Questo mese per i lettori di MC niente «passeggiate amighevoli» del Mangrella ma una rilassante utility atta a cambiare lo stato del led di accensione del vostro Amiga e, conseguentemente, lo stato del filtro passa basso posto dai progettisti Amiga (500 e 2000 delle ultime generazioni) sull'uscita audio. Ricordate? Ci siamo già occupati del problema per gli Amiga non dotati di tale dispositivo un bel po' di numeri fa nella rubrica HardAmiga. Il bello dell'utility presentata questo mese, è che appare all'utente come una finestrella «volante» che potremo facilmente posizionare su qualsiasi schermo programma che generi suoni, per cambiare lo stato del filtro. In un certo senso, niente di nuovo, è vero, ma come scusa per pubblicare come al solito qualcosa di didatticamente valido, va più che bene

SuperLED v1.25

di Claudio Castellini - Genova

Alzi la mano chi, possedendo un Amiga, non ha mai visto una «Guru meditation». Come supponevo, nessuno ha la mano alzata. Dunque, chi ha fatto questa esperienza (vale a dire, tutti) avrà certamente notato che, prima che compaia il fatidico requester (meglio: alert) rosso lampeggiante recante la scritta «Software Failure. Press left mouse button...», il LED indicatore dell'alimentazione contrassegnato «POWER» lampeggia brevemente. Si tratta di un effetto secondario generato dal Sistema di Filtraggio Errori Software altresì «Guru», non so perché. Ma ciò che ci interessa è il fatto che tale LED indica, oltre all'accensione o meno del nostro Amiga, l'attivazione di un filtro audio che taglia le frequenze attorno ai 7 kHz e mezzo in uscita su tutti e quattro i canali. Questo limitatamente ai modelli Amiga 500 e 2000. La differenza sta nel suono prodotto dal computer sensibile anche al meno allenato degli orecchi, e per sperimentarla è sufficiente premere il tasto L durante una partita a «Dragon's Lair»: la musica si fa più ricca, più fredda, e gli effetti sonori divengono più metallici. Questo appunto perché viene eliminato il filtro, e anche le frequenze più alte sono libere di passare.

Il programma SuperLED permette, grazie ad un gadget di tipo TOGGLE, di alterare lo stato di tale filtro, virtualmente aggiungendo un comando a quei programmi di manipolazione sonora che non permettono questa manovra (ad esempio, l'ottimo Sonix della Aegis). Inoltre, la finestrella contenente il gadget può essere spostata da uno schermo all'altro, consentendo la visione più globale possibile della situazione.

Come usarlo

Il programma (che NON vi restituisce il controllo se non usate RUN) può partire sia da WorkBench che da CLI e non necessita di parametri supplementari. È sufficiente invocarlo o eseguire un double-click sulla sua icona. Immediatamente, compare una piccola finestra (150×30) sullo schermo WorkBench, recante, oltre al titolo, tre gadget: quello al centro, di tipo TOGGLE, è appunto l'«interruttore» del filtro, sul

```
/*

 Programma ......... SuperLED.c v1.25
 Linguaggio ........ Lattice C v5.00
 Autore ............ Claudio Castellini
 Inizio stesura .... 1 gennaio 1990
 Fine stesura ...... 2 gennaio 1990
 Inizio debugging .. 2 gennaio 1990
 Fine debugging .... 10 gennaio 1990
 ------------------------------------------------------------
 Questo programma è di pubblico dominio e può pertanto
 essere duplicato e modificato a piacere da chiunque,
 purchè non venga utilizzato a scopo di lucro.

*/


/********************************************** Include files ************/

#include <intuition/intuition.h>
#include <intuition/intuitionbase.h>

/****************************************** Definizioni esterne ************/

#define FILTRO ((BYTE *)0xBFE001)
#define BANNER1 "SuperLED 1.25"
#define BANNER2 "di C. Castellini, Genova, 1990."
#define BANNER3 "Programma di Pubblico Dominio."
#define W_LARGH 150L
#define W_ALT 30L
#define H_LARGH 300L
#define H_ALT 75L

ULONG lock;

struct IntuitionBase *IntuitionBase;
struct Screen *SchermoAttuale;
struct Window *Finestra;

/* ------ Bordo dei gadget HIT ---------------------- */

WORD Coord1[] =
  {
  0, 0, 29, 0, 29, 13, 0, 13, 0, 0
  };

struct Border Bordo_mini =
  {
  -2, -2, 3, 0, JAM1, 5, Coord1, NULL
  };

/* ------- Bordo del gadget TOGGLE ------------------- */

WORD Coord2[] =
  {
  0, 0, 54, 0, 54, 13, 0, 13, 0, 0
  };

struct Border Bordo_maxi =
  {
  48, 13, 3, 0, JAM1, 5, Coord2, NULL
  };
```

```
/* ------- Gadget hit di sinistra ------------------- */

struct IntuiText Testo_Fly =
  {
  1, 0, JAM2, 1, 1, NULL, (UBYTE *)"Fly", NULL
  };

struct Gadget Gadget_Fly =
  {
  NULL, 10, 15, 26, 10, GADGHCOMP, RELVERIFY, BOOLGADGET,
  (APTR)&Bordo_mini, NULL, &Testo_Fly, 0, NULL, 0, NULL
  };

/* ------- Gadget hit di destra --------------------- */

struct IntuiText Testo_About =
  {
  1, 0, JAM2, 1, 1, NULL, (UBYTE *)"Wow", NULL
  };

struct Gadget Gadget_About =
  {
  &Gadget_Fly, 114, 15, 26, 10, GADGHCOMP, RELVERIFY, BOOLGADGET,
  (APTR)&Bordo_mini, NULL, &Testo_About, 0, NULL, 0, NULL
  };

/* ------- Gadget toggle ----------------------------- */

struct Image LED_OFF =
  {
  0, 0, 50, 10, 2, NULL, 0x00, 0x00, NULL
  };

struct Image LED_ON =
  {
  0, 0, 50, 10, 2, NULL, 0x00, 0x03, NULL
  };

struct Gadget Gadget_LED =
  {
  &Gadget_About, 50, 15, 50, 10, GADGHIMAGE|GADGIMAGE,
  GADGIMMEDIATE|TOGGLESELECT, BOOLGADGET, (APTR)&LED_OFF, (APTR)&LED_ON,
  NULL, 0, NULL, 0, NULL
  };

/* ------- Testi del requester ----------------------- */

struct IntuiText Testo_req1 =
  {
  0, 1, JAM2, 23, 28, NULL, BANNER3, NULL
  };

struct IntuiText Testo_req2 =
  {
  0, 1, JAM2, 18, 17, NULL, BANNER2, &Testo_req1
  };

struct IntuiText Testo_req3 =
  {
  0, 1, JAM2, 90, 6, NULL, BANNER1, &Testo_req2
  };

struct IntuiText Testo_OK =
  {
  0, 1, JAM2, 6, 3, NULL, "Vai pure!", NULL
  };
/* ------- Finestra ---------------------------------- */

struct NewWindow nw =
  {
  200, 0, W_LARGH, W_ALT, 2, 3, CLOSEWINDOW|GADGETUP|GADGETDOWN,
  WINDOWDEPTH|WINDOWDRAG|WINDOWCLOSE|NOCAREREFRESH,
  &Gadget_LED, NULL, (UBYTE *)"SuperLED",
  NULL, NULL, 0, 0, 0, 0, WBENCHSCREEN
  };

/************************************** Funzioni secondarie ************/

void Basta (BOOL Status)
  {
  if (Finestra)
    CloseWindow (Finestra);
  if (IntuitionBase)
    CloseLibrary (IntuitionBase);
  _exit (Status);
  }

void ApriLaFinestra ()
  {
  if (!(Finestra = (struct Window *) OpenWindow (&nw)))
    Basta (FALSE);
  DrawBorder (Finestra-> RPort, &Bordo_maxi, 0L, 0L);
  SchermoAttuale = Finestra-> WScreen;
  }

void ApriLaLibreria ()
  {
  if (!(IntuitionBase = (struct IntuitionBase *)
                        OpenLibrary ("intuition.library", 0L)))
    Basta (FALSE);
  }

BOOL Fly ()
  {
  struct Screen *ProssimoSchermo;

  if (!(ProssimoSchermo = SchermoAttuale-> NextScreen))
```

(continua a pag. 264)

quale si può agire col mouse per cambiarne lo stato. È interessante notare che il programma esegue un controllo sul filtro stesso, in modo da cambiare lo stato del gadget in accordo con quello del LED, dieci volte al secondo (può sembrare un valore alto, ma lunghi esperimenti mi hanno dimostrato che è il compromesso ideale tra maggior velocità di ripetizione e minore occupazione del tempo CPU); questo fa sì che, se fate partire una decina di SuperLED assieme e agite su uno dei dieci in modo da cambiare lo stato del filtro, tutti gli altri si «aggiorneranno» al più presto possibile.

Il secondo gadget, quello marcato «Fly», fa appunto «volare» la finestra sullo schermo che sta sotto a quello attuale. Una volta giunti all'ultimo schermo, «Fly» riporta SuperLED sullo schermo in primo piano.

Il terzo gadget mostra un requester con alcune informazioni sul programma e sull'autore.

Il sorgente

Premetto che ho imparato il Linguaggio C da tre mesi e quindi il listato è probabilmente zeppo di inesattezze formali: prego chiunque lo trovi sciatto, infantile, e «messo assieme con lo scotch» di non dirmelo. Ognuno ha bisogno di illusioni!

Prima di tutto bisogna dire che:
1) per azzerare un bit, si effettua l'AND tra il contenuto della locazione di memoria ed il complemento a due (Castellini! Stai più attento: si tratta del complemento ad UNO, non del complemento a DUE, ndadp) del valore del bit stesso, mentre per portarlo a uno si fa l'OR tra il contenuto ed il valore puro. Nel nostro caso, per alzare il filtro è sufficiente l'istruzione:

```
*FILTRO &= ~2;
```

dove FILTRO, definito come ((BYTE *)0xBFE001), rappresenta la locazione del filtro. Per abbassarlo, invece, diremo

```
*FILTRO |= 2;
```

Ricordo infatti che, per una curiosa discordanza, se il bit viene messo a 1, il filtro si abbassa, e viceversa.

2) la lista degli schermi comincia in IntuitionBase->FirstScreen, che rappre-

senta lo schermo in primo piano. La catena prosegue leggendo il parametro NextScreen di ogni schermo successivo, finché non si trova il valore NULL, che significa che quello schermo è l'ultimo, ovvero quello dietro a tutti.

Dunque, dopo un breve banner iniziale e gli include file, ci sono le definizioni esterne. Innanzitutto il filtro, inteso come puntatore ad un byte (prima che mi dicessero che il 68000 non indirizza le word dispari, sono diventato matto a furia di Guru). Seguono le definizioni dei tre banner che appaiono nel requester, delle sue dimensioni e di quelle della finestra. Quindi, nell'ordine:

— il numero ULONG relativo alla funzione LockIBase();
— la struttura IntuitionBase;
— la struttura Screen * indicante lo schermo attuale;
— la struttura Window * relativa alla nostra finestra.

In rapida successione, ho poi messo le strutture statiche Gadget con le relative IntuiText, Image e Border, le IntuiText del requester e, finalmente, la NewWindow necessaria all'apertura delle finestra stessa.

Passiamo alle funzioni esterne. Basta (BOOL) è una funzione di abort su misura, mentre ApriLaFinestra() e ApriLaLibreria() fanno ciò che indica il loro nome. La prima provvede inoltre a tracciare il bordo del gadget centrale e ad associare a SchermoAttuale l'indirizzo della struttura Screen su cui ci si trova (all'inizio è sempre il WB).

Fly() è la funzione che sposta la finestra da uno schermo all'altro. Per farlo esegue questi passi:

— decide il valore del prossimo schermo (IntuitionBase->FirstScreen o SchermoAttuale->NextScreen) in base proprio a quest'ultimo valore; in altre parole, controlla se siamo in fondo alla lista o meno;
— controlla eventuali incompatibilità di dimensioni tra lo schermo di partenza e quello di destinazione. Ad esempio, se si tenta di portare la finestra in posizione (500, 100) da uno schermo HIRES ad uno largo 320 pixel, si otterrà una Guru. Stesso discorso se lo schermo ricevente è più piccolo della finestra stessa (evento improbabile ma possibile). In questi casi, Fly() corregge le coordinate di destinazione o si rifiuta di trasferire;
— a questo punto, assegna alla vecchia struttura NewWindow lo schermo deciso, chiude la Window presente e la riapre. L'uovo di Colombo, no?

Controlla() è una funzione che ho inserito per scrupolo. Mi sono accorto, dopo aver ultimato la versione 1.2, che in teoria è possibile chiudere uno scher-

(segue da pag. 263)

```
    }
    lock = LockIBase (0);
    ProssimoSchermo = IntuitionBase-> FirstScreen;
    UnlockIBase (lock);
    }

    if ((ProssimoSchermo-> Width < 150) || (ProssimoSchermo-> Height < 30))
      {
      DisplayBeep (SchermoAttuale);
      return (FALSE);
      }

    if (((nw.LeftEdge + Finestra-> LeftEdge) + W_LARGH) > ProssimoSchermo-> Width)
      nw.LeftEdge = ProssimoSchermo-> Width - W_LARGH;

    if (((nw.TopEdge + Finestra-> TopEdge) + W_ALT) > ProssimoSchermo-> Height)
      nw.TopEdge = ProssimoSchermo-> Height - W_ALT;

    nw.Screen = ProssimoSchermo;

    CloseWindow (Finestra);
    ApriLaFinestra ();
    }

    BOOL Controlla ()
      {
      struct Screen *temp;

      lock = LockIBase (0);
      temp = IntuitionBase-> FirstScreen;
      UnlockIBase (lock);

      while (temp)
        {
        if (temp == SchermoAttuale)
          return (TRUE);
        temp = temp-> NextScreen;
        }
      return (FALSE);
      }

    void CambiaGadget ()
      {
      if (Controlla ())
        {
        (Gadget_LED.Flags & SELECTED) ?
        (Gadget_LED.Flags &= ~SELECTED) :
        (Gadget_LED.Flags |= SELECTED);
        RefreshGadgets (&Gadget_LED, Finestra, NULL);
        }
      else
        {
        DisplayBeep (IntuitionBase-> FirstScreen);
        CloseLibrary (IntuitionBase);
        _exit (FALSE);
        }
      }

    /************************************** Funzione principale ************/

    void _main ()
      {
      struct IntuiMessage *Messaggio, MioMessaggio;

      ApriLaLibreria ();
      ApriLaFinestra ();

      nw.Type = CUSTOMSCREEN;

      for (;;)
        {
        if (Messaggio = (struct IntuiMessage *) GetMsg (Finestra-> UserPort))
          {
          CopyMem (Messaggio, &MioMessaggio, sizeof (struct IntuiMessage));
          ReplyMsg (Messaggio);

          switch (MioMessaggio.Class)
            {
            case CLOSEWINDOW:
              Basta (TRUE);
              break;

            case GADGETDOWN:
              (*FILTRO & 2) ? (*FILTRO &= ~2) : (*FILTRO |= 2);
              break;

            case GADGETUP:
              if (MioMessaggio.IAddress == (APTR)&Gadget_Fly)
                Fly ();
              else
                AutoRequest (Finestra, &Testo_req3,
                             &Testo_OK, &Testo_OK, NULL, NULL, R_LARGH, R_ALT);
              break;
            }
          }
        else if ((*FILTRO & 2) == ((Gadget_LED.Flags & SELECTED) >> 6))
          CambiaGadget ();
        Delay (5);
        }
      }

    /************************************************ Fine Programma ************/
```

mo SENZA aver chiuso prima le Window associate ad esso. Che succede in quel caso? Le strutture vengono distrutte (ma consumano un po' di memoria che resta, per così dire, «morta») ma il programma non ne sa nulla. Si potrebbe pensare che, restando la finestra inaccessibile all'utente, non ci sia alcun pericolo; invece, su una macchina multitasking come Amiga, i pericoli si nascondono dietro ogni byte... Nel nostro caso, se un evento qualsiasi cambia lo stato del filtro, il SuperLED «in catalessi» se ne accorge, si risveglia e cerca di cambiare lo stato del gadget indicatore, che però ormai non esiste più: bang.... Questa funzione, chiamata ogni qual volta il filtro cambia, controlla che SchermoAttuale sia ancora presente nella lista di schermi: se non lo è, significa che l'utente ha inavvertitamente chiuso, lo schermo del SuperLED. In questo caso, anziché procedere, viene chiusa la libreria ed il programma termina con un flash.

Ora, benché una situazione del genere non sia pericolosa grazie al suddetto sistema di filtraggio, io consiglio, se avete per sbaglio chiuso lo Screen dove era presente SuperLED, di salvare tutto e resettare: non si sa mai.

Segue la funzione CambiaGadget(), che altera lo stato del gadget indicatore invertendo lo stato del flag SELECTED ed effettuando il restauro (W l'italiano! Perché usare «refresh»?), previo uso di Controlla().

E veniamo al Main(). Dopo aver aperto tutto e cambiato il flag WBENCHSCREEN della NewWindow in CUSTOMSCREEN (se no la finestra finirebbe continuamente sul WB!) inizia un ciclo senza fine for (;;) nel quale si alternano rapidamente due controlli: uno per determinare se l'utente ha premuto qualcosa, l'altro che controlla lo stato del filtro.

Nel primo caso, dopo aver «fotocopiato» il messaggio di Intuition e risposto, si controlla quale gadget è stato attivato: se il gadget di chiusura, il programma termina; se il LED gadget, si altera lo stato del filtro; se il Fly gadget, si effettua il salto; se il gadget «Wow», appare il requester. Nel secondo caso, si provvede ad alterare il gadget in modo automatico. Quindi, un ritardo di 5 cinquantesimi di secondo e via daccapo.

Una parola ancora sull'istruzione sui test dello stato del LED, la riporto qui:

```
else if ((*FILTRO & 2) = = ((Gadget_LED.
        Flags & SELECTED) >> 6))
```

Si tratta di confrontare il valore del bit del filtro con il flag di selezione del gadget. Perciò, il confronto avviene in questi termini: ricordando che il valore di SELECTED è 0x80, se questo shiftato 6 volte (ovvero diviso per 0x40) è pari al valore del bit-filtro, bisogna alterare il gadget; infatti ciò significa o che sono entrambi 0x02, cioè gadget attivo e filtro basso, o che sono entrambi nulli, vale a dire gadget disattivato e filtro alto. OK? MC