coordinamento di Andrea de Prisco

Programmare in C su Amiga

di Dario de Judicibus

settima puntata

Se aprire una finestra ci permette di creare uno spazio di lavoro da cui ricevere dati o comandi, e su cui riportare grafica e testi, aprire un nuovo schermo ci dà la possibilità di definire ambienti molto differenti fra di loro, vuoi per la risoluzione grafica, vuoi per il numero di colori.
Vedremo in questa puntata come utilizzare tale possibilità

Abbiamo visto nella 6ª puntata come aprire una finestra sullo schermo del WorkBench. Intuition, tuttavia, ci dà la possibilità di definire diversi modi grafici e di scegliere, entro limiti den definiti, la tavolozza *[palette]* dei colori che desideriamo. Ogni modo grafico ha i suoi vantaggi e svantaggi. Una breve analisi ci aiuterà a utilizzare quello che più si adatta al programma che stiamo scrivendo.

Introduzione

In questa puntata impareremo a:
- conoscere i diversi modi grafici dell'Amiga,
- definire le caratteristiche di uno schermo,
- aprire e chiudere uno schermo.

I modi grafici

Ci sono otto modi grafici nell'Amiga. Alcuni possono essere combinati fra loro, altri sono mutualmente esclusivi.

Bassa risoluzione: è il primo dei due modi grafici orizzontali. Generalmente indica che si hanno 320 pixel nel verso orizzontale, fino a 352 in *overscan*. Si possono utilizzare al massimo 32 colori da una tavolozza di 4096 (cioè 5 piani).

Alta risoluzione: è il secondo dei due modi grafici orizzontali ed è mutualmente esclusivo con la bassa risoluzione. Generalmente indica che si hanno 640 pixel nel verso orizzontale, fino a 704 in *overscan*. Si possono utilizzare al massimo 16 colori da una tavolozza di 4096 (cioè 4 piani).

Modo non interlacciato: è questo il modo verticale base, che permette di avere tipicamente 200 righe nello standard NTSC (Nord e Sud America) e 256 in quello PAL (Europa ed Oceania).

Modo interlacciato: questo modo *[interlace]* permette di raddoppiare la risoluzione verticale dello schermo utilizzando una particolare tecnica di alternanza dell'immagine. Su di un normale monitor a bassa persistenza, quale è ad esempio il Modello 1081 della Commodore, questo crea un fastidioso effetto denominato sfarfallio *[flickering]*. Tale effetto può essere notevolmente ridotto se si usa una opportuna scelta dei colori e del contrasto. Il modo interlacciato è stato ideato principalmente per quei programmi che necessitano di una risoluzione elevata, come ad esempio i CAD o certi prodotti per la titolazione del Video Clip, grazie alle 400 (NTSC) o 512 (PAL) linee che arriva a supportare. Chi tuttavia usa questo genere di prodotti, utilizza generalmente un monitor ad alta persistenza *[high-phosphor-persistense (HPP)]* che elimina il problema dello sfarfallio. Tale monitor, tuttavia, non è adatto ai programmi di animazione (per i quali Amiga è giustamente famosa). Questi infatti richiedono uno schermo a bassa persistenza, per evitare un altro effetto alquanto fastidioso per questo tipo di applicazioni, quello appunto della *persistenza dell'immagine*.

La tecnica di interlacciamento può essere utilizzata sia con l'alta che con la bassa risoluzione in orizzontale, dando così vita a quattro possibili combinazioni che, per i monitor PAL sono:
320×256, 320×512, 640×256, 640×512.

Modo sprite: uno sprite (ovverosia «spiritello», «folletto») è un piccolo oggetto che può essere spostato sullo

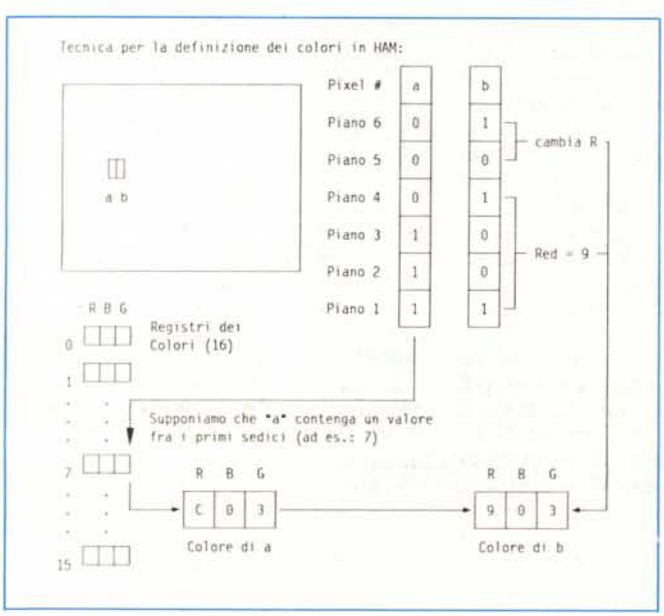

Figura 1
Hold-and-Modify.

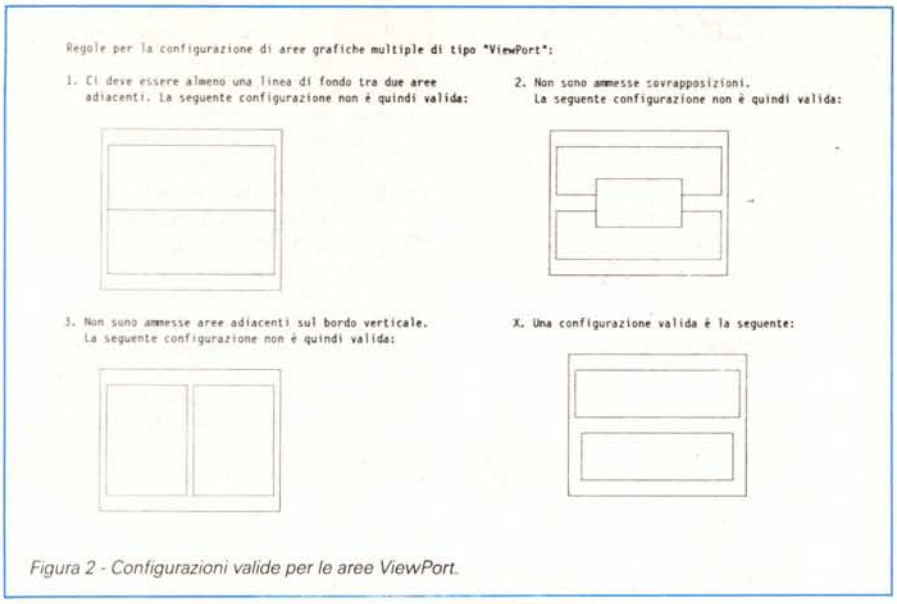

Figura 2 - Configurazioni valide per le aree ViewPort.

«b»: e supponiamo che siano rispettivamente 1 e 0, mentre quelli da 4 ad 1 rappresentino il numero nove (1001).

A questo punto, per definire il colore di «b», basta mantenere (*Hold* appunto) i valori corrispondenti al Blu ed al Verde del registro 7 e modificare (*Modify*)

schermo, come vedremo in una delle prossime puntate. In questo modo è possibile avere fino ad otto oggetti larghi fino a sedici pixel ed alti un qualunque numero di linee. Ognuno di essi può avere un massimo di tre colori (più quello *trasparente*), ma è possibile combinare coppie di sprite per creare un oggetto con quindici colori più il trasparente. Ancora, è possibile utilizzare una tecnica, detta di moltiplicazione *[multiplexing]* che permette di riusare lo stesso oggetto per rendere più sprite nel verso verticale. Questo modo può essere combinato sia con la bassa che con l'alta risoluzione.

Figura 3
Struttura NewScreen.

```
struct NewScreen
{
    SHORT LeftEdge, TopEdge,     /* Angolo superiore sinistro dello schermo */
          Width,    Height,      /* Larghezza ed altezza dello schermo      */
          Depth;                 /* Profondità dello schermo (# piani)      */

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

                                 /* Per i seguenti campi, vedi il testo...  */
    USHORT ViewModes;            /* ...il tipo di modo per lo schermo       */
    USHORT Type;                 /* ...il tipo di schermo (WB o utente)     */
    struct TextAttr *Font;       /* ...il "font" usato (schermo e finestre) */

    UBYTE *DefaultTitle;         /* il titolo iniziale dello schermo        */

    struct Gadget *Gadgets;      /* User Gadgets: vedi il testo             */

    struct BitMap *CustomBitMap; /* BitMap: vedi il testo dell'articolo     */
};
```

Modo doppio-campo: questo modo *[dual-playfield]* permette di sovrapporre due campi grafici indipendenti in modo che il colore 0 del campo in primo piano sia trasparente, e quindi permetta di vedere il campo sullo sfondo. Le due aree grafiche possono essere spostate indipendentemente, dando così la possibilità al programmatore di creare alcuni effetti particolarmente utili per certi giochi, quale, ad esempio, l'impressione di essere nella cabina di un aereo ed osservare fuori dai finestrini il terreno scorrere sotto di noi.

Nel modo «doppio-campo», ogni area grafica può avere al massimo otto colori, mentre il numero totale di colori può assumere solo i seguenti valori:

(Pf1+Pf2):
2+2→4
4+2→6
4+4→8
8+4→12
8+8→16.

Modo HAM: questo modo *[Hold-and-Modify]* permette di avere fino a ben 4096 colori allo stesso tempo, a certe condizioni. Innanzi tutto bisogna essere in bassa risoluzione, non è cioè possibile avere 640 pixel in modo HAM. I colori vengono quindi definiti, per ogni pixel, basandosi sulle seguenti regole:

1) i primi sedici valori definiti dai primi 4 piani, sono resi come in bassa risoluzione, e cioè corrispondono a sedici colori ben definiti;

2) i successivi valori vengono resi in colori differenti a seconda del colore del pixel immediatamente alla sinistra di quello in esame.

Il criterio che permette di definire un colore sulla base del colore del pixel a sinistra è il seguente (vedi figura 1).

Il colore di un pixel, nella rappresentazione RBG, è identificato da una terna che definisce l'intensità per il Rosso (Red), il Blu (Blue), ed il Verde (Green). Supponiamo che un certo pixel, diciamo «a», sia rappresentato nei vari piani, da un valore inferiore a sedici, per esempio sette (7). In tal caso viene utilizzato il valore contenuto nel registro dei colori numero sette che riporta la rappresentazione RBG del colore da visualizzare, mettiamo 0×C03. Supponiamo ora di dover definire il colore del pixel alla destra di «a», cioè «b». In tal caso, andiamo a vedere i bit 6 e 5 associati a quello corrispondente al Rosso con il valore memorizzato nei bit da 4 ad 1 per il pixel «b». Il valore risultante è 0×903. Analogamente nel caso che i bit 6 e 5 di «b» fossero stati differenti, come definito nella seguente tabella:

Bit 6	Bit 5	Modifica	Mantieni	
0	0	—	—	→ Usa il registro di colore
0	1	B	RG	
1	0	R	BG	
1	1	G	RB	

Modo a mezza luminosità: questo modo *[HalfBrite]* non era supportato sui primi modelli dell'Amiga 1000. Esso permette di avere fino a 64 colori in bassa risoluzione. Per far ciò viene usato il sesto bit (i primi cinque definiscono i 32 registri utilizzabili in bassa risoluzione). Quando questo bit è a zero, viene usato il colore contenuto nel registro corrispondente (ad es.: 0×C26), altrimenti si prende tale colore e ne si dimezza l'intensità luminosa (nel nostro caso 0×613). Quindi, in *HalfBrite* i 64 colori non sono indipendenti fra loro, bensì i secondi 32 non sono altri che i primi 32 a cui è stata dimezzata la luminosità.

Schermi utente

Come abbiamo già accennato nella puntata precedente, uno schermo *[screen]* è sostanzialmente uno sfondo su cui possono essere aperte delle finestre, non è fisso ma può scorrere verticalmente ed essere mosso sopra o sotto un altro schermo, ed ha caratteristiche ben definite (colori, risoluzione) che vengono ereditate da tutte le finestre associate.

Anche per gli schermi, come già per le finestre, vengono utilizzate due strutture:
la struttura **NewScreen**, che serve a definire le caratteristiche dello schermo che si vuole aprire, è fornita dall'utente e può essere cancellata una volta utilizzata (tale struttura è riportata in figura 3);
e la struttura **Screen** che mantiene tutte le informazioni aggiornate relative allo schermo una volta apertolo, è gestita da Intuition e viene cancellata da questi solo quando lo schermo viene chiuso (tale struttura è riportata in figura 5).

Analizziamo in dettaglio i campi di cui è composta **NewScreen**.

La struttura NewScreen

I primi quattro campi di **NewScreen** rappresentano le dimensioni geometriche dello schermo, e più precisamente:
LeftEdge l'ascissa dell'angolo superiore sinistro della finestra nel sistema di riferimento dello schermo, partendo da sinistra verso destra; tale campo va sempre impostato a zero (0), ed esiste solo per usi futuri;
TopEdge l'ordinata dell'angolo superiore sinistro della finestra nel sistema di riferimento dello schermo, partendo dall'alto verso 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**.

Il quinto campo (**Depth**) rappresenta invece il numero di piani usati per questo schermo (vedi nota 1 della 6ª punta-

```
/*************************************************************************\
 *                                                                       *
 *    APRISC - Esempio di finestra aperta in uno schermo utente.         *
 *                                                                       *
\*************************************************************************/

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

/*
 *  Prototipi delle funzioni usate (richiesti da molti compilatori)
 */
#include "proto/exec.h"
#include "proto/intuition.h"
extern VOID CloseAll(int);

/*
 *  Qualche definizione...
 */
#define FINE  1000000
#define CARATTERISTICHE (ACTIVATE|NOCAREREFRESH|SMART_REFRESH)
#define INTUINAME "intuition.library"
#define VERSION 0L

/*  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 NewScreen DefSchermo =     /* Inizializza già una parte dei campi  */
{
  0,0,640,256/*PAL*/,2,           /* Angolo sup. sin., dimensioni e piani */
  0,1,                            /* Colori per il fondo ed i particolari */
  HIRES,                          /* Modo grafico (come il WorkBench)     */
  CUSTOMSCREEN,                   /* Tipo dello schermo: utente           */
  NULL,                           /* Font da usare: default               */
  "Schermo non WorkBench",        /* titolo della schermo                 */
  NULL,                           /* Gadgets dello schermo -- non usato   */
  NULL                            /* nessuna BitMap utente                */
};

struct NewWindow DefFinestra =    /* Inizializza gia 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 NON in WorkBench",     /* titolo della finestra                */
  NULL, /* lo assegnamo dopo */   /* Puntatore allo schermo non WorkBench */
  NULL,                           /* Puntatore a BitMap (se SUPERBITMAP)  */
  0, 0,                           /* Dimensioni minime                    */
  0, 0,                           /* Dimensioni massime                   */
  CUSTOMSCREEN                    /* tipo di schermo da utilizzare        */
};

/*
 *  "Schermo" è il puntatore alla struttura restituita da Intuition e che
 *  contiene varie informazioni relative allo schermo creato.
 *  Vedi (5).
 */
struct Screen *Schermo;

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

/*
 *  Solita procedurina per una chiusura pulita!
 */
WORD mask = 0x0000;
#define INTUITION 0x0001
#define SCREEN    0x0002
#define WINDOW    0x0004

VOID CloseAll(rc)
int rc;
{
  if (mask & WINDOW) CloseWindow(Finestra);
  if (mask & SCREEN) CloseScreen(Schermo);
  if (mask & INTUITION) CloseLibrary(IntutionBase);
  Exit(rc);
}


/*  ---------------------------------- *
 *  E qui inzia il programma principale *
 *  ---------------------------------- */
main()
{
  LONG conta;                         /* contatore */

  /*
   *  Apri la libreria che contiene le routine di Intuition
   */
  IntuitionBase = (struct IntuitionBase *)OpenLibrary(INTUINAME,VERSION);
  if (IntuitionBase == NULL) CloseAll(RETURN_ERROR);
  mask |= INTUITION;

  /*
   *  OK, proviamo ora ad aprire prima lo schermo, e poi la finestra
   */
  Schermo = (struct Screen *)OpenWindow(&DefSchermo);
  if (Schermo == NULL) CloseAll(RETURN_ERROR);
  mask |= SCREEN;
                              /* Non dimentichiamoci di dire ad Intuition */
  DefFinestra.Screen = Schermo; /* a quale schermo appartiene la finestra!  */

  Finestra = (struct Window *)OpenWindow(&DefFinestra);
  if (Finestra == NULL) CloseAll(RETURN_ERROR);
  mask |= WINDOW;

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

  /*
   *  Addesso "chiudiamo casa", prima di uscire!
   */
  CloseAll(RETURN_OK);
}
```

Figura 4 - Esempio di finestra aperta in uno schermo utente.

```
struct Screen
{
    /* --- I seguenti campi sono rispettivamente:                    --- */
    /* ---                                                           --- */
    /* --- #1  il puntatore allo schermo successivo nella lista degli --- */
    /* ---     schermi aperti da Intuition                           --- */
    /* --- #2  il puntatore alla prima finestra della lista delle    --- */
    /* ---     finestre associate a questo schermo                   --- */
     struct Screen *NextScreen;  /* #1 puntatore allo schermo successivo    */
     struct Window *FirstWindow; /* #2 puntatore alla prima finestra        */

     SHORT LeftEdge, TopEdge; /* Angolo superiore sinistro dello schermo     */
     SHORT Width, Height;     /* Larghezza ed altezza dello schermo          */

     SHORT MouseY, MouseX;    /* Coord. del mouse (rel. all'angolo sup. sin. */

     USHORT Flags;            /* Vedi sotto: TIPI DI SCHERMI                 */

    /* --- I seguenti campi sono rispettivamente:                    --- */
    /* ---                                                           --- */
    /* --- #1  il titolo dello schermo quando tutte le finestre sono --- */
    /* ---     disattivate                                           --- */
    /* --- #2  il titolo dello schermo, quando è attiva una finestra a --- */
    /* ---     cui NON è associato uno "ScreenTitle" (vedi 6ª puntata) --- */
     UBYTE *Title;            /* #1 Titolo associato allo schermo            */
     UBYTE *DefaultTitle;     /* #2 Titolo associato ad una finestra         */

    /* Dimensioni delle varie "barre" usate nei bordi dello schermo e delle */
    /* finestre associate.                                                  */
     BYTE BarHeight, BarVBorder, BarHBorder, MenuVBorder, MenuHBorder;
     BYTE WBorTop, WBorLeft, WBorRight, WBorBottom;

     struct TextAttr *Font;   /* il "font" usato per questo schermo          */

    /* --- I seguenti quattro campi rappresentano le strutture grafiche  --- */
    /* --- su cui è basato lo schermo. Di interesse solo per i program-  --- */
    /* --- matori avanzati. ATTENZIONE: queste solo proprio le struttu-  --- */
    /* --- re, non i puntatori ad esse.                                  --- */
    /* ---                                                           --- */
     struct ViewPort ViewPort;    /* L'area da usare per questo schermo      */
     struct RastPort RastPort;    /* Le caratteristiche grafiche associate   */
     struct BitMap BitMap;        /* Duplicato della struttura BitMap usata  */
     struct Layer_Info LayerInfo; /* Informazioni sui piani usati            */

     struct Gadget *FirstGadget;  /* Vedi lo stesso campo in "NewScreen"     */

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

     USHORT SaveColor0;           /* Colore di fondo da ripristinare dopo un */
                                  /* "flash" (tecnica usata nel BEEP)        */
     struct Layer *BarLayer;      /* Piano che contiene la barra dei menu e  */
                                  /* quella dello schermo                    */
     UBYTE *ExtData;              /* Per estensioni future....               */
     UBYTE *UserData;             /* Eventuali dati dell'utente              */
};

/* ------------------------- TIPI DI SCHERMI ---------------------------- */
/*                                                                        */
/*  I seguenti valori sono impostati da Intuition per "ricordare" alcune  */
/*  caratteristiche dello schermo. Come si può vedere, tre di essi corri- */
/*  spondono ai valori utilizzati dall'utente per il campo "Type" nella   */
/*  struttura "NewScreen": WBENCHSCREEN, CUSTOMSCREEN e CUSTOMBITMAP.     */
/*                                                                        */
/* ---------------------------------------------------------------------- */

 #define WBENCHSCREEN    0x0001  /* Uno schermo WorkBench (in genere ce n'e */
                                 /* uno solo aperto alla volta)             */
 #define CUSTOMSCREEN    0x000F  /* Uno schermo utente                      */

 #define SHOWTITLE       0x0010  /* Viene impostato quando si chiama        */
                                 /* ShowTitle()                             */
 #define BEEPING         0x0020  /*   Viene impostato quando si fa          */
                                 /*   lampeggiare lo schermo (BEEP)         */
 #define CUSTOMBITMAP    0x0040  /* Se l'utente fornisce una propria        */
                                 /* struttura BitMap                        */
 #define SCREENBEHIND    0x0080  /*   Se lo schermo va aperto dietro a      */
                                 /*   tutti gli altri                       */
 #define SCREENQUIET     0x0100  /* Se non vuoi che Intuition disegni il    */
                                 /* titolo ed i gadget di sistema           */

/* ---------------------------------------------------------------------- */
/*  Gestione automatica delle dimensioni massime dello schermo (PAL/NTSC) */
/* ---------------------------------------------------------------------- */
 #define STDSCREENHEIGHT -1      /* [1.2] Vedi il testo dell'articolo       */
```

Figura 5 - Struttura Screen.

ta), e quindi il numero di colori disponibili.

Questi cinque campi non sono del tutto indipendenti fra di loro ed hanno dei limiti ben definiti. Spendiamo due parole al riguardo.

Innanzi tutto, come abbiamo già detto, l'angolo in alto a sinistra di uno schermo deve essere allineato al bordo sinistro dello schermo fisico del monitor, mentre la larghezza deve essere tale da estendersi fino al bordo opposto dello stesso (quindi 320 in bassa risoluzione e 640 in media risoluzione). In realtà è possibile specificare anche larghezze inferiori a quelle menzionate, ma, dato che comunque un secondo schermo non può iniziare là dove il bordo destro del primo termina (**LeftEdge** non può essere maggiore di zero), non se ne ricava alcun vantaggio. Di fatto l'unico modo per accorgersene, è quello di aprire una finestra dotata di barra per gli spostamenti *[drag bar]* e, spostandola qua e là, utilizzarla come rivelatore dei bordi dello schermo (come richiesto nell'esercizio). Esiste inoltre la possibilità di specificare larghezze superiori a quelle *classiche* già menzionate, sia in bassa che in alta risoluzione. Tale tecnica si chiama *overscan* ed è regolarmente usata da quei prodotti grafici che servono a assemblare o modificare sequenze video per l'Amiga. In questo caso, la larghezza di uno schermo può arrivare finoa 352 pixel in bassa risoluzione e 704 in alta risoluzione.

Altre limitazioni invece, sono le seguenti: non è possibile avere schermi che si sovrappongono a cavallo di altri schermi, oppure adiacenti l'uno all'altro lungo un bordo verticale. Queste ultime due restrizioni sono dovute al fatto che gli schermi di Intuition sono basati su di una struttura di più basso livello chiamata **ViewPort**. L'attuale sistema pone dei limiti ben precisi a come queste aree grafiche possono essere configurate, e quindi alle possibili configurazioni degli schermi di Intuition (vedi figura 2).

Esiste inoltre un'altra restrizione, dovuta questa volta ad Intuition (eliminata nella versione 1.2), che richiede che il bordo inferiore di uno schermo non si trovi al di sopra del bordo inferiore dello schermo fisico del monitor, cioè che **TopEdge + Height > altezza schermo fisico.**

I successivi due campi di **NewScreen** definiscono i colori che devono essere usati per disegnare lo schermo, analogamente a quanto già visto per le finestre:

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

BlockPen il colore da usare per riempire lo sfondo degli elementi che costituiscono la barra superiore dello schermo.

Il campo **ViewModes** definisce invece i vari modi grafici disponibili:

NULL	Bassa risoluzione (320/352 pixel)
HIRES	Alta risoluzione (640/704 pixel)
INTERLACE	Modo interlacciato (400/512 linee)
SPRITES	Modo sprite
DUALPF	Doppio campo
HAM	Hold-and-Modify
HALFBRITE	Mezza luminosità

Il campo **Type** definisce il tipo di schermo:
CUSTOMSCREEN Schermo Utente
CUSTOMBITMAP Usa la BitMap fornita dal programmatore.

Il campo seguente (**Font**) è il puntatore alla struttura **TextAttr** che descrive il tipo di testo che Intuition deve utilizzare per questo schermo e tutte le sue finestre. Ovviamente questo non influisce sui testi direttamente gestiti dal programmatore tramite le funzioni di Intuition. Testi gestiti direttamente da Intuition sono ad esempio quelli nei titoli dello schermo e delle finestre o nei *Requestor* di sistema. Per usare il valore di *default*, basta porre questo campo a zero (**NULL**).

A questo punto c'è il puntatore ad una stringa che contiene il titolo da mettere nella barra superiore dello schermo (**DefaultTitle**), ed il campo **Gadgets** che per il momento non è supportato e che era stato pensato co-

me puntatore ad una eventuale lista di gadget utente per lo schermo in questione, come abbiamo già visto nella struttura **NewWindow**. Va pertanto impostato a **NULL**.

L'ultimo campo (**CustomBitMap**) è il puntatore ad una eventuale **BitMap** fornita dal programmatore, come spiegheremo in una delle prossime puntate. Se questo campo non è nullo, il segnalatore **CUSTOMBITMAP** va impostato nel campo **Types**.

Il programma di esempio

Vediamo ora come va modificato il programmino dimostrativo che abbiamo presentato nella scorsa puntata (figura 4).

Innanzi tutto aggiungiamo alle linee # **include** del programma precedente tre nuove linee. Le prime due vanno aggiunte così come sono solo dagli utenti del *Lattice C 4.0*. Chi possiede una versione precedente del Lattice C oppure un altro compilatore, faccia riferimento al manuale che accompagna il prodotto (vedi nota 1). La terza linea è semplicemente il prototipo della funzione **CloseA11 ()** usata per chiudere gli oggetti aperti nel corso del programma, in caso di errore od al termine dello stesso.

Nel blocco seguente, abbiamo aggiunto una nuova costante (**VERSION**) che corrisponde alla versione richiesta per la libreria da aprire. Ricordo che nel caso di Intuition 1.1 tale valore è 32, per Intuition 1.2 è 33, mentre il valore zero è usato se non esiste motivo di richiedere una versione piuttosto che un'altra. Vedremo più avanti come può essere usata.

Finalmente ecco la struttura **NewScreen** utilizzata per specificare ad Intuition le caratteristiche dello schermo da aprire. Come si può vedere, abbiamo scelto dei valori tali da simulare uno schermo di tipo WorkBench:

- alta risoluzione non interlacciata;
- due piani (quattro colori);
- dimensioni, font e colori di default.

L'unica differenza è il titolo, per distinguerlo dal vero schermo WorkBench (sul quale peraltro sono presenti anche le icone dei dischetti).

Per quello che riguarda la struttura **NewWindow**, abbiamo modificato solo il titolo ed il tipo di schermo utilizzato (CUSTOMSCREEN al posto di WBENCHSCREEN), ovviamente. Notate che il puntatore allo schermo utente è ancora impostato a zero (NULL), dato che qui non va assolutamente quello alla struttura **NewScreen** (errore molto comune), bensì quello alla struttura **Screen** che ci verrà restituito in modo dinamico solo all'apertura dello schermo.

E di fatti, più sotto, ecco la linea che definisce il nuovo puntatore, per ora «vuoto».

Abbiamo colto l'occasione per aggiungere anche l'ormai nota tecnica per chiudere in modo pulito ciò che si è aperto: **mask**, alcune costanti e la funzione **CloseA11()**.

A questo punto entriamo nel programma principale, cioè **main()**.

Come si può vedere in figura 4, dopo l'apertura della libreria di Intuition è stato aggiunto un blocco che prova ad aprire il nuovo schermo, passando il puntatore a **DefSchermo** ad Intuition, verifica il puntatore di ritorno e, se non nullo, lo assegna all'apposito campo in **DefFinestra**, prima di provare ad aprire anche la finestra.

Il tutto termina più o meno come nella scorsa puntata: ciclo a vuoto e chiusura finale.

Conclusione

Come avrete certamente notato, la struttura **NewScreen** nell'esempio riportato definisce uno schermo alto 256 linee, in accordo con lo standard PAL. Ovviamente tale programma male si adatterebbe a girare su di un Amiga americano, che segue invece lo standard NTSC da 200 linee. Come scrivere allora programmi in grado di girare allo steso modo in entrambi gli standard? Certo limitarsi al valore più basso non è la soluzione più elegante. Ebbene, Intuition 1.2 ci mette a disposizione più di un modo per facilitarci nello scrivere questo tipo di programmi. Lo vedremo nella prossima puntata, nella quale parleremo anche di alcuni di quei campi delle strutture **NewScreen** e **NewWindow** un po' più impegnativi, ed inizieremo a fare conoscenza con alcune routine della libreria grafica. Per ora vi basti sapere che, se si assegna il valore **STDSCREENHEIGHT** (vedi figura 5) al campo **NewScreen.Height**, Intuition adatterà automaticamente lo schermo alle dimensioni verticali massime, sia che il programma giri sotto standard PAL, sia che si usi l'NTSC. Naturalmente questo non basta per scrivere programmi portabili al 100%, ma è un ottimo inizio (attenzione: leggi la nota 2). Per sicurezza inoltre, è bene modificare la costante **VERSION** utilizzata nel programmino di esempio come segue:

```
# define VERSION 33L
```

in modo da essere sicuri di aprire la versione 1.2 di Intuition, dato che la 1.1 non supporta le funzioni suddette.

Per la prossima volta abbiamo pensato ad un esercizio semplice semplice, ma a suo modo istruttivo. Provate ad aprire uno schermo utente di piccole dimensioni (diciamo 100×100), ricordando che comunque esso va allineato al bordo sinistro dello schermo fisico. Aprite quindi una piccola finestra (60×40) dotata di gadget per la chiusura (utilizzate le costanti CLOSEWINDOW e WINDOWCLOSE) e di barra per gli spostamenti (WINDOWDRAG). A questo punto mettetevi in attesa sulla porta IDCMP associata alla finestra, utilizzando la funzione:

```
Wait(1<<w—>UserPort—>mp_SigBit);
```

di modo che, quando selezionate il gadget di chiusura della finestra, il programma termini. Compilate e lanciate il tutto. Usate la barra di spostamento della finestra per muoverla qua e là per lo schermo. Vedrete che, anche se i limiti dello schermo non sono visibili, non riuscirete a muovere la finestra più in là di un piccolo rettangolo in alto a sinistra. Avrete così costrito una finestra sonda! Buon lavoro!

Note

1. Il *Lattice C 4.0* è oramai disponibile anche in Italia. Inoltre, proprio in questi giorni la Lattice Inc. ha annunciato una nuova versione del *Lattice C*. Il *Lattice C 5.0* includerà tra l'altro, un ottimizzatore globale del codice ed un *source debugger*. Quest'ultimo è un programma che permette di analizzare il codice mentre gira direttamente sul file sorgente, aumentando di molto la capacità di analisi degli errori. Un analogo prodotto era già presente nella versione 3.6 del *Manx Atzec C*.

Questi due sono di fatto i due più importanti compilatori C per Amiga. Per avere informazioni a riguardo, riporto qui di seguito gli indirizzi delle corrispondenti società produttrici:

Lattice, Incorporated
2500 S. Highland Avenue
Lombard IL 60148 (USA)

Manx Software Systems
One Industrial Way West
Eatontown NY 07724 (USA)

2. Prima di lanciare il programma, vi consigliamo di verificare che i vostri *include file* siano aggiornati per il sistema operativo 1.2. Altrimenti il compilatore non sarà in grado di trovare la definizione di costanti quali MODE_READWRITE (AmigaDOS) o STDSCREENHEIGHT (Intuition). Naturalmente dovrete anche girare sotto 1.2, ma oramai non dovrebbero esserci problemi, no? Tra parentesi, ieri (21 ottobre) mi è arrivata la notizia che la versione 1.3 del sistema operativo su dischetto è già in vendita negli Stati Uniti al modico prezzo di circa $25. Si tratta della versione ufficiale, ovviamente, non di una versione *gamma*. Conto quindi di vederla in Italia quanto prima.

S.C.COMPUTERS s.a.s.

via E.Fermi 4, 40024 Cast.S.Pietro T. (BO)
tel. 051 - 943500 (2 lin.ric.aut. + fax)

Confrontate attentamente queste configurazioni e questi prezzi con altre inserzioni di questa rivista:

PC XT: L. 1.550.000

8088/2, clock a 5 e **8 MHz**, zero wait, 512 Kbytes di RAM, 1 **drive** da 360Kb 1 **Hard Disk** da 20 Mbytes, **Controllers** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Scheda **Hercules** Hi Res, **Tastiera** Avanzata 101/2 tasti, Cabinet tipo ATcon chiave, Aliment. 200 W, **Monitor** 12" TTL Hi-Res, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

E' inoltre disponibile un modello analogo, ma con clock a 10 MHz e microprocessore NEC V20

PC AT: L. 2.150.000

80286, clock a 6 e **12 MHz**, zero wait, 512 Kbytes di RAM, 1 **drive** da 1,2 Mbytes, 1 **Hard Disk** da 20 Mbytes, **Controller** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Porta **Seriale** Doppia RS 232, Scheda **Hercules** Hi Res, **Tastiera** Avanzata 101/2 tasti, Cabinet con chiave, Aliment., **Monitor** 12" TTL Hi-Res, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

PC AT/VGA: L. 3.150.000

80286, clock a 6 e **12 MHz,** zero wait, 512 Kbytes di RAM, 1 **drive** da 1,2 Mbytes, 1 **Hard Disk** da 20 Mbytes, **Controller** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Porta **Seriale** Doppia RS 232, Scheda **VGA, Tastiera** Avanzata 101/2 tasti, Cabinet con chiave, Aliment., **Monitor 14" Philips per VGA**, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

PC 386/40: L. 5.800.000

80386, clock **16 MHz** , 1 Mbytes di RAM, 1 **drive** da 1.2 Mbytes, 1 **Hard Disk** da **40 Mbytes** veloce (29 ms), **Controller** per 2 drives e 2 Hard Disks, Porta **Parallela** Centronics, Porta **Seriale** Doppia RS 232, Scheda **Hercules** Hi Res, **Tastiera** Avanzata 101/2 tasti, Cabinet, Aliment., **Monitor** 12" TTL Hi-Res, Tutti i cavi e manuali, 1 Anno di **Garanzia TOTALE**

TOSHIBA

PROMOZIONALE sul **T 1600:**
prenotateVi SUBITO !!!

COPROCESSORI & RAM

RAM 100-120-150 ns, 64K -256K-1Mb	TELEFONARE
8087	L. 249.000
8087/2	L. 320.000
80287	L. 349.000
80287/8	L. 535.000
80287/10	L. 610.000
80387/20	L. 1.060.000

Condizioni particolari per
☆☆ RIVENDITORI !!! ☆☆

Tutti i prezzi sono da intendersi IVA 19% esclusa, ma comprendono un anno di garanzia TOTALE f.co ns. sede. Siamo in grado di spedire la merce a mezzo corriere in tutta Italia entro 36 ore dal ricevimento di un acconto pari al 10% dell'importo totale tramite Vaglia Telegrafico. Spedizione gratuita, se effettuata a mezzo posta.

OFFERTE del MESE

Nuovissima Stampante **EPSON LQ 500**, 24 AGHI, 80 col., 150 car.per sec., doppia velocita' in Letter Quality rispetto alla LX800, grafica bidire-z., 8 Kb di buffer, foglio singolo e modulo continuo, int. parallela: . . **L. 699.000**

Mouse emulaz. MICROSOFT e Mouse System Mouse, con porta-mouse, tappetino e lo splendido Dr.HALO III originale e mauali . . . . . **L. 120.000**

Hard Disk 20 Mbytes . . . . . **L. 390.000**

TELEFAX TOSHIBA:

OMOLOGATI!!!!!
nuovissimi: TF 111, 211, 311,
a partire da L. 2.500.000

COMMODORE - ATARI

❄❄❄ *OFFERTE NATALIZIE !!!* ❄❄❄
I migliori prezzi, ma con
GARANZIA ITALIANA.
☎☎ TELEFONATECI !!! ☎☎