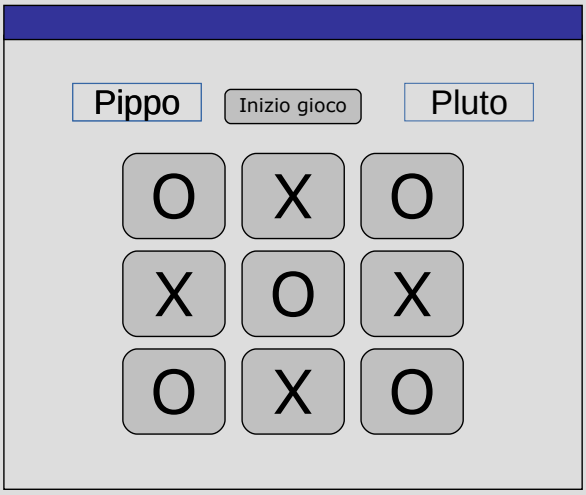
Laboratorio di informatica – classe 3ª – data 2/2/2017

TRIS

Implementare il gioco del tris.

Layout della UI



Requisiti

L'interfaccia del programma deve implementare le seguenti funzionalità:

- Due *textbox* per l'inserimento dei nomi giocatori. (Si presuppone che al primo giocatore corrisponda il simbolo O mentre al secondo il simbolo X)
- Un bottone che determini l'inizio del gioco.
- Nove bottoni che identificano le "caselle" del gioco. Cliccando su un bottone, un giocatore segnerà la casella con il proprio simbolo, ma soltanto se questa non è già stata occupata. Inizialmente, i bottoni non presenteranno alcun simbolo.
- Il programma dovrà visualizzare mediante una *message box* l'esisto del gioco, e cioè il nome del giocatore che ha vinto, oppure se c'è stato un pareggio.
- Per semplicità, si può supporre che il giocatore (O) muova sempre per primo.

Note sull'implementazione

Lo stato del gioco deve essere memorizzato in una matrice 3x3. (matrice `string`, `int` o `bool`, non ha particolare importanza). Un problema da risolvere, dunque, è quello di associare i bottoni alle celle corrispondenti della matrice. Tale problema è connesso alla modalità di creazione dei nove bottoni. Si possono individuare due soluzioni, che si collocano agli estremi.

Soluzione "ingenua"

I nove bottoni e i relativi eventi *click* sono creati a *design-time*. Ogni gestore di evento gestisce in modo appropriato il bottone corrispondente.

Ad esempio, supponendo che i bottoni siano denominati: `btnUno`, `btnDue`, `btnTre`... e collocati da sinistra a destra e dall'alto in basso:

```
...
string[,] tris = new string[3, 3];
string mossaCorrente = "O";  //cambia ogni volta ("O"->"X"->"O"...)

private void btnUno_Click(object sender, EventArgs e)
{
    tris[0, 0] = mossaCorrente;
    ...
}

private void btnDue_Click(object sender, EventArgs e)
{
    tris[0, 1] = mossaCorrente;
    ...
}
...
```

Soluzione sofisticata

I bottoni vengono creati dinamicamente (da codice) e associati allo stesso gestore di evento. Ad esempio, si supponga, nell'evento `Form_Load`, di collocare i nove bottoni in un *panel*:

```
private void Form1_Load(object sender, EventArgs e)
{
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            Button btn = new Button();
            btn.SetBounds(j * 50, i * 50, 50, 50);   //-> 50: dimensione bottone
            pnlTris.Controls.Add(btn);  //-> aggiunge bottone al panel
            btn.Click += Btn_Click;     //-> associa bottone a gestore di evento
        }
    }
}

private void Btn_Click(object sender, EventArgs e)
{
    //... gestisce click dei nove bottoni
    // (ma quale bottone è stato premuto? lo stabilisce il parametro sender)
}
```

Viene prodotto un risultato simile al seguente:

Poiché i nove bottoni condividono lo stesso gestore di evento (`btn_Click`), occorre determinare quale bottone viene premuto (il primo, il secondo, il terzo, ...) e conseguentemente trovare la cella corrispondente della matrice.

Vi sono varie possibilità; una è quella di determinare innanzitutto la posizione del bottone all'interno del *panel*. Per farlo occorre prima ottenere un riferimento del bottone utilizzando il parametro `sender`. ; quest'ultimo rappresenta di fatto il bottone cliccato. Successivamente, si cerca il bottone nella collezione `Controls` del *panel*:

```
private void Btn_Click(object sender, EventArgs e) //->nota bene: sender è di tipo object
{
    Button b = sender as Button;  // -> "obbliga" C# a considerare sender di tipo Button
    int indice = pnlTris.Controls.IndexOf(b);  // -> cerca posizione del bottone
    // ... partendo da "indice" trova posizione corrispondente nella matrice
    ...
}
```

Nota bene: perché il processo di ricerca dell'indice del bottone funzioni correttamente è necessario che il *panel* contenga soltanto i nove bottoni. (Da qui l'importanza di collocare i bottoni nel *panel* invece che crearli direttamente nel *form*, il quale contiene anche altri controlli.)

Soluzioni a confronto

La prima appare più semplice ma è ripetitiva e richiede di scrivere più codice. La seconda è apparentemente più complicata, ma più compatta e, soprattutto, generalizzabile ad altri scenari nei quali occorre gestire una griglia di controlli, siano essi bottoni o altro.