coordinamento di Andrea de Prisco

Programmare in C su Amiga (18)

Forse la caratteristica più conosciuta ed apprezzata dell'Amiga è la sua interfaccia grafica, facile ed intuitiva, da cui appunto il nome: Intuition. Uno dei punti forti di tale interfaccia sono i menu. A partire da questa puntata vedremo come si definisce e si usa una interfaccia a menu, fino ad arrivare a tecniche particolarmente sofisticate e potenti

Nella scorsa puntata abbiamo visto come gestire un generico evento ed il messaggio ad esso relativo che Intuition spedisce per informarci della cosa, insieme ad eventuali dati legati a quel particolare evento. Abbiamo inoltre costruito uno scheletro di una procedura il cui compito è quello di creare una copia locale del messaggio, rilasciare quella originale dando ricevuta ad Intuition, identificare il tipo di messaggio ricevuto ed in base a tale codice, passare il controllo ad un'altra procedura che si occupi di gestire l'evento in questione.

Da questa puntata metteremo in pratica quanto fin qui appreso introducendo i menu e la loro gestione. Vedremo come si informa Intuition che vogliamo associare dei menu ad una finestra, vedremo come utilizzare lo scheletro menzionato nel caso che gli unici eventi siano quelli di chiusura della finestra e di selezione di un elemento da un menu, vedremo come costruire un menu a discesa e molto altro ancora. Il tutto ci porterà via più di una puntata, ovviamente, ma alla fine saremo in grado di padroneggiare una tecnica di interfacciamento con l'utente molto più sofisticata e flessibile di quelle viste finora.

Nella seconda parte di questa puntata, continueremo a parlare di *GREP* e delle sue opzioni. Tornano infine, per la prima volta insieme, la *Casella Postale* e la *Scheda Tecnica*, quest'ultima dedicata alle operazioni di controllo che Amiga effettua alla partenza del sistema.

I menu

Un menu è una lista di azioni o di attributi che possono essere selezionati dall'utente tramite tastiera o mouse. Una *azione* è in sostanza un comando, corrisponde cioè ad una operazione che il programma deve effettuare. Viceversa un *attributo* è una caratteristica di un oggetto del sistema, ad una opzione relativa ad un certo comando. Mentre si può selezionare una sola azione alla volta, è possibile selezionare più attributi contemporaneamente, sempre che essi siano compatibili fra loro, come vedremo in seguito (vedi nota 1). In Amiga, i menu sono organizzati come una struttura gerarchica formata al massimo da tre livelli, descritti qui di seguito.

1. Il primo livello è rappresentato dalla *barra dei menu [menu bar]*. Questa contiene una lista di nomi a ciascuno dei quali corrisponde un gruppo di azioni e/o attributi. Diremo che ad ogni nome corrisponde un *menu. [menu]*.

2. Il secondo livello è rappresentato dai *menu* veri e propri, i cui elementi sono detti *voci [item]*. Le voci sono generalmente riportate una di seguito

Note

1. La possibilità di due o più attributi di coesistere è *sempre* definita dal programmatore. Intuition non ha la possibilità di prendere alcuna decisione relativamente alla logica che sta alla base di una certa struttura di menu, ma solo riguardo la loro gestione in termine di posizionamento e dimensioni, come vedremo nella prossima puntata.

2. Vedremo più avanti che è possibile costruire anche un tipo molto differente di menu, chiamati *menu a comparsa [pop-up menu]*, in quanto appaiono più o meno nel punto in cui si trova il cursore nel momento in cui l'utente li richiede tramite il tasto destro del mouse. Sono più rari in Amiga, ma Intuition ci permette comunque di definirli.
Per il momento ci occuperemo solo dei menu a discesa. Riprenderemo il discorso con quelli a comparsa dopo aver parlato di gadget e quadri.

3. Questo per quello che riguarda il bottone sinistro del mouse, ovviamente, ma un discorso analogo vale anche per quello destro.

all'altra e disposte verticalmente. Da qui il nome di *menu a discesa [pull-down menu]* che viene dato a questo tipo di rappresentazione (vedi nota 2).

3. Il terzo livello è rappresentato dai *sottomenu [submenu]*, i cui elementi sono detti *sottovoci [subitem]*, e serve a dettagliare ulteriormente il secondo livello, se necessario.

Nel momento in cui tre livelli non bastassero, oppure fosse necessario da parte dell'utente di fornire altri dati per poter eseguire l'azione selezionata o specificare più in dettaglio un determinato attributo, è possibile utilizzare i cosiddetti *quadri [requester]*, che vedremo in seguito.

Ci sono due modi di selezionare una voce (od una sottovoce) di un menu. Il primo consiste nel premere il bottone destro del mouse, detto appunto *bottone dei menu [menu button]* in modo che appaia la barra dei menu al posto del titolo (a meno che non sia già visibile). Quindi, sempre tenendo premuto tale bottone, spostare il cursore sul menu che ci interessa, e «tirare giù» la lista delle voci, posizionare il cursore sulla voce che ci interessa e rilasciare il bottone. Se ad una voce corrisponde un sottomenu, questo comparirà automaticamente quando il cursore è posizionato nella voce in questione. Le sottovoci si selezionano come le voci. In genere, mentre il cursore si trova su un elemento di un determinato livello, sia esso menu, voce o sottovoce, questa viene evidenziata in qualche modo. Il secondo modo consiste nell'associare ad un elemento selezionabile (voce o sottovoce) una particolare combinazione di tasti, e precisamente uno dei due tasti Amiga ed un qualunque tasto alfanumerico.

Notare che mentre voci e sottovoci sono detti *elementi selezionabili*, in quanto l'utente può selezionare tali elementi con uno dei due metodi appena descritti, i menu veri e propri possono essere aperti, ma non selezionati.

Se si cambia idea e non si vuole più selezionare una voce basta rilasciare il bottone destro del mouse in un qualun-

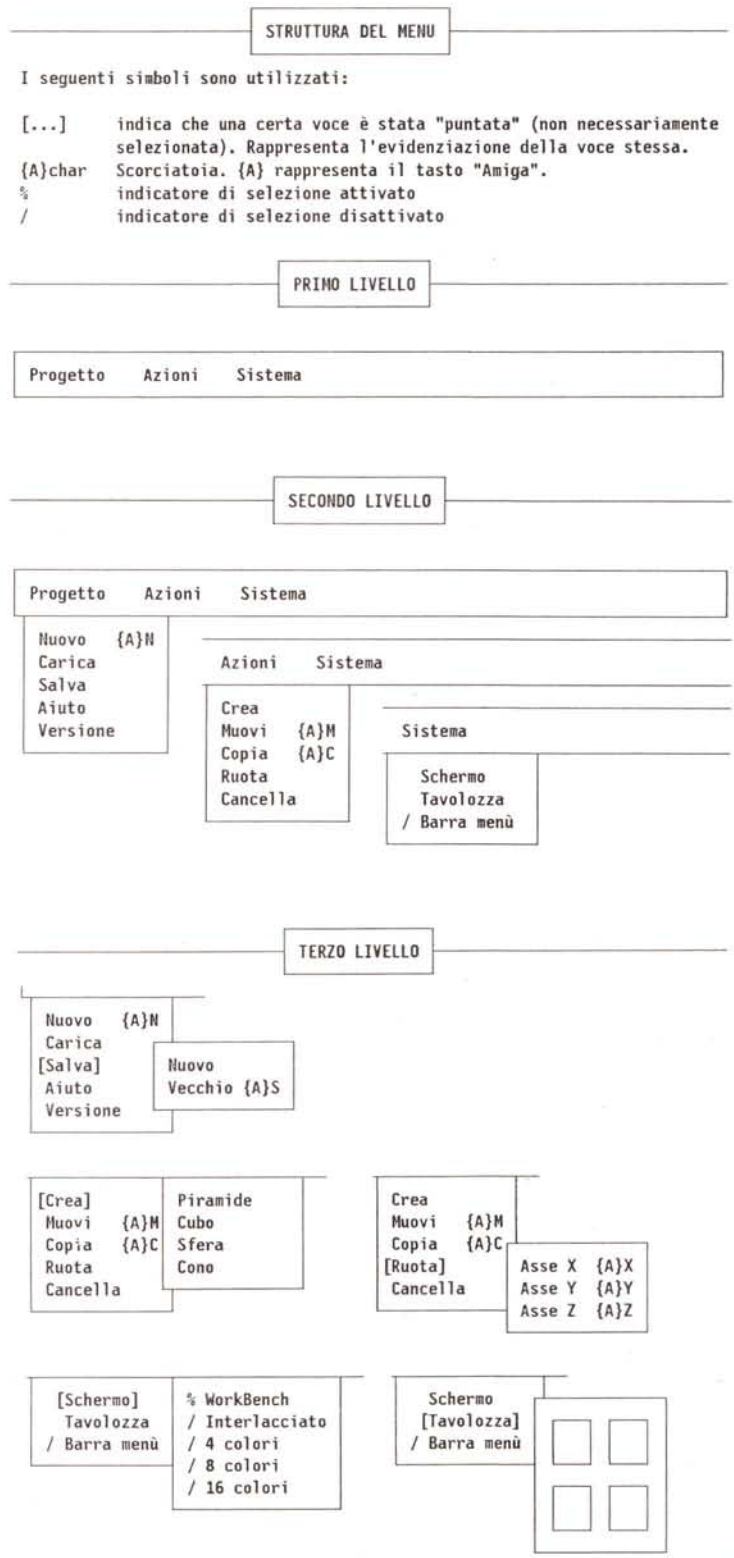

Figura 1 - Un esempio di interfaccia a menu.

que punto dello schermo non interessato dai menu.

Al contrario delle azioni, in genere, un attributo può rimanere nello stato *selezionato* anche dopo che il menu è stato chiuso. In realtà nulla vieta di fare la stessa cosa con una azione, ma nella maggior parte dei casi la cosa non ha senso. A causa di questa particolarità è possibile che più attributi siano *attivi* contemporaneamente. Di solito tale stato è indicato da un simbolo *[checkmark]* che si «accende» di lato al nome dell'attributo. Quando questi è inattivo, il simbolo è spento, ma è possibile vederlo ugualmente *[fake checkmark]* in modo da far sapere all'utente quali attributi possono mantenere lo stato di attivazione.

A questo punto è importante dire una cosa. Tutti voi che state leggendo questo articolo avrete riconosciuto in questa descrizione il modo in cui la maggior parte dei programmi per Amiga gestisce l'interfaccia a menu. Questo nasce dal fatto che la Commodore ha dato certe regole di comportamento per garantire una certa uniformità nell'interfaccia utente. Tuttavia solo in pochissimi casi Intuition forza il disegno dei menu. Per il resto sta al programmatore definire certe cose. Ad esempio, nulla impedisce di costruire un menu in cui gli attributi attivi, invece di essere marcati da un simbolo, abbiano un certo nome, mentre se disattivati ne prendono un altro (ad esempio, **HIRES ON <-> HIRES OFF).**

È inoltre possibile definire se un attributo possa essere attivato o meno in funzione di quali lo siano già, e selezionare più voci in sequenza facendo uso di entrambi i bottoni del mouse contemporaneamente. Quest'ultima operazione può essere effettuata in due modi:

1. tenendo premuto il bottone destro e premendo quello sinistro solo sulle voci che ci interessa selezionare (in questo modo si possono anche selezionare più voci da menu distinti);
2. oppure premendo contemporaneamente entrambi i bottoni del mouse e muovendo il cursore sulle voci che ci interessano (così vengono selezionati blocchi di voci contingue).

Una voce (o sottovoce) può essere rappresentata sia da un testo che da una immagine. Usando una tecnica speciale di sovrapposizione è possibile avere elementi misti testo/grafica.

Un esempio di interfaccia a menu è riportato in figura 1. Nel caso riportato ci sono tre menu: Progetto, Azioni e Sistema. Nel primo menu ci sono due voci che aprono quadri: Carica e Versione. Salva ha un sottomenu, di cui una sottovoce può essere selezionata via tastiera con Amiga-S. Le voci precedute dal simbolo / rappresentano gli attributi selezionabili non attivi, quelle precedute dal simbolo % quelli attivi (ovviamente è solo un esempio). La voce Tavolozza del menu Sistema ha come sottomenu una tavolozza in cui le sottovoci sono rappresentate da quadrati colorati piuttosto che da testo.

Questa lunga introduzione aveva lo scopo di schematizzare i concetti che stanno alla base dell'interfaccia a menu. Sicuramente molti di voi queste cose le sapevano già, quantomeno perché quasi tutti i programmi Amiga si comportano in questo modo, tuttavia molti magari non sapevano quanti e quali livelli sono possibili, quanti e quali tipi di elementi esistono, ed in generale cosa Intuition ci mette a disposizione e fino a che punto siamo liberi di disegnare la nostra interfaccia. Alla fine di questo blocco di puntate dedicate ai menu spenderemo due parole su come si disegna un'interfaccia di questo tipo e quali tecniche sono consigliabili sia da un punto di vista di maggiore usabilità, sia per una migliore integrazione nel sistema ed uniformità con gli altri prodotti.

Vediamo ora come si definisce una finestra alla quale vogliamo associare una interfaccia a menu, e come si attiva e si disattiva questa interfaccia.

```
/* ------------------------------------------------------------------ *
 * --- Associa una struttura menù ad una finestra                 --- *
 * ------------------------------------------------------------------ */
void SetMenuStrip(struct Window *, struct Menu *);

/* ------------------------------------------------------------------ *
 * --- Elimina la struttura menù associata ad una finestra        --- *
 * ------------------------------------------------------------------ */
void ClearMenuStrip(struct Window *);

/* ------------------------------------------------------------------ *
 * --- Ritorna il puntatore alla struttura relativa ad una voce   --- *
 * ------------------------------------------------------------------ */
struct MenuItem *ItemAddress(struct Menu *, long);

/* ------------------------------------------------------------------ *
 * --- Disattiva un menù, una voce od una sottovoce               --- *
 * ------------------------------------------------------------------ */
void OffMenu(struct Window *, long);

/* ------------------------------------------------------------------ *
 * --- Attiva un menù, una voce od una sottovoce                  --- *
 * ------------------------------------------------------------------ */
void OnMenu(struct Window *, long);
```

Figura 2 - Prototipi delle funzioni relative ad i menu.

Finestre e menu

Una volta aperta una finestra e costruite le strutture necessarie a definire i menu che intendiamo associare a quella determinata finestra (in che modo, lo vedremo nelle prossime puntate), è necessario passare il tutto ad Intuition per mezzo della funzione SetMenuStrip() (vedi figura 2).

Analogamente la funzione ClearMenuStrip() può essere usata per cancellare l'associazione tra una finestra ed una struttura a menu.

È possibile avere più chiamate in sequenza a questa coppia di funzioni, secondo lo schema di tabella A.

È sempre bene cancellare l'associazione tra un menu ed una finestra *prima* di chiudere la finestra, per evitare che il menu sia aperto dopo che la finestra è stata già chiusa.

Associare una struttura menu ad una finestra non è sufficiente, tuttavia. Se si vogliono ricevere le informazioni relative alle operazioni effettuate dall'utente del programma sui menu da noi definiti, è necessario avvertire Intuition di notificarci tali eventi. Per far questo si utilizza una costante chiamata MENUPICK. Tale costante va impostata nel campo della struttura NewWindow che contie-

Tabella A

```
- Apri la finestra con OpenWindow()
  ...
- Associale una struttura a menù con SetMenuStrip()   ┐ può essere
  ...                                                 ├ ripetuto
- Cancella l'associazione con ClearMenuStrip()       ┘ più volte
  ...
- Chiudi la finestra con CloseWindow()
```

ne i segnalatori IDCMP, e cioè IDCMPFlags, nel caso che si intenda gestire tali eventi fin dall'inizio. Altrimenti si può sempre aprire una finestra senza preoccuparci degli eventi di tipo menu, salvo poi attivare la richiesta di notifica utilizzando la funzione ModifyIDCMP() già vista nelle scorse puntate, come nell'esempio seguente:

```
#define MYFLAGS (CLOSEWINDOW|MOUSEBUTTONS)
...
nw.IDCMPFlags = MYFLAGS;
w = (struct Window *)OpenWindow(&nw);
...
ModifyIDCMP(w,MYFLAGS|MENUPICK);
SetMenuStrip(w,mymenu);
...
ClearMenuStrip(w);
ModifyIDCMP(w,MYFLAGS);
...
CloseWindow(w);
```

Un'altra costante molto importante è MENUVERIFY, che ci permette di essere avvisati del fatto che un utente sta per aprire un menu *prima* che questo effettivamente venga aperto da Intuition. Tale evento permette di completare, ad esempio, le operazioni di grafica che stavamo effettuando sulla finestra prima che il menu vada a coprire le zone interessate. Nella prossima puntata vedremo come e perché sia necessario modificare leggermente la logica della procedura HandleEvent() nel caso che si voglia utilizzare questa funzione.

Attenzione però, se volete ricevere eventi di tipo MENUPICK non usate mai la costante RMBTRAP nel campo Flags della struttura NewWindow, altrimenti non sarete in grado di gestire gli eventi relativi ad i menu. Tale valore, infatti, notifica ad Intuition che non siete assolutamente interessati agli eventi di tipo menu, ma che volete ricevere una normale notifica di tipo MOUSEBUTTONS anche nel caso che l'utente prema il tasto destro del mouse. In genere tali eventi sono riportati ogni qual volta l'utente preme un bottone del mouse in un contesto tale da non avere alcun significato per Intuition. Ad esempio, se l'utente seleziona il gadget di chiusura di una finestra, l'evento viene riportato come CLOSEWINDOW; se invece il cursore si trova in un punto qualunque della finestra l'evento è riportato come MOUSEBUTTONS, appunto, sempre che IDCMPFlags sia stato impostato opportunamente (vedi nota 3).

Tenete presente inoltre che RMBTRAP non è una costante IDCMP, e quindi non può essere annullata per mezzo della funzione ModifyIDCMP(). È tuttavia possibile utilizzare il seguente codice per ottenere lo stesso risultato:

```
/*
**   "w" è il puntatore ad una finestra
*/
w->Flags &= ~RMBTRAP;
```

od anche

```
#define RMBtrapON(w)   {(w)->Flags |=  RMBTRAP}
#define RMBtrapOFF(w)  {(w)->Flags &= ~RMBTRAP}
```

E questo è quanto, almeno per questa puntata.

GREP

Nella scorsa puntata abbiamo introdotto il programma di utilità *GREP* (Global Regular Expression Search and Print), cioè un programma in grado di analizzare uno o più file alla ricerca di una o più stringhe di caratteri più o meno definite. Queste infatti possono contenere delle sequenze di caratteri speciali che permettono di definire delle categorie di stringhe dette «espressioni regolari». Ad esempio P.+o rappresenta tutte le stringhe che iniziano con P e terminano con o, con almeno un altro carattere nel mezzo. Abbiamo visto i caratteri e le sequenze speciali, e come definire sia le espressioni regolari che i modelli di ricerca per i file. La sintassi, ricordo, è la seguente:

GREP [opzioni] espressione file1..filen

Adesso vediamo le opzioni. Queste sono sempre precedute dal segno meno (–), e possono essere fornite una ad una o raggruppate in una sola òpzione (ad esempio, si può scrivere sia –pn che –p –n).

- **c** Visualizza il totale delle linee che soddisfano l'espressione regolare specificata.
- **f** Visualizza il nome dei file nei quali è stata individuata almeno una linea che soddisfi l'espressione regolare specificata.
- **n** Non visualizza il numero di linea prima di ogni singola linea visualizzata.
- **p** Blocca l'emissione di caratteri non visualizzabili, come i caratteri di controllo ASCII.
- **q** Non visualizza né i nomi dei file, né i numeri di linea.
- **s** Visualizza il nome di tutti i file interessati dalla ricerca, sia quelli nei quali è stata individuata almeno una linea che soddisfi l'espressione regolare specificata, sia quelli in cui la ricerca è stata negativa.
- **v** Visualizza tutte le linee che *non* soddisfano l'espressione regolare specificata.
- **V** Visualizza la versione corrente di *GREP*.
- **$** Non distingue tra caratteri maiuscoli e minuscoli nelle espressioni regolari.

Nella prossima puntata vedremo come si può usare *GREP* da C.

Conclusione

Bene, anche questa volta siamo arrivati alla fine. Immagino che molti di voi già speravano di essere in grado di lavorare con i menu fin da questa puntata, ma l'argomento richiede un certo spazio, specialmente se si vuole arrivare a padroneggiare con sicurezza la materia. Certo in Basic le cose sono più semplici, ma anche più limitate. Vedremo comunque di disegnare uno scheletro di programma da riutilizzare ogni qual volta vogliamo lavorare con i menu, semplice da usare e sufficientemente flessibile.

Nella prossima puntata vedremo come si modifica la procedura HandleEvent() presentata lo scorso mese, per gestire gli eventi relativi ai menu. Vedremo inoltre le strutture che ci consentiranno di definire i vari elementi di una interfaccia a menu e come è strutturato l'*identificativo di menu*. Torneremo inoltre a parlare di *GREP*, ed in particolar modo della libreria grep.lib.

La Scheda Tecnica

La scheda tecnica di questo mese è dedicata alle verifiche di sistema che l'Amiga fa all'accensione della macchina.
Le seguenti operazioni sono effettuate quando fate partire il sistema:

```
[ 1]  Cancellazione di eventuali vecchi dati da TUTTI i chip.
[ 2]  Disabilitazione dei canali DMA e di TUTTI i segnali di interruzione.
[ 3]  Cancellazione dello schermo.
[ 4]  Controllo HARDWARE (verifica del 68000).
[ 5]  Cambia il colore dello schermo (vedi sotto).
[ 6]  Verifica di tutte le ROM via "checksum".
[ 7]  Cambia il colore dello schermo (vedi sotto).
[ 8]  Inizio partenza sistema.
[ 9]  Verifica della RAM all'indirizzo $C0000. Carica a tale indirizzo SYSBASE.
[10]  Verifica di tutta la RAM di tipo CHIP.
[11]  Cambia il colore dello schermo (vedi sotto).
[12]  Controllo SOFTWARE di sistema.
[13]  Cambia il colore dello schermo (vedi sotto).
[14]  Prepara la RAM di tipo CHIP a ricevere dati.
[15]  Aggancia le librerie.
[16]  Verifica se esiste memoria addizionale ed in caso agganciala.
[17]  Riabilita i canali DMA ed i segnali di interruzione.
[18]  Fa partire un "task" base.
[19]  Verifica l'esistenza di eventuali 68010, 68020 e/o 68881.
[20]  Controlla se è presente un segnale di eccezione (errore di processore)
[21]  Se [20] dà esito positivo, fa ripartire il sistema di nuovo.
```

Durante le verifiche di sistema, lo schermo può assumere vari colori, ad indicare se la verifica è andata a buon fine o meno:

Colore	Tipo	Descrizione
Grigio scuro	Successo	L'HARWARE iniziale è a posto, il 68000 funziona regolarmente, ed i registri sono leggibili.
Grigio chiaro	Successo	Il SOFTWARE di sistema è a posto.
Bianco	Successo	Tutte le verifiche iniziali hanno dato esito positivo.
Rosso	Errore	Un errore è stato trovato nelle ROM.
Verde	Errore	Un errore è stato trovato nella RAM CHIP.
Blu	Errore	Un errore è stato trovato nei chip speciali.
Giallo	Errore	Un errore è stato trovato dal 68000 prima che il software di filtraggio degli errori (GURU) potesse intervenire.

La tastiera ha il suo processore, RAM e ROM. Quando il sistema viene acceso, le seguenti verifiche automatiche sono effettuate da questo processore:

```
[1]  Effettua il controllo di "checksum" sulle ROM di tastiera.
[2]  Comtrolla i 64 byte della RAM.
[3]  Controlla il contatore.
[4]  Verifica della comunicazione con il computer e riporta i risultati delle
     prove effettuate, se negative.
```

Nel caso le prove di tastiera dessero risultato negativo, questo verrebbe indicato per mezzo del LED presente sul tasto di CAPS-LOCK, e precisamente:

# lampeggii	Significato
1	Verifica della ROM di tastiera negativa.
2	Verifica della RAM di tastiera negativa.
3	Verifica del contatore (timer) negativa.
4	Corto circuito fra due serie di tasti o con i tasti di controllo speciali.

Casella Postale

Questo mese risponderò a due brevi lettere ricevute alla fine di ottobre.

Elenco funzioni

Spettabile redazione di MC,
sono uno studente di informatica che legge MC da quando possedeva un mai dimenticato Spectrum. Adesso sono passato all'Amiga e apprezzo molto lo spazio che gli riservate, in particolare per la qualità degli articoli.

Vi scrivo riguardo alla serie «Programmare in C su Amiga» che reputo molto interessante e stimolante. È mio desiderio approfondire maggiormente la conoscenza delle possibilità offerte da SO, ma mi rendo conto delle esigenze della rivista, allora ecco la mia proposta: perché non pubblicate un elenco completo (anche in più puntate e/o come inserto da staccare...) delle funzioni offerte dalle varie componenti del SO con una breve descrizione di ognuna?

Dico ciò perché sono in possesso del Lattice_5.0 in cui, nel quarto dischetto, sono presenti tutti gli header file contenenti tra l'altro le varie strutture dati necessarie per interagire col SO, ma non c'è nulla o quasi sulle funzioni. È vero che si trova tutto sui «ROM Kernel» ma non tutti (me compreso) possono/vogliono spendere più di 100.000 lire per scrivere qualche ludico programmino che si diverta a creare processi comunicanti, ecc. (semmai l'acquisto dei RK può avvenire in futuro se necessario!).

Questo darebbe la possibilità un po' a tutti di sperimentare le varie funzioni in modo abbastanza immediato, e comunque un «function cross reference» fa sempre comodo!

Ringraziandovi per l'attenzione rinnovo i complimenti e saluto cordialmente nella speranza di veder presto esaudita la mia richiesta.

Stefano Tirabassi - Lucca

Caro Stefano,
la tua idea è indubbiamente interessante, anche se probabilmente ci vorrebbero varie decine di puntate prima di riuscire a coprire tutte le funzioni che il SO ci mette a disposizione. Non a caso i RKMs sono due volumi di considerevole *peso* e dimensioni...

In effetti, in seguito alla tua e ad altre richieste, ho iniziato la sottorubrica denominata *La scheda tecnica*. Certamente, delle schede inserto da raccogliere poi in un volumetto a schede staccabili sarebbe una soluzione interessante e più pratica di quella di riportare tali funzioni all'interno della rubrica. Tuttavia non è una decisione che spetta a me, dato che comporta una spesa aggiuntiva in termini di rilegatura della rivista, sia che sia riportata come foglio di car-

toncino da staccare, sia che sia unita alla rivista in una confezione apposita in cellophane. L'idea mi piace, ma a questo punto penso che la palla debba passare ad altri. Da parte mia mi dichiaro disposto a supportare una tale iniziativa qualora si ritenesse il costo ragionevole. Se ciò non avvenisse, cercherò comunque di riportare, come ho sempre fatto finora d'altra parte, le liste complete delle funzioni e delle strutture di cui mi occupo volta per volta nei vari articoli. Certo, così le informazioni sono un po' più sparpagliate, ma se sei un assiduo lettore di MC, basterà stare attenti a non perdere neanche un numero... giusto?

Scherzi a parte, cercherò in futuro di venire incontro sempre più a tutti coloro che non hanno la possibilità di accedere ai *RKMs* ed alla documentazione tecnica in generale per Amiga.

Richiesta d'aiuto

Help me! Dario de Judicibus
Sono un felicissimo possessore di un Amiga 2000 nonché lettore dei tuoi interessantissimi articoli.

Dopo aver imparato a programmare in «C», grazie anche ai tuoi articoli, mi sono imbarcato nel fantascientifico (impossibile) progetto di un programma che permetta all'Amiga di riconoscere comandi vocali.

Vorrei sapere (se possibile) come emulare col suddetto linguaggio i comandi Basic peek & poke e come identificare il tasto premuto in una window creata tramite la funzione OpenWindow dell'Intuition.library (ovviamente dopo aver messo negli IDCMP flags RAWKEY o VANILLAKEY).

Spero che mi risponderai al più presto perché queste informazioni sono di vitale importanza per realizzare il mio pazzo, pazzo, progetto.

Detto questo ti saluto e ti faccio i miei complimenti; saluto anche tutto lo staff di MC.

Distinti saluti

Fabio Alessi

Caro Fabio,
se ho ben capito quello che stai cercando di fare (*riconoscimento di comandi vocali*), ho l'impressione che, per aver iniziato da poco a programmare in C su Amiga, ti sei imbarcato in un progetto alquanto impegnativo. Il tuo programma dovrebbe essere in grado di gestire un digitalizzatore audio, riconoscere un certo numero di comandi vocali, e quindi operare come richiesto. Vada per la gestione del digitalizzatore, per quella di più processi in parallelo, necessaria per questo genere di programmi (puoi usare le semplici e potenti funzioni dell'ADPmttb 2.0), per la gestione dell'interfaccia utente tramite Intuition. È tutto possibile anche se non facile. Ma hai idea di cosa significhi fare del *riconoscimento della voce*? Ci sono gruppi in tutto il mondo che lavorano da anni su questo genere di progetti. Sono necessarie conoscenze approfondite degli algoritmi per il riconoscimento dei fonemi, per la soppressione degli elementi non significativi e del rumore di fondo, per la ricostruzione delle parole in un contesto definito. Insomma, se ci riesci, sei più che bravo... Non voglio scoraggiarti, ma da quel poco che conosco della materia, so che è estremamente complessa. Un consiglio: procedi a blocchi. Disegna cioè il tuo programma in tanti moduli, ognuno responsabile di una attività specifica ed interagente con gli altri tramite un protocollo ben definito. Creati inoltre una serie di funzioni base, un po' come ha fatto Andrea con *ADPmttb*: rende più semplice scrivere il codice successivo.

Ti faccio tutti i miei auguri e passo alle tue domande.

In Basic peek e poke servono rispettivamente a leggere e scrivere in determinate locazioni di memoria del sistema. Questo è molto utile se si vuole gestire direttamente i registri hardware del sistema od altre aree di memoria utilizzate ad esempio per il video piuttosto che per l'audio. Il fatto è che il Basic, non avendo il concetto di *puntatore* e di *indirizzamento diretto*, non è in grado di operare su tali elementi in altro modo. Si tratta di una estensione necessaria nel momento in cui si richiede ad un linguaggio semplice nell'uso ma limitato nella sintassi base, di effettuare operazioni più complesse come ad esempio la gestione diretta dell'hardware. In C questo non è necessario, perché le aree di memoria possono essere indirizzate direttamente, per cui, per mettere il carattere A, ad esempio, nella locazione di memoria 0×21FF70, basta fare:

```
#define RIEMPIMI ((UBYTE *)0x21FF70)

*RIEMPIMI = (UBYTE)'A';
```

In generale possiamo dire che, se BaseAdrs contiene il puntatore ad un certo indirizzo di memoria ed Offset rappresenta la posizione della *parola* che ci interessa calcolata in byte rispetto alla base, la PEEKW e a la POKEW del Basic possono essere rappresentate in C nel modo seguente: ▼

```
/*
**  Attenzione: BaseAdrs è in realtà un numero
**  fosse di tipo puntatore, l'aggiunta di un v
**  verrebbe a tener conto delle dimensioni in l
**  puntato:
**
**    UBYTE *ptr;
**    ptr += 5  ;   -- sposta di 5 byte
**
**    ULONG *ptr;
**    ptr += 5  ;   -- sposta di 20 byte
**
*/
 #define BaseAdrs 0xCF67A0
 #define MEMCELL ((UWORD *)(BaseAdrs+Offset))

 int   Offset  = 0;
 UWORD myvalue = 0;

 ...
 myvalue = *MEMCELL;  /* Questa è una PEEKW */
 ...
 *MEMCELL = myvalue;  /* Questa è una POKEW */
 ...
```

In realtà non è consigliabile codificare direttamente *[hardcode]* gli indirizzi, piuttosto è meglio ottenerli direttamente dal sistema utilizzando quelle funzioni C e delle librerie Amiga che appunto ritornano il puntatore ad aree ben definite. Ecco quindi che, se si apre una finestra, si ottiene il puntatore alla struttura Window relativa, di cui quello alla struttura RastPort, e quindi l'accesso, ad esempio, alle variabili che contengono i colori delle penne usate per la grafica, come già visto nelle scorse puntate.

Per quanto riguarda IDCMP e la gestione della tastiera, ne parleremo in una prossima puntata. Avrai comunque già capito dalla 17ª puntata, che basta andarsi a leggere i campi Code e Qualifiers nel messaggio ricevuto da Intuition. In che modo lo vedremo in seguito. Abbi pazienza, tanto, per mettere su un progetto come quello che hai descritto non saranno questi i problemi più impegnativi che incontrerai. MC

PC-SPEED

Trasformate il vostro computer Atari in un compatibile MS-DOS!!
Scheda hardware di emulazione MS-DOS per Atari ST

ATARI

A proposito di emulatori MS-DOS.....

Tralasciamo le barzellette, e analizziamo i fatti.

PC-SPEED è il primo computer MS-DOS che è stato integrato all'interno di qualsiasi modello di computer Atari serie ST MEGA, 1040 e 520.
Esso conta attualmente circa 10.000 installazioni in tutta Europa, ed è stato premiato dalla stampa del settore come "La novità più innovativa del 1989", sia per quanto riguarda il funzionamento, sia per quanto riguarda la sua affidabilità.

PC-SPEED viene montato all'interno dei computer Atari, in modo da non dover sempre portarsi dietro scatole e bussolotti, permettendo così anche una vera trasportabilità.

PC-SPEED permette di utilizzare la RAM già esistente, e senza bisogno di chip aggiuntivi, può supportare direttamente anche 4 Mbyte, e tutto senza spendere nemmeno 5 lire!!!!

PC-SPEED utilizza, nel caso sia presente, il coprocessore matematico MC68881 (prezzo di listino Atari £ 239.000) in emulazione 8087, ciò vuol dire che è inutile comprare un nuovo e costoso coprocessore matematico!!!

PC-SPEED, con l'ausilio di un driver in dotazione, permette di utilizzare, dove richiesto, il mouse Atari...

PC-SPEED, utilizzando le stesse partizioni sia per MS-DOS che per TOS-GEM, evita di dover uscire da una applicazione per dover trasferire dei file o altro, permettendo uno scambio diretto di dati!!!

PC-SPEED, utilizza come processore, un NEC V 30 a 8 Mhz, pertanto la sua velocità, è pari a quella di qualsiasi sistema utilizzante lo stesso processore (Norton utility fattore 4.0).....

PC-SPEED, a differenza di altri prodotti, può funzionare emulando, indifferentemente, la scheda grafica CGA, la scheda HERCULES, la scheda HERCULES in OVERSCAN, la scheda EGC OLIVETTI (640 × 400), e nella nuova versione, anche la scheda EGA.....

PC-SPEED, è in continuo aggiornamento per migliorare sempre più le sue già ottime prestazioni, e gli aggiornamenti, costano esattamente O lire!!!

Possiamo anche anticiparvi alcune novità:
Utilizzo della stampante LASER ATARI, emulazione scheda ega, collegamento di altre tastiere, via porta MIDI 1, Utilizzo della porta MIDI ATARI 1, Utilizzo di Hard Disk SCSI.

Quanto costa PC-SPEED?

Chiedete al vostro rivenditore, sarà una grandiosa sorpresa....

P.S. Questo, FUNZIONA!!

CARATTERISTICHE PRINCIPALI:

Processore NEC V30 a 8 Mhz 0 wait state, Norton utility fattore velocità 4, Memoria libera 704 Kb, Supporto Memoria Estesa!! (3.9 Mbyte con Mega 4). Emulazione schede grafiche CGA, Hercules, Hyper Hercules, Olivetti, Autoboot da Hard disk con partizioni definibili a piacere, Gestione mouse, Utilizzo di mouse PC in ambiente Atari, Supporto monitor monocromatico, Supporto monitor colore, Gestione porta seriale, Gestione porta parallela, Gestione interrupts. Supporto drive 3.5" e 5.25", Compatibile con tutti i modelli ST.

Dove trovare PC-SPEED

ALT.SERVICE - VIA NAPOLI 112, 63100 ASCOLI PICENO
BOLLO SAVERIO - VIA LESSI 35, 33014 GEMONA DEL FRIULI (UD)
CASA MUSICALE SCAVINO - VIA ORMEA 66, 10125 TORINO
CENTRO INFORMATICA - VIA ZNOJMO 41, 50065 PONTASSEVE (FI)
CHOPIN INFORMATICA - VIA CHOPIN 28, 00144 ROMA
CIPOLLA ANTONIO - VIA V. VENETO 26, 55100 LUCCA
COMPUTER SHOP - VIA A. DA BRESCIA 2, 21013 GALLARATE (VA)
COMPUSERVICE - VIA SIRACUSA 30 A, 90141 PALERMO
EASY COMPUTER - VIA LAGOMAGGIO 50, 47037 RIMINI (FO)
EASY DATA - VIA A. OMODEO 21/29, 00179 ROMA
EUROSOFT - VIA DEL ROMITO 1Dr, 50134 FIRENZE
FUTURA 2 - VIA L. GAMBINI 19, 57127 LIVORNO
INTELLIGENT GAMES - VIA E. XIMENES 9A, 00197 ROMA
HARD & SOFT - VIA CARRARA 16, 05100 TERNI
HOME & PERSONAL COMPUTER - P.ZZA MELOZZO 1, 47100 FORLI
HPE INFORMATICA - VIA N. BIXIO 46 BIS, IS1, 80126 NAPOLI
LUCKY - VIA PASSERONI 2, 20135 MILANO
MAGLIOLA - VIA PORPORA 1, 10155 TORINO
OFFICE POINT - CORSO FRANCIA 92A, 10093 COLLEGNO TORINO
ORSA MAGGIORE - P.ZZA MATTEOTTI 20, 41100 MODENA
PASSI HI-FGI - VIA TRENTO NUNZI 72/74, 63023 FERMO (AP)
PCC COMPUTER HOUSE - VIA CASILINA 283A, 00176 ROMA
RVE - CORSO CAVOUR 196, 70121 BARI
SIDESTREET - VIA S. D'ACQUISTO, 831044 MONTEBELLUNA (TV)
SIMEDATA - L. ARIOSTO 3, 63100 ASCOLI PICENO
TAULINO COMPUTERS - P.ZZA CARDUCCI 13 15100 ALESSANDRIA
TRIA ELETTRONICA - VIA E. ZACCONI 28 A, 43100 PARMA
VIDEOFUTURO PRATO- V.LE MONTEGRAPPA 15, 50047 PRATO (FI)
ZUCCATO HI-FI - CORSO PALLADIO 78, 36100 VICENZA

Distribuito in Italia da:
EuroSoft, via del Romito 1 Dr
50134 FIRENZE, tel. 055-496455-474959