Laboratorio di informatica – classe 3ª – data 22/12/2015

Memory

Realizzare il gioco “Memory”, con un solo giocatore.

Svolgimento del gioco

Su una griglia sono disposte casualmente “n” coppie di carte. Inizialmente tutte le carte sono coperte.

Il giocatore scopre temporaneamente una carta cliccando su di essa. Quindi ne scopre un'altra, alla ricerca della carta gemella. Se la trova, entrambe restano scoperte, in caso contrario vengono nuovamente coperte.

Il gioco termina quando tutte le carte sono scoperte. L'obiettivo è quello di riuscirci con il numero minore di mosse possibile.

Note sull'implementazione

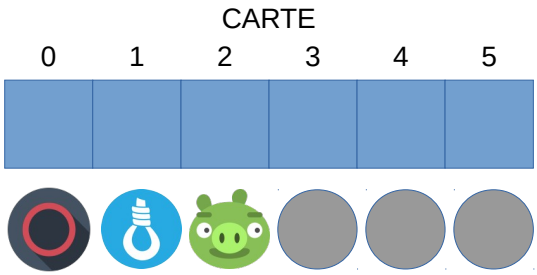
Si possono immaginare due tipi di implementazione: *base* e *avanzata* (vedi più avanti). In entrambi i casi occorre gestire le informazioni relative alle carte del gioco.

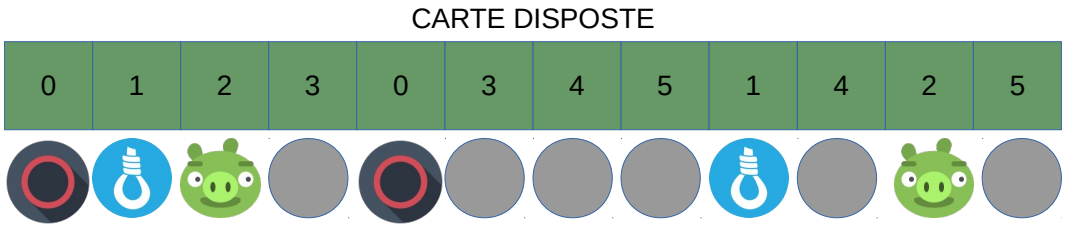
1. Gestire la collezione delle carte (ad esempio mediante un vettore).
2. Gestire “n” coppie di carte, posizionandole casualmente.
3. Identificare le carte, (ad esempio con un numero), ed eventualmente il loro stato, coperta o scoperta. (Quest'ultimo dato non è detto che sia necessario; dipende da scelte progettuali.)

Strutture dati

Si possono immaginare due strutture dati principali:

- la collezione delle carte; ogni carta è identificata dalla sua posizione nella collezione e dal nome del file contenente l'immagine.
- La “griglia”, e cioè la collezione delle “n” coppie di carte.





Nota bene: ciò che viene chiamata “griglia” è in realtà un vettore. Ogni elemento contiene il n° di carta e, eventualmente, lo stato della carta: coperta/scoperta.

Un problema fondamentale è quello di memorizzare casualmente i numeri di carta nel secondo vettore.

Generazione numeri casuali

Per la generazione di numeri casuali si utilizza la classe **Random**. Il seguente codice mostra come generare un numero compreso tra 0 e 5.

```
Random rnd = new Random();
int numero = rnd.Next(6);  //-> numero tra 0 e 5 compresi
```

Visualizzazione delle immagini

Sorvolando sull'implementazione della interfaccia utente (vedi: versione *base* e *avanzata*), si suggerisce di collocare le immagini su una sotto cartella del progetto e di caricare il loro nome nella collezione delle carte all'avvio del programma.

Per visualizzare l'immagine del pilota si utilizza un **PictureBox**. Si carica l'immagine sulla proprietà **Image** del controllo:

```
picCarta.Image = Image.FromFile(<nome del file>);
```

Supponendo che le immagini siano caricare nella sotto cartella (nel progetto) “Carte”, per visualizzare un'immagine possiamo scrivere:

```
picCarta.Image = Image.FromFile(@"../../Carte/Triangolo.png");
```

Naturalmente, i nomi effettivi dei file immagine sono memorizzati nel vettore delle carte.

Per nascondere l'immagine visualizzata è sufficiente assegnare **null**:

```
picCarta.Image = null
```

Interfaccia utente: versione base

Stabilito un numero fisso di carte, ad esempio 12, si può pensare di posizionare 12 **PictureBox**, assegnando un numero progressivo al loro nome: **pic0**, **pic1**, **pic2**, **pic3,…** I gestori di evento Click dei **PictureBox** saranno tutti uguali, eccetto che gestiranno ognuno la carta corrispondente .

Il gioco può essere visto come la gestione di una successione di tentativi.

Ogni tentativo comincia con un il click su una carta. (Qualsiasi click su una carta scoperta dovrà essere ignorato). Se è la prima carta della coppia, si visualizza e si "segna" il suo numero. Se è la seconda si verifica se il suo numero è uguale a quello segnato e si procede di conseguenza: se è così, entrambe la carte restano scoperte; altrimenti, entrambe saranno nuovamente coperte.

Interfaccia utente avanzata

La disposizione dei **PictureBox** avviene dinamicamente da codice, sulla base del numero di carte e del numero di colonne che si desidera utilizzare. Tutti i gestori di evento click punteranno allo stesso metodo.

Quando l'utente clicca su un **PictureBox** occorre cercare la carta corrispondente. Si possono immaginare varie tecniche, ma tutte partono dall'accesso al **PictureBox** cliccato.

Supponiamo che tutti i (click dei) **PictureBox** siano gestiti dal seguente metodo:

```
private void picCarta_Click(object sender, EventArgs e)
{
    PictureBox pic = sender as PictureBox;
    ...
}
```

L'istruzione evidenziata memorizza in **pic** un riferimento al controllo effettivamente cliccato.

Il procedimento da utilizzare per ottenere il numero di carta a partire dal controllo dipende da alcune scelte.

Un'idea è quella di memorizzare i **PictureBox** dentro un vettore della stessa lunghezza della "griglia" (il vettore contenente le coppie di carte). Ottenuto **pic**, si cerca all'interno del vettore: la posizione trovata corrisponde alla posizione della carta nel vettore "griglia".

Posizionamento dei PictureBox

Ogni PictureBox creato deve essere posizionato in base alla riga/colonna che occupa nella griglia. A questo proposito occorre impostare opportunamente le proprietà **Left** (colonna) e **Top** (riga). I calcoli devono essere eseguiti in base alle dimensioni del PictureBox, date dalle proprietà **Width** e **Height**.