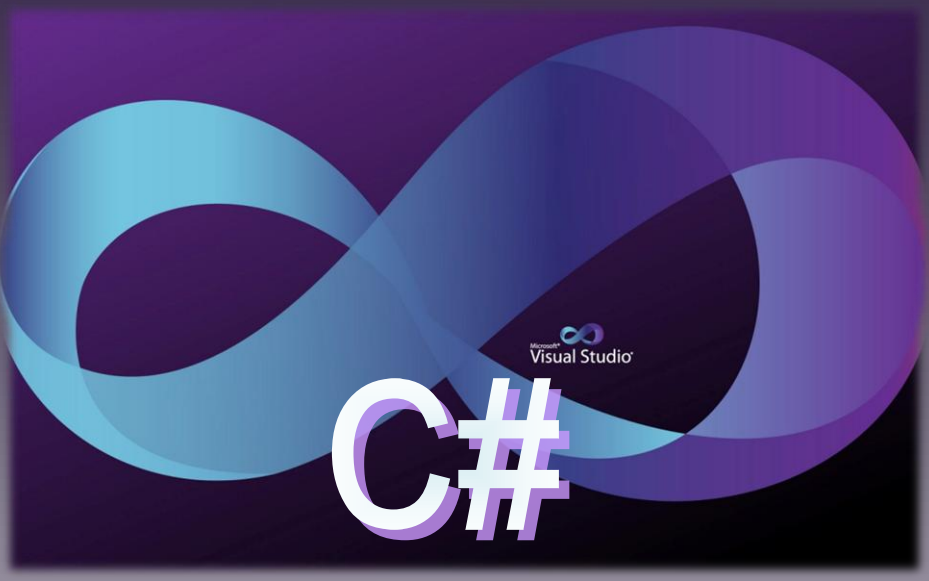
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

IL FORM E I SUOI EVENTI

DISPENSA 04.02

04-02_Il_Form_e_gli_Eventi_[ver_15]

Versione del: 07/11/2015
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

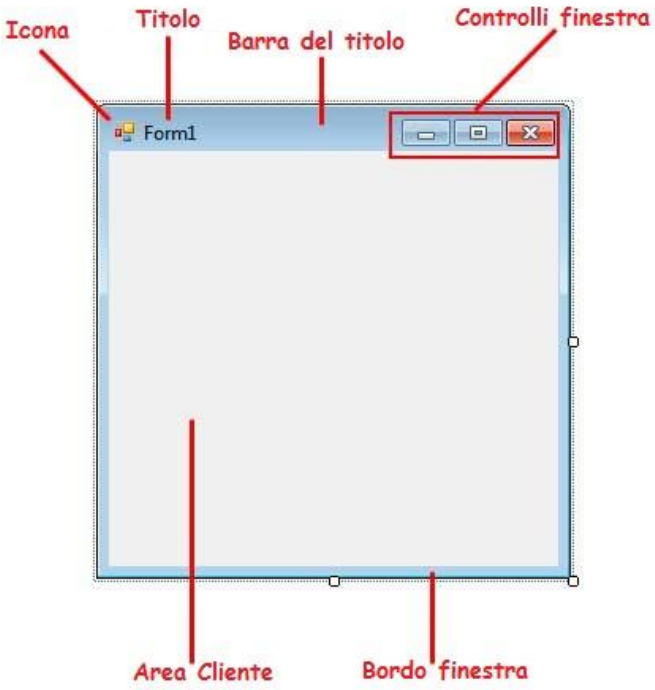
IL CONTROLLO FORM

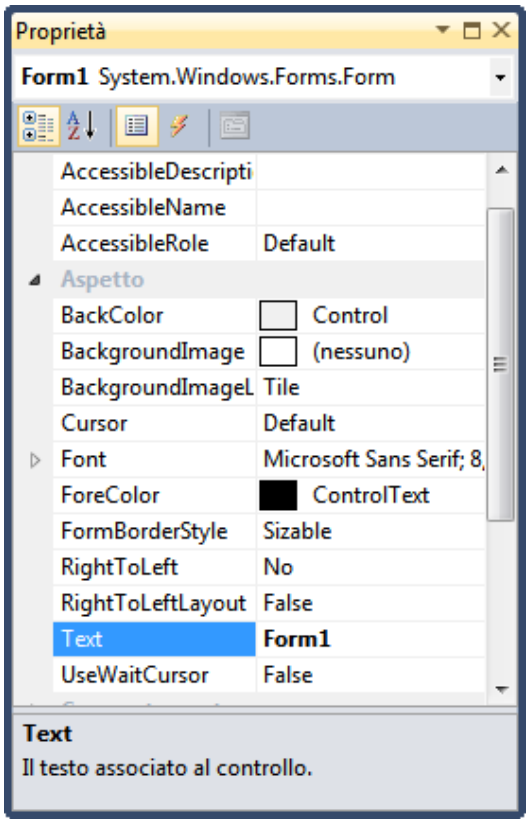
IL FORM

UNA FINESTRA DA SCHIUDERE

IL FORM

Il Form è un controllo e possiede le caratteristiche comuni ai controlli. Come ogni controllo possiede proprietà ed eventi. Il Form è un controllo contenitore, nel senso che è in grado di contenere altri controlli e di gestirli come un insieme.





Quando si seleziona il Form la finestra delle proprietà ne visualizza le sue. Si osservi che la finestra delle Proprietà indica il controllo selezionato

Scopri la Guida Microsoft

Riferimenti: http://msdn.microsoft.com/it-it/library/dd492171

Scopri come creare un visualizzatore di immagini, un labirinto, un quiz matematico o il gioco delle coppie.

RIFLESSIONI SUL FORM E SUGLI EVENTI

Il Form è un controllo e possiede le caratteristiche comuni ai controlli. Come ogni controllo possiede proprietà ed eventi. Le proprietà del Form si possono modificare sia a tempo di progettazione (design-time) per impostarne l'aspetto di partenza, sia con istruzioni di programma (run-time) come le normali assegnazioni.

In questa occasione non esamineremo tutte le proprietà e gli eventi del Form, ma solo quelli ritenuti principali ai fini del corso.

Le proprietà importanti del Form sono:

PROPRIETÀ	DESCRIZIONE
Text	Testo visualizzato sul titolo della finestra (barra di intestazione)
BackColor	Colore di sfondo della finestra
ForeColor	Colore di primo piano della finestra, il colore del testo visualizzato
Visible	Indica se della finestra è visibile mentre è in esecuzione
Enabled	Indica se della finestra può reagire ai comandi dell'utente
Width	Larghezza della finestra
Height	Altezza della finestra
Left	Distanza della finestra dal bordo sinistro dello schermo
Top	Distanza della finestra dal bordo superiore dello schermo

Altre proprietà interessanti del Form che useremo in seguito sono:

PROPRIETÀ	DESCRIZIONE
Icon	Icona che compare nell'angolo superiore sinistro della finestra
FormBorderStyle	Stile del bordo della finestra che determina anche il comportamento
StartPosition	Posizione iniziale all'avvio della finestra
Controls	Ottiene l'insieme di controlli contenuti nel controllo.

Molte delle proprietà del Form sono elencate nella finestra delle Proprietà dove possono essere modificate a tempo di progettazione (design-time).

FILE DEL FORM

Form significa letteralmente Modulo, come quello che compili per iscriverti a scuola. In effetti il Form nasce per proporre all'utente un'interfaccia con cui interagire e compiere le sue scelte.

Per impostazione predefinita, quando si crea un progetto Windows Form in Visual C# questi aggiunge un Form al progetto, denominandolo **Form1**.

Ci sono due file che rappresentano il Form e sono denominati **Form1.cs** e **Form1.designer.cs**.

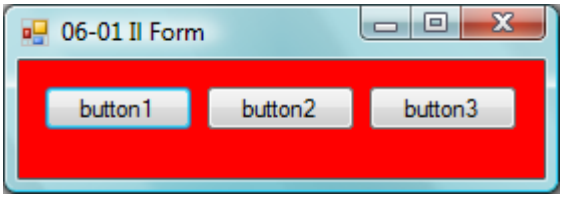
Nel file Form1.cs è possibile scrivere il codice personalizzato, mentre nel file designer.cs viene prodotto e scritto automaticamente il codice che implementa tutte le azioni eseguite aggiungendo e modificando visualmente i controlli selezionati dalla Casella degli strumenti.

È possibile aggiungere un ulteriore nuovo Form scegliendo dal menu Progetto l'opzione Aggiungi Windows Form. Ad ogni Form sono associati i rispettivi due file già descritti.

Il file denominato Form1.cs (che il progettista può rinominare in qualsiasi altro modo) contiene il codice sorgente che viene scritto per configurare il Form e i relativi controlli, ad esempio caselle di riepilogo e caselle di testo, e risponde agli eventi quali selezioni di pulsanti e sequenze di tasti. Se un progetto è abbastanza semplice, quasi tutte le operazioni di codifica vengono attuate solo in questo file.

Il file designer.cs contiene il codice sorgente generato dalla progettazione del Form sia quando si modifica il suo aspetto che quando si inseriscono o si modificano i controlli nel Form, anche impostando le proprietà nella finestra Proprietà. Generalmente, non è necessaria alcuna modifica manuale in questo file.

PROGETTO GUIDATO (PROPRIETÀ)



- Si prepari un Form1 simile alla figura:
- Si faccia un doppio clic sul Form, ma non sui pulsanti né sul bordo né sul titolo: appare un gestore di evento simile al seguente:

```
private void Form1_Load(object sender, EventArgs e)
{
      BackColor = Color.Aqua;
      Text = "Dante";
      button1.Text = "Alighieri";
      button2.Visible = false;
      MessageBox.Show("testo del messaggio", "titolo del messaggio",
            MessageBoxButtons.RetryCancel,
            MessageBoxIcon.Warning,
            MessageBoxDefaultButton.Button3,
            MessageBoxOptions.DefaultDesktopOnly           );
      button1.Enabled = false;
      button2.Enabled = false;
      button3.Enabled = false;
}
```

- Ritorniamo alla finestra e selezioniamo il Form; nella finestra delle proprietà appaiono le diverse proprietà del Form; tra queste individua e modifica le seguenti:
  - **Icon** carica un'immagine a piacere;
  - **FormBorderStyle** None
  - **StartPosition** CenterScreen
- Questa è una buona occasione per modificare le proprietà che ti incuriosiscono, verificare l'effetto e controllare se sia possibile modificarle anche da programma (per es. col codice dei pulsanti)
- Adesso osserva che la finestra delle proprietà mostra anche un pulsante di controllo, in alto, con un fulmine: è la paletta dei gestori di evento: clic e osserva come muta la finestra della proprietà
- In particolare nota che uno degli eventi, Load, ha un valore Form1_Load, che significa che l'evento è agganciato ad un gestore; in effetti quando abbiamo fatto doppio clic e scritto l'evento, l'ambiente ha creato il gestore di evento e lo ha agganciato all'evento e significa che alla partenza il Form invoca un evento da gestire
- Prova il progetto

GLI EVENTI DEL FORM

Per evento, come accennato all'inizio della dispensa, intendiamo una situazione che il controllo è in grado di rilevare e di gestire.

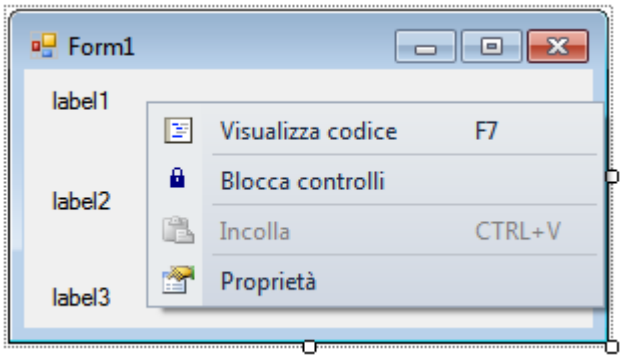
Gli eventi sono elencati nella finestra delle proprietà, in un'altra paletta, come mostrato nella figura qui di lato. Il pulsante appare così:

Gli eventi principali del Form sono:

EVENTI	DESCRIZIONE
Load	Quando si avvia il Form (caricamento)
Click	Quando si fa clic sulla finestra
KeyDown	Quando si preme un tasto della tastiera
KeyPress	Quando si tiene premuto un tasto della tastiera
KeyUp	Quando si rilascia un tasto della tastiera
FormClosing	Quando si tenta di chiudere il Form

PROGETTO GUIDATO (EVENTI)

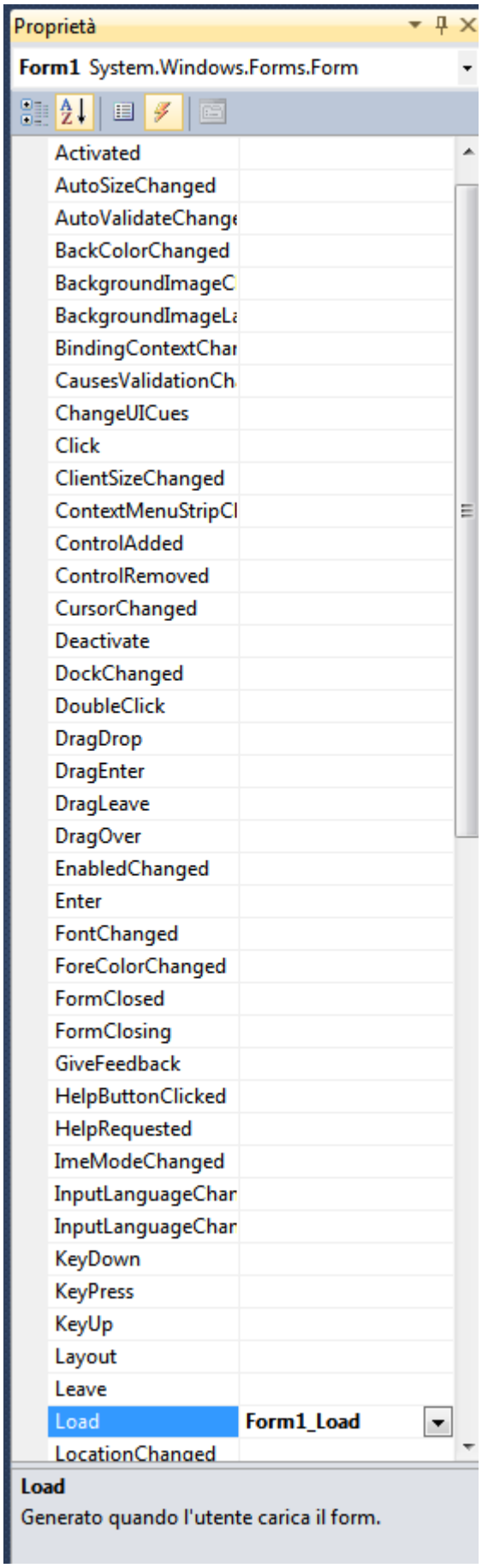
- Si prepari un Form1 (col clic destro sul Form è possibile far comparire il menu contestuale per visualizzare sia il codice che le proprietà);



- Doppio clic sul Form1 per predisporre il metodo Form_Load
- Nel metodo scrivere:

```
BackColor = Color.Red;
```

Dalla finestra delle proprietà del Form1 apri la paletta degli eventi



- Doppio clic su evento Click e nel metodo scrivere:

```
BackColor = Color.White;
```

- Doppio clic su evento KeyDown e nel metodo scrivere:

```
Text = "tasto premuto";
label1.Text =    Convert.ToString(e.KeyCode);
label2.Text =    Convert.ToString(e.KeyData);
label3.Text =    Convert.ToString(e.KeyValue);
```

- Doppio clic su evento KeyPress e nel metodo scrivere:

```
MessageBox.Show
            (Convert.ToString(e.KeyChar);
```

- Doppio clic su evento KeyUp e nel metodo scrivere:

```
Text = "tasto lasciato";
label1.Text =    Convert.ToString(e.KeyCode);
label2.Text =    Convert.ToString(e.KeyData);
label3.Text =    Convert.ToString(e.KeyValue);
```

- Doppio clic su evento FormClosing e nel metodo scrivere:

```
MessageBox.Show("si chiude!");
```

- Salva e avvia il progetto
- Fai clic sulla finestra in un qualsiasi punto;
- Mentre è in funzione prova a premere i seguenti tasti
  - è (tasto con la E accentata)
  - 1 (dalla fila di tasti in alto)
  - 2 (dal tastierino numerico)
  - Tasto Alt (in basso a sinistra)
  - Tasto Alt + Maiuscolo (insieme)
  - Tasto Alt + Ctrl (insieme)
- A volte mantieni premuti i tasti, mentre altre volte rilasciali velocemente;
- Infine chiudi col mouse

AGGANCIARE GESTORI DI EVENTI

PROSECUZIONE DEL PROGETTO GUIDATO

- Si ritorni al progetto appena realizzato e si osservi la finestra delle proprietà, che adesso mostra gli eventi:
- Si scelga l'evento KeyDown e si faccia doppio clic affianco
- Appare un gestore di evento, in cui scrivere il codice proposto:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
     bool premutoAlt = e.Alt;
     bool premutoCtrl = e.Control;
     int tastoPremuto = e.KeyValue;
     string frase = "Stai premendo: ";     //NB: lo spazio !!!!
```

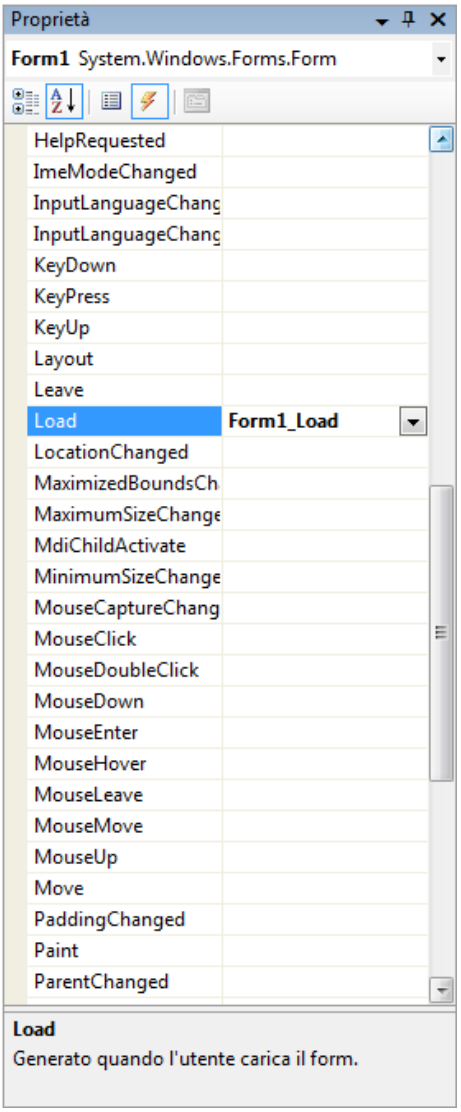
```
        if (premutoAlt)
              frase += "Alt+";
        if (premutoCtrl)
              frase += "Ctrl+";
        frase += tastoPremuto;
        Text = frase;      /*visualizza la frase */
                           /*nel titolo del Form */
}
```

- Prova il progetto
- Si scelga l'evento KeyPress e si faccia doppio clic affianco, e scrivi il codice proposto:

```
private void Form1_KeyPress(object sender,
KeyPressEventArgs e)
{
      char tastoPremuto = e.KeyChar;
      string frase =
                  Convert.ToString(tastoPremuto);
      MessageBox.Show("Hai premuto:" + frase);
}
```

- Prova il progetto

Osserva che la lettera e nei gestori ha un significato speciale!



IL GESTORE PREFERITO

Ogni controllo ha un suo evento preferito. L'evento preferito è quello che l'ambiente prepara quando si esegue doppio clic sul controllo. Per esempio col doppio clic sul Form, si apre il gestore di evento per Load_Form, che è l'evento preferito di Form.

Ovviamente altri controlli hanno altri eventi preferiti.

AVVIO E CHIUSURA DELLA FINESTRA

Quando si avvia o si chiude la finestra scattano diversi eventi che è possibile gestire. Per esempio il gestore Form_Load consente di intercettare la creazione della finestra Form) e di eseguire istruzioni prima che compaia a video. Attento però, che non tutti i controlli sono stati creati e quindi potrebbero non esistere!

GESTIONE DELLA TASTIERA

La gestione della tastiera avviene mediante gli eventi del Form. Quando il Form è attivo può ricevere i segnali provenienti dalla tastiera (interruzioni). Visual C# propone tre diversi gestori di eventi per gestire la tastiera che sono:

EVENTI	DESCRIZIONE
KeyDown	Quando si preme un tasto della tastiera
	Rileva anche tasti speciali, come Alt, Ctrl, Shift, e altri
KeyPress	Quando si tiene premuto un tasto della tastiera
	Non rileva tasti speciali se non sono premuti insieme tasti ordinari
KeyUp	Quando si rilascia un tasto della tastiera
	Rileva anche tasti speciali, come Alt, Ctrl, Shift, e altri

EVENTI DEL MOUSE

Oltre alla gestione della tastiera, ovviamente è possibile gestire il mouse. Gli eventi offerti che analizzeremo sono:

EVENTI	DESCRIZIONE
MouseDown	Quando si fa click sinistro del mouse e lo si tenga premuto
Click	Quando avviene un generico clic del mouse
MouseClick	Viene generato quando il controllo viene selezionato mediante il mouse.
MouseUp	Quando si fa lascia il tasto sinistro del mouse dopo averlo premuto
MouseEnter	Quando il mouse entra nella zona del controllo visuale
MouseHover	Quando il mouse sosta (abbastanza) nella zona del controllo visuale
MouseLeave	Quando il mouse esce dalla zona del controllo visuale

Movimento del cursore

Gli eventi mouse associati al suo movimento e alle sue azioni si innescano solitamente nel seguente ordine:

1. Evento MouseEnter
2. Evento MouseMove
3. Evento MouseHover / MouseDown / MouseWheel
4. Evento MouseUp
5. Evento MouseLeave

Pressione dei tasti

La pressione di un pulsante del mouse quando il cursore è posizionato su un controllo genera, di solito, una serie di eventi nell'ordine riportato di seguito.

1. Evento MouseDown.
2. Evento Click.
3. Evento MouseClick.
4. Evento MouseUp.

Affinché questa serie si verifichi, non è possibile disabilitare i vari eventi nella classe del controllo.

Due singoli clic che si verificano in rapida sequenza, come determinato dalle impostazioni del mouse del sistema operativo dell'utente, genereranno un evento MouseDoubleClick anziché un secondo evento MouseClick.

MouseEnter

L'Evento `Control.MouseEnter` si verifica quando il puntatore del mouse entra nell'area del controllo. Questo evento è comune a tutti i controlli perciò può essere associato al Form, ma anche a Button, textBox e altri.

Esempio (associate all'evento MouseEnter del Form1)

```
private void Form1_MouseEnter(object sender, EventArgs e)
{
    BackColor = Color.Pink; //colora di rosa se il puntatore va sul Form1
}
```

MouseMove

L'Evento `Control.MouseMove` si verifica quando il puntatore del mouse subisce uno spostamento restando sul controllo. Questo evento è comune a tutti i controlli.

Esempio (associate all'evento MouseMove del Form1)

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    BackColor = Color.Red;    //colora di rosso se il cursore si muove
    Top = e.X;                //usa l'argomento di tipo MouseEventArgs
    Left = e.Y;               //usa l'argomento di tipo MouseEventArgs
}
```

MouseHover

L'Evento `Control.MouseHover` si verifica quando il puntatore del mouse permane abbastanza a lungo sopra il controllo. Questo evento è comune a tutti i controlli.

Esempio (associate all'evento MouseHover del Form1)

```
private void Form1_MouseHover(object sender, EventArgs e)
{
    BackColor = Color.Aqua; //colora di azzurro se permane sulla finestra
}
```

MouseDown

L'Evento `Control.MouseDown` si verifica quando il puntatore del mouse si trova sul controllo mentre viene premuto un pulsante del mouse. Questo evento è comune a tutti i controlli.

Esempio (associate all'evento MouseDown del Form1)

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button.ToString() == "Left") //usa l'argomento di tipo MouseEventArgs
        BackColor = Color.Tan; //colore per tasto sinistro
    else
        BackColor = Color.Lime; //colore per tasto de4stro
}
```

MouseUp

L'Evento `Control.Up` si verifica quando si rilascia il tasto del mouse restando sopra il controllo. Questo evento è comune a tutti i controlli.

Esempio (associate all'evento MouseUp del Form1)

```
private void Form1_MouseUp(object sender, MouseEventArgs e)
{
   BackColor = Color.Black; //colora di nero al rilascio del tasto
}
```

MouseLeave

L'Evento `Control.Leave` si verifica quando si rilascia il tasto del mouse restando sopra il controllo. Questo evento è comune a tutti i controlli.

Esempio (associate all'evento MouseLeave del Form1)

```
private void Form1_MouseLeave(object sender, EventArgs e)
{
   BackColor = Color.Teal; //colora quando il mouse esce dal controllo
}
```

PARAMETRI DEI GESTORI DI EVENTO

Osserva attentamente i gestori degli eventi del mouse. Puoi notare che nell'intestazione tra le parentesi alcuni di essi riportano la scritta **EventArgs** ed altri la scritta **MouseEventArgs**? Ebbene queste due parole contraddistinguono un particolare parametro di nome e che puoi usare nel blocco dei metodi gestori.

EVENTI	DESCRIZIONE
`EventArgs e`	Gli eventi con questo parametro di solito non specificano elementi che interessano specificamente il mouse. Sono gestori utili ma non riescono a gestire casi particolari.
`MouseEventArgs e`	Gli eventi con questo parametro possono aiutarti a gestire nel dettaglio elementi che interessano specificamente il mouse. Sono gestori che possono avere cura di casi particolari.

Gli eventi MouseUp, MouseDown, e MouseMove possiedono il parametro di tipo `MouseEventArgs` e possono usarlo per ottenere dettagli sul mouse. La lettera `e` è una locazione di questo tipo e può essere usata con la notazione puntata consueta; vediamo come.

MouseEventArgs

Ripetiamo che la locazione `e` rappresenta un pacchetto di informazioni relative alla gestione del mouse; per poter accedere queste informazioni si usano solitamente alcune interessanti proprietà tra cui:

Proprietà	Descrizione
e.Button	Restituisce quale una stringa che indica quale tasto del mouse è stato premuto. I valori che può rendere sono «Left», «Right», «Middle»
e.Clicks	Rende il numero di volte che il tasto è stato premuto e rilasciato. Se un evento interrompe i click spesso rende 1.
e.Delta	Rende un intero con segno che indica quanti scatti ha compito la rotellina del mouse, moltiplicati per la costante WHEEL_DELTA. Uno scatto è un passo della rotellina del mouse che puoi percepire mentre la ruoti avanti o indietro.
e.Location	Rende la posizione del mouse quando ha generato l'evento.
e.X	Rende la coordinata X del mouse quando ha generato l'evento.
e.Y	Rende la coordinata Y del mouse quando ha generato l'evento.

Per esempio per verificare quale tasto è stato premuto possiamo scrivere un codice come:

```
MessageBox.Show (e.Button.ToString() );
```

il codice precedente può essere inserito in un gestore di evento come MouseUp, MouseDown, e MouseMove associato a controlli come un Button o il Form1.

Per esempio per sapere il punto esatto in cui è stato premuto il mouse possiamo scrivere:

```
MessageBox.Show ("Coordinate: "+e.X+" ; "+e.Y);
```

anche il codice precedente può essere inserito in un gestore di evento come MouseUp, MouseDown, e MouseMove associato a controlli come un Button o il Form1.

GESTIRE IL SEMPLICE CLIC

Click

L'evento `Control.Click` è uno degli eventi più comunemente usati. Nelle dispense precedenti era il gestore preferito del button, ma anche il Form può usarlo.

Questo evento è comune a tutti i controlli. L'evento è generato quando si fa clic sul controllo.

L'evento Click passa un semplice oggetto EventArgs al relativo gestore eventi; pertanto, al gestore viene soltanto indicato che si è verificato un clic. Per poter avere più specifiche informazioni relative al modo con cui l'utente ha usato il mouse (ad esempio quale tasto ha premuto, se destro o sinistro) conviene utilizzare l'evento MouseClick.

Attenzione però: l'evento MouseClick non verrà generato se il clic è determinato da un'azione diversa da quella del mouse, ad esempio dalla pressione del tasto INVIO. Per generare questo evento, è necessario impostare il valore StandardClick di ControlStyles su true.

Esempio (associate all'evento Click del Form1)

```
void Form1_Click(object sender, EventArgs e)
{
  Width -= 10; //riduce la larghezza della finestra
}
```

GESTIRE IL DOPPIO CLIC

Un doppio-clic è determinato dalle impostazioni relative al mouse del sistema operativo dell'utente. L'utente può impostare un intervallo tra i clic effettuati con un pulsante del mouse in modo che debbano essere considerati come un doppio clic anziché come due clic distinti. Ogni volta che l'utente fa doppio clic su un controllo, verrà generato l'evento Click. Se, ad esempio, si dispone di gestori eventi per gli eventi Click e DoubleClick di un oggetto Form, gli eventi Click e DoubleClick vengono generati quando si fa doppio clic sul Form e vengono chiamati entrambi i metodi. Se l'utente fa doppio clic su un controllo che non supporta l'evento DoubleClick, è possibile che l'evento Click venga generato due volte.

DoubleClick

L'evento `Control.DoubleClick` è usato di frequente. Si verifica quando si fa doppio clic sul controllo. Questo evento è comune a tutti i controlli.

Anche l'evento DoubleClick passa un semplice oggetto EventArgs al relativo gestore eventi e pertanto, al gestore viene soltanto indicato che si è verificato un clic.

Esempio (associate all'evento DoubleClick del Form1)

```
void Form1_DoubleClick(object sender, EventArgs e)
{
   Height += 25; //aumenta la altezza della finestra
}
```

DRAG AND DROP

Drag & Drop significa letteralmente **trascina e rilascia**; questa tecnica permette di prendere un elemento visuale e, mantenendo il mouse premuto, trascinarlo fino alla destinazione, dove rilasciarlo. Il rilascio dell'elemento dovrebbe causare un effetto, generalmente una scomparsa dalla sorgente e una comparsa nella destinazione.

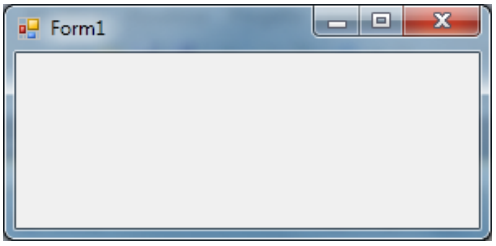
Per utilizzare la tecnica del Drag & Drop in un programma C# si devono compiere numerosi passi: preparare il controllo di destinazione, gestire il metodo **MouseDown** del controllo di partenza, gestire il metodo **DragEnter** del controllo di destinazione.

Qui di seguito saranno proposti diversi esempi di utilizzo dei gestori degli eventi discussi finora. In particolare si illustra uno tra i possibili esempi di gestione del DragDrop che però può anche essere usato con diverse modalità. Purtroppo lo studente giunto fino a questa parte del corso si suppone abbia una competenza ancora troppo modesta per poter discutere con maggiore profondità e precisione la gestione degli eventi e degli oggetti ad essi correlati.

Verso la fine del corso, comunque, affronteremo la questione con maggiore dettaglio.

ESERCITAZIONI GUIDATE

ESERCITAZIONI CLIC DESTRO E SINISTRO



- Si prepari un Form1 come nella figura:
- Si selezioni il Form1 e si visualizzino gli eventi nella finestra delle Proprietà
- Si faccia doppio clic sull'evento MouseClick
- Nel gestore di evento si scriva il seguente codice:

```
private void Form1_MouseClick(object sender, MouseEventArgs e)
 {
      Char aCapo = Convert.ToChar(13);
      string tastoMouse = e.Button.ToString();
      string volte = e.Clicks.ToString();
      string coordinataX = "" + e.X;
      string coordinataY = "" + e.Y;
      string messaggio = "Hai premuto: " + aCapo + tastoMouse;
      messaggio += aCapo + volte + " volte ";
      messaggio += aCapo + "nel punto: " + coordinataX + ";" + coordinataY;
      textBox1.Text = messaggio;
      textBox2.Text = "click sul Form1";
      if (tastoMouse == "Left")
            BackColor = Color.Blue;
      else
            BackColor = Color.Red;
 }
```

- Si provi il progetto, poi lo si termini e si torni in fase di progettazione.

ESERCITAZIONI ENTRATA E USCITA

- Si prepari un Form1 come nell'esercizio precedente.
- Si selezioni il Form1 e si visualizzino gli eventi nella finestra delle Proprietà
- Si faccia doppio clic sull'evento MouseEnter
- Nel gestore di evento si scriva il seguente codice:

```
private void Form1_MouseEnter(object sender, EventArgs e)
{
     BackColor = Color.Green;

}
```

- Si faccia doppio clic sull'evento MouseEnter e nel gestore di evento si scriva il codice:

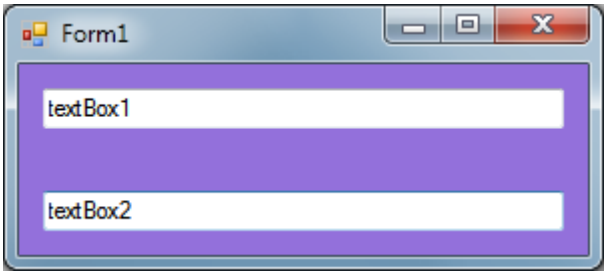
```
private void Form1_MouseEnter(object sender, EventArgs e)
{
     BackColor = Color.Green;
}
```

- Si provi il progetto, poi lo si termini e si torni in fase di progettazione.
- Si faccia doppio clic sull'evento MouseHover e nel gestore di evento si scriva il seguente codice:

```
private void Form1_MouseHover(object sender, EventArgs e)
{
      BackColor = Color.Orange;
}
```

- Si provi il progetto: spostare il mouse sulla finestra, poi fuori, e ancora. Si termini.

ESERCITAZIONI DRAG AND DROP



- Si prepari un Form1 con due caselle di testo rispettivamente textBox1 e textBox2.
- Per la casella textBox1 dobbiamo associare un gestore per l'evento **MouseDown** (doppio clic nella paletta eventi) e scriverci il codice seguente:

```
private void textBox1_MouseDown(object
sender, MouseEventArgs e)
{
      textBox1.DoDragDrop(textBox1.Text, DragDropEffects.Copy);
}
```

- La funzione DoDragDrop appartiene alla classe Control e come primo parametro accetta i dati da usare nell'operazione di Drag & Drop, come secondo la modalità con cui dovrà usarli, in questo caso Copy.
- Nella casella che riceverà i dati (textBox2) dobbiamo impostare la proprietà AllowDrop a True (possiamo farlo design-time).
- Ora dobbiamo gestire la situazione che si verifica quando il mouse entra nel textBox2; scegliamo dalla paletta eventi del textBox2 l'evento **DragEnter**. Il gestore dovrà controllare che i dati siano di tipo stringa, altrimenti impostare come operazione di Drag & Drop None, per non incorrere in errori:

```
private void textBox2_DragEnter(object sender, DragEventArgs e)
{
      if (e.Data.GetDataPresent(DataFormats.StringFormat))
      {
            e.Effect = DragDropEffects.Copy;
      }
      else
      {
            e.Effect = DragDropEffects.None;
      }
}
```

- Infine provvediamo all'effettiva scrittura dei dati nel textBox2; scegliamo dalla paletta eventi del textBox2 l'evento **DragDrop** e scriviamo il seguente codice:

```
private void textBox2_DragDrop(object sender, DragEventArgs e)
{
      String str = (String) e.Data.GetData(DataFormats.StringFormat);
      textBox2.Text = str;
}
```

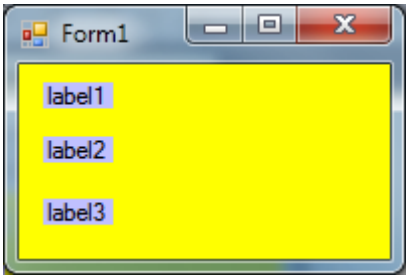
- Prova a lanciare in esecuzione il programma e ad usarlo scrivendo del testo nella prima casella per poi trascinarlo nella seconda casella.

ESERCIZI

ESERCIZI SU: GESTIONE DELLA TASTIERA

GESTIONE DELLA TASTIERA

- Prepara un Form con tre etichette (label).
- Associa a ciascuno dei tre metodi `KeyDown`, `KeyPress`, `KeyUp` rispettivamente tre gestori ciascuno dei quali visualizza in una etichetta l'esito rilevato dall'evento.

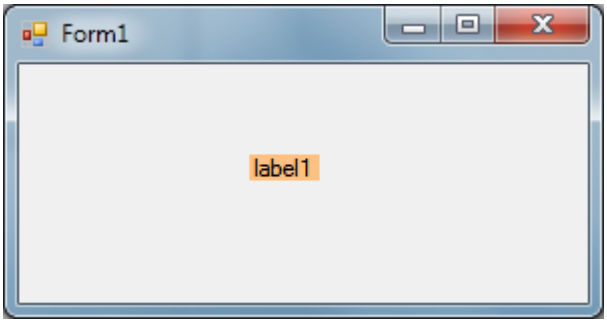


MOVIMENTO DA TASTIERA

- Prepara un Form con una etichetta (label1).
- Costruisci una applicazione che, quando l'utente preme le frecce della tastiera, la etichetta si muove nella direzione indicata di 1 pixel.

MOVIMENTO DA MOUSE

- Prepara un Form con una etichetta (label1).
- Costruisci una applicazione che, quando l'utente il mouse sulla etichetta questa si muove casualmente di 10 pixel in entrambe le direzioni verticali ed orizzontali.
- Quando l'utente preme il mouse sulla finestra, l'etichetta si muove per avvicinarsi al mouse.



TASTI DESTRO E SINISTRO

- Prepara un Form con una etichetta (label1).
- Costruisci una applicazione che, quando l'utente preme il tasto destro del mouse l'etichetta va a sinistra, ma se preme il sinistro l'etichetta va a destra
- Se l'utente usa la rotella a salire allora l'etichetta va in alto, ma se la usa a scendere allora l'etichetta va in basso

SOMMARIO

IL FORM 2
UNA FINESTRA DA SCHIUDERE 2
Il Form 2
Riflessioni sul Form e sugli eventi 2
File del Form 3
Progetto guidato (proprietà) 4
Gli eventi del Form 5
Progetto guidato (eventi) 5
AGGANCIARE GESTORI DI EVENTI 6
Prosecuzione del Progetto guidato 6
Il gestore preferito 7
Avvio e chiusura della finestra 7
Gestione della tastiera 7
Eventi del mouse 8
Parametri dei gestori di evento 10
Gestire il semplice clic 11
Gestire il doppio clic 12
Drag and drop 12
ESERCITAZIONI GUIDATE 13
Esercitazioni clic destro e sinistro 13
Esercitazioni entrata e uscita 13
Esercitazioni Drag and drop 14
ESERCIZI SU: GESTIONE DELLA TASTIERA 15
Gestione della tastiera 15
Movimento da tastiera 15
Movimento da Mouse 15
Tasti destro e sinistro 15