste avuto in C, se non maggiori). Ad esempio, il Basic non permette di associare un elemento grafico ad un menu, o di avere dei sottoelementi per un certo elemento di un menu [*subitem*].

Schermi e finestre

Uno schermo [*screen*] è lo sfondo su cui possono essere aperte delle finestre. Non è un supporto fisso, ma può scorrere verticalmente (ma non orizzontalmente) ad essere mosso sopra o sotto un altro schermo. Uno schermo ha caratteristiche ben definite (colori, risoluzione) che vengono ereditate da tutte le finestre associate. Quindi, se si vuole aprire una finestra in bassa risoluzione, bisogna prima aprire uno schermo a bassa risoluzione, dato che quello del WorkBench è in media risoluzione. Analogamente, per poter utilizzare otto colori, bisogna aprire uno schermo basato su tre piani [*layer*] come vedremo nel secondo esempio più avanti. Uno schermo non può modificare le proprie dimensioni e, a parte il fatto che può essere «tirato giù», occupa sempre tutto il quadro del monitor, almeno orizzontalmente.

Una finestra [*window*], viceversa, non solo può essere spostata sopra e sotto ad altre finestre, ma può essere mossa in una qualunque direzione su di uno schermo, ingrandita o rimpicciolita. Naturalmente il programmatore può decidere quante e quali di queste caratteristiche possono essere associate ad una certa finestra. Una finestra non può essere spostata da uno schermo ad un altro, anche se hanno le stesse caratteristiche. In pratica una finestra rappresenta il mezzo principale attraverso il quale un utente può richiedere o ricevere informazioni, siano esse testo o grafica.

Se è stato caricato il Work Bench possiamo usare lo schermo associato per aprire una finestra ovviamente in media risoluzione e con quattro colori, cioè due piani (vedi nota 1). Sarà questo il nostro primo esempio di utilizzo di Intuition (vedi figura 3). Per poter aprire una finestra, tuttavia, è necessario prima passare ad Intuition tutta una serie di informazioni che la caratterizzano. Per far questo si usa una struttura chiamata **NewWindow** (vedi nota 3). Tale struttura è riportata in figura 2. Analizziamo in dettaglio i campi di cui è composta.

La struttura NewWindow

I primi quattro campi di **NewWindow** rappresentano le dimensioni geometriche della finestra, e più precisamente:

LeftEdge l'ascissa dell'angolo superiore sinistro della finestra nel sistema di riferimento dello schermo, partendo da sinistra verso destra;

TopEdge l'ordinata dell'angolo superiore sinistro della finestra nel sistema di riferimento dello schermo, partendo dall'alto vero il basso;

Width la larghezza della finestra;

Height l'altezza della finestra.

Queste dimensioni sono tutte calcolate in *pixel [picture element]* e devono essere fornite come **short integer**. L'angolo superiore sinistro dello schermo viene assunto (0,0).

I successivi due campi definiscono i colori che devono essere usati per disegnare la finestra, tra quelli associati allo schermo a cui la finestra appartiene:

DetailPen il colore da usare per «dettagli» quali *gadget* (vedi nota 2) od il testo nella barra del titolo [*title bar*];

BlockPen il colore da usare per riempire lo sfondo degli elementi che costituiscono il bordo della finestra (vedi nota 4).

Se ad esempio si è scelta la coppia **5, 7** corrispondente, per lo schermo che contiene la finestra, ad i colori nero e giallo, il titolo della finestra sarà disegnato in nero su sfondo giallo. Dato che anche lo schermo ha associati due colori che vengono utilizzati per disegnare gli elementi dello schermo stesso (come ad esempio la barra che serve a farlo scivolare verticalmente), si può specificare per uno od entrambi i campi gli stessi colori usati per lo schermo. Per far questo basta porli a **(-1)**, come mostrato in figura 3.

Il campo successivo è quello che ci permette di definire che tipo di comunicazione desideriamo mantenere con Intuition. IDCMP sta infatti per *Intuition's Direct Communications Message Port*, cioè «Porta di Comunicazione Diretta con Intuition per lo scambio di Messaggi». E a questo punto, chi non l'ha già fatto, vada subito a leggersi la quinta puntata per le definizioni di «Porta» e «Messaggio».

IDCMPFlags è una doppia voce che viene usata per segnalare ad Intuition gli eventi su cui il programma vuole venire informato. Ad esempio, se la finestra ha associato un gadget per la chiusura (vedremo dopo come definirlo), e questo viene selezionato, il programma ne viene a conoscenza solo se

ci si è ricordati di porre ad uno in **IDCMPFlags** il bit relativo. Per far questo si utilizza una costante predefinita in **intuition/intuition.h**: CLOSEWINDOW.

Analogamente, se le dimensioni della finestra sono state modificate e, per un qualche motivo è importante che il programma lo sappia dopo che la modifica è stata effettuata, si può usare il segnalatore [*flag*] NEWSIZE. Dato che nell'esempio in figura 3 il corpo del programma non prevede alcuna interazione con l'utente, ma si limita a girare a vuoto per un milione di volte per poi terminare da solo, il valore di **IDCMPFlags** è **NULL**.

Analoga a **IDCMPFlags** è la doppia voce **Flags**. Questa serve a specificare ad Intuition che caratteristiche deve avere la finestra da aprire:

- se, ad esempio, deve aver gadget di sistema, e se sì quali;
- se deve essere attiva fin dall'inizio;
- se deve venir automaticamente ricostruita nel caso sia stata coperta da un'altra finestra, o è piuttosto responsabilità del programma occuparsi di tale operazione; e così via. La figura 5 e la figura 6 riportano una lista di tutti i segnalatori predefiniti in **intuition.h**. Naturalmente, se il programmatore desidera specificare più di un segnalatore contemporaneamente, questi vanno addizionati logicamente (OR, cioè |), come nell'esempio già menzionato.

Il campo seguente (**FirstGadget**) è il puntatore ad una lista di gadget utente, cioè di quei gadget disegnati e definiti dal programmatore per gestire operazioni di tipo applicativo. Come già detto, le operazioni «classiche» di gestione della finestra sono effettuate attraverso i gadget di sistema definiti tramite il campo **Flags**.

Il campo **CheckMark** è invece il puntatore ad una struttura **Image** (che vedremo nelle prossime puntate) che descrive il simbolo da utilizzare per segnalare gli elementi di un eventuale menu, selezionati dall'utente. Se la finestra non ha menu, o se si desidera utilizzare il simbolo standard, basta porre questo campo a NULL. Per chi non lo ricordasse, il simbolo usato dal sistema è:

A questo punto ci sono il puntatore ad una stringa che contiene il titolo da mettere nella barra superiore della finestra (**Title**), e quello dello schermo al quale la finestra appartiene. Se la finestra va aperta sullo schermo del Work Bench quest'ultimo è nullo.

Il puntatore successivo va usato solo per finestre SUPERBITMAP, cioè finestre il cui contenuto si trova in un'area di memoria fornita e gestita direttamente dal programmatore. Torneremo in seguito su questa tecnica particolare di gestione delle finestre.

I successivi quattro campi sono analoghi al terzo ed al quarto della struttura **NewWindow** e definiscono le dimensioni minime e massime della finestra. Il valore zero ha un significato speciale per questi campi: esso indica che il campo assume lo stesso valore usato per definire la dimensione corrispondente all'apertura della finestra. Quindi, se ad esempio **MinWidth** = **0** e **Width** = **200**, tale valore sarà anche quello assunto come la larghezza minima alla quale la finestra può essere ridotta. I campi in questione sono:

MinWidth la larghezza minima che può essere assunta dalla finestra;

MinHeight l'altezza minima che può essere assunta dalla finestra;

MaxWidth la larghezza massima che può essere assunta dalla finestra;

MaxHeight l'altezza massima che può essere assunta dalla finestra.

Anche queste dimensioni sono tutte calcolate in *pixel [picture element]* e

```
/*************************************************************************\
 *                                                                       *
 *    APRIWB - Esempio di finestra aperta sullo schermo del WorkBench.   *
 *                                                                       *
\*************************************************************************/

 #include "exec/types.h"
 #include "intuition/intuition.h"

/*
 *  Qualche definizione...
 */
 #define FINE  1000000
 #define CARATTERISTICHE (ACTIVATE|NOCAREREFRESH|SMART_REFRESH)
 #define INTUINAME "intuition.library"

/*  Usiamo le stesse costanti del DOS, ma non includiamo libraries/dos.h   */
 #define RETURN_OK      0
 #define RETURN_ERROR  10

 struct IntuitionBase *IntuitionBase;

/*
 *  Se mettiamo qui queste definizioni, saranno disponibili anche ad altre
 *  eventuali routine definite fuori da main().
 */
 struct NewWindow DefFinestra =    /* Inizializza già una parte dei campi  */
 {
   20,20,320,120,                  /* angolo sup. sinistro e dimensioni    */
   -1,-1,                          /* colori per il fondo ed i particolari */
   NULL,                           /* IDCMP flags...   per ora niente      */
   CARATTERISTICHE,                /* caratteristiche di questa finestra   */
   NULL,                           /* Primo Gadget...  per ora niente      */
   NULL,                           /* CheckMark...     per ora niente      */
   "Esempio in WorkBench",         /* titolo della finestra                */
   NULL,                           /* Puntatore allo schermo non WorkBench */
   NULL,                           /* Puntatore a BitMap (se SUPERBITMAP)  */
   0, 0,                           /* Dimensioni minime                    */
   0, 0,                           /* Dimensioni massime                   */
   WBENCHSCREEN                    /* tipo di schermo da utilizzare        */
 };

/*
 *  "Finestra" è il puntatore alla struttura restituita da Intuition e che
 *  contiene varie informazioni relative alla finestra creata.
 *  Vedi Figura 4.
 */
 struct Window *Finestra;

main()
{
  LONG conta;                          /* contatore */

 /*
  *  Apri la libreria che contiene le routine di Intuition
  */
  IntuitionBase = (struct IntuitionBase *)OpenLibrary(INTUINAME,0);
  if (IntuitionBase == NULL) exit(RETURN_ERROR);

 /*
  *  OK, proviamo ora ad aprire la finestra
  *  >>> ATTENZIONE: nessun controllo viene effettuato sull'esistenza <<<
  *  >>>             o meno dello schermo del WorkBench!             <<<
  */
  Finestra = (struct Window *)OpenWindow(&DefFinestra);
  if (Finestra == NULL) exit(RETURN_ERROR);

 /******************** Questo è il corpo del programma ******************/
 /************/                                                /************/
 /************/     for (conta=0; conta<FINE; conta++);        /************/
 /************/                                                /************/
 /********************** Ovviamente non fà niente ***********************/

 /*
  *  Non dimentichiamoci di "chiudere casa", prima di uscire!
  */
  CloseWindow(Finestra);
}
```

Figura 3 - Esempio di finestra aperta nello schermo del Work Bench.

```
{
    struct Window *NextWindow;      /* Puntatore alla finestra successiva   */
                                    /*   appartenente allo stesso schermo   */
    SHORT LeftEdge, TopEdge;        /* Angolo superiore sinistro della fin. */
    SHORT Width, Height;            /* Dimensioni della finestra            */
    SHORT MouseY, MouseX;           /* Posizione del mouse NELLA finestra   */
    SHORT MinWidth, MinHeight;      /* Dimensioni minime della finestra     */
    USHORT MaxWidth, MaxHeight;     /* Dimensioni massime della finestra    */
    ULONG Flags;                    /* Informazioni varie... in seguito!    */
    struct Menu *MenuStrip;         /* Eventuali menù associati alla fin.   */
    UBYTE *Title;                   /* Titolo della finestra                */
    struct Requester *FirstRequest; /* Lista dei Requester ATTIVI           */
    struct Requester *DMRequest;    /* Lista dei Requester DOPPIO-CLICK     */
    SHORT ReqCount;                 /* Contatore                            */
    struct Screen *WScreen;         /* La struttura SCREEN associata        */
    struct RastPort *RPort;         /* La struttura RASTPORT associata      */

   /*
    *  Le cinque variabili qui di seguito descrivono il bordo della finestra.
    *  Vedremo la loro importanza quando parleremo di finestre GIMMEZEROZERO,
    *  SUPERBITMAP e vedremo come viene memorizzata una finestra nell'Amiga.
    */
    BYTE BorderLeft, BorderTop, BorderRight, BorderBottom;
    struct RastPort *BorderRPort;

   /*
    *  Lista dei GADGET UTENTE eventualmente utilizzati con questa finestra.
    *  Quelli di SISTEMA sono definiti attraverso la variabile Flags.
    */
    struct Gadget *FirstGadget;

    struct Window *Parent, *Descendant; /* Usate per l'apertura/chiusura    */

   /*
    *  Eventuale PUNTATORE utente definito tramite SetPointer()
    *  DOPO l'apertura della finestra. Di fatto è uno SPRITE.
    */
    USHORT *Pointer;            /* definizione dello SPRITE-PUNTATORE       */
    BYTE PtrHeight;             /* altezza dello SPRITE-PUNTATORE           */
    BYTE PtrWidth;              /* larghezza dello SPRITE-PUNTATORE (<=16)  */
    BYTE XOffset, YOffset;      /* posizione dello SPRITE-PUNTATORE (offset) */

   /*
    *  Strutture di comunicazione Intuition/Utente
    */
    ULONG IDCMPFlags;                       /* Flags specificati dall'utente */
    struct MsgPort *UserPort, *WindowPort; /* Porta Utente e di Intuition   */
    struct IntuiMessage *MessageKey;        /* Struttura Messaggio-Intuition */

    UBYTE DetailPen, BlockPen;      /* Colori usati per disegnare la finestra */
    struct Image *CheckMark;        /* Segno di selezione per gli elementi di */
                                    /* un menù. Se NULL, usa il default.      */

    UBYTE *ScreenTitle;         /* Titolo dello schermo QUANDO LA FINESTRA E' */
                                /* ATTIVA.                                    */

    SHORT GZZMouseX; /* Coordinate del mouse relative alla finestra INTERNA */
    SHORT GZZMouseY; /* nel caso di finestra GIMMEZEROZERO. MouseX e MouseY */
    SHORT GZZWidth;  /* sono invece relative all'angolo superiore sinistro  */
    SHORT GZZHeight; /* della finestra vera e propria. Vedremo in seguito.  */

    UBYTE *ExtData;     /* Altre eventuali estensioni della struttura...    */
    BYTE *UserData;     /* Eventuali estensioni aggiunte dall'UTENTE        */

    struct Layer *WLayer;       /* = Window.RPort->Layer                    */
    struct TextFont *IFont;     /* Font usata [DOS 1.2]                     */
};
```

Figura 4 - Struttura Window.

Note

1. Intuition utilizza la libreria per i piani [*layers library*] per la gestione di schermi e finestre. Ogni punto su di uno schermo è rappresentato da un bit in una matrice le cui dimensioni dipendono dalla risoluzione dello schermo. Chiamiamo piano [*layer*] tale matrice (vedi figura 1). Se avessimo un solo piano, ogni pixel potrebbe avere solo due colori, uno se il bit associato è 0 (ad esempio nero) ed uno quando il bit è 1 (ad esempio bianco). Supponiamo ora di avere due piani e di associare ad ogni pixel un bit della prima matrice più un bit della seconda posto nella stessa posizione del primo. Abbiamo adesso quattro combinazioni: {**00 01 10 11**}. Associando ad ognuna di esse un colore, possiamo definire fino a quattro colori per pixel. Se avessimo tre piani potremmo definire otto colori e così via, secondo le potenze di due: il numero di colori permessi è uguale a due elevato il numero di piani utilizzati. In teoria questo vuol dire che potremmo avere anche 1024 colori, o 262144, a condizione di usare, rispettivamente 10 e 18 piani. Tuttavia esiste un limite ben preciso a riguardo, dovuto principalmente a questioni relative alla memoria. Pensate che uno schermo in bassa risoluzione PAL (320 × 256) e 32 colori (5 piani), richiede ben 409600 bit, cioè 50 KByte! E questo solo per memorizzare lo schermo stesso. Se si aggiungono tutte le strutture di controllo, eventuali finestre aperte ed altri oggetti grafici, lo spazio occupato aumenta di molto. Uno schermo ad alta risoluzione PAL a 16 colori occupa 80 KByte.

Il numero massimo di piani utilizzabile nell'Amiga è di cinque, ma può arrivare a sei se si utilizzano alcune tecniche speciali. Possiamo considerare il numero di piani di un certo schermo, come una sorta di risoluzione in profondità analogamente a quella in larghezza ed altezza. Ad ogni schermo poi, è legata una tabella che associa ad ogni combinazione di bit un certo colore, scelto da una palette di 4096. Quando cambiamo i colori di uno schermo, come si può fare in un qualunque programma grafico, modifichiamo semplicemente la tabella di correlazione tra combinazioni di bit e colori, non le matrici di bit che formano i piani.

2. Un *gadget* (in italiano potremmo utilizzare il termine *aggeggio*) è qualunque strumento, oggetto, «coso» insomma, che ci permette di controllare una macchina od un altro strumento. Esempi di gadget sono leve, pulsanti, cursori, manopole ed in generale tutto ciò che è stato inventato per regolare il volume di una radio o il programma di lavaggio di una lavatrice. Dato che Intuition è una interfaccia *object-oriented*, non poteva mancare una riproduzione grafica di tali aggeggi. Grazie ad essi possiamo gestire finestre modificandone le dimensioni o spostandole sullo schermo, possiamo mandare ad Intuition qualsiasi tipo di comando od informazione, rispondere alle richieste del sistema di introdurre un dischetto, e via dicendo. Torneremo sull'argomento nelle prossime puntate, ed impareremo a costruire i nostri gadget e a dar loro la forma e le caratteristiche che vogliamo grazie alle enormi possibilità che Intuition ci mette a disposizione.

3. La struttura **NewWindow** serve a trasmettere ad Intuition tutte quelle informazioni necessarie per costruire la finestra da aprire. Esse vengono quindi utilizzate solo all'apertura della finestra. Al contrario, la struttura **Window**, il cui puntatore viene restituito da Intuition al programma chiamante, rappresenta la finestra vera e propria, e viene aggiornato da Intuition man mano che le informazioni contenute vengono modificate. Una volta utilizzata, quindi, la struttura **NewWindow** non serve più, a meno che non si voglia aprire una seconda finestra con caratteristiche analoghe alla precedente. Viceversa, il puntatore alla struttura **Window** va gelosamente conservato, dato che rappresenta il punto di aggancio per tutte quelle operazioni che servono alla gestione della finestra associata. Attenzione però: se volete modificare uno qualunque dei valori iniziali forniti ad Intuition in **NewWindow**, vi sconsigliamo caldamente di mettere le mani nella struttura **Window**. Intuition ci mette a disposizione delle apposite funzioni per fa ciò, come vedremo nelle prossime puntate.

4. Questi due campi definiscono solo i colori dei bordi della finestra, non dell'area interna. Se si vuole modificare il colore dello sfondo di una finestra, bisogna «riempirlo» con il colore desiderato utilizzando la funzione grafica **SetRast()** .

5. Attenzione: se aprite una finestra in uno schermo utente (CUSTOMSCREEN), dovete aver prima aperto lo schermo nel quale va la finestra, come vedremo nella settima puntata.

```
SEGNALATORI UTILIZZATI NELLA GESTIONE DELLE FINESTRE

  Segnalatore      Significato

  SIZEVERIFY       Avvertimi PRIMA che l'utente varii le dimensioni della
                   finestra ed aspetta ad effettuare la modifica finchè non
                   ti dò via libera (cosa da fare VELOCEMENTE).
  NEWSIZE          Avvertimi DOPO che l'utente ha variato le dimensioni della
                   finestra.
  REFRESHWINDOW    Avvertimi se la finestra ha bisogno di una "rinfrescata".
                   Ha senso solo se usato con finestre SIMPLE_ o SMART_REFRESH.
  MOUSEBUTTONS     Avvertimi se l'utente ha schiacciato un tasto del mouse, a
                   meno che tale evento non rientri in uno di quelli previsti da
                   un altro dei segnalatori specificati.
  MOUSEMOVE        Informami di tutti i movimenti del mouse. Funziona solo per
                   finestre in cui sia stato specificato REPORTMOUSE, o gadget
                   con FOLLOWMOUSE attivato. Usare con parsimonia!
  DELTAMOVE        Riporta i movimenti del mouse relativamente all'ultima
                   posizione rilevata.
  CLOSEWINDOW      Avvertimi se l'utente ha selezionato il gadget di chiusura
                   della finestra (se WINDOWCLOSE è attivo, ovviamente).
  GADGETDOWN       Avvertimi se l'utente ha selezionato un gadget tra quelli che
                   hanno GADGIMMEDIATE attivo.
  GADGETUP         Avvertimi se l'utente ha rilasciato un gadget tra quelli che
                   hanno RELVERIFY attivo.
  MENUPICK         Dimmi quale menù, elemento e/o sottoelemento è stato
                   selezionato.
  MENUVERIFY       Avvertimi PRIMA di aprire un menù.
  REQVERIFY        Avvertimi se stai per aprire un requester nella finestra.
  REQCLEAR         Avvertimi quando l'ultimo requester è stato chiuso nella
                   finestra. ATTENZIONE: in [1.2] questo segnale viene mandato
                   alla chiusura di OGNI requester, non solo dell'ultimo.
  REQSET           Avvertimi quando il primo requester è stato aperto nella
                   finestra. ATTENZIONE: in [1.2] questo segnale viene mandato
                   alla apertura di OGNI requester, non solo del primo.
  ACTIVEWINDOW     Avvertimi se la finestra viene attivata.
  INACTIVEWINDOW   Avvertimi se la finestra viene disattivata.
  RAWKEY           Se l'utente preme un tasto, dammi il codice relativo.
  VANILLAKEY       Se l'utente preme un tasto, dammi il carattere associato
                   utilizzando la mappa dei caratteri attualmente attiva.
  INTUITICKS       Segnalami ad intervalli regolari di circa un decimo di
                   secondo mentre la finestra è attiva.

ALTRI SEGNALATORI

  Segnalatore      Significato

  NEWPREFS         Avvertimi se l'utente od un altro programma ha modificato
                   le caratteristiche del sistema memorizzate in Preferences.
  DISKINSERTED     Avvertimi se è stato inserito un dischetto in un drive.
  DISKREMOVED      Avvertimi se è stato rimosso un dischetto da un drive.
  WBENCHMESSAGE    *** Usato da Intuition per comunicare col WorkBench ***
```

◄ *Figura 5 Segnalatori per Intuition's Direct Communications Message Port.*

```
                   Aggiungi il gadget di sistema per
  WINDOWSIZING       cambiare le dimensioni della finestra.
  WINDOWDRAG         spostare la finestra nello schermo.
  WINDOWDEPTH        muovere la finestra di fronte o sotto altre finestre.
  WINDOWCLOSE        chiudere la finestra.

                   Poni il gadget per la modifica delle dimensioni della finestra
  SIZEBRIGHT         all'interno del bordo di destra.
  SIZEBBOTTOM        all'interno del bordo inferiore.

  REPORTMOUSE      Avvertimi non appena il mouse si muove.
  RMBTRAP          Avvertimi se l'utente ha schiacciato il bottone di destra
                   del mouse.

  ACTIVATE         Quando apri la finestra, attivala subito.

*************************************************************************
* Questi segnalatori saranno analizzati in dettaglio nelle prossime puntate *
*************************************************************************
  BORDERLESS       Questa finestra non ha bordi.
  BACKDROP         Questa finestra sta sempre sotto a tutte le altre.
  GIMMEZEROZERO    Questa finestra ha bordi gestiti independentemente dalla
                   zona interna.

                   Se una parte della finestra (o tutta) è stata oscurata da
                   un'altra finestra o da un altro oggetto, il compito di
                   restaurarne il contenuto è
  SIMPLE_REFRESH   del programma
  SMART_REFRESH    di Intuition
  SUPER_BITMAP     di Intuition, ma le informazioni sono in RAM.

  NOCAREREFRESH    Non avvertirmi se c'è da restaurare una finestra.
*************************************************************************
*************************************************************************
```

▲ *Figura 6 - Segnalatori per la definizione delle caratteristiche della finestra.*

devono essere fonite come **short integer**. L'angolo superiore sinistro dello schermo viene assunto (0,0).

L'ultimo campo (**type**) definisce il tipo di schermo nel quale va aperta la finestra:
WBENCHSCREEN apri nello schermo del Work Bench;
CUSTOMSCREEN apri nello schermo puntato dal campo **Screen** (vedi nota 5).

Il programma di esempio

Veniamo ora al nostro programmino dimostrativo.

Innanzi tutto è necessario mettere in testa al programma almeno i seguenti #**include** file:

```
#include "exec/tipes.h"
#include "intuition/intuition.h"
```

Quindi bisogna aprire la libreria che contiene le routine di Intuition, cioè **intuition.library** (ricordate la prima puntata?).

Dato che apriremo una finestra sullo schermo del Work Bench non è necessario definire ed aprire un nuovo schermo. Definiremo invece la finestra basandoci su un insieme minimale di informazioni: a parte le dimensioni della finestra, il titolo e l'indicazione che deve essere aperta nello schermo del Work Bench infatti, l'unico altro campo specificato è quello che definisce le caratteristiche della finestra stessa. ACTIVATE indica che vogliamo che la finestra sia attivata subito dopo l'apertura (cosa che appare dal fatto che il titolo viene reso con un tratto marcato piuttosto che con il tratto «fantasma» tipico delle finestre e degli schermi non attivi [*ghost title*]); SMART_REFRESH indica che Intuition si deve prendere carico della gestione delle eventuali zone della finestra «oscurate» da altri oggetti, mentre NOCAREREFRESCH avverte Intuiton che non si vuole ricevere alcun messaggio relativo a tali operazioni di *refresh*.

Per quello che riguarda i colori e le dimensioni minime e massime della finestra, utilizziamo i valori dello schermo del Work Bench per i primi (–1–1) e le dimensioni iniziali della finestra per le seconde (0,0,0,0).

A questo punto non resta che aprire la finestra stessa usando la funzione **OpenWindow()** e passandogli il puntatore alla struttura **NewWindow** che abbiamo definito. Se tutto è stato fatto secondo le regole e se non ci sono altri problemi, Intuition aprirà la finestra e ci restituirà il puntatore alla struttura **Window**, puntatore che useremo in fondo al programma per richiuderla.

Il corpo del programma altro non è, per il momento, che un giro a vuoto [*empty loop*] da un milione di passi. Troppo? Provare per credere.

Chiudere la finestra è anche più semplice che aprirla: basta passare a **CloseWindow()** il puntatore alla struttura **Window** associato alla finestra da chiudere.

Ultimo avvertimento. Il programma riportato assume che il Work Bench sia già attivo al momento di venir eseguito, e quindi non fa nessun controllo in merito. Non lanciatelo perciò da CLI se nella vostra **Startup-Sequence** non è presente il comando **LoadWB** (od alternativamente avete già eseguito in precedenza **Run LoadWB**).

Conclusione

Bene, per il momento è tutto. Niente esercizio per stavolta, ci vediamo tra un mese per imparare ad aprire e chiudere gli schermi. MC

FLOPPERIA®

srl

Viale Monte Nero, 31
20135 Milano

Tel. (02) 55.18.04.84
Fax (02) 55.18.81.04

Vendita per corrispondenza in tutta Italia
Evasione ordini in 24 ore
Assistenza hardware/software, riparazioni e consulenza

AMIPRINT

Novità assoluta

E' finalmente disponibile l'interfaccia per collegare una qualsiasi stampante seriale per Commodore 64 a tutti gli Amiga, come MPS 801, 802 e 803; Okimate 20, NL-10C, Riteman, Seikosha, e tutte le altre compatibili seriali 64.

£. 69.000

PREZZI IVA 19% INCLUSA
I prezzi potranno variare a seconda dell'andamento delle valute estere

ANTIDRIVE
Dispositivo hardware da collegare alla porta drives per Amiga, che permette di sconnettere ogni unità esterna senza dover spegnere ogni volta il computer e rischiare di danneggiarlo.. Molto utile per recuperare memoria RAM preziosa per i giochi e le applicazioni grafiche.
£. 25.000

VIDEON
Digitalizzatore video a colori per Amiga, dotato di convertitore PAL-RGB con banda passante di 15 KHz per ottenere fantastiche immagini a colori dalla stupefacente qualità e risoluzione; collegabile con una qualsiasi fonte video PAL, come ad esempio videoregistratori, televisori, telecamere, computer, ecc., senza l'uso di filtri.
£. 420.000

STEREON
Nuovo digitalizzatore stereofonico per Amiga, con banda passante di 20.000 Hz, per sfruttare al massimo le capacità del computer.
£. 249.000

PAL GENLOCK
Genlock amatoriale per tutti gli Amiga, per ottenere ottime sovrapposizioni di titoli, animazioni, ecc.
£. 650.000

MINIRACK
Mobiletto per Amiga 500 o per Atari ST 520, sostiene il monitor e fornisce spazio per contenere drive, alimentatore, cavi, riviste, ecc.
£. 69.000

ROM KICKSTART 1.3
La nuova versione del sistema operativo Per Amiga 500 e 2000, non necessita di saldature.
£. 199.000

64 EMULATOR
La nuova versione del famoso emulatore C64, con gestione dell'audio, sprite, stampanti e drive dedicati; utilizza i drives Amiga, hard disk compresi.
£. 29.000

SUPPORTO TOWER
Sistema l'Amiga 2000 in verticale sul pavimento, per risparmiare spazio sulla scrivania e dare un tocco di professionalità al vostro sistema.
£. 59.000

ESPANSIONI PER AMIGA
Espansione per A-500 da 512 KB ..... telefonare
Gigatron 1.8 MB interna per A500/1000.....telefonare
Espansione interna autoconfigurante,
0 wait state, da 1 MB per A-1000 ..... 699.000
Espansione per A-2000 da 2 MB 1.199.000

DRIVES AMIGA
Drive esterno per Amiga passante 230.000
Drive interno per A-2000 ..... 199.000

STAMPANTI
Panasonic KX-1081, 160 cps, NLQ 439.000
Commodore MPS 1250 per C-64 ed Amiga.....499.000
Commodore MPS 1500 Centronics a colori 579.000
Commodore MPS 1550 C-64 a colori 599.000
Star LC-10, 140 cps, 80 col., bidirez., NLQ.....529.000
Star LC-10 versione a colori 629.000
Star LC-10 24 aghi 999.000
Nec P-2200, 170 cps, 80 col., 24 aghi,
bidirez., con 5 fonts NLQ residenti.....949.000

Super offerte natalizie per Amiga, Atari ST, Amstrad e compatibili MS-DOS. Telefonare

JITTER-RID
Filtro antiriflesso per monitor, riduce lo sfarfallio ed aumenta contrasto e definizione.
12" mono 35.000
14" color 40.000

X-RAYS PROTECTOR
Filtro per monitor o televisori, blocca totalmente l'emissione dei pericolosi raggi X dal cinescopio. Indispensabile per chi usa intensivamente il computer.
£. 299.000

Disponibile l'intera libreria di software Public Domain di Fred Fish
Richiedeteci il catalogo su disco che vi sarà spedito in contrassegno di £. 10.000

DUST REMOVER
Maneggevole mini-aspirapolvere er rimuovere la polvere che si accumula in tastiere, schede, ecc.
£. 25.000

FINAL IV
La prima cartuccia a finestre! Un'innovativo e completo sistema operativo tipo Geos con sprotettore programmi, hardcopy, calcolatrice, game-killer, Word Processor, gestione stampanti seriali/parallele e soprattutto è un ottimo velocizzatore per drive.
£. 79.000

DUST COVER
Copertina trasparente antistatica, protegge da polvere e liquidi dannosi.
per A-2000 ..... 18.000
per stampanti 80 col 15.000
per stampanti 132 col ... 18.000

SUPPORTI MONITOR
Robustissima base rotante su 360 gradi, inclinabile di 25 gradi, con piedini antivibrazione ed antislittamento.
9-13 pollici 35.000
14-18 pollici 40.000

PORTASTAMPANTI
Disegno funzionale, robusta costruzione in metallo, con supporto angolato per consentire la lettura durante la stampa.
80 col 29.000
132 col 39.000

KIT MPS 803 TURBO
Aggiunge 4 nuovi Fonts di caratteri con discendenti ed aumenta notevolmente la velocità di stampa.
£. 39.000

LEGGII
Per facilitare la battitura di lettere e di listati, anche da riviste, con righello per una rapida lettura.
portatile 29.000
con morsetto 39.000
con base basculante 49.000

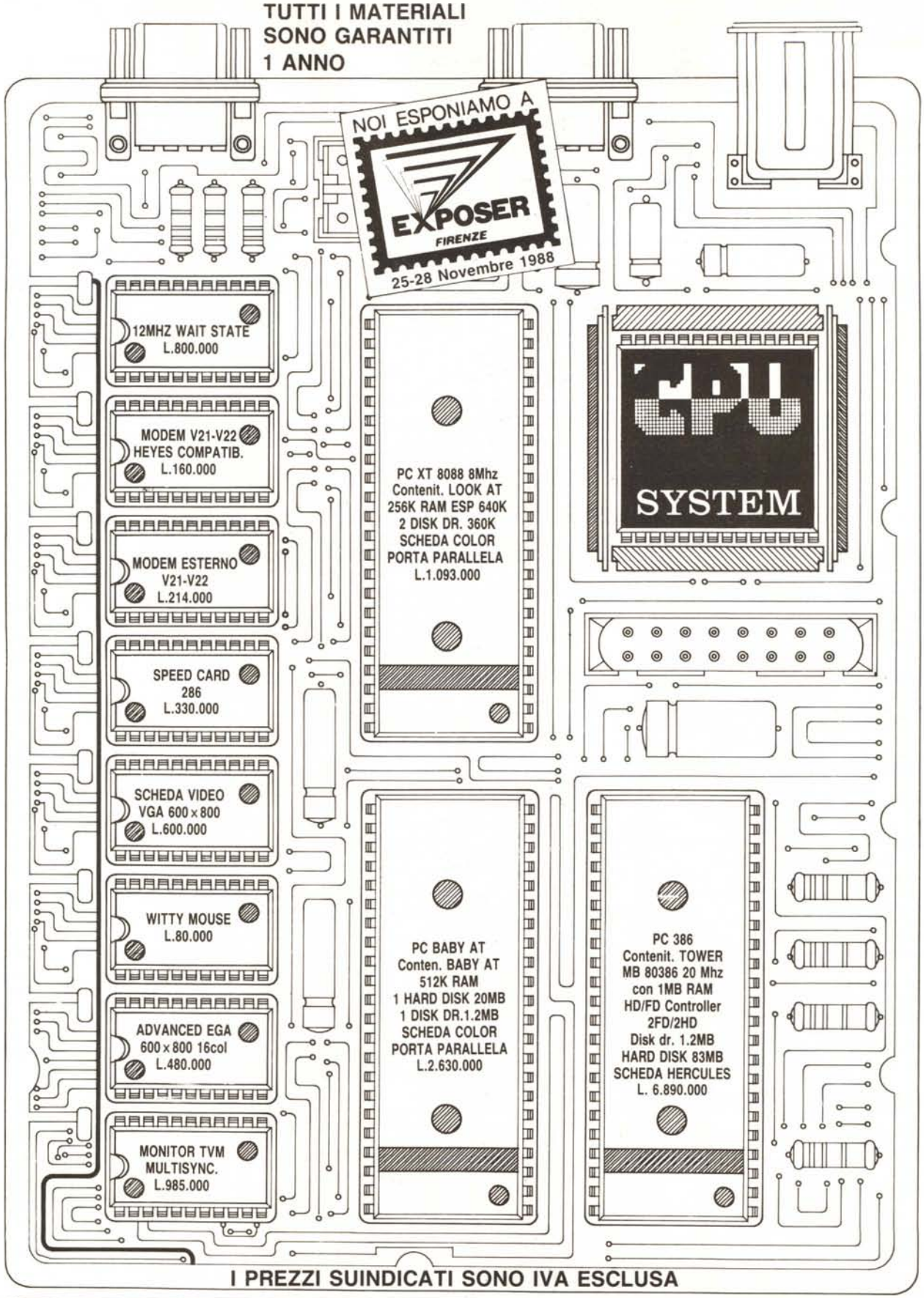
SUPERCOPY SYSTEM 88
Non permettere che il tuo prezioso programma originale si rovini! Con SuperCopy System 88 é possibile effettuare copie di sicurezza di tutto il software protetto, compresi gli "impossibili" come Geos 64 e 128. Per C64/128 e drive 1541, 1541C ed OC-118.
£.45.000

PARTI DI RICAMBIO COMMODORE

ROM Basic 901226	25.000
ROM Kernal 901227	25.000
ROM char. gen. 901227	25.000
6526 CIA	27.000
6581 SID	29.000
6510 microprocessor	20.000
6569 int. 906111/01	59.000
82S100 PLA	32.000
8701 PAL C64	18.000
7406/7416 IC buffer inverter	5.000
4066 IC quad switch	5.000
7805 IC voltage regulator	4.000
7812 IC voltage regulator	4.000
Quartz crystal 17,734 MHz	8.000
8501 CPU C16	21.000
8360 video controller C16	41.000
6502 microprocessor	19.000
6522 floppy	16.000
8722 MMU	20.000
8721 PLA	31.000
8563	40.000
8502	24.000
8566 PAL video	41.000
8564 NTSC VIC	41.000
4164 RAM	5.000
41464 RAM	29.000
ROM floppy 1541	22.000
Logic array 1541	40.000
2364-130 ROM	40.000
7700-010 PLA C16	30.000
6529 B C16/Plus 4	4.000
Ted Kernal C16	55.000
Ted Basic C16	55.000
8371 PAL Amiga	55.000
8520 CIA	25.000
68000 MPU	39.000
68010 MPU	49.000
8364 Paula	69.000
315093 IC	49.000
5719 IC	53.000
8362 Denise	59.000
Alimentatore C16	49.000
Alimentatore C64	59.000
Alimentatore C128	79.000

Richiedete il nostro catalogo GRATUITO

TUTTI I MATERIALI
SONO GARANTITI
1 ANNO



I PREZZI SUINDICATI SONO IVA ESCLUSA

CPU - 50127 FIRENZE - Via M. Ulivelli 39/r - Tel. 055-4361096 - TELEX 574354 SEAC I - FAX 055/4361096
CPU - 50047 PRATO (FI) - Via Settesoldi 32 - Tel. 0574/434554