

Trascinamento OLE

http://www.vbsimple.net/howto/ht_042.htm

Difficoltà: 3 / 5

Una delle caratteristiche più apprezzate dei sistemi Windows è il [trascinamento OLE](#) tra oggetti di diverso genere: files, testi ed immagini possono essere trasferite, copiate da una posizione ad un'altra con poche semplici operazioni. Il termine trascinamento OLE indica il trasferimento mediante trascinamento del **contenuto di un controllo** e non il trascinamento del controllo stesso; per trascinare un controllo sulla superficie del form vedi l'[HowTo dedicato allo spostamento di un controllo a runtime](#).

Quasi tutti i controlli di Visual Basic sono in grado di eseguire operazioni di trascinamento OLE, alcuni dei quali, se desiderato, in maniera automatica. Il progetto che svilupperemo potrebbe sembrare parecchio complesso ma non è affatto così. Per evitare ripetizioni, dopo aver visto il funzionamento del sistema di trascinamento OLE, racchiuderemo le operazioni interessate a più parti del codice in tre Sub separate.

Cominciamo disponendo sulla superficie del nostro form i seguenti controlli: nell'alto in angolo a sinistra un controllo *Image* di nome **imgFreccia1** contenente l'immagine della freccia verso destra mostrata nella figura a fianco; nell'angolo opposto abbiamo una seconda *Image* di nome **imgFreccia2** con l'immagine della freccia verso sinistra.

Entrambe serviranno da indicatori durante l'utilizzo del programma.

Tra le due frecce abbiamo 4 controlli disposti verticalmente uno sopra l'altro: il primo di questi è un controllo *Image* di nome **Image1** contenente l'immagine del cane *Soichiro* ^_^ e con la proprietà *BorderStyle* impostata su **1 - Fixed Single**; nel nostro esempio questa proprietà non è strettamente necessaria ma sarà utile quando in seguito amplieremo il discorso con le operazioni di trascinamento automatico.



Subito sotto il cane Soichiro abbiamo una *Label* di nome **Label1**, un *Frame* di nome **Frame1** ed una *ListBox* di nome (indovina un po') **List1**. Abbiamo preferito non assegnare un nome significativo a questi controlli per evitare confusioni con le altre parti del codice; i quattro controlli con nome predefinito saranno quelli interessati nelle operazioni di trascinamento OLE. Tutti e quattro questi controlli (**Image1**, **Label1**, **Frame1** e **List1**) avranno la proprietà *OLEDropMode* impostata su **1 - Manual**. Ciò significa che sarà compito del programmatore occuparsi delle operazioni di trascinamento e pertanto non saranno gestite automaticamente e verranno generati degli eventi in occasione delle operazioni di trascinamento.

Sotto la *ListBox* avremo una *Label* descrittiva di nome **lblEffetto** e due *OptionButton*  di nome **optEffetto** in una [matrice](#) di controlli, quindi con indice 0 per quello con *Caption* "Copia" ed indice 1 per quello con *Caption* "Sposta".

In fondo al form un'altra *Label* descrittiva di nome **lblGestioneAutomatica**, una *TextBox*  di nome **txtAutomatica** ed una *PictureBox*  di nome **picAutomatica**. Queste ultime due verranno utilizzate per il trascinamento OLE automatico e pertanto avranno la proprietà *OLEDropMode* impostata su **2 - Automatic**.

Prima di vedere il codice spiegheremo il funzionamento generale del procedimento: per le operazioni di trascinamento manuale il programma in una situazione particolare, tipicamente la pressione di un pulsante del mouse sopra un controllo lancia l'esecuzione del metodo  **OLEDrag** dell'oggetto di cui si richiede il trascinamento OLE. In risposta l'oggetto utilizzato per il trascinamento lancia l'evento  **OLEStartDrag** fornendo come argomenti un oggetto *DataObject* di nome **Data** ed un numero di nome **AllowedEffects**.

Il primo di questi servirà come contenitore dei dati da trasferire in cui ogni dato è identificato da un formato; il valore del formato sarà un numero dell'enumerazione  [ClipboardConstants](#). L'oggetto *DataObject* potrà essere riempito tramite il metodo **SetData** fornendo il valore che assumerà tale dato ed il numero del formato. Sarebbe anche possibile non fornire immediatamente il valore del dato ma solo il suo formato: in tal caso al momento della lettura dei dati sarà lanciato l'evento **OLESetData** per richiedere il completamento di tali dati. Tralasciamo per il momento questo caso limite ed assumiamo che il programma fornisca sempre il valore per ogni dato.

L'altro argomento passato all'evento **OLEStartDrag** sarà **AllowedEffects** e dovrà essere un numero dell'enumerazione [OLEDropEffectConstants](#). Esso determinerà innanzitutto la forma del puntatore durante il trascinamento ed il comportamento dei dati al termine del rilascio. I tre valori principali dell'enumerazione sono:

- $vbDropEffectNone = 0$ 
Indica che il trascinamento non verrà effettuato.
- $vbDropEffectCopy = 1$ 
Indica un'operazione di copiatura dei dati che verranno trascinati; al termine dell'operazione i dati originali non dovrebbero essere rimossi.
- $vbDropEffectMove = 2$ 
Indica un'operazione di spostamento dei dati trascinati. Al termine dell'operazione i dati originali dovrebbero essere rimossi dalla locazione originale.

La guida in linea di Visual Basic raccomanda l'utilizzo dei suddetti valori in combinazione di bit sia durante la composizione del valore (tramite operatore OR) che durante la lettura dei dati in esso contenuti (tramite operatore AND); vedi anche le [Informazioni aggiuntive sull'estrazione bit a bit](#).

Personalmente non sono d'accordo ma utilizzeremo comunque tale metodologia. 

L'operazione di trascinamento sarà effettuata soltanto se entrambi gli argomenti verranno completati: riempito il contenitore **Data** tramite `SetData` ed impostato il valore **AllowedEffects**. La fase di trascinamento termina quando l'utente rilascia il pulsante del mouse oppure quando scatta qualche evento codice che ne interrompe l'esecuzione.

Durante il trascinamento se l'oggetto che si trova sotto il puntatore ha la proprietà *OLEDropMode* impostata su **1 - Manual** verrà lanciato l'evento **OLEDragOver** ad indicare che è in atto un trascinamento sopra l'oggetto cui si riferisce l'evento. Se il trascinamento sopra tale oggetto non è voluto, ad esempio per la differenza dei dati trattati oppure perché si tratta dello stesso oggetto che fornisce i dati, è possibile disabilitare temporaneamente il trascinamento, senza tuttavia interrompere l'intera fase di trascinamento, impostando l'argomento *Effect* su **vbDropEffectNone**. Ciò renderà inutile il rilascio del pulsante del mouse sul controllo e l'evento successivo che conferma il rilascio non verrà generato.

Le ultime due fasi dell'operazione di trascinamento sono identificate dal rilascio effettivo con successo del pulsante del mouse. Il controllo su cui è effettuato il rilascio lancerà l'evento **OLEDragDrop** per richiedere il trasferimento dei dati. Anche in questo caso saranno forniti gli argomenti **Data** di tipo *DataObject* ed **Effect** per indicare il tipo di trascinamento. A parte questi due argomenti l'evento presenta gli stessi dati dell'evento **MouseMove**. All'interno dell'evento sarà necessario controllare la presenza di dati idonei al tipo di controllo e l'eventuale recupero tramite il metodo **GetData**.

L'ultimo evento che verrà generato sarà **OLECompleteDrag** relativo all'oggetto da cui inizia il trascinamento e che fornisce i dati. La sua funzione è esclusivamente quella di permettere al controllo originale la rimozione dei dati trasferiti mediante operazione (Effetto) di spostamento.

Vediamo il codice del progetto alla luce di quanto detto finora cominciando dal primo controllo trascinabile, `Image1`:

```
1. Option Explicit
2.
3. Private Sub Image1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
4.     If Button = 1 Then Image1.OLEDrag
5. End Sub
6.
7. Private Sub Image1_OLEStartDrag(Data As DataObject, AllowedEffects As Long)
8.     AttivaTrascinamento Image1, Data, AllowedEffects
9.     If Image1.Picture.Handle <> 0 Then Data.SetData Image1.Picture, vbCFBitmap
10. End Sub
11.
```

Nel momento in cui l'utente preme un tasto del mouse scatta il normale evento `MouseDown`. Controlleremo pertanto che il pulsante premuto sia il sinistro ed in tal caso avvieremo l'operazione di trascinamento sul controllo `Image1` (riga 4).

In funzione del controllo che esegue il metodo `OLEDrag` (riga 4) scatta l'evento `OLEStartDrag` per inizializzare l'operazione di trascinamento. All'interno di questo lanceremo la Sub **AttivaTrascinamento** che si occuperà di fornire dati validi per gli argomenti `Data` ed `AllowedEffects`, che vedremo più avanti. Ci basti sapere per ora che essa fornirà al contenitore `Data` una stringa.

Alla riga 9 oltre ai dati forniti dalla Sub **AttivaTrascinamento** abbiamo aggiunto un ulteriore dato al contenitore **Data**. Essendo **Image1** un controllo grafico diamo la possibilità di trasferire l'immagine in esso contenuta. Basterà controllare che vi sia un'immagine e richiamare il metodo **SetData** a cui forniremo il dato ovvero la proprietà **Picture** dell'oggetto **Image1** e come formato dei dati specificheremo che si tratta di una **Bitmap**.

```

12. Private Sub Image1_OLEDragOver(Data As DataObject, Effect As Long, Button As
    Integer, Shift As Integer, X As Single, Y As Single, State As Integer)
13.     TrascinamentoSu Image1, Data, Effect
14. End Sub
15.
16. Private Sub Image1_OLEDragDrop(Data As DataObject, Effect As Long, Button As
    Integer, Shift As Integer, X As Single, Y As Single)
17.     RilasciaTrascinamento Image1, Data, Effect
18. End Sub
19.
20. Private Sub Image1_OLECompleteDrag(Effect As Long)
21.     If (Effect And vbDropEffectMove) = vbDropEffectMove Then Set Image1.Picture =
    Nothing
22. End Sub
23.
24. Private Sub Image1_DblClick()
25.     Set Image1.Picture = LoadPicture(App.Path & "\soichiro.bmp")
26. End Sub
27.

```

Il funzionamento dell'evento **OLEDragOver** sarà controllato dalla Sub **TrascinamentoSu** che fondamentalemente impedirà il rilascio sullo stesso controllo che ha fornito i dati. Non che sia proibito farlo ma chi avrebbe interesse a trasferire dei dati da un controllo allo stesso controllo?

In maniera analoga, l'evento **OLEDragDrop** sarà controllato dalla Sub **RilasciaTrascinamento** che si occuperà di mostrare una **MsgBox** con i dati stringa e di mostrare le eventuali immagini **Bitmap** contenuti nell'oggetto **DataObject**.

Il completamento del trascinamento è identificato dal richiamo dell'evento **OLECompleteDrag** sul controllo che ha fornito i dati. Nel caso dell'immagine sarà possibile eliminare l'originale a seguito di un'operazione di spostamento dei dati.

La funzione alle righe 24-26 consentono di ricaricare l'immagine del cane **Soichiro** nel caso essa venga eliminata perché richiesto lo spostamento dei dati.

Sarà omesso il codice relativo agli altri 3 controlli trascinabili (**Label1**, **Frame1** e **List1**) perché del tutto identici a quello appena visto, eccetto per le righe 9, 21 e 25 in quanto essi non tratteranno l'uso di immagini.

```

28. Private Sub AttivaTrascinamento(ByVal Controllo As Control, ByRef Data As
    DataObject, ByRef AllowedEffects As Long)
29.     Data.SetData Controllo.Name, vbCFText
30.     If optEffetto(0).Value = True Then
31.         AllowedEffects = AllowedEffects Or vbDropEffectCopy
32.     Else
33.         AllowedEffects = AllowedEffects Or vbDropEffectMove
34.     End If
35.     imgFreccial.Move imgFreccial.Left, Controllo.Top
36. End Sub

```

37.

Avevamo accennato che la funzione **AttivaTrascinamento** fornirà al contenitore **Data** una stringa e tale operazione sarà svolta dalla riga 29. Il dato che sarà fornito a scopo di esempio sarà il nome del controllo che effettua il trascinamento.

In funzione del pulsante di opzione scelto sarà impostato l'effetto desiderato, copia o sposta (righe 30-34).

Alla riga 35 viene infine posizionata la prima freccia **imgFreccia1** in corrispondenza del controllo che ha fornito i dati. Questo indicatore punterà l'origine dei dati.

```

38. Private Sub TrascinamentoSu(ByVal Controllo As Control, ByVal Data As DataObject,
    Effect As Long)
39.     If Data.GetFormat(vbCFText) Then
40.         If Data.GetData(vbCFText) = Controllo.Name Then Effect = vbDropEffectNone
41.     End If
42.     If imgFreccia2.Top <> Controllo.Top Then imgFreccia2.Top = Controllo.Top
43. End Sub
44.

```

La Sub **TrascinamentoSu** effettua una serie di controlli sui dati trasferiti. Infatti se il trascinamento viene effettuato sullo stesso controllo che ha fornito i dati disabilitiamo il rilascio semplicemente impostando l'effetto su **vbDropEffectNone** (riga 40).

Prima di richiedere la lettura dei dati trasferiti è opportuno verificare che sia presente il formato di dati che intendiamo leggere, nel nostro esempio il formato stringa corrispondente al valore **vbCFText**. Tale controllo è effettuato utilizzando il metodo **GetFormat** e restituisce un valore booleano corrispondente alla presenza di dati nel formato richiesto (riga 39).

Avendo la sicurezza che i dati trasferiti siano del formato corretto effettueremo la lettura tramite il metodo **GetData** e nel caso che tali dati corrispondano al nome del controllo in esame bloccheremo temporaneamente il rilascio (riga 40).

Infine sposteremo la seconda freccia **imgFreccia2** in corrispondenza del controllo su cui i dati sono trascinati. Questo secondo indicatore punterà pertanto la destinazione dei dati.

```

45. Private Sub RilasciaTrascinamento(ByVal Controllo As Control, ByVal Data As
    DataObject, ByVal Effect As Long)
46.     Dim Messaggio As String
47.     If Data.GetFormat(vbCFText) Then
48.         If (Effect And vbDropEffectCopy) = vbDropEffectCopy Then
49.             Messaggio = "Copia "
50.         Else
51.             Messaggio = "Sposta "
52.         End If
53.     Messaggio = Messaggio & Data.GetData(vbCFText)
54.     Messaggio = Messaggio & " su " & Controllo.Name
55.     MsgBox Messaggio, vbInformation Or vbOKOnly, "Trascinamento OLE"
56. End If
57. End Sub
58.

```

L'ultima Sub creata sarà quella dedicata alla fase di rilascio dei dati sopra un controllo: **RilasciaTrascinamento** effettua il controllo della presenza dei dati nel formato corretto (riga 47) ed in tal caso mostra all'utente un messaggio.

Il messaggio verrà costruito in base all'effetto richiesto (righe 48-54) e saranno recuperati i dati trasferiti tramite il metodo **GetData**. Tale messaggio sarà mostrato mediante una **MsgBox** alla riga 55.

Lanciamo il progetto, avendo cura di aver inserito anche il codice relativo agli altri controlli in maniera analoga a quanto fatto per il controllo **Image1**, e vediamo cosa succede trascinando il **Frame** sopra la **ListBox**:

Se il pulsante di opzione selezionato per la scelta dell'effetto di trascinamento è il primo dei due (**Copia**) il puntatore del mouse assumerà la forma relativa all'operazione di copia (vedi figura 2) e sarà mostrata la finestra con il messaggio "**Copia Frame1 su List1**" come mostrato nella figura 3.

La stessa operazione può essere effettuata con gli altri tre controlli per i quali è abilitato il trascinamento. I dati sul controllo originale non saranno mai modificati.



Figura 2



Figura 4

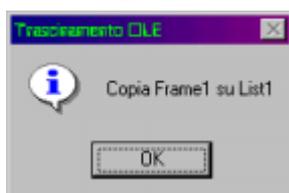


Figura 3

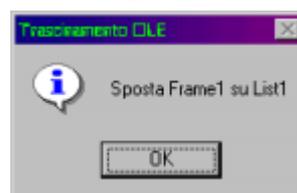


Figura 5

Se invece il pulsante di opzione scelto fosse il secondo (**Sposta**) il puntatore del mouse assumerà una forma differente ad indicare l'operazione di spostamento dei dati (vedi figura 4). Il messaggio mostrato nella **MsgBox** indicherà l'operazione di spostamento (vedi figura 5). In tali situazioni al termine del trasferimento i dati nel controllo di origine dovrebbero essere eliminati

Nel nostro progetto l'unico controllo per cui è abilitata l'operazione di spostamento è **Image1**.

Per dimostrare il funzionamento selezionare innanzitutto il pulsante di opzione **Sposta**, trascinare l'immagine **Image1** sulla **PictureBox** in basso come mostrato nella figura a fianco e rilasciare il pulsante del mouse.

L'immagine originale sarà trasferita nella **PictureBox**.

Per fare apparire nuovamente l'immagine del cane basterà un semplice click doppio sulla casella dell'immagine in alto.

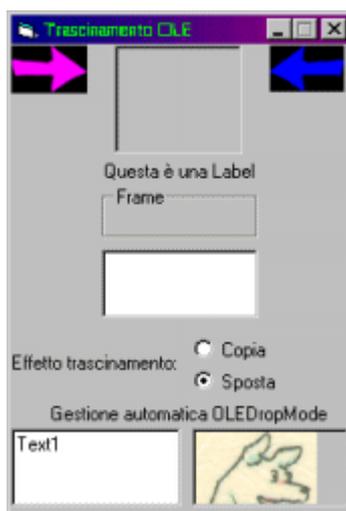


Figura 6

Abbiamo volutamente relegato alla fine del discorso i due controlli in basso con proprietà *OLEDropMode* impostata su **2 - Automatic**: **txtAutomatica** e **picAutomatica**.

Tali controlli effettuano il ricevimento dei dati in maniera automatica senza l'esecuzione degli eventi relativi *OLEDragOver*, *OLEDragDrop* e quindi non richiedono una gestione manuale come per i controlli precedenti. Il controllo **txtAutomatica** leggerà i dati di tipo stringa se presenti mentre il controllo **picAutomatica** recupererà l'immagine in maniera automatica.

Il discorso è ancora ampio e richiederebbe ben più di una singola pagina web. Mediante questo meccanismo possono essere effettuate tantissime operazioni utili rendendo l'applicativo semplice per l'utente. La maggioranza dei programmi sia per piattaforme Windows che per altri sistemi operativi utilizzano questa caratteristica.

Il progetto in prima analisi potrebbe sembrare complesso e lungo ma uno sguardo al codice del progetto scaricabile potrebbe chiarire molti punti a prima vista oscuri.

Si consiglia anche la consultazione della guida in linea di Visual Basic sull'uso della proprietà *Files* della classe  *DataObject*.

Fibia FBI
3 Marzo 2002



[Torna all'indice degli HowTo](#)
