


Determinare il colore di un pixel sullo schermo


http://www.vbsimple.net/howto/ht_035.htm

Difficoltà:  2 / 5

Gli schermi dei nostri computer si fanno sempre più colorati. Come fare per determinare il colore di un punto sullo schermo? In questo How-To estrarremo il colore del pixel indicato dal cursore del mouse.

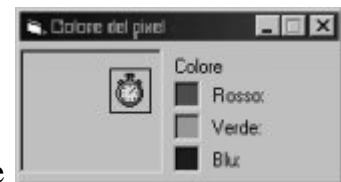
Il progetto si compone di un solo form e poche righe di codice.

Inseriamo nel nostro form una *PictureBox*  di nome

ColorePixel, un *Timer*  di nome **Tempo**, con la proprietà

Enabled impostata a True ed *Interval* a 100, una *Label* **A** di nome

Colore ed altre tre *Label* in una [matrice](#) di nome **NumeriRGB** con i loro indici rispettivamente di 0, 1 e 2. Abbiamo voluto aggiungere anche altre tre *PictureBox* colorate con i tre colori fondamentali: rosso, verde e blu, ma esse non serviranno per i nostri scopi.




Il funzionamento è molto semplice: mediante una funzione API otteniamo le coordinate del punto in cui è posizionato il mouse e mediante un'altra funzione leggiamo il colore del pixel in quel punto. Fatto questo scomponiamo il numero ottenuto nella tripletta [RGB](#) ed estraiamo i valori dei singoli colori.

Tali operazioni saranno ripetute ogni decimo di secondo per assicurare l'aggiornamento dei colori estratti. Vediamo il codice:

```

1. Option Explicit
2.
3. Private Type POINTAPI
4.     x As Long
5.     y As Long
6. End Type
7.
8. Private Declare Function GetCursorPos Lib "USER32" (lpPoint As POINTAPI) As Long
9. Private Declare Function GetDC Lib "USER32" (ByVal hwnd As Long) As Long
10. Private Declare Function ReleaseDC Lib "USER32" (ByVal hwnd As Long, ByVal hdc As Long) As Long
11. Private Declare Function GetPixel Lib "GDI32" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long
12.
13. Private ScreenDC As Long
14. Private Coord As POINTAPI
15.

```

Alle righe 3-6 definiamo un nuovo tipo di dati  di nome **POINTAPI**. È un tipo di dati API e viene richiesto da svariate funzioni grafiche.

Alla riga 8 abbiamo la dichiarazione della funzione API **GetCursorPos** che restituisce le coordinate del cursore del mouse. Richiede appunto il tipo di dati **POINTAPI**.

Le righe 9 e 10 contengono le dichiarazioni delle funzioni **GetDC** e **ReleaseDC**. La prima serve per ottenere il [DC \(Device Context\)](#) di una finestra, mentre la seconda serve per

rilasciare ([deallocare](#)) un DC.

La riga 11 include la dichiarazione della funzione API **GetPixel** che restituisce in uscita il colore del pixel indicato alle coordinate specificate. La funzione richiede anche il DC cui si fa riferimento.

Alle righe 13 e 14 definiamo due variabili che utilizzeremo più volte all'interno del nostro progetto. La prima è **ScreenDC**, ovvero l'handle del DC dello schermo; ciò significa che il punto di origine sarà l'angolo in alto a sinistra dello schermo; se avessimo scelto un altro DC l'origine sarebbe stato l'angolo in alto a sinistra della finestra cui appartiene il DC. Il Device Context dello schermo contiene tutto ciò che è visibile all'utente, ma non contiene la grafica che al momento è nascosta.

La seconda variabile è **Coord**, di tipo **POINTAPI**, che utilizzeremo per contenere le coordinate del cursore del mouse.

```
16. Private Sub Form_Load()  
17.     ScreenDC = GetDC(0)  
18. End Sub  
19.  
20. Private Sub Form_Unload(Cancel As Integer)  
21.     Call ReleaseDC(0, ScreenDC)  
22. End Sub  
23.
```

All'avvio del programma estrarremo il DC dello schermo tramite la funzione **GetDC**. La finestra 0 denota lo schermo. Memorizzeremo tale DC nella variabile **ScreenDC** (riga 17).

Altresì alla chiusura del programma rilasceremo tale [handle](#) DC, tramite la funzione **ReleaseDC** (riga 21). Il mancato rilascio di un DC comporta il blocco di un handle e alla lunga anche il blocco del sistema.

```
24. Private Sub Tempo_Timer()  
25.     Dim ColoreEstratto As Long  
26.     Call GetCursorPos(Coord)  
27.     ColoreEstratto = GetPixel(ScreenDC, Coord.x, Coord.y)  
28.     ColorePixel.BackColor = ColoreEstratto  
29.     Colore.Caption = Right("000000" & Hex(ColoreEstratto), 6)  
30.     NumeriRGB(0).Caption = "Rosso: " & Mid(Colore.Caption, 5, 2)  
31.     NumeriRGB(1).Caption = "Verde: " & Mid(Colore.Caption, 3, 2)  
32.     NumeriRGB(2).Caption = "Blu: " & Mid(Colore.Caption, 1, 2)  
33. End Sub
```

Tutto il codice principale è contenuto nella funzione legata all'evento **Timer** ⚡ di **Tempo**.

Alla riga 25 dichiariamo la variabile **ColoreEstratto** che utilizzeremo per memorizzare il valore del colore ottenuto.

Alla riga 26 reperiamo le coordinate del cursore tramite la funzione **GetCursorPos**. Possiamo adesso estrarre il colore del pixel alle coordinate indicate nella struttura **Coord**. Utilizzeremo allora la funzione **GetPixel**: essa richiede l'handle del DC per specificare l'origine dei punti e le due coordinate. Il risultato della chiamata alla funzione sarà memorizzato nella variabile **ColoreEstratto**.

Ottenuto il colore possiamo procedere alla sua scissione. Alla riga 28 mostriamo dentro la **PictureBox ColorePixel** il colore del pixel estratto, impostando per essa la proprietà

BackColor.

Alla riga 29 convertiamo il colore estratto in esadecimale tramite funzione Hex. L'operazione di concatenazione usata assieme alla funzione Right è una soluzione molto comune per assicurarsi che il risultato della conversione in esadecimale sia esattamente 6 caratteri. Il risultato sarà mostrato nella Label **Colore**.

La scomposizione del colore in tripletta RGB sarà effettuato mediante estrazione di coppie di caratteri dalla Label **Colore**. La prima coppia indicherà il blu, la seconda il verde e la terza il rosso (righe 30-32).

Avremmo potuto adottare mille soluzioni per effettuare quest'operazione ma questa sembra essere una delle più semplici. Una soluzione per la scomposizione si trova nell'[How-To dedicato alla gestione dei colori RGB](#) ed in quello dedicato all'[estrazione dei singoli bytes da un dato di tipo multibyte](#).

La prova del programma è semplicissima, infatti il programma farà tutto da solo tramite il *Timer*. Lanciamo il programma, muoviamo il mouse sullo schermo e vedremo nella *PictureBox* il colore del punto che stiamo indicando.



Il progetto è molto semplice e non pone alcun rischio particolare. È anche il principio di base per l'[How-To dedicato all'ingrandimento o riduzione di una parte dello schermo](#).

[Fibia FBI](#)

17 Maggio 2001



[Torna all'indice degli HowTo](#)
