di Pierluigi Panunzi

Il Mouse

quarta parte

In questa quarta parte della serie di articoli riguardanti il mouse analizzeremo altre funzioni, alcune delle quali più complesse di quelle viste finora, più complesse in quanto richiedono un «maggiore studio» prima di poterle applicare nei nostri programmi: innanzitutto richiamiamo l'attenzione sull'onnipresente figura 1, che rappresenta un quadro riassuntivo delle funzioni disponibili in genere all'interno del driver MOUSE.COM (o MOUSE.SYS) fornito assieme al mouse

La funzione OA: Set Text Cursor

Prima di parlare della funzione in esame, vediamo quali tipi di «cursore» si possono usare con un mouse.

Si tratta di tre tipi di cursore, due di testo ed uno in grafica: i due relativi alla modalità testo sono uno software (programmabile) e l'altro hardware (in pratica cursore «normale»), mentre quello grafico è completamente programmabile. Di quest'ultimo ci occuperemo nel prossimo paragrafo: ora vediamo dunque i cursori nel modo testo.

Il caso più semplice si riferisce al cursore di tipo «hardware», rappresentato con una matrice in genere di 8×14 pixel, organizzati come 14 linee di pixel (dette «scan lines») che possono essere accese in svariati modi: il cursore usuale del DOS (in pratica una sottolineatura lampeggiante) è formato da due scan line, la tredicesima e la quattordicesima, mentre per ottenere un cursore «pieno» (un rettangolo lampeggiante) le scan line interessate saranno tutte e cioè dalla prima alla quattordicesima.

In particolare dunque per definire un cursore (spostabile muovendo il mouse) in modo testo, bisogna specificare la scan line d'inizio e quella di fine.

Facendo riferimento alla figura 2, dunque, in BX bisogna porre il valore «1» (hardware cursor) ed in CX e DX, rispettivamente, la scan line iniziale e quella finale: dal momento che la massima scan line dipende in genere dalla scheda grafica utilizzata, consigliamo di effettuare delle semplicissime prove per vedere appunto quali sono i valori ammissibili.

In questo caso dunque il cursore che si ottiene è sempre lampeggiante e bianco intensificato, indipendentemente dal colore che si trova «sotto» al cursore.

Invece l'altro tipo di cursore in modo testo, quello software, è fino ad un certo punto programmabile, nel senso che si capirà nel prosieguo.

Per capire come funziona il cursore software, bisogna ritornare un attimo indietro per ricordare come sono codificate in memoria le informazioni relative al carattere scritto sullo schermo nonché sulla sua colorazione e sul colore dello sfondo: in particolare il colore del foreground e del background sono codificati in quello che si chiama «byte degli attributi».

In figura 3 vediamo dunque la codifica degli attributi di un qualunque carattere, formata dai seguenti bit:

— BL, «BLink», se settato indica che il carattere sarà lampeggiante

— R G B, (maiuscoli!), «Red», «Green» e «Blue», indicano la presenza o meno del rispettivo colore primario nella formazione del colore dello sfondo (background)

— IN, «INtensity», se posto ad uno indica che il carattere avrà un colore intensificato

— r g b, analoghi ad R G B, questa volta relativi alla colorazione del carattere vero e proprio.

A beneficio di chi non fosse al corrente di quali sono i 16 colori che si possono usare in modo testo, abbiamo riportato in tabella 2 l'elenco delle possibili combinazioni, distinguendo il caso di colori intensificati o meno.

Dalla codifica del byte di attributi si vede dunque che i colori possibili per il carattere stesso sono 16, mentre per lo sfondo si riducono ad 8, essendo stato stabilito che il bit più significativo si

AX = 0H	Mouse Reset
AX = 1H	Cursor Enable
AX = 2H	Cursor Disable
AX = 3H	Get Mouse Position and Button Status
AX = 4H	Set Mouse Position
AX = 5H	Get Button Press Information
AX = 6H	Get Button Release Information
AX = 7H	Set min & max horizontal position
AX = 8H	Set min & max vertical position
AX = 9H	Set Graphic Cursor Block
AX = 0AH	Set Text Cursor
AX = 0BH	Read Motion Counters
AX = 0CH	Set User-defined subroutine
AX = 0DH	Enable Light Pen Emulation
AX = 0EH	Disable Light Pen Emulation
AX = 0FH	Set Mickey/Pixel Ratio
AX = 10H	Window Conditional Off
AX = 12H	Set Large Graphic Cursor
AX = 13H	Set Speed Threshold

Tabella 1 - Per comodità di consultazione, abbiamo ancora una volta riportato un quadro sinottico delle funzioni gestibili in un mouse per mezzo dell'INT 33H: ad eccezione della funzione 12H, si tratta di funzioni standard implementate nel file MOUSE.COM fornito assieme al mouse.

riferisce alla possibilità o meno di visualizzare un carattere lampeggiante.

Fin qui per quel che riguarda la colorazione dei caratteri nello schermo quando lavoriamo in modo testo, ad esempio con un word processor: il cursore di tipo «software» a disposizione con un mouse può dunque essere programmato in modo da assumere una colorazione differente a seconda del colore dello sfondo, in modo tale che risulti sempre e comunque distinguibile dallo sfondo stesso. Un cursore blu su di uno sfondo blu non è utile a nessuno...

Ecco che dunque si possono gestire colori del carattere e dello sfondo nel punto in cui si trova il cursore mosso con il mouse: il meccanismo è abbastanza semplice, ma non del tutto intuitivo. Tra l'altro lo ritroveremo anche nel caso della gestione del cursore grafico.

La funzione in esame prevede (oltre al valore 0 da porre nel registro BX, che indica dunque il «software cursor») due valori a 16 bit da porre nei registri CX e DX, valori che prendono il nome di «screen mask» e «cursor mask», in quanto rappresentano delle maschere logiche da applicare al byte di attributi, nonché al byte che contiene la codifica ASCII del carattere interessato. In particolare la funzione in esame prende la word formata dal byte di attributo e dalla codifica ASCII (rispettivamente come parte più significativa o meno significativa della word) e la pone in AND con la «screen mask» ed il risultato lo pone successivamente in XOR con la «cursor mask».

Per capire meglio forniamo subito un esempio, supponendo di avere in una certa locazione dello schermo video un asterisco («*», di codice ASCII esadecimale 2AH) verde chiaro su sfondo magenta e per giunta lampeggiante: il byte di attributi sarà espresso in binario, dato da:

«**1** (=lampeggiante) **101** (=magenta) **1010** (=verde chiaro)»

e perciò la word considerata dalla funzione in esame sarà data dal valore **DA2AH**, dove appunto «DAH» è il byte di attributi e «2AH» è il codice ASCII.

Supponiamo ora che la screen mask valga **77FFH** e che la cursor mask valga **FF00H**: l'AND tra DA2AH e 77FFH fornisce il valore **522AH** che posto poi in XOR con FF00H dà un valore finale pari a **AD2AH**: analizzando i bit che compongono la parte più significativa (ADH) si vede che si tratterà di un carattere «*» (in quanto il 2AH meno significativo è rimasto inalterato) di colore magenta su sfondo verde, per di più blinkante... Ecco che perciò con questa coppia di «mask», i colori dello sfondo e del carattere risultano scambiati, con in più le seguenti differenze:

— se il colore del carattere era intenso allora il risultato sarà un carattere lampeggiante

— se il carattere era lampeggiante allora diventerà di colore intensificato.

Effetti che vanno tra il divertente ed il disgustoso si ottengono ponendo per le due «mask» dei valori a caso che interessino soprattutto la codifica ASCII del carattere: in particolare nell'esempio di cui sopra la prima maschera per il carattere ASCII era FFH e perciò con l'AND lasciava inalterata la codifica, come pure accadeva facendo lo XOR con un valore nullo.

Viceversa ponendo valori differenti da FFH e da 00H per le due maschere (nei byte meno significativi!!!) ecco che verrà modificato anche il codice ASCII del carattere evidenziato dal cursore del mouse.

Invitiamo i lettori a provare il semplice programmino di figura 4, cambiando di volta in volta i valori esadecimali da porre nei registri CX e DX, magari dopo aver aggiunto altre scritte di colori differenti e con diversi sfondi.

AX = 0AH	Set Text Cursor
INPUT	BX = cursor selection CX = screen mask or scan line start DX = cursor mask or scan line end
OUTPUT	-

Figura 2 - Questa funzione serve per selezionare il tipo di cursore da usare nei modi di testo, a scelta tra quello hardware (quello usuale, bianco lampeggiante) e software, programmabile e di colore differente a seconda dello sfondo in cui si muove.

7	6	5	4	3	2	1	0
BL	R	G	B	IN	r	g	b

Figura 3 - Formato del byte degli attributi, che codifica il colore di un carattere e dello sfondo quando si lavora in modo testo.

```
uses crt,dos;
var reg : registers;
    gd,gm,i : integer;

procedure mouse(ax : word);
begin
 reg.AX := ax;
 intr($33,reg);
end;

begin
 textcolor(blink + lightgreen);
 textbackground(magenta);
 gotoxy(10,10);
 writeln('****************');
 mouse(0);
 with reg do
  begin
   BX := 0;
   CX := $77ff;
   DX := $ff00;
  end;
 mouse(10);
 mouse(1);
 repeat
 until keypressed;
 mouse(2);
end.
```

Figura 4 - Programmino di prova per saggiare la potenza e le caratteristiche del «software cursor» mosso dal mouse: per provare vari effetti consigliamo di aggiungere altre scritte sullo schermo, di colori e sfondi via via differenti. In alcuni casi i risultati saranno imprevedibili, specialmente se si va a modificare anche la codifica ASCII del carattere sul quale si muove il cursore.

La funzione 9: Set Graphic Cursor

Questa funzione, come già si può capire dal nome, serve a settare la forma del cursore nei modi grafici: in particolare il cursore grafico è costituito da un insieme di pixel accesi e spenti posti all'interno di un quadrato di 16×16 pixel ed analogamente al caso precedente, viene disegnato a partire da due maschere (una **screen mask** ed una **cursor mask**), stavolta formate ognuna da 16 word (infatti una word possiede 16 bit...).

In particolare il cursore in esame viene definito con 16 righe ognuna larga 16 pixel, come detto: vediamo innanzitutto, come primo esempio, come è definita la «freccetta» che per default compare lavorando in grafica.

In figura 6 abbiamo rappresentato questa freccetta in tre modi:
— sulla sinistra vediamo come essa appare in termini di pixel accesi («*») e spenti («.»)
— in centro è riportata la **screen mask**, espressa in binario
— a destra è riportata la **cursor mask**, sempre espressa in binario.

Vediamo subito che la freccetta, nella screen mask, è leggermente più grande (un pixel in più nel contorno) di quella nella cursor mask (che coincide con la rappresentazione a sinistra): dal momento che anche nel caso del cursore grafico viene effettuata l'operazione di AND seguita dallo XOR, si può vedere come la freccetta sarà sempre presente e visibile (infatti agli «uni»; della cursor mask corrispondono degli «zeri» nella screen mask) ed inoltre avrà sempre tutt'intorno a sé un pixel comunque spento (si controlli infatti nella screen mask la posizione degli «zeri»).

```
+
.............   0011111111111111   0000000000000000
.*...........   0001111111111111   0100000000000000
.**..........   0000111111111111   0110000000000000
.***.........   0000011111111111   0111000000000000
.****........   0000001111111111   0111100000000000
.*****.......   0000000111111111   0111110000000000
.******......   0000000011111111   0111111000000000
.*******.....   0000000001111111   0111111100000000
.********....   0000000000111111   0111111110000000
.*********...   0000000000011111   0111111111000000
.*****.......   0000000111111111   0111110000000000
.*...**......   0001000011111111   0100011000000000
.....**......   0011000011111111   0000011000000000
......**.....   1111100001111111   0000001100000000
......**.....   1111100001111111   0000001100000000
.......**....   1111110000111111   0000000110000000
```

Figura 6 - Rappresentazione schematica della freccetta che è genericamente visibile per default «accendendo» il cursore in uno schermo gestito in modalità grafica: a sinistra sono rappresentati con «» i pixel accesi, mentre al centro e a destra sono indicate rispettivamente la screen mask e la cursor mask. Il «+» indica il pixel «puntato» dalla freccia stessa.*

AX = 9H	Set Graphic Cursor Block
INPUT	BX = cursor x hot spot CX = cursor y hot spot ES:DX = pointer to screen/cursor mask
OUTPUT	-

Figura 5 - Questa funzione serve per definire un cursore nei modi grafici.

Tutto questo si può vedere più facilmente (e lo stesso discorso vale esattamente per il cursore in modo testo) considerando la tabellina di verità che abbiamo posto in tabella 3: in essa abbiamo riportato tutte e quattro le combinazioni possibili di «0» ed «1» relative all'esecuzione di un AND seguita da uno XOR.

Si può vedere così in un sol colpo d'occhio che laddove nella screen mask appare uno «0» allora il pixel risultante sarà pari a quello presente nella cursor mask, mentre laddove nella screen mask appare un «1» (è generalmente il caso più interessante), allora il pixel risultante rimarrà inalterato, se nella cursor mask c'è uno «0», mentre verrà **invertito** se nella cursor mask c'è un «1».

Quest'ultimo caso è proprio quello del cursore che rimane visibile comunque (avendo istante per istante sempre il colore inverso dei pixel sui quali si muove), senza «alterare» il disegno sottostante: il caso invece della freccetta consente di far comunque vedere la freccia bianca qualsiasi sia la colorazione dei pixel attraverso i quali si muove.

C'è da dire anche che il valore «0» ed il valore «1» nella colonna «bit risultante» in generale significano «pixel di colore nero (spento)» e «pixel di colore bianco intenso»: ciò è dovuto alla particolare configurazione hardware delle schede grafiche, sulla quale torneremo ben presto (N.d.r.: è una promessa!).

Visto dunque come viene definito il cursore grafico, vediamo ora un'altra caratteristica del cursore grafico, dopodiché vedremo come si usa la funzione 9 in esame.

La caratteristica di cui sopra è quella che viene definita in gergo «hot spot» è rappresentata convenzionalmente il punto dello schermo video «puntato» dal mouse: la freccetta di cui abbiamo più volte parlato, «punta» al pixel indicato con un «+» nella figura 6 ed infatti istante per istante le coordinate del mouse fornite dalle funzioni che ben conosciamo saranno proprio quelle dell'hot spot.

Se ad esempio decidiamo di tracciare un cursore a forma di crocetta, ecco

Figura 7 - Programma di prova che consente di creare come cursore grafico una crocetta di colore sempre invertito rispetto allo sfondo: cambiando le prime 8 word del vettore «v» con i valori posti tra le graffe è possibile creare un'altra crocetta che si staglierà comunque nello schermo, indipendentemente dal colore dei pixel attraversati.

```
uses graph,crt,dos;
const v : array[0..31] of word =
(
      ($fc7f,$fc7f,$fc7f,$e00f,$e00f,$e00f,$fc7f,$fc7f,
       $fc7f,$ffff,$ffff,$ffff,$ffff,$ffff,$ffff,$ffff,
);
      ($ffff,$ffff,$ffff,$ffff,$ffff,$ffff,$ffff,$ffff,
       $ffff,$ffff,$ffff,$ffff,$ffff,$ffff,$ffff,$ffff,
       $0000,$0100,$0100,$0100,$0fe0,$0100,$0100,$0100,
       $0000,$0000,$0000,$0000,$0000,$0000,$0000,$0000);
var reg : registers;
    gd,gm,i,x,y,c : integer;

procedure mouse(ax : word);
begin
 reg.AX := ax;
 intr($33,reg);
end;

begin
 gd := detect;
 initgraph(gd,gm,'');
 setlinestyle(solidln,0,thickwidth);
 for i := 0 to 15 do
  begin
   setcolor(i);
   rectangle(i * 20 + 40,200,i * 20 + 50,210);
   setcolor(15 - i);
   rectangle(i * 20 + 40,240,i * 20 + 50,250);
  end;
 mouse(0);
 mouse(1);
 with reg do
  begin
   BX := 7;
   CX := 4;
   ES := seg(v);
   DX := ofs(v);
  end;
 mouse(9);
 repeat
 until keypressed;
 mouse(2);
end.
```

R G B oppure r g b	colore normale	colore intensificato
0 0 0	nero	grigio scuro
0 0 1	blu	blu chiaro
0 1 0	verde	verde chiaro
0 1 1	ciano	ciano chiaro
1 0 0	rosso	rosso chiaro
1 0 1	magenta	magenta chiaro
1 1 0	marrone	giallo
1 1 1	grigio chiaro	bianco

Tabella 2 - Codifica dei vari colori disponibili per lo sfondo (solo la colonna «colore normale») e per un carattere (entrambe le colonne «colore normale» e «colore intensificato») nei modi di testo.

bit nella screen mask	bit nella cursor mask	bit risultante
0	0	0
0	1	1
1	0	stesso colore
1	1	colore invertito

Tabella 3 - Tabella di verità che consente di vedere subito l'effetto di un AND e di uno XOR (rispettivamente con la screen mask e con la cursor mask) sui pixel dello schermo video.

che l'**hot spot** deve trovarsi necessariamente nel punto di intersezione dei due tratti verticali ed orizzontali che compongono la crocetta.

Ecco che dunque la posizione dell'hot spot si indicherà in modo relativo (con valori compresi tra −16 e 16), rispetto al pixel posto in alto a sinistra della matrice 16×16: nel caso della freccetta di default l'hot spot ha coordinate (−1, −1) ed infatti è posto proprio un pixel verso l'alto ed uno verso sinistra rispetto al primo pixel della matrice.

Un hot spot di coordinate (15, 15) corrisponde esattamente al pixel posto in basso a destra nella matrice di pixel.

Tornando dunque alla funzione 9, ecco che nei registri BX e CX andranno le coordinate (tra −16 e 16 dell'hot spot (rispettivamente la **x** e la **y**), mentre nella coppia di registri ES:DX bisognerà porre rispettivamente il segmento e l'offset della locazione a partire della quale è posta in memoria la screen mask, che deve essere subito seguita dalla cursor mask, pena malfunzionamenti nella definizione del cursore stesso.

In figura 7 vediamo infine un programma di esempio nel quale definiamo come cursore una crocetta sempre visibile perché comunque di colore invertito rispetto a quello dei pixel sui quali si muove: cambiando poi le prime 8 word del vettore «v» con le 8 word che abbiamo posto tra parentesi graffe, si otterrà invece una crocetta comunque bianca e che si staglia nettamente rispetto allo sfondo in quanto ha tutt'intorno una «cortina di pixel di sbarramento». Lasciamo ai lettori la verifica che le due «mask», rappresentate dal vettore «v» corrispondono effettivamente a quanto richiesto.

Con questo abbiamo terminato e diamo l'appuntamento alla prossima puntata. MC