Tricks & Tips

[Access]

- Michele de Nittis -

C.I.S.A. 1 - I Controlli Treeview e List View

Versione: 1 Data Versione: martedì 24 maggio 2005

Indice

Introduzione 2
Il controllo TreeView 2
L'evento NodeClick() 5
Il controllo ImageList 5
Impiego delle Immagini nel controllo Tree View 7
Il Controllo ListView 8
Aggiunta delle voci al controllo 9
La modalità Report View 11
Come leggere i valori del controllo selezionati dall'utente 13
La funzionalità Drag and Drop (D&D) 13
OleStartDrag() 15
OleDragOver() 15
OleDragDrop() 16
Il metodo HitTest() 19
La proprietà DropHighLight 19
Abilitare lo scrolling durante il trascinamento di un nodo all'interno di un controllo TreeView 20

Introduzione

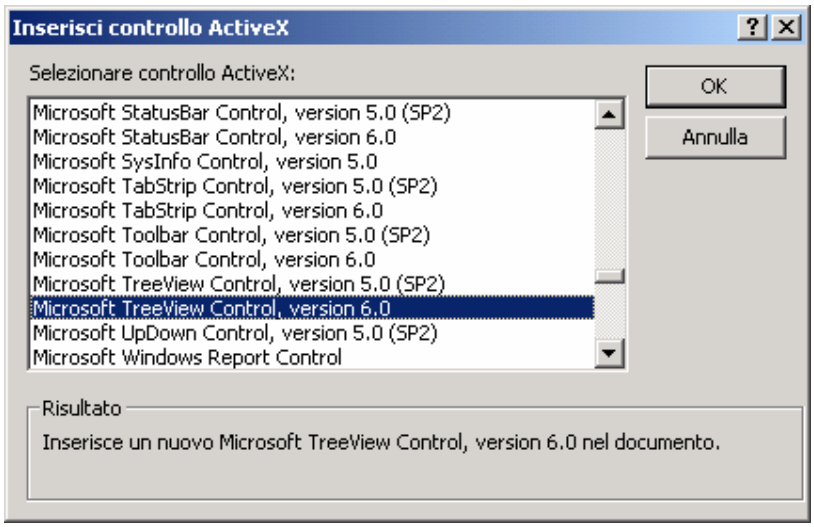
Questo documento tratterà, in modo sintetico, l'uso dei controlli ActiveX all'interno di una maschera (*Form*) di Access.

Maggiori dettagli sono disponibili in quattro tutorial scaricabili dalla sezione Documentazione del sito internet dell'autore (http://www.mdenittis.it/Documentazione.htm).

Il controllo TreeView

Questo controllo si presenta come un insieme di voci organizzate secondo una struttura ad albero. Ogni voce corrisponde ad un nodo dell'albero.

Per inserire questo controllo, entrare in modalità *Struttura* nella maschera dove si vuol collocare il controllo ActiveX. Selezionare la voce "Controllo ActiveX" dal menu "Inserisci" e scegliere la voce "*Microsoft TreeView Control, version 6.0*" dalla lista della finestra "Inserisci controllo ActiveX". Premere il bottone OK.



Inizialmente l'albero risulta privo di nodi, cosicché è necessario sfruttare un evento per eseguire un procedura (Sub) di popolamento dell'albero. Possiamo, ad esempio, sfruttare l'evento **Form_Load**() della maschera che contiene il componente.

L'oggetto di classe *TreeView*, appartenete alla libreria *MSComCtlLib*, è in relazione *Has Many* con l'oggetto di classe *Node* appartenente alla medesima libreria. Infatti tra le sue proprietà figura la collezione *Nodes* di oggetti *Node* che stabilisce la predetta relazione.

Esaminiamo alcuni dettagli della classe ***Node*** e della collezione ***Nodes***. Le proprietà più significative dell'oggetto *MSComCtlLib.Node* sono le seguenti:

Checked	Boolean	Determina (r/w) se l'oggetto risulta selezionato, nel qual caso restituisce il valore true.
Child	(Property) Node	E' l'oggetto classe *Nodo* gerarchicamente inferiore all'oggetto corrente. Disponibile solo a tempo di esecuzione.
Expanded	Property (Boolean)	Determina (r/w) se il noto è collassato o espanso nei suoi sottonodi: true per espanso, false per collassato.
Index	(Property) Integer	E' l'identificatore numerico univoco per l'oggetto di classe Nodo corrente. Coincide con il numero d'ordine del nodo all'interno della collezione *Nodes* e può variare dinamicamente con l'inserimento e la rimozione di nodi dalla predetta collezione.
Key	(Property) String	E' una stringa identificativa univoca di un nodo all'interno di un controllo TreeView.
Next	(Property) Node	E' l'oggetto classe *Nodo* che segue l'oggetto corrente nel medesimo livello dell'albero.
Parent	(Property) Node	E' l'oggetto classe *Nodo* gerarchicamente superiore all'oggetto corrente. Disponibile solo a tempo di esecuzione.
Previous	(Property) Node	E' l'oggetto classe *Nodo* che precede l'oggetto corrente nel medesimo livello dell'albero.
Root	(Property) Node	E' il nodo radice dell'albero di cui l'oggetto di classe Nodo corrente è parte.
Text	String	E' la stringa di testo con cui l'oggetto viene graficamente visualizzato nell'albero.
Visibile	Boolean	Determina se l'oggetto risulta visibile (true) o nascosto (false)

I metodi più significativi della collezione *Nodes* sono i seguenti:

Add(…)	Node	Aggiunge un nuovo nodo alla collezione di nodi '*Nodes*' del componente TreeView
Remove(item)	Void	Rimuove un l'elemento indicato dal parametro <item> dalla collezione di nodi '*Nodes*' del controllo TreeView. <item> è l'indicatore univoco del nodo da rimuove e può essere un numero (prop. Index) o una stringa (prop. Key).
Clear()	void	Cancella tutto l'albero.

Esaminiamo la sintassi del metodo **Add**() che è quello che ci consente di aggiungere voci al controllo TreeView:

TreeView.Nodes.**Add**(*relative, relationship, key, text, image, selectedimage*)

Questo metodo consente di creare un nuovo nodo e di aggiungerlo alla collezione *Nodes*, restituendo al chiamante un riferimento al nodo stesso.

Il dettaglio dei parametri formali del metodo **Add()** è il seguente:

Relative	Opzionale. Rappresenta l'identificatore (numero o stringa) univoco di un nodo già presente nella collezione *Nodes* con il quale quello che da aggiungere è legato da una relazione (vedi parametro *Relationship*).
Relationship	Opzionale. Rappresenta il tipo di relazione che lega il nodo da aggiungere ad uno o più nodi della collezione *Nodes (vedere tabella successiva*).
Key	Opzionale: corrisponde alla omonima proprietà della classe *Node*.
Text	Obbligatorio: corrisponde alla omonima proprietà della classe *Node*
Image	Opzionale: è l'indice dell'immagine, presente nel controllo **ImageList** associato all'albero, che viene usata per visualizzare il nodo nell'albero;
Selectedimage	Opzionale: come il parametro precedente ma l'immagine è quella che viene mostrata quando il nodo è selezionato.

I valori ammissibili per il parametro *Relationship* sono:

tvwFirst	0	Il nodo viene posizionato nell'albero prima di tutti gli altri nodi di livello pari a quello del nodo specificato nel parametro *Relative*.
tvwLast	1	Il nodo viene posizionato nell'albero dopo tutti gli altri nodi di livello pari a quello del nodo specificato nel parametro *Relative*.
tvwNext	2	Il nodo viene posizionato nell'albero dopo il nodo specificato nel parametro *Relative* e nello stesso livello.
tvwPrevious	3	Il nodo viene posizionato nell'albero prima del nodo specificato nel parametro *Relative* e nello stesso livello.
tvwChild	4	Il nodo diventa uno dei figli del nodo specificato nel parametro *Relative*.

Nella sottostante figura, è riportato un esempio di applicazione del metodo **Add**.

```
Option Compare Database
Public tmpnode As MSComctlLib.Node
Private Sub Form_Load()
' Aggiunta delle finanziarie
Set tmpnode = Me.TreeView2.Nodes.Add(, , "Fin01", "Finanziaria 1")
tmpnode.Expanded = True
Set tmpnode = Me.TreeView2.Nodes.Add(, , "Fin02", "Finanziaria 2")
tmpnode.Expanded = True
Set tmpnode = Me.TreeView2.Nodes.Add(, , "Fin03", "Finanziaria 3")
tmpnode.Expanded = True
```

Osservazione:

Qualora non venga specificato alcun valore per il parametro *Relative*, il nodo creato viene aggiunto come ultimo tra i nodi del livello più alto.

L'evento NodeClick()

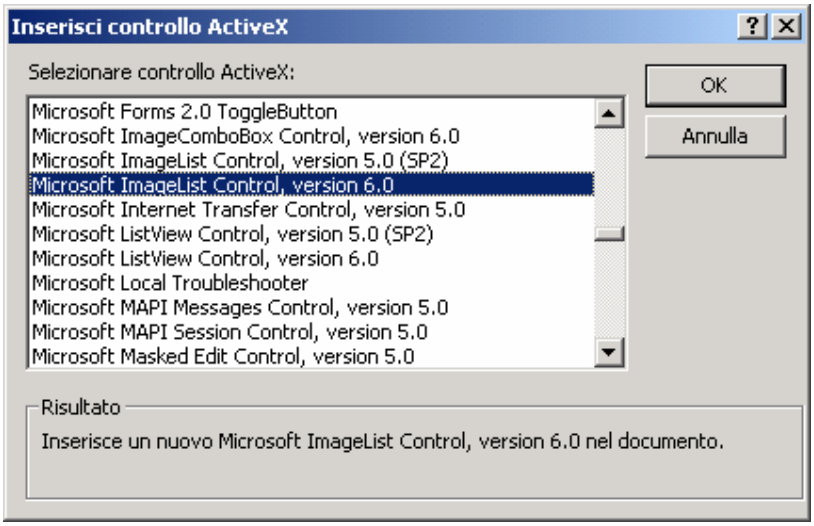
Questo evento occorre ogni volta che l'utente *seleziona* un nodo del controllo TreeView. La procedura (Sub) che intrappola e gestisce l'evento ha la seguente sintassi dichiarativa:

Private Sub **TreeViewObject_NodeClick**(*ByVal Node as Object*)

Il controllo ImageList

Per associare delle immagini alle voci di un controllo Tree View o di un controllo List View (descritto più avanti) si deve far uso dell'oggetto ausiliario ImageList. Tale oggetto è essenzialmente una collezione di immagini che possono essere usate all'interno di altri controlli della libreria *Windows Common Controls*.

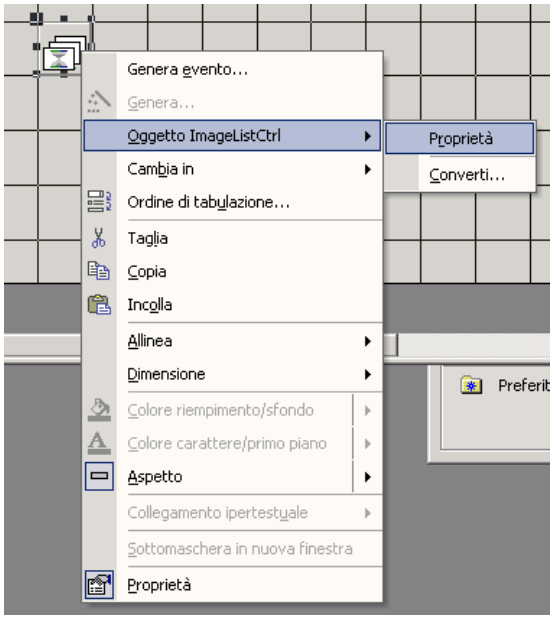
Spieghiamo l'uso di questo particolare controllo all'interno di Access[1] mediante un esempio pratico. Si apra in modalità struttura una maschera di Access che verrà usata a titolo esemplificativo (la si chiami *Esempio* per semplicità), e vi si aggiunga il controllo Active X '*Microsoft ImageList Control, version 6.0*'.



Si selezioni l'oggetto, si prema il tasto destro del mouse e si scelga la voce 'Oggetto ImageListCtrl → Proprietà'

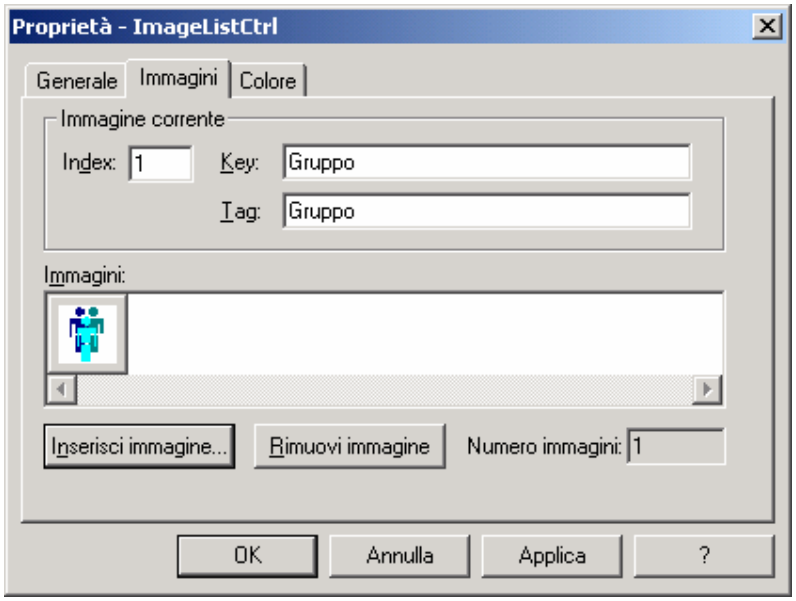
---

[1] L'esempio illustrato nel presente documento è stato realizzato con Access 2002 ma è facilmente portabile sulle versioni precedenti di Access.



A questo punto si possono immettere nella collezione le immagini che dovranno essere visualizzate in altri controlli ActiveX della maschera.

L'inserimento delle immagini è semplicissimo, basta premere sul bottone '*Inserisci Immagine*' e cercare il relativo file.



Una volta inserita un'immagine è possibile assegnarle un'etichetta ed una **chiave di ricerca**, come mostrato nella sovrastante figura.

Le immagini possono essere aggiunte al controllo anche a tempo di esecuzione (*runtime*) del codice VBA. A tal proposito si catturi l'evento '*Form_Load()*' della predetta maschera *Esempio* e si inserisca il seguente codice per caricare l'immagine 'Divieto.gif':

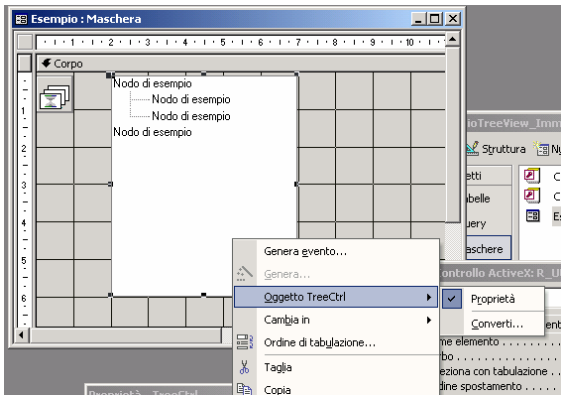
```
Private Sub Form_Load()
Me.R_ListaImmagini.ListImages.Add , "Negato", _
LoadPicture("c:\temp_Mike\Lavori\CreditoAgevolato\Divieto.gif")
End Sub
```

Il caricamento di un'immagine a tempo di esecuzione deve essere compiuto una sola volta per ogni esecuzione del programma, altrimenti VBA genera un errore di tipo "**chiave duplicata**".

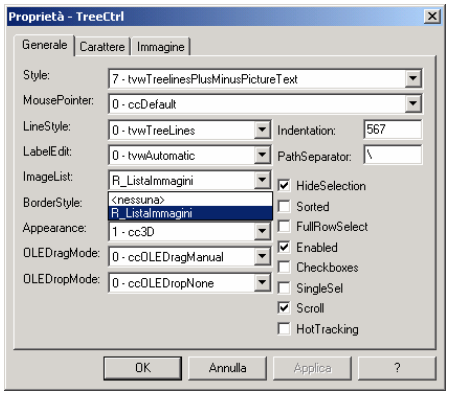
Impiego delle Immagini nel controllo Tree View

Si inserisca nella maschera d'esempio un controllo ActiveX di tipo Tree View (vedi [1]).

Per inserire delle immagini in corrispondenza dei nodi dell'albero è necessario associare all'oggetto **TreeView** un oggetto **ImageList** tramite la proprietà *ImageList* dell'oggetto **TreeView** stesso. Questa proprietà viene impostata **<u>in fase di progetto (*Design Time*)</u>** mediante la maschera '*proprietà*' dell'oggetto **TreeView**, richiamabile selezionando il controllo **TreeView** dalla maschera *Esempio* in modalità 'Struttura', premendo il tasto destro del mouse e selezionando la voce 'Oggetto TreeCtrl→Proprietà', come illustrato nella seguente figura:



Selezionare, quindi, la scheda 'Generale' delle proprietà dell'oggetto **TreeView** ed impostare la proprietà **ImageList**, come illustrato nella seguente figura.



A questo punto è semplice associare un'icona a ciascun nodo del controllo TreeView: al momento dell'invocazione del metodo ***Add()*** della collezione *Nodes* si passa, come ultimo parametro (vedi la sintassi del metodo *Add()*), la chiave identificativa dell'icona scelta all'interno del controllo *ImageList.*

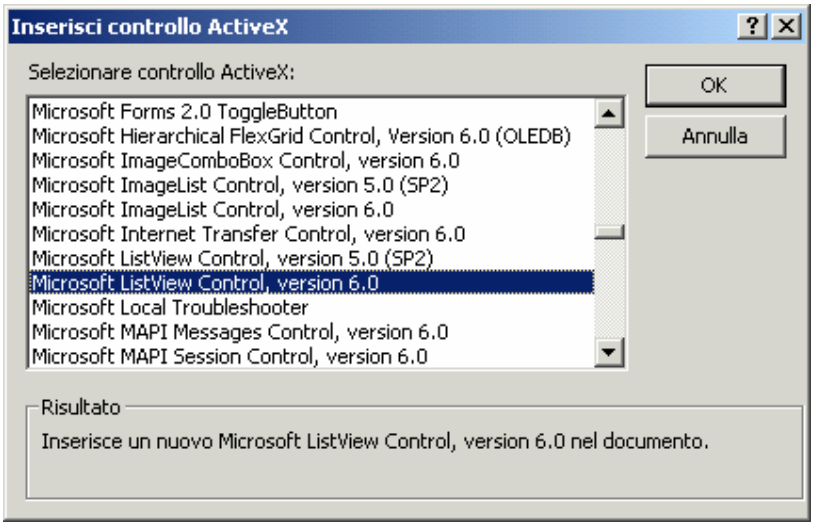
Il procedimento è Identico nel caso di assegnazione di icone alle righe di un controllo di tipo *ListView*.

Il Controllo ListView

Questo controllo si presenta come un insieme di voci organizzate in uno dei seguenti quattro modi:

Modo	**Descrizione**
Icon view	Ogni voce è un elemento grafico costituito da un'icona e da un'etichetta descrittiva. L'utente può spostare ogni elemento in un qualsiasi punto del controllo.
Small icon view	Come il precedente modo ma le icone sono più piccole.
List view	Ogni voce è rappresentata da un elemento testuale accompagnato da un'eventuale piccola icona. Gli elementi vengono mostrati incolonnati dall'alto verso il basso e da sinistra verso destra ed allineati a sinistra.
Report view	Questo modo mostra le voci in modo tabellare. Ogni voce è una riga di una tabella ed è costituita da molteplici informazioni testuali e grafiche disposte lungo le colonne.

Per inserire il controllo all'interno di una maschera, entrare in modalità *Struttura*, selezionare la voce "Controllo ActiveX" dal menu "Inserisci" e scegliere la voce "*Microsoft ListView Control, version 6.0*" dalla lista della finestra "Inserisci controllo ActiveX". Premere il bottone OK.



Trascinare la sagoma del controllo nella posizione desiderata e farli assumere le dimensioni preferite.

Il controllo ListView *usa* due collezioni: *ColumnHeaders*, che è la collezione di oggetti di classe *ColumHeader* che rappresentano le intestazioni delle colonne del controllo in modalità *Report View*, e *ListItems,* che è la collezione di oggetti *ListItem*, cioè gli elementi testuali e grafici che rappresentano le voci da visualizzare. In modalità *Report View*, inoltre, è possibile specificare, per ogni voce, tutte le rispettive sottovoci mediante l'array *SubItems*.

Aggiunta delle voci al controllo

Esaminiamo prima le modalità *Icon View*, *Small Icon View* e *List View*, più semplici, per poi approfondire la modalità *Report View* più articolata e completa. Per impostare la modalità di visualizzazione si deve assegnare alla proprietà pubblica *View* uno dei seguenti valori costanti:

Costante	Valore	Visualizzazione
lvwReport		Report View
lvwIcon		Icon View
lvwList		List View
lvwSmallIcon		Small Icon View

Per aggiungere un elemento di tipo ListItem al controllo ListView si deve usare il metodo Add() della collezione ListItems, la cui segnatura è la seguente:

```
Object.Add(index, key, text, icon, smallicon)
```

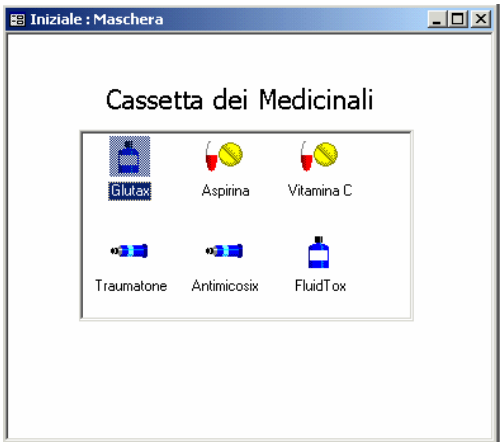
Index	**Opzionale.** È un numero che indica la posizione della voce all'interno della collezione di voci. Se omesso, la voce viene inserita in coda alla collezione. Il suo valore varia in funzione di

	successivi inserimenti e cancellazioni.
Key	**Opzionale.** E' una stringa che rappresenta la chiave identificativa **univoca** di ogni voce nella collezione. Utile per la ricerca di una voce.
Text	**Opzionale.** Rappresenta la parte testuale della voce. Viene mostrata nel controllo, eventualmente associata ad una icona.
Icon	**Opzionale.** Può essere un numero o una stringa e rappresenta, rispettivamente, l'indice o la chiave identificativa per fare riferimento ad un'immagine del controllo ImageList associato all'oggetto ListView (proprietà **Icon**).
SmallIcon	**Opzionale.** Come il precedente parametro ma fa riferimento alla proprietà **SmallIcon.**

Come già fatto per il controllo TreeView, per popolare il controllo di voci impieghiamo l'evento **Form_Load()** (su Caricamento) della finestra "*Iniziale*":

```
Private Sub Form_Load()
Dim LI As ListItem
' Aggiunta di uno sciroppo
Set LI = ListView0.ListItems.Add(, "Glu001", "Glutax", "Sciroppo", "SmallSciroppo")
' Aggiunta di Pillole
Set LI = ListView0.ListItems.Add(, "Asp001", "Aspirina", "Pillole", "SmallPillole")
' Aggiunta di Pillole
Set LI = ListView0.ListItems.Add(, "VtC001", "Vitamina C", "Pillole", "SmallPillole")
' Aggiunta di una Pomata
Set LI = ListView0.ListItems.Add(, "Tmt001", "Traumatone", "Pomata", "SmallPomata")
' Aggiunta di una Pomata
Set LI = ListView0.ListItems.Add(, "Antm001", "Antimicosix", "Pomata", "SmallPomata")
' Aggiunta di uno sciroppo
Set LI = ListView0.ListItems.Add(, "Flu001", "FluidTox", "Sciroppo", "SmallSciroppo")
' modo di visualizzazione di default - Icon View
Me.ListView0.View = lvwIcon
End Sub
```

Il risultato del predetto codice è mostrato nella seguente figura dove gli *Item* vengono mostrati in modalità *Icon View* (lvwIcon).



La modalità Report View

Questa modalità di visualizzazione è la più articolata ed è anche quella con maggior potere espressivo. In sintesi è possibile mostrare ciascuna delle voci del controllo mediante informazioni di dettaglio testuali e grafiche.

Come abbiamo visto nei precedenti modi di visualizzazione, per inserire una nuova voce nel controllo ListView è necessario creare un apposito oggetto **ListItem** ed aggiungerlo alla collezione **ListItems** mediante il metodo **Add().** Per aggiungere informazioni di dettaglio ad una voce è possibile usare l'array **SubItems** del corrispondente oggetto **ListItem**, come illustra il seguente frammento di codice.

```
' Aggiunta di uno sciroppo
Set LI = ListView0.ListItems.Add(, "Glu001", "Glutax", "Sciroppo", "SmallSciroppo")
LI.SubItems(1) = "Lassativo"
LI.SubItems(2) = "12/12/2004"

' Aggiunta di Pillole
Set LI = ListView0.ListItems.Add(, "Asp001", "Aspirina", "Pillole", "SmallPillole")
LI.SubItems(1) = "Febbrifugo"
LI.SubItems(2) = "04/09/2003"

' Aggiunta di Pillole
Set LI = ListView0.ListItems.Add(, "VtC001", "Vitamina C", "Pillole", "SmallPillole")
LI.SubItems(1) = "Ricostituente"
LI.SubItems(2) = "19/05/2005"

' Aggiunta di una Pomata
Set LI = ListView0.ListItems.Add(, "Tmt001", "Traumatone", "Pomata", "SmallPomata")
LI.SubItems(1) = "Unguento anti trauma"
LI.SubItems(2) = "25/02/2005"

' Aggiunta di una Pomata
Set LI = ListView0.ListItems.Add(, "Antm001", "Antimicosix", "Pomata", "SmallPomata")
LI.SubItems(1) = "Antimicotico"
LI.SubItems(2) = "25/02/2005"

' Aggiunta di uno sciroppo
Set LI = ListView0.ListItems.Add(, "Flu001", "FluidTox", "Sciroppo", "SmallSciroppo")
LI.SubItems(1) = "Fluidificatore Muco"
LI.SubItems(2) = "14/11/2003"
```

Si devono, poi, aggiungere alla collezione **ColumnHeaders** dell'oggetto **ListView,** mediante il metodo **Add()**, tanti oggetti di classe **ColumnHeader** quante sono le colonne che si vogliono mostrare. Il metodo **Add()** della collezione **ColumnHeaders** ha la seguente sintassi:

```
Object.Add(indice, chiave, testo, larghezza, allineamento, icona)
```

indice	**Opzionale.** È un numero che indica la posizione della colonna.
chiave	**Opzionale.** E' una stringa che rappresenta la chiave identificativa **univoca** di ogni voce nella collezione. Utile per la ricerca di una voce.
testo	**Opzionale.** Rappresenta la parte testuale di un'intestazione di colonna. Può essere associata ad una icona.

larghezza	**Opzionale.** Espressione numerica che specifica la larghezza dell'oggetto, in base all'unità di scala utilizzata dal contenitore del controllo.
allineamento	**Opzionale.** Facoltativa. Valore intero che determina l'allineamento del testo nell'oggetto **ColumnHeader.**

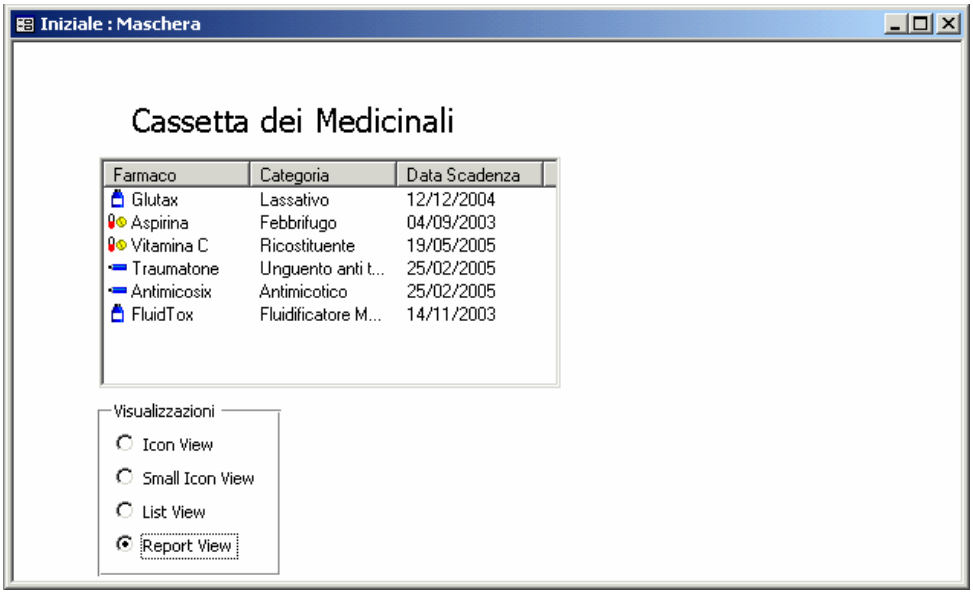
lvwColumnLeft	0	(Predefinita) Sinistra. Il testo viene allineato a sinistra.
lvwColumnRight	1	Destra. Il testo viene allineato a destra.
lvwColumnCenter	2	Centrato. Il testo viene centrato.

icona	**Opzionale.** Può essere un numero o una stringa e rappresenta, rispettivamente, l'indice o la chiave identificativa di un'immagine del controllo ImageList associato all'oggetto ListView (proprietà **ColumnHeaderIcons**).

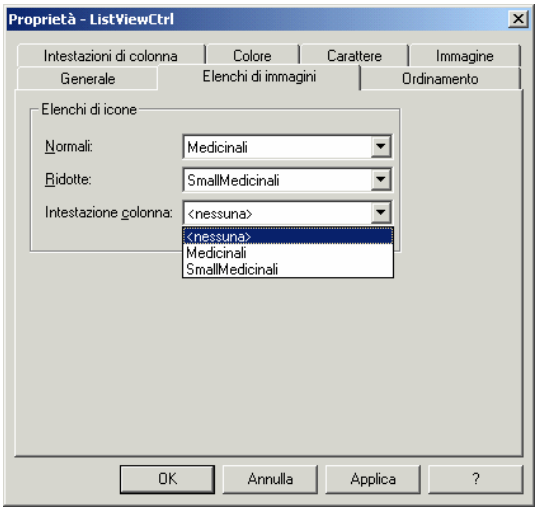
Esempio:

```
' impostiamo tre intestazioni di colonna
Set CH = ListView0.ColumnHeaders.Add(, , "Farmaco")
Set CH = ListView0.ColumnHeaders.Add(, , "Categoria")
Set CH = ListView0.ColumnHeaders.Add(, , "Data Scadenza")
```

Il risultato del nuovo codice aggiunto è il seguente:



Per associare a ciascun'intestazione un'icona, si deve impostare la proprietà **ColumnHeaderIcons** del controllo **ListView** in fase di definizione della struttura della maschera (*Design Time*).



Come leggere i valori del controllo selezionati dall'utente

Per accedere ai valori della voce del controllo (riga) selezionata dall'utente è necessario recuperare il corrispondente oggetto di classe Listitem.

Set LI= ListView0.selectedItem

Una volta ottenuto l'oggetto di classe ListItem è possibile leggere i valori delle sue proprietà:

LI.text;

LI.subitems(x), con x ∈ { 1, 2, 3, … N}

Per conoscere il numero N di oggetti di classe ListItem associati ad un oggetto di classe ListView, basta ricordare che la proprietà ListItems dell'oggetto ListView è un oggetto di classe Collection. Dunque:

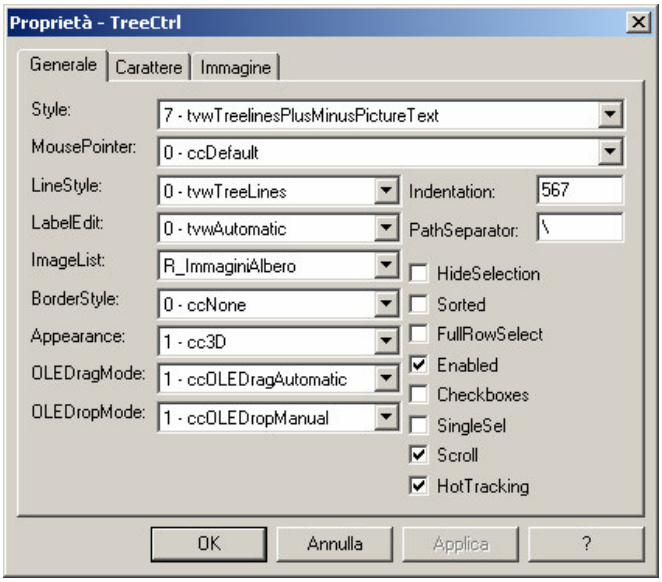
N = ListView0.ListItems.count()

La funzionalità Drag and Drop (D&D)

Esaminiamo la funzionalità Drag and Drop all'interno di un controllo *TreeView* e tra due controlli diversi, ad esempio uno di tipo *TreeView* ed uno di tipo *ListView*.

Inseriamo in una maschera di prova due controlli, uno di tipo *TreeView* ed uno di tipo *ListView*, come descritto nei precedenti paragrafi e chiamiamo **R_AlberoTest** il controllo *TreeView* e **R_ListaTest** il controllo di tipo *ListView*. Analizziamo prima il caso di trascinamento di un nodo all'interno del controllo *TreeView.*

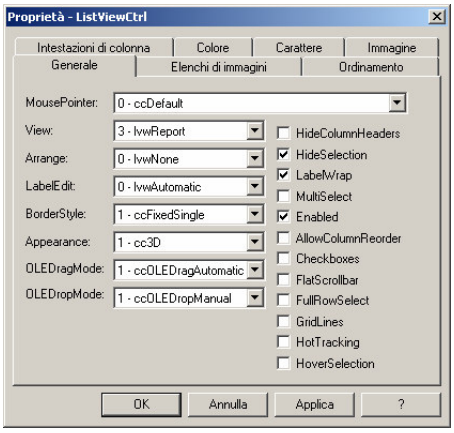
Per abilitare la funzionalità Drag – and – Drop (D&D) nel controllo di tipo *TreeView*, selezionare il controllo con la maschera (*form*) in modalità struttura e col tasto destro del mouse far apparire la maschera delle proprietà (vedi figura seguente).

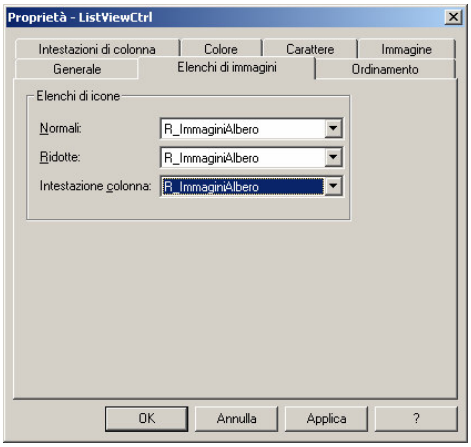


Impostare, quindi, le seguenti proprietà come di seguito indicato:

- OLEDragMode: 1- ccOLEDragAutomatic;
- OLEDropMode: 1 – ccOLEDropManual.

Per abilitare la funzionalità D&D sul controllo di tipo *ListView*, selezionare il controllo quando la maschera (*form*) è in modalità struttura e col tasto destro del mouse far apparire la maschera delle proprietà (vedi figure seguenti).





Una volta attivata le funzionalità D&D, per controllarne il funzionamento a tempo di esecuzione è possibile impiegare i seguenti eventi:

1. *_OLEStartDrag*(): invocato all'inizio del trascinamento;
2. *_OLEDragOver()*: invocato quando un oggetto è trascinato sopra il controllo;

3. *_OLEDragDrop()*: invocato al termine del trascinamento, quando si rilascia il tasto sinistro del mouse;

OleStartDrag()

Questo evento viene invocato prima che inizi l'azione di trascinamento del nodo selezionato e viene impiegato, solitamente, per operazioni di *controllo* e/o di *inizializzazione*. La sintassi completa dell'evento è la seguente:

Oggetto_OleStartDrag(Data As Object, AllowedEffect as Long)

Oggetto	È l'oggetto da cui ci sia aspetta l'invocazione dell'evento, nel caso in esempio è il controllo **R_AlberoTest**.
Data	E' un oggetto contenente I formati che il controllo fornirà ed, eventualmente, i dati per quei formati.
AllowedEffect	E' un intero lungo che specifica gli effetti che il controllo supporta. I valori possibili sono i seguenti: (0) – **vbDropEffectNone**: L'elemento di destinazione dell'operazione di trascinamento (Drag) non può accettare dati; (1) – **vbDropEffectCopy**: L'esecuzione dell'operazione di Drop si conclude con una **copia** dell'elemento sorgente trascinato sull'elemento di destinazione, lasciando comunque l'elemento sorgente inalterato; (2) – **vbDropEffectMove**: L'esecuzione dell'operazione di Drop si conclude con uno **spostamento** dell'elemento sorgente trascinato sull'elemento di destinazione.

OleDragOver()

Questo evento viene invocato durante l'operazione di trascinamento dell'elemento (Node/Item) selezionato sugli altri elementi del controllo ActiveX. Questo evento è utile per controllare la posizione del puntatore del mouse all'interno del controllo stesso.

La sintassi completa dell'evento è la seguente:

Oggetto_OleDragOver (*data *As Object,* effect *As Long,* button *As Integer,* shift *As Integer,* x *As Single,* y *As Single,* state *As Integer)

Oggetto	È' l'oggetto da cui ci sia aspetta l'invocazione dell'evento, nel caso in esempio è il controllo **R_AlberoTest**.
Data	E' un oggetto contenente I formati che il controllo fornirà ed, eventualmente, i dati per quei formati.
Effect	Un intero lungo (Long) inizializzato dall'oggetto origine per identificare tutti gli *effetti* supportati.
button	Valore intero che agisce come un campo di bit corrispondente allo stato di uno dei tasti del mouse quando è premuto. Il tasto sinistro corrisponde al bit in posizione 0, il tasto

	destro corrisponde al bit in posizione 1 ed il tasto centrale corrisponde al bit in posizione 2. Per ciascun bit il valore 1 indica che il tasto è premuto.
Shift	Valore intero che agisce come un campo di bit corrispondenti allo stato dei tasti *Shift*, *ctrl* e *alt* quando sono premuti. Il tasto *shift* corrisponde al bit 0, il tasto *ctrl* corrisponde al bit 1 ed il tasto *alt* corrisponde al bit 2. Per ciascun bit il valore 1 indica che il tasto è premuto
x, y	Sono valori interi che specificano l'attuale posizione del puntatore del mouse in coordinate orizzontali (x) e verticali (y).
State	E' un valore intero che corrisponde allo stato della transazione del Controllo in fase di trascinamento con riferimento alla maschera o al controllo di destinazione: (0) – Enter (il Controllo sorgente è trascinato all'interno della destinazione); (1) – Leave (il Controllo sorgente è trascinato all'esterno della destinazione); (2) – Over (il Controllo sorgente è stato spostato da una posizione ad un'altra all'interno della destinazione).

OleDragDrop()

Questo evento viene invocato al termine dell'operazione di trascinamento di un elemento di un controllo su un altro elemento. L'operazione di trascinamento ha termine quando l'elemento spostato viene posizionato per trascinamento (con il tasto sinistro del mouse premuto) su un altro elemento e viene rilasciato il tasto sinistro del mouse. Possiamo sfruttare questo evento per aggiornare lo stato del controllo ActiveX, per controllare la consistenza delle informazioni che si stanno spostando mediante trascinamento e per compiere le operazioni associate al rilascio (*Drop*) di un oggetto (*Item/Node*) su un altro oggetto (*Item/Node*).

La sintassi completa dell'evento è la seguente:

OLEDragDrop(*data *As Object,* Effect *As Long,* button *As Integer,* shift *As Integer,* x *As Single,* y *As Single,* state *As Integer)

Oggetto	È' l'oggetto da cui ci sia aspetta l'invocazione dell'evento, nel caso in esempio è il controllo **R_AlberoTest**.
Data	E' un oggetto contenente I formati che il controllo fornirà ed, eventualmente, i dati per quei formati.
Effect	Un intero lungo (Long) inizializzato dall'oggetto origine per identificare tutti gli *effetti* supportati.
Button	Valore intero che agisce come un campo di bit corrispondente allo stato di uno dei tasti del mouse quando è premuto. Il tasto sinistro corrisponde al bit in posizione 0, il tasto destro corrisponde al bit in posizione 1 ed il tasto centrale corrisponde al bit in posizione 2. Per ciascun bit il valore 1 indica che il tasto è premuto.

Shift	Valore intero che agisce come un campo di bit corrispondenti allo stato dei tasti *Shift*, *ctrl* e *alt* quando sono premuti. Il tasto *shift* corrisponde al bit 0, il tasto *ctrl* corrisponde al bit 1 ed il tasto *alt* corrisponde al bit 2. Per ciascun bit il valore 1 indica che il tasto è premuto
x, y	Sono valori interi che specificano l'attuale posizione del puntatore del mouse in coordinate orizzontali (x) e verticali (y).

Affronteremo ora la problematica del trascinamento di un oggetto *Node* o *ListItem* da un controllo ad un altro all'interno della medesima maschera. In particolare vedremo operativamente come spostare le informazioni collezionate in un controllo di tipo *List View* per riorganizzarle in un controllo di tipo *TreeView*.

Quanto illustreremo è di validità generale, potendosi applicare con poche e semplici modifiche al trascinamento di informazioni da un controllo di tipo *Tree View* verso uno di tipo *List View*, oppure da un controllo di tipo *List View* verso un altro controllo di tipo *List View* o, ancora, tra due controlli di tipo *Tree View.*

Partiamo facendo due considerazioni:

1) il controllo *ListView* supporta la gestione degli eventi OLEDragStart, OLEDragOver ed OLEDragDrop in modo totalmente identico al controllo *TreeView*;
2) Nel trascinamento d'informazioni tra due controlli diversi abbiamo la necessità di identificare il controllo d'origine, cioè il controllo dal quale inizia il trascinamento (ovviamente questo problema non sussiste in operazioni D&D all'interno del medesimo controllo ActiveX).

Nella sintassi dei gestori degli eventi OLEDragStart, OLEDragOver ed OLEDragDrop compare, come primo parametro, un oggetto di Classe *DataObject*. L'oggetto *DataObject* può essere pensato come una specie di *clipboard*.

Attenzione: *se all'interno di un modulo VBA si invocano i predetti gestori con una firma in cui il primo parametro è dichiarato come oggetto di classe DataObject, si ottiene un errore. Per la sintassi che non genera errori nell'ambiente VBA, vedere gli esempi che seguono*.

Nella procedura che gestisce l'evento di inizio del trascinamento (*StartDrag*) nel controllo *Tree View*, scriviamo un'informazione identificativa del controllo stesso nell'oggetto di classe *DataObject* usando il metodo *SetData().*

Esempio:

```
Private Sub R_ListaTest_OLEStartDrag(Data As Object, AllowedEffect As Long)
' Salviamo l'informazione sul controllo d'origine
Data.SetData "R_ListaTest", 1   ' 1 = vbCfText indica che il tipo di informazioni portato dall'oggetto
                                ' DataObject è una stringa di testo
End Sub
```

Nel codice d'esempio abbiamo impiegato il metodo *SetData()* per scrivere come stringa di testo (vbCfText=1) il nome del controllo *ListView* dal quale inizia l'operazione di trascinamento. Nella procedura che gestisce l'evento *DragDrop* degli altri controlli che possono accogliere l'informazione in corso di trascinamento si deve leggere dall'oggetto *DataObject* la stringa identificativa del controllo di partenza.

Esempio:

Il seguente frammento di codice, aggiunto alla procedura ***OLEDragDrop()*** del controllo *TreeView*, preleva l'informazione sul controllo d'origine, legge le informazioni necessarie alla creazione di un nuovo nodo (*Node*) dall'elemento *ListItem* selezionato, rimuove il predetto elemento *ListItem* del controllo *List View* ed esegue il refresh dei due controlli.

```
Private Sub R_AlberoTest_OLEDragDrop(Data As Object, Effect As Long, Button As Integer, shift As
Integer, x As Single, y As Single)
Dim Dest_Node As MSComctlLib.Node       'Nodo di destinazione
Dim Src_Node As MSComctlLib.Node        'Nodo di origine
Dim counter As Long                     'contatore
' Parte II
Dim Controllo As String                 'Stringa di appoggio
Dim Src_Item As MSComctlLib.ListItem    'ListItem d'Origine
Dim NomeIntero As String                'Stringa di Appoggio

' Verifica della provenienza del controllo
Controllo = Data.GetData(1)     ' 1 vbCFText

If Controllo = "R_ListaTest" Then
    ' PARTE DI CODICE NON OTTIMIZZATA PER RAGIONI DIDATTICHE
    ' Le informazioni provengono per trascinamento
    ' dal controllo ListView.

    ' Prelievo del Nodo di Destinazione
If Me.R_AlberoTest.DropHighlight Is Nothing Then Exit Sub
    Set Dest_Node = Me.R_AlberoTest.DropHighlight
    ' Prelievo dell'Item d'origine
    Set Src_Item = Me.R_ListaTest.SelectedItem
```

```
    ' Composizione del Nome Intero
    NomeIntero = Src_Item.SubItems(1) & " " & Src_Item.Text
    ' Rimozione dell'Item dal controllo ListView
    Me.R_ListaTest.ListItems.Remove Src_Item.Key
    ' Se il Nodo di destinazione è di secondo livello, prelievo il relativo nodo
    ' di primo livello
    If (Not Dest_Node.Parent Is Nothing) Then Set Dest_Node = Dest_Node.Parent
    ' Aggiunta dell'Item al controllo TreeView
    Set Src_Node = Me.R_AlberoTest.Nodes.Add(Dest_Node.Key, tvwChild, Src_Item.Key, NomeIntero,
Src_Item.Icon, Src_Item.Icon)
    ' Refresh dei controlli
    Me.R_AlberoTest.Requery
    Me.R_ListaTest.Requery
    Exit Sub
End If
… contitnua …
```

Tratteremo, infine, la sintassi del metodo **HitTest()** e la proprietà ***DropHighLight*** e spiegheremo come superare uno dei maggiori limiti del controllo TreeView che si manifesta durante il trascinamento di un nodo.

Il metodo HitTest()

Questo metodo degli oggetti TreeView e ListView restituisce, rispettivamente, l'oggetto di classe Node e l'oggetto di classe ListItem corrispondente alle coordinate x, y del puntatore del mouse all'interno del controllo. Le coordinate del mouse vengono passate alla funzione come parametri.

Node TreeView_Object.HitTest(x as Single, y as Single);

ListItem ListView_Object.HitTest(x as Single, y as Single).

La proprietà DropHighLight

Questa proprietà degli oggetti *TreeView* e *ListView* consente di evidenziare un oggetto Node o ListItem. Con la riga di codice

```
' Evidenziazione del cursore
Me.R_AlberoTest.DropHighlight = Me.R_AlberoTest.HitTest(x, y)
```

Si evidenzia l'oggetto sopra il quale sta passando il puntatore del cursore durante le operazioni di trascinamento.

Abilitare lo scrolling durante il trascinamento di un nodo all'interno di un controllo TreeView

Uno dei più grandi limiti del controllo *Tree View*, così come documentato dall'articolo Q177743 del 13 Luglio 2004 della Microsoft Knowledge Base, è il fatto che durante l'operazione di trascinamento oltre i confini del controllo, il controllo stesso non esegue automaticamente lo scrollino verticale.

Per superare questo limite si deve ricorrere necessariamente alle API di Windows (libreria user32.dll) mandando un messaggio al controllo *Tree View* in modo che effettui lo scrolling verticale. La funzione delle API di windows da invocare per inviare un messaggio al controllo *Tree View* è ***SendMessageA***, la cui dichiarazione all'interno di un modulo VBA è del tipo:

```
Private Declare Function sendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Any) As Long
```

Abbiamo ritenuto conveniente invocare la funzione all'interno di una procedura nella quale sia possibile ottenere informazioni sulla posizione del puntatore le mouse.

Esempio:

Nell'esempio esaminato nel presente documento, abbiamo invocato la funzione ***sendMessage()*** all'interno della procedura che gestisce l'evento ***DragOver***:

```
Private Sub R_AlberoTest_OLEDragOver(Data As Object, Effect As Long, Button As Integer, shift As Integer, x As Single, y As Single, state As Integer)
' Evidenziazione del cursore
Me.R_AlberoTest.DropHighlight = Me.R_AlberoTest.HitTest(x, y)
If y < 10 Then
    ' scroll verso l'alto
    sendMessage Me.R_AlberoTest.hwnd, 277&, 0&, vbNull
End If
If y > Me.R_AlberoTest.Height - 100 Then
    ' scroll verso il basso
    sendMessage Me.R_AlberoTest.hwnd, 277&, 1&, vbNull
End If
End Sub
```

Come si può osservare, la funzione di sistema viene invocata passandogli un *handler* al controllo *TreeView*, il codice del messaggio (277&) e la direzione in cui effettuare lo scrolling (0& verso l'alto, 1&verso il basso) quando la coordinata verticale (y) supera due soglie: una inferiore ed una superiore.