

04-07_Keyevents

Stampa del: lunedì 3 febbraio 2014

04.07

GESTIONE DELLA TASTIERA

Microsoft®
Visual Studio®

Andrea Zoccheddu

CORSO INFORMATICA ITI ANGIOY SASSARI



Sintesi

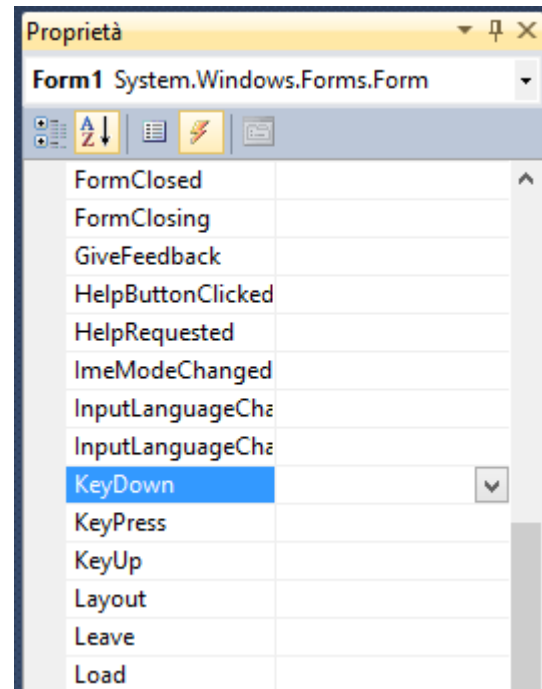
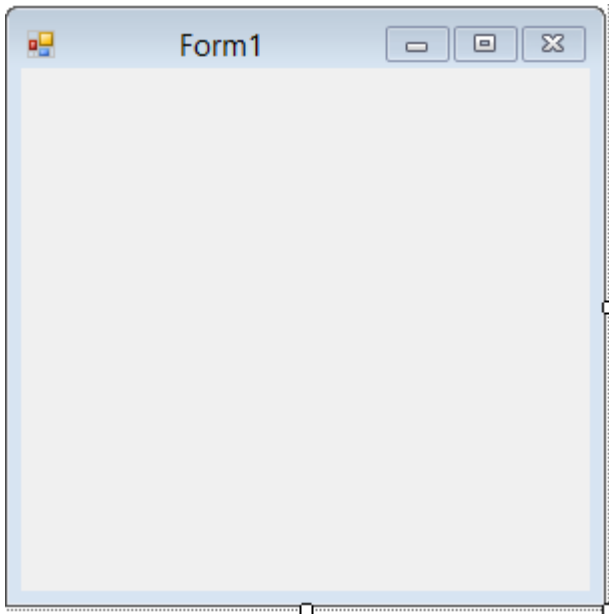
In questa dispensa si approfondisce il funzionamento del Form e la gestione di mouse e tastiera.

GLI EVENTI KEY

INTERCETTARE LA TASTIERA DAL FORM

PROGETTO GUIDATO

- Si prepari un Form1 vuoto:



- Si apra l'elenco dei gestori di evento (si osservi che la figura a destra mostra l'elenco dei gestori di evento associati al Form1, che è l'unico controllo esistente nel progetto).
- Si individui il gestore di evento **KeyDown** e si faccia doppio clic per associarvi un gestore di evento, in cui scrivere il codice seguente:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    int codice = e.KeyValue;
    string ascii = "" + codice;
    Text = "Hai premuto: " + codice;
}
```

- Si lanci in esecuzione l'applicazione e si provi a premere tasti sulla tastiera. Si osservino i codice che si ottengono premendo lettere minuscole e maiuscole, i numeri in alto e i numeri del tastierino numerico sulla destra. Si prenda nota delle differenze.
- Si ritorni in fase di progettazione e si modifichi il codice del gestore di evento **KeyDown** col codice seguente:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    int codice = e.KeyValue;
    string ascii = "" + codice;
    MessageBox.Show("DOWN - Hai premuto: " + codice);
}
```

- Adesso si individui il gestore di evento **KeyPress** e si faccia doppio clic per associarvi un gestore di evento, in cui scrivere il codice seguente:

```
private void Form1_KeyPress(object sender, KeyPressEventArgs e)
{
    int codice = e.KeyChar;
    string ascii = "" + codice;
    MessageBox.Show("PRESS - Hai premuto: " + codice);
}
```

- Si lanci in esecuzione l'applicazione e si provi a premere tasti sulla tastiera. Si osservino i codice che si ottengono premendo lettere minuscole e maiuscole, i numeri in alto e i numeri del tastierino numerico sulla destra. Si prenda nota delle differenze.
- Si noti che la precedenza degli eventi è:

(1) KEYPRESS → (2) KEYDOWN

- Si ritorni in fase di progettazione e si individui il gestore di evento **KeyUp** e si faccia doppio clic per associarvi un gestore di evento, in cui scrivere il codice seguente:

```
private void Form1_KeyUp(object sender, KeyEventArgs e)
{
    int codice = e.KeyValue;
    string ascii = "" + codice;
    MessageBox.Show("UP - Hai premuto: " + codice);
}
```

- Si lanci in esecuzione l'applicazione e si provi a premere tasti sulla tastiera. Si osservino i codice che si ottengono premendo lettere minuscole e maiuscole, i numeri in alto e i numeri del tastierino numerico sulla destra. Si prenda nota delle differenze.
- Si noti che la precedenza degli eventi è:

(1) KEYPRESS → (2) KEYDOWN → (3) KEYUP

COMMENTO AL CODICE

La prima osservazione è che il Form1 prevede tre diversi gestori di evento per rilevare i tasti dalla tastiera: (1) KEYPRESS □ (2) KEYDOWN □ (3) KEYUP

Questi eventi sono scatenati in un preciso ordine quando si preme un tasto:

- 1) KeyPress
- 2) KeyDown
- 3) KeyUp

Inoltre gli eventi prevedono tutti un elemento speciale (un parametro) che è individuato dalla lettera e; quando usi il simbolo e seguito da un punto, il RAD offre tutte le scelte possibili per questo elemento.

Il simbolo e rappresenta tutte le informazioni sul tasto premuto, a seconda del tipo di evento che si sta gestendo.

Quando si è dentro il gestore di evento di KeyDown e KeyUp è possibile rilevare il codice del tasto premuto usando e.KeyValue.

ESERCIZIO

Si associ al gestore KeyDown un codice che colora la finestra nel seguente modo:

- se e.KeyValue è 37 allora colora di rosso
- se e.KeyValue è 38 allora colora di verde

- se e.KeyValue è 39 allora colora di blu
- se e.KeyValue è 40 allora colora di giallo
- quando si prova il progetto si usino i tasti freccia della tastiera

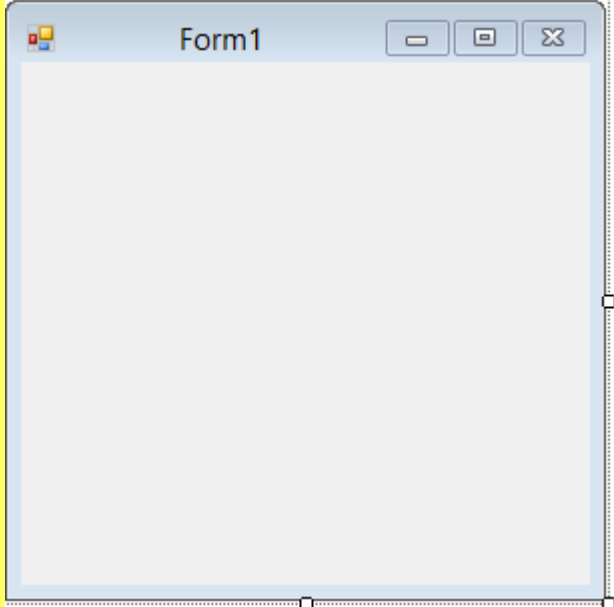
PROGETTO GUIDATO (IL PARAMETRO DI TIPO KEYEVENTARGS)

- Si prepari un Form1 vuoto come nella figura sottostante
- Si apra l'elenco dei gestori di evento e si individui il gestore di evento **KeyDown** e si faccia doppio clic per associarvi un gestore di evento, in cui scrivere il codice seguente:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    bool alt = e.Alt;
    bool ctr = e.Control;
    Keys cod = e.KeyCode;
    Keys dat = e.KeyData;
    int val = e.KeyValue;
    Keys mod = e.Modifiers;
    bool shf = e.Shift;
    string frase = "";
    if (alt)
        frase += "ALT+";
    if (ctr)
        frase += "CTRL+";
    if (shf)
        frase += "SHIFT+";
    Text = "Hai premuto: "
        + frase + val;

    //
    if (cod == Keys.CapsLock)
        BackColor = Color.Red;
    if (cod == Keys.Escape)
        BackColor = Color.Blue;
    if (dat == Keys.F1)
        BackColor = Color.Yellow;
    if (mod == Keys.Insert)
        BackColor = Color.Green;
}

```



- Si lanci in esecuzione l'applicazione e si provi a premere tasti sulla tastiera.
- Si premano anche i tasti ESC (in alto a sinistra), il tasto CAPSLOCK (blocca maiuscole sulla sinistra), il tasto F1 (funzione 1 in alto sopra i numeri), il tasto INS (modo inserimento / sovrascrittura sulla destra)
- Si prema anche la combinazione dei tasti ALT+A o simili
- Si prema anche la combinazione dei tasti CTRL+A o simili
- Si prema anche la combinazione dei tasti SHIFT+A o simili
- Si prema anche la combinazione dei tasti ALT+SHIFT+A o simili
- Si prema anche la combinazione dei tasti ALT+CTRL+A o simili
- Si prema anche la combinazione dei tasti ALT+CTRL+ SHIFT+A o simili

COMMENTO AL CODICE

La prima osservazione è sul codice del gestore di evento dove si notano le dichiarazioni seguenti:

```
bool alt = e.Alt;
bool ctr = e.Control;
Keys cod = e.KeyCode;
Keys dat = e.KeyData;
int val = e.KeyValue;
Keys mod = e.Modifiers;
bool shf = e.Shift;
```

Il parametro `e` propone diverse proprietà tra cui nell'esempio si sono usate `Alt` (che rende il valore `true` se si è premuto il tasto `Alt` anche in combinazione con altri tasti), `Control` (che rende il valore `true` se si è premuto il tasto `Control` anche in combinazione con altri tasti), `Shift` (che rende il valore `true` se si è premuto il tasto `Shift` anche in combinazione con altri tasti).

La proprietà `KeyCode` rende un valore speciale di tipo `Keys` che può essere confrontato con tutti i possibili valori della classe `Keys`, come si è fatto per esempio nel controllo:

```
if (cod == Keys.CapsLock)
    BackColor = Color.Red;
```

Analogamente è possibile usare le proprietà `KeyData` e `Modifiers`.

Infine la proprietà `KeyValue` che abbiamo già discusso all'inizio del capitolo.

ESERCIZIO

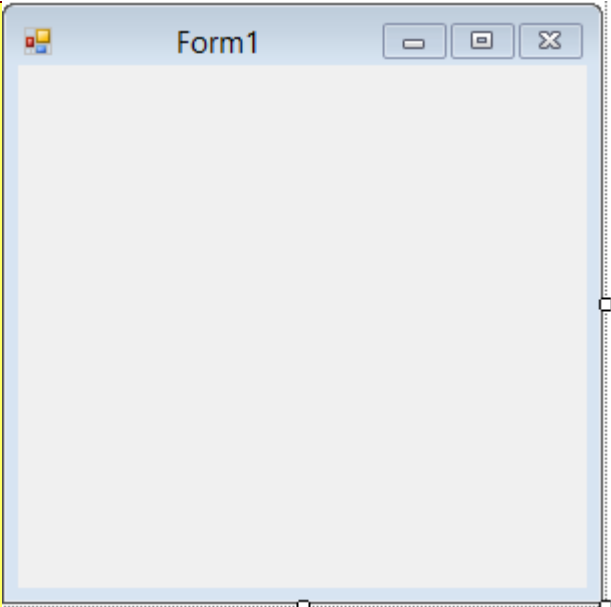
Si associ al gestore `KeyDown` un codice che sposta la finestra nel seguente modo:

- se `e.KeyValue` è 37 allora va verso sinistra di un pixel
- se `e.KeyValue` è 38 allora va verso l'alto di un pixel
- se `e.KeyValue` è 39 allora va verso destra di un pixel
- se `e.KeyValue` è 40 allora va verso il basso di un pixel
- se si preme uno dei precedenti tasti in combinazione con `Control` allora lo spostamento è di 5 pixel
- se si preme uno dei precedenti tasti in combinazione con `Alt` allora lo spostamento è di 10 pixel
- se si preme uno dei precedenti tasti in combinazione con `Shift` allora lo spostamento è di 20 pixel
- se si preme uno dei precedenti tasti in combinazione con `Alt+Control` allora lo spostamento è di 30 pixel
- se si preme uno dei precedenti tasti in combinazione con `Shift+Control` allora lo spostamento è di 100 pixel

PROGETTO GUIDATO

- Si prepari un Form1 vuoto come nella figura sottostante
- Si apra l'elenco dei gestori di evento e si individui il gestore di evento **KeyDown** e si faccia doppio clic per associarvi un gestore di evento, in cui scrivere il codice seguente:

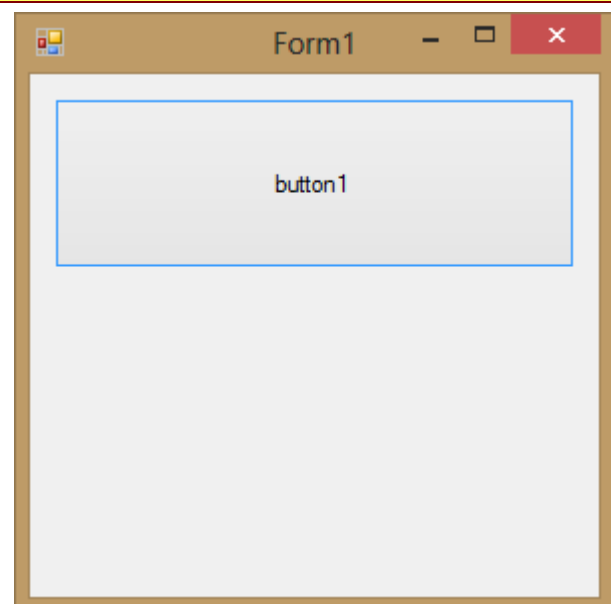
```
private void Form1_KeyDown(object sender,
{
    bool alt = e.Alt;
    bool ctr = e.Control;
    Keys cod = e.KeyCode;
    Keys dat = e.KeyData;
    int val = e.KeyValue;
    Keys mod = e.Modifiers;
    bool shf = e.Shift;
    string frase = "";
    if (alt)
        frase += "ALT+";
    if (ctr)
        frase += "CTRL+";
    if (shf)
        frase += "SHIFT+";
    Text = "Hai premuto: " + frase + val;
    //
    if (cod == Keys.CapsLock)
        BackColor = Color.Red;
    if (cod == Keys.Escape)
        BackColor = Color.Blue;
    if (dat == Keys.F1)
        BackColor = Color.Yellow;
    if (mod == Keys.Insert)
        BackColor = Color.Green;
}
```



- Si lanci in esecuzione l'applicazione e si provi a premere tasti sulla tastiera.
- Adesso si torni in fase di progettazione e si aggiunga un button1 al form1 come nella seguente figura:
- Si lanci in esecuzione l'applicazione e si provi a premere tasti sulla tastiera. **NON FUNZIONA PIÙ?**

COMMENTO AL CODICE

Come si è potuto constatare il codice di rilevamento della tastiera non ha funzionato quando nella finestra è stato inserito un pulsante. Il motivo è che il flusso della tastiera è canalizzato verso il controllo che ha il focus in quel momento. Ma cos'è un focus? È la capacità di intercettare gli eventi standard, nel nostro caso della tastiera.



Quando in un Form1 si inserisce un pulsante, normalmente lo si fa per consentire all'utente di usarlo per inserire i dati (mouse e tastiera); quindi per default il RAD conferisce al button1 il focus dell'applicazione e il flusso è intercettato da questo pulsante.

Purtroppo gli eventi che abbiamo usato (KeyDown) è un gestore associato al form1 e quindi si attiva solo quando il focus è sul form1, ma non se è su un altro controllo, per esempio button1.

PROGETTO GUIDATO

- Si torni al Form1 col pulsante e si selezioni il button1
- Si apra l'elenco delle proprietà di button1 e si imposti la sua proprietà Enabled a false
- Si provi di nuovo il progetto

COMMENTO AL CODICE

Nella ultima modifica al progetto si può constatare che il programma funziona nuovamente; il motivo è che quando un controllo non è abilitato (ha la proprietà Enabled a false) non può ricevere il focus che quindi passa necessariamente al primo controllo disponibile, che in questo caso è nuovamente il Form1.

Quindi il flusso della tastiera è intercettato dal Form1 invece che dal button1.

PROGETTO GUIDATO

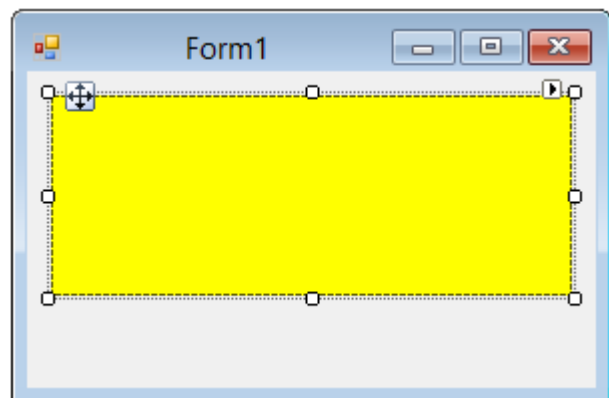
- Si torni al Form1 col pulsante e si selezioni il button1
- Si apra l'elenco delle proprietà di button1 e si imposti la sua proprietà Enabled a true
- Tra le proprietà di button1, si imposti la sua proprietà TopStop a false
- Si provi di nuovo il progetto

COMMENTO AL CODICE

Con questa ultima modifica al progetto si ottiene che fino a quando il button1 non è cliccato o selezionato allora il focus resta al Form1, ma appena si clicca il pulsante button1 il focus passa a questi e non lo lascia più!

PROGETTO GUIDATO

- Si prepari un Form1 vuoto
- Si inserisca un controllo panel1 come nella figura di lato e lo colori per evidenziarlo
- Si selezioni il Form1 e si associ al suo evento **KeyDown** un gestore di evento, col codice seguente:



```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    switch (e.KeyValue)
    {
        case 37:    Left -= 25; break;
        case 38:    Top -= 25; break;
        case 39:    Left += 25; break;
        case 40:    Top += 25; break;
    }
}
```

COMMENTO AL CODICE

In questo progetto il focus resta al Form1, poiché il panel1 non può mai ricevere un focus; di conseguenza il flusso della tastiera è intercettato in modo utile.

Per sperimentare è possibile sostituire il panel1 con una label1.

PROGETTO GUIDATO

- Si prepari un Form1 vuoto
- Si inserisca un controllo button1 come nella figura di lato
- Si selezioni il button1 e si associ al suo evento **KeyDown** un gestore di evento, col codice seguente:

```
private void button1_KeyDown(object sender, KeyEventArgs e)
{
    MessageBox.Show("" + e.KeyValue);
}
```

- Si noti che il codice somiglia al primo della dispensa, ma il gestore di evento NON è legato al Form1 bensì al button1

COMMENTO AL CODICE

- In questo progetto il focus è del button1 che può gestire il flusso della tastiera, anche se alcuni tasti non sono intercettati correttamente (si provino le frecce)

PROGETTO GUIDATO

- Si prepari un Form1 vuoto
- Si inserisca un controllo button1 come nella figura di lato
- Si selezioni il button1 e si associ al suo evento **KeyDown** un gestore di evento, col codice seguente:

```
private void button1_KeyDown(object sender, KeyEventArgs e)
{
    MessageBox.Show("" + e.KeyValue);
}
```

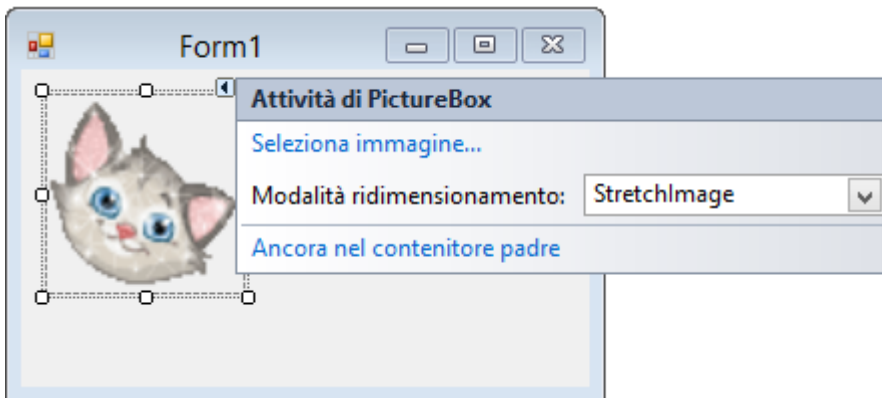
- Si noti che il codice somiglia al primo della dispensa, ma il gestore di evento NON è legato al Form1 bensì al button1

COMMENTO AL CODICE

- In questo progetto il focus è del button1 che può gestire il flusso della tastiera, anche se alcuni tasti non sono intercettati correttamente (si provino le frecce)

PROGETTO GUIDATO

- Si prepari un Form1 vuoto
- Si inserisca un controllo pictureBox1 come nella figura qui sotto



- Si carichi una immagine e si imposti il ridimensionamento a StretchImage, come nella figura
- Si selezioni il Form1 e si associ al suo evento **KeyDown** un gestore di evento, col codice seguente:

```
private void button1_KeyDown(object sender, KeyEventArgs e)
{
    Text = "" + e.KeyValue;
}
```

- Si esegua il progetto

COMMENTO AL CODICE

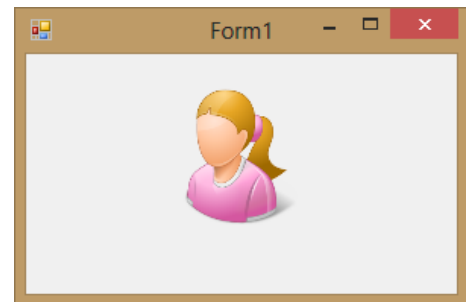
- In questo progetto il focus è del Form1 che può gestire il flusso della tastiera, poiché di base il controllo PictureBox1 non riceve il focus anche quando cliccato
-

ESERCIZI

ESERCIZI SU: GESTIONE DELLA TASTIERA

ESERCIZIO 1. MOVIMENTO DA TASTIERA I

- Prepara un Form con una PictureBox1 con una immagine a piacere (64x64 pixel).
- Costruisci una applicazione che, quando l'utente preme le frecce della tastiera, l'immagine si muova di 5 pixel nella direzione indicata dalla freccia.



ESERCIZIO 2. MOVIMENTO DA TASTIERA II

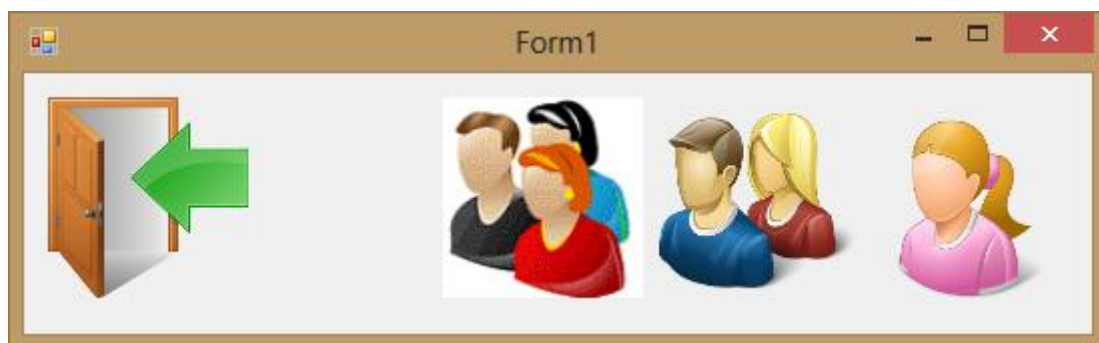
- Modificare l'applicazione precedente in modo che quando l'utente preme una freccia insieme al tasto Control, il movimento sia di 15 pixel
- Modificare l'applicazione precedente in modo che quando l'utente preme una freccia insieme al tasto Alt, il movimento sia di 1 solo pixel

ESERCIZIO 3. MOVIMENTO DA TASTIERA III

- Prepara un Form con due PictureBox con delle immagini a piacere (64x64 pixel).
- Costruisci una applicazione che, quando l'utente preme i numeri del tastierino numerico muove opportunamente la prima immagine
- Quando l'utente preme il tasto Alt la seconda immagine si sovrappone alla prima
- Quando l'utente preme il tasto Shift la prima immagine raggiunge la seconda



ESERCIZIO 4. GESTIONE DELLA TASTIERA



- Prepara un Form con quattro immagini
 - Quando l'utente preme il tasto Alt+1 una immagine si avvicina alla porta di 10 pixel
 - Quando l'utente preme il tasto Alt+2 un'altra immagine si avvicina alla porta di 10 pixel
 - Quando l'utente preme il tasto Alt+3 un'altra immagine si avvicina alla porta di 10 pixel
- Se una immagine raggiunge la porta la finestra si chiude

SOMMARIO

Gli eventi key	2
Intercettare la tastiera dal form	2
PROGETTO GUIDATO	2
Commento al codice	3
Esercizio.....	3
PROGETTO GUIDATO (il parametro di tipo EventArgs).....	4
Commento al codice	5
Esercizio.....	5
PROGETTO GUIDATO	6
Commento al codice	6
PROGETTO GUIDATO	7
Commento al codice	7
PROGETTO GUIDATO	7
Commento al codice	7
PROGETTO GUIDATO	7
Commento al codice	8
PROGETTO GUIDATO	8
Commento al codice	8
PROGETTO GUIDATO	8
Commento al codice	8
PROGETTO GUIDATO	9
Commento al codice	9
Esercizi	10
Esercizi su: gestione della tastiera	10
Esercizio 1. Movimento da tastiera I	10
Esercizio 2. Movimento da tastiera II	10
Esercizio 3. Movimento da tastiera III.....	10
Esercizio 4. Gestione della tastiera.....	10
Sommario	11