Programmare in C su Amiga

di Dario de Judicibus

nona parte

La canzone dice che le stelle sono tante... e tanti sono i tipi di finestre che Intuition ci permette di aprire. Ovviamente ancora di più sono le possibilità che si aprono al programmatore esperto

Vi siete esercitati ad aprire e chiudere schermi e finestre di varie dimensioni, risoluzione e colore? Se sì siete pronti ad affrontare un discorso un pochino più avanzato, altrimenti vi suggeriamo di mettere da parte quest'articolo e giocare ancora un po' con il vostro Amiga, in modo da evitare che quanto stiamo per dirvi finisca per restare un bel discorso teorico senza risvolti pratici.

Introduzione

Finora abbiamo visto come aprire schermi e finestre di tipo «classico». Anche se siamo in grado già da adesso di dar vita ad un numero elevato di combinazioni giocando sulle dimensioni, la risoluzione, i colori, i gadget di sistema e via dicendo, si tratta sempre dello stesso modello: cambiano gli optional ma il motore è quello. Intuition ci mette tuttavia a disposizione molti più modelli di finestre di quanto potremmo pensare, con caratteristiche molto differenti fra loro e, di conseguenza, utilizzi molto diversi.

In questa puntata analizzeremo in dettaglio i vari tipi di finestre che Intuition ci permette di aprire, le loro caratteristiche ed il loro utilizzo.

```
/* ------------------------------------------------------------------------ */
/* - Programmino d'esempio -- Dario de Judicibus -- 21 Dicembre 1988 - Pta 9 */
/* ------------------------------------------------------------------------ */
/*  Il seguente programma serve ad dimostrare come, in una finestra senza   */
/*  bordi, questi possono essere evidenziati dalla presenza di elementi che */
/*  tipicamente vanno a posizionarsi nei bordi di una finestra.             */
/* ------------------------------------------------------------------------ */

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

struct IntuitionBase *IntuitionBase;

#define WFLAGS WINDOWSIZING|WINDOWCLOSE|WINDOWDRAG
#define RFLAGS SMART_REFRESH|ACTIVATE|NOCAREREFRESH

struct NewWindow nw = /* struttura di definizione di una finestra 200x100   */
{
  20,20,200,100,0,1,CLOSEWINDOW,WFLAGS|RFLAGS|BORDERLESS,
  NULL,NULL,"Finestra",NULL,NULL,0,0,400,200,WBENCHSCREEN
};

#define INTLIB "intuition.library"
#define INTVER 33L

main()
{
  struct Window *w;

  /* --- apri la libreria di Intuition ------------------------------------ */
  IntuitionBase = (struct IntuitionBase *)OpenLibrary(INTLIB,INTVER);
  if (IntuitionBase == NULL) exit(FALSE);

  /* --- apri la finestra 200x100 con l'angolo sup. sin. a (20,20) --------- */
  if ((w = (struct Window *)OpenWindow(&nw)) == NULL) exit(FALSE);

  /* --- aspetta un qualunque segnale attraverso IDCMP --------------------- */
  Wait(1<<w->UserPort->mp_SigBit);

  /* --- chiudi tutto ed esci ---------------------------------------------- */
  CloseWindow(w);
  exit(TRUE);
}
```

Figura 1 - Programmino d'esempio.

Personalizzazione delle Finestre

Tutti sappiamo come è fatta una finestra. Si tratta di un'area rettangolare, delimitata da un bordo che la differenzia dal resto dello schermo, spesso dotata di un titolo ed una serie di gadget che permettono di operare su di essa. In genere il gadget di chiusura è in alto a sinistra, quelli di profondità (vedi l'ottava puntata) in alto a destra, e quello per il ridimensionamento in basso a destra. Spesso è possibile «agganciare» il titolo (barra di spostamento) o far scorrere il contenuto della finestra per mezzo di un cursore [*slider*] posto in una barra verticale, in genere posizionata sul bordo destro della finestra stessa.

In realtà una finestra può essere molto differente da quella appena descritta. Ad esempio può non avere bordo, oppure essere agganciata al fondo dello schermo in modo che altre finestre non possano posizionarvisi dietro. Può non avere nessuno dei gadget sopra nominati od averne viceversa di completamente diversi. L'unica cosa sicura che accomuna tutte le finestre è quella di essere un'area rettangolare associata ad uno schermo e sulla quale si possono effettuare operazioni in ingresso ed uscita.

Vediamo ora appunto alcune tipologie di finestra che possono essere richieste ad Intuition. Tenete presente che, anche se qui di seguito parleremo di finestra «X» piuttosto che «Y», queste caratteristiche possono essere combinate fra loro in modo da creare tutta una serie di og-

getti estremamente personalizzati ed opportunamente configurati in relazione all'uso che il programmatore ha previsto.

Il bordo delle Finestre

Il bordo di una finestra [*border*] ricopre un ruolo fondamentale nella gestione delle stesse. Esso infatti non solo rappresenta l'elemento chiave per distinguere la finestra dal resto dello schermo, ma può servire anche a contenere tutta una serie di oggetti [*border gadgets*] che servono ad operare sia sul contenuto della finestra che sulla finestra stessa. In realtà dobbiamo distinguere tra linee di bordo [*border lines*] ed area del bordo [*border area*]. Se non si specifica altrimenti, Intuition disegna il bordo utilizzando una doppia cornice: quella più esterna è resa nel colore definito in **BlockPen** (vedi MCmicrocomputer n. 79 — 6ªpuntata: *La struttura NewWindow*), mentre per quella interna viene utilizzato il valore nel campo **DetailPen**. L'area compresa tra le due cornici viene utilizzata da Intuition per i gadget di sistema e quelli utente, se ce ne sono. Tale area quindi non ha lo stesso spessore dappertutto ma è costante solo lungo un singolo bordo. Ad esempio, una finestra dotata solo di gadget di chiusura, di profondità e di spostamento, avrà il bordo superiore molto più spesso (diciamo 10 pixel) degli altri tre (solo due pixel, uno per cornice).

In genere il bordo è disegnato automaticamente da Intuition, ed è contenuto all'interno delle dimensioni specificate nella struttura **NewWindow**. Questo vuol dire che, se si apre una finestra 120×120 dotata di bordo, l'area grafica effettivamente disponibile al programmatore sarà inferiore a quella specificata. Lo spessore dei quattro lati del bordo viene mantenuto da Intuition in quattro campi della struttura **Window**, come riportato in figura 4 della 6ª puntata, e precisamente:

BorderLeft = Bordo di sinistra
BorderRight = Bordo di destra
BorderTop = Bordo superiore
BorderBottom = Bordo inferiore

Quindi, se si vuole tracciare una linea diagonale che parte dal bordo superiore sinistro fino a quello inferiore destro, e supponendo che le dimensioni della nostra finestra siano appunto 120×120, bisognerà specificare come coordinate di partenza e di arrivo rispettivamente **(0+BorderLeft, 0+BorderTop) e (120-BorderRight, 120-BorderBottom).**

Finestra senza bordo

Come dice il nome, una finestra senza bordo [*borderless window*] altro non è che una normale finestra in cui però Intuition non disegna automaticamente il bordo. Per far questo basta usare la costante **BORDERLESS** nel campo **Flags** della struttura **NewWindow**. Attenzione però: non è affatto detto che una finestra senza bordo non abbia affatto bordi! In realtà, se il programmatore ha specificato il titolo della finestra, gadget di sistema o gadget utente che vanno posizionati nel bordo della finestra (vedi nota 1), questi rappresenteranno comunque una sorta di bordo parziale per quella determinata finestra. Provate a scrivere e compilare il programmino in figura 1. Vedrete che, anche se il bordo è invisibile, quello superiore è comunque evidenziato dal gadget di chiusura, dal titolo e dalla barra di spostamento mentre l'angolo inferiore destro è rivelato dal gadget di ridimensionamento. Naturalmente, se togliete questi elementi della finestra, dato che uno schermo e le finestre che gli appartengono condividono la stessa tavolozza [*palette*] dei colori, non sarete più in grado di dire dove esattamente è posizionata la finestra.

Un possibile uso di una finestra senza bordi è quello di simulare la classica *console* da terminale utilizzando una finestra di fondo (vedi più avanti), senza bordi ed a tutto schermo. Un esempio può essere fornito dal programma PD (vedi nota 2) **border.**

I fortunati possessori di un Sidecar o di una scheda BridgeBoard conoscono molto bene una finestra di questo tipo. Se infatti nella finestra *PC Color Display*

```
/*
 *  Struttura ColorMap: tabella dei colori per una certa ViewPort.
 *  Se Type == 0 allora ColorTable e una tabella di UWORDS xRGB.
 */
struct ColorMap
{
   UBYTE Flags;
   UBYTE Type;
   UWORD Count;
   APTR  ColorTable;
};

/*
 *  Struttura ViewPort: definisce una vista ritagliata da un raster di
 *  dimensioni uguali o maggiori (BitMap).
 */
struct ViewPort
{
   struct   ViewPort *Next;         /* puntatore alla succesiva in lista  */
   struct   ColorMap  *ColorMap;    /* tabella dei colori                 */
   struct   CopList  *DspIns;       /* struttura usata da MakeView()      */
   struct   CopList  *SprIns;       /* struttura usata per gli sprite     */
   struct   CopList  *ClrIns;       /* struttura usata per gli sprite     */
   struct   UCopList *UCopIns;      /* Lista "copper" dell'utente         */
   SHORT    DWidth,DHeight;         /* dimensioni della vista             */
   SHORT    DxOffset,DyOffset;      /* posizione nello schermo            */
   UWORD    Modes;                  /* modi di visualizzazione            */
   UWORD        SpritePriorities;   /* usata da MakeVP [1.2]              */
   UBYTE        reserved;
   struct   RasInfo *RasInfo;       /* --- vedi sotto ------------------   */
};

/*
 *  Struttura View: definisce la struttura completa dello schermo.
 */
struct View
{
   struct ViewPort *ViewPort;  /* puntatore alla lista delle ViewPort     */
   struct cprlist *LOFCprList; /* usato x interlacciato/non-interlacciato */
   struct cprlist *SHFCprList; /* usato solo se interlacciato             */
   short DyOffset,DxOffset;    /* posizione della intera vista nello      */
                               /* schermo, relativamente a quella base    */
   UWORD   Modes;              /* modi: INTERLACE, GENLOC, ...            */
};

/*
 *  Struttura RasInfo: indirizzo e posizione della BitMap
 */
struct RasInfo  /* used by callers to and InitDspC() */
{
   struct   RasInfo *Next;     /* secondo raster solo per dualplayfield   */
   struct   BitMap *BitMap;    /* struttura BitMap (raster)               */
   SHORT    RxOffset,RyOffset; /* offset per lo scrolling della vista     */
};

/*
 *  modi di visualizzazione usati in IVPargs
 */
#define PFBA            0x40
#define DUALPF          0x400
#define HIRES           0x8000
#define LACE            4
#define HAM             0x800
#define SPRITES         0x4000
#define VP_HIDE         0x2000
#define GENLOCK_AUDIO   0x100
#define GENLOCK_VIDEO   2
#define EXTRA_HALFBRITE 0x80
```

Figura 2 - Strutture View, ViewPort, ColorMap, RasInfo.

che si ottiene aprendo l'icona *PC Color* si fa un doppio *click* col bottone di selezione del mouse, questa si allarga a prendere tutto lo schermo nel senso orizzontale: i bordi verticali scompaiono, quello orizzontale inferiore si riduce ad una sottile linea, mentre quello superiore è l'unico che rimane come prima, dato che contiene i vari gadget di chiusura, di spostamento, di profondità ed il titolo.

Finestra Doppio Zero

Una finestra Doppio Zero [*GimmeZeroZero (GZZ)*] è in realtà formata da due finestre sovrapposte ed incollate insieme. Quella di sotto è più grande, è chiamata *esterna* ed ha le dimensioni specificate dal programmatore, mentre quella di sopra è più piccola, prende il nome di *interna* e le sue dimensioni sono tali che lo spazio tra le due finestre consenta l'inserzione di quegli elementi che nelle finestre normali vanno nel bordo. Come quindi avrete già capito, una finestra Doppio Zero non si differenzia da una finestra normale tanto per l'aspetto, quanto per il fatto che l'area di bordo non è ricavata dalla finestra stessa, ma dal perimetro visibile della finestra esterna. Dato che tuttavia le operazioni grafiche vengono effettuate sulla finestra interna, non sarà più necessario adottare le coordinate su cui si opera in modo da evitare di disegnare sul bordo stesso, in quanto questi non appartiene allo stesso *raster* nel quale si agisce graficamente. L'angolo superiore destro (origine) della finestra interna avrà quindi sempre le coordinate (0,0), da cui il nome *GimmeZeroZero*, «dammi un'origine nel punto (0,0)» appunto.

D'altra parte, essendo la finestra interna più piccola di quella specificata nella struttura **NewWindow**, può rendersi necessario sapere le dimensioni di tale area, dimensioni che dipendono dal

```
typedef UBYTE *PLANEPTR;

struct BitMap
{
   UWORD      BytesPerRow;  /* Numero di BYTEs per ogni riga            */
   UWORD      Rows;         /* Numero di righe                          */
   UBYTE      Flags;
   UBYTE      Depth;        /* Numero di piani                          */
   UWORD      pad;
   PLANEPTR   Planes[8];    /* Fino a ben NOVE piani! Non si sa mai...  */
};
```

Figura 3
La struttura BitMap.

```
==== InitBitMap() == Inizializza una struttura BitMap ====================

---- prototipo ----------------------------------------------------------

void InitBitMap(struct BitMap *, SHORT, SHORT, SHORT);

---- utilizzo -----------------------------------------------------------

InitBitMap(&mappa, piani, larghezza, altezza);

dove:   mappa      è la struttura da inizializzare (&mappa ne è quindi il
                   puntatore). Attenzione: se definite

                   >>> struct BitMap *mappa;

                   dovete usare mappa, e non &mappa nella inizializzazione.

        piani      è il numero di piani desiderati (e quindi di colori,
                   secondo la formula

                                 piani
                   >>> colori = 2

        larghezza  larghezza in pixel di un singolo piano (deve essere
                   un multiplo di 16)

        altezza    altezza in pixel di un singolo piano


==== AllocRaster() == Riserva memoria per allocare un piano ===============

---- prototipo ----------------------------------------------------------

APTR AllocRaster(SHORT, SHORT);

---- utilizzo -----------------------------------------------------------

piano = AllocRaster(larghezza, altezza);

dove:   piano      è il puntatore all'area di memoria corrispondente al
                   piano da allocare.

        larghezza  larghezza in pixel del piano da allocare (deve essere
                   un multiplo di 16)

        altezza    altezza in pixel del piano da allocare

 ========================================================================

 Attenzione: se usate il Lattice C 4.00, 4.01, o 5.00, ed includete i
             prototipi delle funzioni grafiche fornite nel file
             <proto/graphics.h>, i due prototipi per le funzioni riportate
             sono definiti come segue:

             void InitBitMap(struct BitMap *, long, long, long);

             PLANEPTR AllocRaster(long, long);

 ------------------------------------------------------------------------
```

Figura 4 - Funzioni per l'inizializzazione di una struttura BitMap.

```
#include "exec/types.h"
#include "intuition/intuition.h"
#include "graphics/gfxmacros.h"
#include "proto/graphics.h"

#define NPIANI      2
#define LARGHEZZA 380
#define ALTEZZA   380

struct BitMap mappa;
int conta;

.
.
.
/*
*   Inizializza la struttura BitMap
*/
InitBitMap(&mappa,NPIANI,LARGHEZZA,ALTEZZA);

/*
*   Alloca i piani necessari
*/
for (conta = 0; conta < NPIANI; conta++)
{
   mappa.Planes[conta] = AllocRaster(LARGHEZZA,ALTEZZA);
   if (mappa.Planes[conta] == 0)
   {
      --- processa la condizione di errore ---
   }
}

------------------------------------------------------------------------
--- OK, adesso riempi i piani come meglio desideri (per esempio caricando ---
--- un file IFF, decifrandolo e ricomponendolo sotto forma di sequenze di ---
--- bit) oppure inizializza una struttura RastPort per potervi disegnare  ---
--- utilizzando le funzioni grafiche elementari della graphics.library    ---
---                                                                       ---
--- Esempio di inizializzazione di una RastPort:                          ---
---                                                                       ---
---    struct RastPort rp;                                                ---
---                                                                       ---
---    InitRastPort(&rp);                                                 ---
---    rp.BitMap = &mappa;                                                ---
------------------------------------------------------------------------

/*
*   Fine: libera i piani allocati
*/
for (conta = 0; conta < NPIANI; conta++)
{
   if (mappa.Planes[conta] != 0)
   {
      FreeRaster(mappa.Planes[conta],LARGHEZZA,ALTEZZA);
   }
}
```

Figura 5 - Preparazione di una struttura BitMap.

tipo di gadget utilizzati e dalle opzioni specificate in **NewWindow.Flags**. Tali valori possono essere trovati nella struttura **Window** nelle variabili **GZZWidth** e **GZZHeight**. Inoltre, dato che in un programma grafico il mouse spesso rappresenta lo strumento principale per disegnare, può tornare utile conoscerne la posizione nel sistema di riferimento della finestra interna. Questa informazione è aggiornata da Intuition in altri due campi della stessa struttura: **GZZMouseX** e **GZZMouseY**.

Il vantaggio di non dover adattare tutte le operazioni grafiche alla presenza di un bordo disegnato da Intuition *nella stessa finestra* in cui opera, ha però un prezzo: quello della memoria addizionale che serve a gestire due finestre invece di una sola e quello della riduzione di performance negli spostamenti e ridimensionamenti di una doppia finestra, prezzo che si fa sentire con il crescere del numero di finestre GZZ aperte.

La finestra GZZ è particolarmente utile per programmi di grafica, mentre, nel caso di testi, non è necessario utilizzare una GZZ per evitare di «toccare» i bordi, dato che, come avremo modo di vedere molto più avanti, la *console device* gatantisce un corretto posizionamento dei caratteri del testo.

Per ottenere da Intuition una finestra Doppio Zero è sufficiente impostare il valore **GIMMEZEROZERO** in **NewWindow.Flags.** Tutti i gadget di sistema richiesti verranno automaticamente posizionati nella finestra esterna, mentre quelli utente andranno in quella interna a meno che il programmatore non specifichi la costante **GZZGADGET** nell'apposito campo della struttura **Gadget**, come vedremo quando parleremo di gadget utente.

Un ultimo avvertimento, frutto di una esperienza personale e di una certa dose di lavoro di *debug*. Alcuni mesi fa scrissi un programmino che utilizzava una finestra GZZ ed aveva un gadget utente nel bordo inferiore. Molto probabilmente riporterò il codice sorgente quando incominceremo a parlare di gadget. Si trattava di un programmino basato su un articolo de *Le Scienze*, la versione italiana della nota rivista *Scientific American*. In una settimana misi a punto il tutto e dopo averci giocato un po' mi dedicai ad altro. Il mese dopo lanciai di nuovo il programma e mi accorsi con stupore che non funzionava più. Dato che non avevo cambiato nulla e che aveva sempre funzionato regolarmente la cosa mi lasciò alquanto perplesso. Così, usando il buon vecchio metodo del rasoio di Occam mi dissi, dopo essermi fatto venire un bel po' di mal di testa a onor del vero, che se il baco [*bug*] non stava *nel* programma (vedi nota 3) doveva stare *fuori*. Geniale, no? In realtà quello che era successo è la seguente cosa. Qualche giorno prima avevo sostituito il buon vecchio **PopCLI**, un programma PD che serve a far comparire una finestra **CLI** premendo una opportuna combinazione di tasti sulla tastiera (Amiga sinistro + Escape), con il più potente **DMouse** di Matt Dillon, anche questo PD. Come tutti i programmi di Matt, **DMouse** è una di quelle «utilità» [*utilities*] che non possono mancare in un Amiga serio. Si tratta di un sistema integrato che combina una serie di funzionalità estremamente interessanti:

- acceleratore dei movimenti del mouse;
- autoattivazione delle finestre sotto il mouse;
- finestra CLI a comparsa (come **PopCLI**);
- movimenti in profondità di finestre e schermi senza dover usare i gadget di profondità;

Note

1. Mentre i gadget di sistema sono sempre posizionati all'interno del bordo, quelli utente possono sia essere posti nel bordo che nel corpo vero e proprio della finestra. Un esempio di gadget utente nel bordo di una finestra è quello utilizzato per ridurre le dimensioni di una finestra e riportarla in seguito a pieno schermo in **WordPerfect (tm)**, mentre gadget nel corpo della finestra sono, ad esempio, quelli di **CliMate**.

2. Da qui in poi, useremo la seguente terminologia nel classificare i programmi:
Prodotti - sono tutti quei programmi che vengono venduti sotto particolari condizioni d'uso (licenza d'utilizzo) e non possono essere copiati o duplicati anche parzialmente se non per uso personale (copie di riserva o *backup*);
PD - sono tutti quei programmi detti di «pubblico dominio» *[public domain]* che possono essere duplicati e sono soggetti a condizioni molto meno restrittive dei prodotti commerciali. Ce ne sono di tre tipi:
a. *freeware* senza alcuna restrizione: possono essere copiati, modificati, utilizzati all'interno di altri programmi od addirittura prodotti (e quindi venduti). In quest'ultimo caso però l'autore si riserva comunque il diritto di utilizzare il programma a scopi commerciali in futuro, e comunque il suo nome deve comparire sempre assieme al programma stesso;
b. *freeware con copyright*: come nel caso precedente, ma in ogni caso i programmi non possono essere venduti od inclusi in un prodotto senza l'autorizzazione dell'autore stesso, che ne conserva i diritti;
c. *shareware*: si tratta di programmi liberamente distribuibili, come i precedenti; tuttavia, chi dovesse farne un uso frequente, ha l'obbligo morale di mandare un contributo (dai 5 ai 15 dollari) all'autore, in modo da permettergli di garantire supporto al programma stesso. Mmmm.... vedo già qualcuno scuotere la testa dubbioso. Beh, vi assicuro per esperienza personale che molte persone negli Stati Uniti lo fanno veramente. D'altrocanto spesso chi spedisce questo contributo riceve gratuitamente un numero di licenza, ulteriore documentazione, una versione più potente e/o nuovi rilasci *[release]* quando disponibili.

Tutti i programmi PD devono essere distribuiti al solo prezzo della duplicazione, cioè quello del dischetto più un tot per il tempo impiegato (intorno ai $5, in USA). Non possono essere commercializzati senza l'autorizzazione dell'autore e, in tal caso, cessano di essere PD. Un caso eclatante è quello di **blink**, un *linkage editor* per il linguaggio C che ha preso il posto dell'**alink** nel prodotto della Lattice Inc.

3. I lettori più attenti e con una qualche infarinatura di inglese avranno notato come, a volte, i termini italiani da me usati in corrispondenza di quelli inglesi non ne rappresentino sempre la traduzione letterale. Questo è dovuto principalmente a due motivi: uno pratico, quando in qualche modo il termine italiano corrispondente esiste e si è affermato pur non avendo necessariamente una relazione col termine inglese; l'altro estetico, qualora una eventuale traduzione letterale non renda intuitivamente il significato del termine, mentre un'opportuna scelta può dare risultati a mio avviso più accettabili.

Un esempio del primo tipo è *baco*, oramai ampiamente utilizzato pur non essendo la traduzione di *bug*, che vuol dire piuttosto *cimice* o *piccolo insetto*. Per quello che riguarda il secondo tipo, un esempio può essere trovato nella scorsa puntata dove, a fronte del termine *pattern* che vuol dire *modello, disegno per stoffa o tappezzeria*, ho usato il termine *tassellatura*, molto più vicino in italiano al significato informatico. Ovviamente il tutto è soggettivo. Sono aperto ad ogni suggerimento a riguardo.

4. Le strutture che stiamo descrivendo (**View, ViewPort, RasInfo, BitMap**) sono strutture base situate molto *al di sotto* di quelle di Intuiton. Esse sono usate da Intuiton infatti, sia per gli schermi che per le finestre. Se andate a vedere le varie strutture usate da Intuition, vi renderete conto subito, ad esempio, che è possibile trovare un puntatore ad una struttura **BitMap** non solo nelle strutture **Screen** e **Window**, ma anche in strutture come **Requester**, senza contare che Intuition usa internamente raster propri per mantenere i dati relativi ad altri elementi, come ad esempio le tendine a discesa dei menu.

Il termine finestra usato nella frase in questione quindi, deve intendersi nel senso più classico della parola, e cioè un'apertura per lasciare intravedere parzialmente qualcosa di più grande che sta dall'altra parte.

5. Useremo d'ora in poi il termine *restauro*, piuttosto che *ristoro* o *rinfesco*, per tradurre quello inglese *refresh*, in quanto più adatto a renderne il significato pur non essendo la traduzione letterale di tale parola.

e varie altre cose.

L'effetto collaterale era dotuto proprio all'ultima funzione della lista.

Per portare una finestra di fronte a tutte le altre, come ben sapete, bisogna usare il gadget di retrofronte, sempre che sia disponibile. **DMouse** permette di fare questo semplicemente posizionando il mouse in un punto qualunque della finestra e premendo il bottone di sinistra del mouse stesso (notate, lo stesso che si usa per selezionare un gadget). Tale meccanismo ha due vantaggi: il primo è quello di non dover spostare il mouse sull'angolo in alto a destra della finestra per poi riportarlo nel punto voluto (piuttosto comodo se si sta disegnando su di una finestra parzialmente coperta da qualcos'altro e si arriva nella zona occultata), il secondo è quello di poter portare davanti anche quelle finestre che non prevedono l'uso dei gadget di profondità. Il guaio era che, quando posizionavo il mouse sul gadget che avevo fatto porre nel bordo tra le due finestre e lo selezionavo premendo il bottone di selezione, **DMouse** intercettava l'azione e portava la finestra esterna *di fronte* a quella interna. Dato che le operazioni grafiche avvenivano comunque su quella interna, ora occultata, sembrava che di fatto non accadesse niente. In realtà quella che vedevo era l'area «nascosta» della finestra esterna, che normalmente non è mai esposta.

Per fortuna Matt aveva dato la possibilità di eliminare una o più funzionalità a piacere in modo del tutto indipendente l'una dall'altra, o di definire un eventuale tasto da premere in combinazione col bottone di selezione del mouse per attivare i vari meccanismi disponibili nel programma. Avevo quindi più di un modo di aggirare il problema senza peraltro rinunciare ad un ottimo programma. Quanto detto vi dovrebbe dare un'idea di quanto difficile sia a volte l'analisi di un problema in un ambiente multiprogramma.

Finestra di fondo

Una finestra di fondo [*BackDrop Window*] ha la caratteristica di rimanere sempre agganciata allo sfondo dello schermo su cui viene aperta, indipendentemente da come vengono arrangiate in profondità altre eventuali finestre appartenenti allo stesso schermo. Nel caso che già esistano una o più finestre dello stesso tipo, cioè di fondo, essa si posiziona comunque sotto tutte le altre. L'unico tipo di finestra quindi che può essere posizionata sotto una finestra di fondo, è un'altra finestra di fondo aperta successivamente.

L'unico gadget di sistema che si può attaccare ad una finestra di fondo è quello di chiusura. Viceversa è possibile attaccarvi qualunque gadget utente.

Uno degli utilizzi più comuni di una finestra di fondo è quello di creare uno sfondo su cui si può disegnare liberamente senza incorrere negli effetti collaterali che si possono avere quando si disegna direttamente su di uno schermo, come ad esempio rovinare un menu o perdere l'area corrispondente alla barra di scorrimento dello schermo, come già accennato nella scorsa puntata.

Per far questo, oltre a specificare **BACKDROP** nel campo **Flags** in modo da ottenere una finestra di fondo, si può utilizzare anche **BORDERLESS** per togliere i bordi, utilizzare le dimensioni massime dello schermo sia orizzontalmente che verticalmente (utilizzando i campi della struttura **GFxBase** già descritti la volta scorsa) ed inserire l'istruzione **«ShowTitle(FALSE);»** prima di dare il controllo all'utente. Naturalmente non devono essere specificati gadget di sistema, sempre che si voglia uno sfondo uniforme senza oggetti estranei.

Finestra a mappa

Nella scorsa puntata abbiamo detto che tutto ciò che appare sul nostro schermo può essere descritto tramite una struttura detta **View**. Tale struttura contiene, fra l'altro (vedi figura 2), una lista di **ViewPort** che rappresentano le varie aree grafiche presenti sullo schermo. In realtà una **ViewPort** altro non è che una «finestra» (Attenzione! Vedi nota 4) aperta su di un'area chiamata *raster*, di dimensioni non inferiori alle sue. Il raster è appunto l'immagine prodotta dalle informazioni memorizzate in memoria sotto forma di una mappa di bit [*bit-map*].

La mappa è formata da più piani [*bitplane*], come già accennato un po' di tempo fa (vedi MCmicrocomputer n. 80 pag. 214 figura 1), ognuno dei quali contiene un bit per pixel e la cui sovrapposizione permette di gestire immagini a più colori. Una struttura **BitMap** (vedi fig. 3) è appunto quella struttura che contiene tutti i bit necessari per formare un'immagine grande fino a 1024×1024 pixel. Ovviamente, a meno di non avere uno schermo così grande un'immagine di tali dimensioni non potrà essere visualizzata tutta allo stesso momento, ma sarà sempre possibile muovere la nostra finestra su e giù, a destra ed a sinistra, in modo da vedere un pezzo alla volta tutto il raster.

La figura 4 mostra come va inizializzata questo particolare tipo di struttura.

Tornando ai tipi di finestre disponibili in Intuition quindi, una finestra a mappa [*SuperBitMap*] altro non è che una normale finestra in cui il raster associato non viene fornito da Intuition ma dallo stesso programmatore. Questo comporta due vantaggi: il primo è legato ad una particolare tecnica di restauro (vedi nota 5) di cui parleremo nella decima puntata, il secondo alla possibilità da parte del programmatore di caricare immagini preparate in precedenza come fondo per le proprie finestre.

La figura 5 mostra un semplice scheletro di codice per la preparazione di una mappa.

Per definire una finestra a mappa, basta impostare in **Flags** il valore predefinito **SUPER_BITMAP** inizializzare mappa e piani come mostrato in figura, assegnare a **NewWindow.BitMap** il puntatore alla struttura così preparata ed aprire la finestra come al solito. In genere si preferisce utilizzare finestre GZZ quando si hanno finestre a mappa, in modo da evitare che Intuition disegni nel raster utente i bordi della finestra. In tal caso **Flags** conterrà **SUPER_BITMAP|GIMMEZERO-ZERO**. Da notare che in questo caso non è necessario definire una **RastPort** che, ricordo, serve come aggancio ad ogni operazione grafica che si voglia effettuare nel raster. Viceversa si può utilizzare una mappa utente come area grafica fuori linea [*off line*] ed una normale finestra (non mappa) a cui è associata un raster in linea [*on line*]. Questa tecnica permette di disegnare nella prima e trasferire poi il tutto alla seconda in modo da evitare che l'utente veda il disegno mentre si forma. Ma di questo parleremo in un'altra serie di articoli, se l'interesse suscitato da questa prima serie giustificherà lo sforzo. Affronteremo allora le tecniche di *programmazione avanzata*: vedremo il meccanismo a pacchetti dell'AmigaDos, entreremo più in dettaglio nel multitasking con i *device* e gli *handler*, parleremo di librerie utente, grafica di base (Blitter, Copper) e molto altro ancora. Almeno se avrò il tempo di impararle prima io tutte queste cose... Ma tutto ciò è nel futuro.

Conclusione

Abbiamo visto i vari tipi di finestre e le loro caratteristiche. Nella prossima puntata vedremo le tecniche di restauro, come cambiare il puntatore associato ad una finestra, ed alcune primitive grafiche per operare in essa.

Nel frattempo andate a riprendere i programmini che avete fatto per esercitarvi nel mese scorso e, su quella base, provate ad aprire alcune finestre descritte in questo articolo. Vi consigliamo di non modificare direttamente il codice vecchio che, almeno si spera, funziona già. Piuttosto duplicatelo, cambiategli nome (ad es..: **Prova.001.c, Prova-.002.c...**) e modificate la copia. Vi può sempre essere utile tenervi degli scheletri di codice già fatti per i vari tipi di finestre. Imparerete presto, se già non lo avete capito, che più si scrivono programmi e... meno codice si scrive. Semplice, no?

Quotha32

software & hardware

SOFTWARE

Originale, sigillato con garanzia ufficiale e possibilità di aggiornamento

SPREADSHEETS/INTEGRATI

Microsoft Excel 2.0 (It.)	**750.000**
Microsoft Works (It.)	295.000
Lotus 1-2-3 Rel. 2.01 (It.)	650.000
Lotus Symphony Rel. 2.0 (It.)	850.000
Borland Quattro	**350.000**

Microsoft Excel 2.0 italiano
+
Microsoft Mouse 7

850.000

WORD PROCESSING

Microsoft Word 4.0 (It.)	**750.000**
Lotus Manuscript (It.)	650.000
MicroPro WordStar Professional 5.0	**595.000**
MicroPro WordStar Professional 4.0 (It.)	595.000
MicroPro WordStar 2000 Rel. 3.0 (It.)	890.000
WordPerfect	990.000
Borland Sprint	**350.000**

DATABASE MANAGEMENT

Ashton-Tate dBASE III Plus (It.)	890.000
Borland Paradox 2.0 Single (It.)	1.090.000

NOVITA' ASHTON-TATE

FRAMEWORK III (It.)	**990.000**
dBASE IV	**1.090.000**
dBASE IV Developer's Edition	**1.890.000**

GRAFICA

Microsoft Chart 3.0	550.000
Lotus Freelance Plus (It.)	650.000
Micrografx Designer per Windows	**1.850.000**
CADKey One (CAD tridimensionale)	**850.000**

Quotha 32
PUNTO DI RIFERIMENTO PER IL SOFTWARE PACCHETTIZZATO MANTIENE A MAGAZZINO LE PIU' RECENTI RELEASE

DESKTOP PUBLISHING

Aldus PageMaker 3.0 (It.)	**1.390.000**
Rank-Xerox Ventura Publisher 1.2 (It.)	1.390.000
Rank-Xerox Ventura Publisher 2.0	**1.490.000**
Microsoft Pageview (It.)	70.000

LINGUAGGI

Microsoft QuickBASIC 4.5	**150.000**
Microsoft QuickC	160.000
Borland Turbo Pascal 5.0 (It.)	**250.000**
Borland Turbo C 2.0 (It.)	250.000
Borland Turbo Prolog 2.0	195.000
Microsoft Macro Assembler 5.1	250.000
Microsoft BASIC Compiler 6.0	390.000
Microsoft Pascal Compiler 4.0	390.000
Microsoft FORTRAN Compiler 4.1	595.000
Microsoft C Compiler 5.1	595.000
Microsoft COBOL 3.0	1.150.000
Microsoft OS/2 Programmer's Toolkit	495.000

UTILITIES

Microsoft Windows 286 (It.)	195.000
Microsoft Windows 2 Toolkit	650.000
Fastback Plus	295.000

Microsoft Windows 386 (It.)
Un vero ambiente multitasking che rompe la barriera RAM dei 640 KB utilizzando applicativi DOS. Interfaccia compatibile OS/2 ed emulazione standard LIM

295.000

HARDWARE

Originale, imballato con garanzia TOTALE di 1 anno

Personal Computer IBM	Telefonare
Personal Computer Olivetti	Telefonare

Portatile ZENITH SupersPORT/20
640 KB RAM, 1 FDU 3.5",1 HDU 20 MB, schermo LCD retroilluminato
L. 3.750.000

CONDIZIONI AGEVOLATE PER ENTI PUBBLICI, SCUOLE UNIVERSITA', C.N.R.

STAMPANTI

Panasonic KX-P1081	450.000
Altre stampanti Panasonic a magazzino	Telefonare

Stampanti NEC 24 aghi P2200, P6 Plus, P7 Plus Laser LC-866 Plus e LC-890 PostScript
Prezzi fantastici e consegna immediata

Stampanti OKI	**Telefonare**

MONITOR

NEC MultiSync GS	499.000
NEC MultiSync II	**1.190.000**
NEC Monograph DTP Full Page	2.790.000
Rank-Xerox Full Page Display	1.890.000

HARD DISK

Hardcard PLUS 20 MB	1.250.000
Hardcard PLUS 40 MB	1.550.000
Passport PLUS 20 MB	950.000

SCHEDE GRAFICHE, UPGRADE ED ESPANSIONE

Video Seven VEGA VGA	690.000
Intel Inboard 386/AT 0 KB RAM	1.990.000

Intel
produttore dei processori 8088, 8086, 80286 e 80386
trasforma il tuo PC in un potente 386
a 16 MHzcon 1 MB di RAM installata con
Inboard 386/PC

L. 1.790.000

SPEDIZIONI GRATUITE IN 24 ORE INTUTTA ITALIA VIA CORRIERE

Scheda Ram Intel	Telefonare
Above Board	
Altre schede	Telefonare

Scheda ORCHID DESIGNER VGA
standard VGA con 3 modalità addizionali:
640 X 480 a 256 colori
800 X 600 a 16 colori
1024 X 768 a 16 colori
Supporta gli standards
VGA, EGA, CGA, MDA,MCGA ed Hercules

680.000

CHIP-MOUSE-VARIE

Cop. Mat. Originale	650.000
Intel 80287-10 10 MHz	
Altri Cop. Mat.	Telefonare
Microsoft Mouse 7	**250.000**
con Paintbrush o EasyCAD	
Microsoft Mouse per PS/2	**250.000**

Tutti i prezzi sono al netto di I.V.A.

TERMINI E CONDIZIONI DI VENDITA: Tutti i prezzi sono al netto di I.V.A. - Pagamento in contrassegno con assegno circolare NT intestato a Quotha 32 s.r.l. o contante. - Sconto del 3% per pagamento anticipato. - Ci riserviamo di accettare ordini di importo inferiore a 500.000 lire. - La merce si intende salvo il venduto. - Ulteriori sconti per quantità. - La presente offerta è valida sino al 15 marzo 1989 ed annulla e sostituisce ogni nostra precedente offerta.

per ordini o informazioni telefonare allo

055 - 23.20.240

oppure spedire il tagliando compilato a:

Quotha32 s.r.l.
Via Accursio, 2 - 50125 FIRENZE
Telefax 055 - 20.80.674

Ragione Sociale: _____

Nominativo: _____ Qualifica: _____

Indirizzo: _____

C.A.P.: _____ Città: _____ Prov: _____

Tel. _____ Telefax: _____

☐ Desidero essere contattato da un vostro funzionario commerciale

☐ Desidero ricevere informazioni su: _____

☐ Inseritemi nella vostra mailing list

ZQ