Proprio in extremis sono arrivati in redazione alcuni programmi interessanti per l'Apple II; quello che ha vinto la pool position è questo «Modulo Base» specificatamente sviluppato per il IIGS.
Il programma si occupa di creare il background necessario al funzionamento di un qualsiasi programma per Apple IIGS con il classico aspetto MAC_like. Peccato che la descrizione dei vari tool chiamati non sia molto didattica sicché il «Modulo» va un po' preso così com'è (o modificato leggermente).
Comunque se l'argomento dovesse interessare più di due o tre lettori (come certamente sarà) spero di pubblicare, quanto prima, qualcosa sui TOOLS GS e sul loro uso in programmazione.
A tal proposito sarà certamente gradita ogni forma di collaborazione da quanti già utilizzano nei loro programmi le chiamate al TOOL_BOX, perché un esempio pratico vale sempre più di dieci lezioni astratte!

Modulo Base

Francesco Meschia - Asti

Sono un fedelissimo della mela iridata da cinque anni, vostro lettore da altrettanto tempo, ora fortunato possessore di un Apple IIGS. Potete ben immaginare in quale modo mi abbia freddato l'apprendere che la mia rubrica preferita stava cadendo in rovina; così ho deciso di scrivervi perché spero di potervi aiutare a salvarla. Vi invio infatti per la pubblicazione qualcosa che nessuna altra rivista ha mai pubblicato: un programma dedicato all'Apple IIGS, da me scritto interamente in APW C, che funziona totalmente in modo nativo. E, almeno nelle mie intenzioni, il programma ha funzione didattica, dal momento che è un «modulo base», cioè un'intelaiatura, funzionante, sulla quale il programmatore può costruire la sua applicazione. Il programma si occupa infatti della manovalanza, cioè di svolgere quei compiti (inizializzazione del Toolbox, setup dei menu e delle finestre, ciclo eventi, routine varie) che sono comuni a tutti i programmi, o che comunque si possono adattare al caso specifico con poche varianti.

Il programma, come ho detto, è scritto in C: deve perciò essere compilato con l'Apple IIGS Programmer's Workshop 1.0 e l'APW C Compiler 1.0, e poi può essere lanciato dal Finder o dal Program Launcher; è commentato, e comprenderlo richiede solo una modesta conoscenza del C e del Toolbox di Apple IIGS. È diviso, come tutti i buoni programmi C, in più file: tre sorgenti e due header.

BASE.C

Funzione main(): richiama le routine per il caricamento e l'inizializzazione del Toolbox, pulisce il buffer degli eventi, disegna il desktop, mostra il cursore e richiama le routine che inizializzano fine-

BASE.C

```
/* base.c : main base per applicazioni C */

#include <TYPES.H>
#include <MENU.H>
#include <WINDOW.H>
#include "BASE.H"

boolean finito=FALSE;                          /* flag di fine programma */
GrafPortPtr window1;

main()
{
   InitTool();                                 /* carica e sveglia il Toolbox */
   FlushEvents(everyEvent,0);                  /* spazza via gli eventi spuri */
   RefreshDesktop(NULL);                       /* disegna il desktop */
   ShowCursor();                               /* mostra il cursore */
   SetUpMenus();                               /* inizializza i menu' */
   SetUpWindows();                             /* e le finestre */

   while (!finito)                             /* finche' il programma va... */
       Eventi();                               /* aspetta e processa eventi */

   Chiudi();                                   /* quindi spegne il toolbox */
}

/* Eventi : attende e processa gli eventi, utilizzando la funzione
            TaskMaster del Window Manager, che si occupa di gestire
            menu', grow, move, scroll, update ed un'infinita' di altre
            cosette.
*/


Eventi()
{
   WmTaskRec ev;
   int evento;

   ev.wmTaskMask=0x1fffL;                      /* taskmaster fa tutto */
   evento=TaskMaster(everyEvent,&ev);          /* pre-processa l'evento */
   switch(evento) {                            /* e poi sceglie: */
   case wInSpecial:                            /* se nei menu' edit... */
   case wInMenuBar:                            /* o in qualunque altro menu' */
       Menu(ev.wmTaskData);                    /* passa l'evento a Menu() */
```

È disponibile, presso la redazione, il disco con i programmi pubblicati in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 258.

stre e menu. Quindi, finché il valore della variabile finito è FALSE, richiama la funzione di event processing ed infine chiama Chiudi() che si occupa dello shutdown del Toolbox.

Funzione Eventi(): chiama TaskMaster() dicendogli di fare tutto quanto è in suo potere, quindi, se TaskMaster() restituisce un codice WinSpecial (significa che è stato selezionato un menu item speciale, p.es. Taglia, Copia o Incolla) o WinMenuBar, chiama Menuche si occuperà di scoprire quale item è stato scelto e di richiamare le routine appropriate.

Il programma, e questa funzione in particolare, non è previsto per trattare casi di chiusura delle finestre, dato che il compito da svolgere può variare di parecchio da programma a programma. È sufficiente comunque aggiungere il case WinGoAway allo switch seguito dalla propria routine di chiusura finestre qualora si volesse trattare questo tipo di evento.

Funzione Draw1(): è di tipo Pascal perché viene chiamata dal Window Manager ogni volta che la finestra 1 ha bisogno di essere ridisegnata. Tutto quello che fa è scrivere «Base 2.0» nella finestra, usando QuickDraw.

Funzione Menu(): trasforma il long integer che proviene dal campo TaskData dell'event record in due short integer, dei quali uno identifica il menu scelto e l'altro l'item selezionato. Quindi passa in rassegna agli item ID che conosce, chiamando le routine di conseguenza, e infine toglie l'inverse al titolo del menu scelto.

BASE.H

```
/* base.h : definizioni base per applicazioni su Apw C 1.0 */

#ifndef __dialog__                      /* se non c'e il dialog manager */
#include <DIALOG.H>                     /* lo #include */
#endif

#define normale 0                       /* tipi di alert */
#define stop 1
#define note 2
#define caution 3
#define cSItem longStatText+itemDisable      /* Dialog item : stringa C */
#define cSItem2 longStatText2+itemDisable    /* idem, ma formattata */

extern GrafPortPtr window1;
extern pascal void Draw1();
extern int CrossCurs[],IBeamCurs[];
```

```
        break;
    default:                                /* se no, non fa niente */
        break;
    }
}

/* Draw1 : procedura di disegno del contenuto della finestra. E' di tipo
           pascal perche' viene chiamata direttamente dal Toolbox in
           risposta ad un update event proveniente dalla finestra 1.
*/

pascal void Draw1()
{
    MoveTo(20,20);                          /* si sposta */
    DrawCString("Base 2.0");                /* e disegna */
}

/* Menu : riconosce il menu' e l'item selezionato, e si comporta di
          conseguenza. Prima di finire toglie l'highlight al titolo.
*/

Menu(mn)
long mn;
{
    int item=mn&0xffff,                     /* trasforma il long in una */
        menu=mn/0xffff;                     /* coppia di interi */
    switch(item) {                          /* controlla l'item: */
    case 1001:
        Info();                             /* se 1001, mostra le */
        break;                              /* informazioni sul prog. */
    case 2999:                              /* se 2999, finisce il */
        finito=TRUE;                        /* programma */
        break;
    default:
        break;
    }
    HiliteMenu(FALSE,menu);                 /* toglie highlight al menu' */
}
```

BASE.H

Questo è il file di header che gli altri moduli # includono per avere conoscenza delle variabili globali e degli identificatori predefiniti. Definisce i tipi di alert, due dialog item per usare più agevolmente le stringhe in formato C, e comunica agli altri moduli l'esistenza delle variabili globali window1, CrossCurs[], |- BeamCurs[] e della funzione Draw1().

CURSORE.C

Questo piccolo file contiene due array di int, che contengono le definizioni di due cursori di uso comune: quello a croce e quello tipico dei word processor, detto «I-Beam». Dato che sono due array globali, è sufficiente fare un SetCursor(nome array) per cambiare la forma del cursore.

SUB.C

Funzione SetUpMenus(): scrive in tre array di caratteri le descrizioni dei menu ed esegue le chiamate al Toolbox necessarie per creare i menu.

Funzione SetUpWindows(): scrive in una struttura ParamList (typedef-inita nell'header file WINDOW.H del compilatore) gli attributi essenziali della finestra e quindi esegue la chiamata New-

Window() per creare la finestra vera e propria.

UTIL.C

Funzione InitTool(): carica in memoria il Toolbox, chiede al Memory Manager la memoria necessaria agli altri tool e quindi li inizializza uno per uno.

Funzione Chiudi(): si limita ad eseguire le chiamate xxShutDown per ciascun tool del Toolbox.

Funzione Info(): fa comparire la finestra di informazioni sul programma. La finestra è creata come una dialog box modale, con del testo esplicativo ed un bottone OK. La funzione aspetta fino a quando non si clicca OK e poi chiude la finestra.

Funzione Attento(): serve per creare una alert box. I suoi primi 7 parametri sono int: prima viene il tipo dell'alert (i valori sono definiti nell'header BASE.H), poi l'indentificatore della alert, quindi quattro int specificanti i quattro stadi dell'alert (vedere il Toolbox Reference Manual, volume 1) e il numero di bottoni che deve contenere l'alert. Se si sceglie una alert con un solo bottone, esso sarà «OK», se i bottoni sono due saranno «OK» e «Annulla» se sono tre «Sì», «No» e «Annulla». L'ottavo e ultimo parametro è un puntatore a carattere, che identifica la stringa di testo esplicativa.

Funzione CheckToolError(): verifica se c'è stato un errore del Toolbox. Se c'è stato realmente, interrompe il programma, mostrando la mela rimbalzante, il numero dell'errore e il punto del programma in cui l'errore si è verificato (quest'ultimo viene passato in input dalla funzione chiamante). La funzione è stata adattata dall'omologa descritta nell'Apple IIGS Programmer's Introduction.

CURSORE.C

```
/* cursore.c : definizioni dei di uso comune */

int CrossCurs[] = {
   9,3,                                    /* dimensioni */
   0x0000,0x00f0,0x0000,                   /* forma del cursore */
   0x0000,0x00f0,0x0000,
   0x0000,0x00f0,0x0000,
   0x0000,0x00f0,0x0000,
   0xffff,0xff0f,0x00f0,
   0x0000,0x00f0,0x0000,
   0x0000,0x00f0,0x0000,
   0x0000,0x00f0,0x0000,
   0x0000,0x00f0,0x0000,
   0,0,0,                                  /* maschera */
   0,0,0,
   0,0,0,
   0,0,0,
   0,0,0,
   0,0,0,
   0,0,0,
   0,0,0,
   0,0,0,
   4,8                                     /* hot spot */
   };

int IBeamCurs[]={
   12,2,                                   /* dimensioni */
   0x0fff,0x00f0,                          /* forma */
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0xf000,0x0000,
   0x0fff,0x00f0,
   0,0,                                    /* maschera */
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   0,0,
   8,8                                     /* hot spot */
   };
```

SUB.C

```
/* sub.c : subroutines di base per applicazioni C */

#include <DESK.H>
#include <WINDOW.H>
#include <MENU.H>
#include "BASE.H"

/* SetUpMenus : inizializza i menu', aggiunge gli accessori di scrivania
                e disegna la barra di menu'.
*/

SetUpMenus()
{
   static char menu1[] =
"++@\\N1X\0\
--Info su Base\\N1001V\0\
+";
   static char menu2[] =
"++  Archivio  \\N2\0\
--Esci\\N2999*Qq\0\
+";
   static char menu3[]=
"++  Composizione  \\N3\0\
--Annulla\\N250D*Zz\0\
---\\N9999D\0\
--Taglia\\N251*XxD\0\
--Copia\\N252*CcD\0\
--Incolla\\N253*VvD\0\
--Cancella\\N254D\0\
+";

   SetMTitleStart(10);                     /* fissa l'inizio barra */
   InsertMenu(NewMenu(menu3),0);           /* crea i menu' */
   InsertMenu(NewMenu(menu2),0);
   InsertMenu(NewMenu(menu1),0);
   FixAppleMenu(1);                        /* aggiunge gli NDA */
   FixMenuBar();                           /* calcola le dimensioni */
   DrawMenuBar();                          /* e disegna la menu bar */
}

/* SetUpWindows : creazione di una nuova finestra, partendo da una struttura
                  dati messa a disposizione dal window manager.
*/
SetUpWindows()
{
   static ParamList Parm1 = {
      sizeof (Parm1),
      fVis+fTitle+fClose+fMove+fZoom+fGrow+fRScroll+fBScroll,
      "\pFinestra 1",                      /* pointer al titolo */
      0L,                                  /* RefCon finestra */
      0,0,0,0,                             /* Rect per zoom (non usato) */
      NULL,                                /* tabella colori */
      0,0,                                 /* origine */
      400,650,                             /* area dati */
      0,0,                                 /* max growth */
      10,10,                               /* arrow scroll */
      0,0,                                 /* page scroll */
      NULL,                                /* Info Bar RefCon */
      0,                                   /* Info Bar Height */
      NULL,                                /* frame DefProc */
      NULL,                                /* Info Bar DefProc */
      Draw1,                               /* Content DefProc */
      40,20,180,598,                       /* rettangolo della finestra */
      (GrafPortPtr)topMost,                /* piano */
      NULL                                 /* memoria da usare */
   };

   window1 = NewWindow(&Parm1);            /* crea la finestra */
}
```

UTIL.C

```
/* util.c : utilities varie, di uso generale. */

#include <LOCATOR.H>
#include <MEMORY.H>
#include <DESK.H>
#include <INTMATH.H>
#include <WINDOW.H>
#include <MENU.H>
#include <QDAUX.H>
#include <LINEEDIT.H>
#include <DIALOG.H>
#include <SCRAP.H>
#include <STDFILE.H>
#include <FONT.H>
#include <PRINT.H>
#include "BASE.H"

/* InitTool : carica il Toolbox e lo inizializza. Cosi' com'e' carica tutti
              i tools su disco: la si puo' facilmente modificare se qualche
              tool non serve.
*/

InitTool()
{
   static int MieiTool[]={
      15,                       /* Ci servono 15 tools: */
       3, 0x100,                /* Miscellaneous Tool */
       4, 0x100,                /* QuickDraw II */
       5, 0x100,                /* Desk Manager */
       6, 0x100,                /* Event Manager */
      14, 0x100,                /* Window Manager */
      15, 0x100,                /* Menu Manager */
      16, 0x100,                /* Control Manager */
      18, 0x100,                /* Auxiliary QuickDraw */
      19, 0x100,                /* Print Manager */
      20, 0x100,                /* LineEdit */
      21, 0x100,                /* Dialog Manager */
      22, 0x100,                /* Scrap Manager */
      23, 0x100,                /* Standard File Operations */
      27, 0x100,                /* Font Manager */
      28, 0x100                 /* List Manager */
   };
   int zp;

   LoadTools(MieiTool);
   zp=(int)(*NewHandle(0xB00L,_ownerid,0xc005,NULL));
   ScrapStartUp();
   QDStartUp(zp,0x80,160,_ownerid);
   QDAuxStartUp();
   EMStartUp(zp+0x300,20,0,640,0,200,_ownerid);
   WindStartUp(_ownerid);
   CtlStartUp(_ownerid,zp+0x500);
   MenuStartUp(_ownerid,zp+0x400);
   RefreshDesktop(NULL);
   LEStartUp(_ownerid,zp+0x600);
   DialogStartUp(_ownerid);
   DeskStartUp();
   SFStartUp(_ownerid,zp+0x700);
   FMStartUp(_ownerid,zp+0x800);
   PMStartUp(_ownerid,zp+0x900);

      /* Il List Manager non richiede inizializzazione */
   }

   /* Chiudi : esegue lo shutdown dell'intero apparato del Toolbox, chiudendo
               anche gli eventuali NDAs.
   */

   Chiudi()
   {
      DeskShutDown();
      PMShutDown();
      FMShutDown();
      SFShutDown();
      DialogShutDown();
      LEShutDown();
      MenuShutDown();
      WindShutDown();
      CtlShutDown();
      EMShutDown();
      QDAuxShutDown();
      QDShutDown();
      ScrapShutDown();
   }

   /* Info : dialog box richiamato dall'item "Info su...". Fa uso di un item
             LongStatText2 per visualizzare il testo formattato
   */

   Info()
   {
      GrafPortPtr dial;                         /* puntatore al dialog */
      Rect r;
      int lft;
      static char message[]="\1J\1\0\1S\020\0Base 2.0\r\r\1S\0\0\
   base per applicazioni Apple IIGS\r\
   di Francesco Meschia - Asti 1988\r";
      lft=(640-300)/2;                          /* calcola la posizione */
      SetRect(&r,lft,40,lft+300,140);
      dial=NewModalDialog(&r,1,0L);             /* crea la dialog... */
      SetRect(&r,100,75,200,90);
      NewDItem(dial,1,&r,buttonItem,"\pOk",0,0,NULL);     /* ...il bottone OK */
      SetRect(&r,10,15,280,70);                 /* ...e il messaggio */
      NewDItem(dial,2,&r,cSItem2,message,sizeof(message)-1,0,NULL);
      ModalDialog(NULL);                        /* aspetta che si scelga OK */
      CloseDialog(dial);                        /* e chiude la dialog */
   }

   /* Attento : alert box con bottone OK e testo esplicativo. Vengono supportati
                i quattro tipi di alert forniti dal toolbox di Apple IIGS.
                Il testo della alert deve essere in formato C.
   */

   Attento(tipo,id,s1,s2,s3,s4,opt,d)
   int tipo,id,s1,s2,s3,s4,opt;
   char *d;
   {
      static ItemTemplate okbut = (1,0,0,0,0,buttonItem,"\pOk",0,0,NULL);
      static ItemTemplate si = (2,0,0,0,0,buttonItem,"\pSi",0,0,NULL);
      static ItemTemplate no = (3,0,0,0,0,buttonItem,"\pNo",0,0,NULL);
      static ItemTemplate annul = (4,0,0,0,0,buttonItem,"\pAnnulla",0,0,NULL);
      static ItemTemplate testo = (10,0,0,0,0,cSItem,NULL,0,0,NULL);
      static AlertTemplate alt = (0,0,0,0,0,0,0,0,0,NULL,NULL,NULL,NULL,NULL);
      int l,lm,res;

      alt.atStage1=s1; alt.atStage2=s2; alt.atStage3=s3; alt.atStage4=s4;
      l=170+CStringWidth(d);                    /* calcola la larghezza */
      l=(l<400 ? 400 : l);                      /* della stringa e le */
      l=(l>600 ? 600 : l);                      /* dimensioni della alert */
      lm=(640-l)/2;
      SetRect(&alt.atBoundsRect,lm,40,lm+l,110);
      switch (opt) {                            /* quanti bottoni ? */
      case 1:                                   /* uno... */
          SetRect(&okbut.itemRect,l-120,48,l-20,63);
          alt.atItemList[0]=&okbut;
          break;
      case 2:                                   /* ...due... */
          SetRect(&okbut.itemRect,l-120,38,l-20,53);
          SetRect(&annul.itemRect,l-120,18,l-20,33);
          alt.atItemList[0]=&okbut;
          alt.atItemList[1]=&annul;
          break;
      case 3:                                   /* ...o tre? */
          SetRect(&si.itemRect,l-120,48,l-20,63);
          SetRect(&no.itemRect,l-120,28,l-20,43);
          SetRect(&annul.itemRect,l-120,8,l-20,23);
          alt.atItemList[0]=&si;
          alt.atItemList[1]=&no;
          alt.atItemList[2]=&annul;
          break;
      }
      SetRect(&testo.itemRect,20,30,l,65);
      alt.atAlertID=id;                         /* fissa la item template */
      testo.itemDescr=d;                        /* per la stringa esplicativa */
      testo.itemValue=strlen(d);
      alt.atItemList[opt]=&testo;
      switch (tipo) {                           /* e crea l'alert, */
      case 0:                                   /* scegliendone il tipo */
          res=Alert(&alt,NULL);                 /* e ritornando in res l'ID */
          break;                                /* del bottone premuto */
      case 1:
          res=StopAlert(&alt,NULL);
          break;
      case 2:
          res=NoteAlert(&alt,NULL);
          break;
      case 3:
          res=CautionAlert(&alt,NULL);
          break;
      }
      return(res);
   }

   /* CheckToolError : se c'e' un errore del Toolbox chiama il System Fail Mgr.
                       Adattata dall'Apple Programmer's Introduction.
   */

   CheckToolError(dove)
   int dove;
   {
      static char Testamento[]="\pIn $XXXX: Toolbox error #";
      int errore;

      errore=_toolErr;                          /* salva il codice d'errore */
      if (errore) {                             /* se c'e' davvero... */
          Int2Hex(dove,Testamento+5,4);         /* scrive dove e' avvenuto */
          SysFailMgr(errore,Testamento);        /* e fallisce */
      }
```