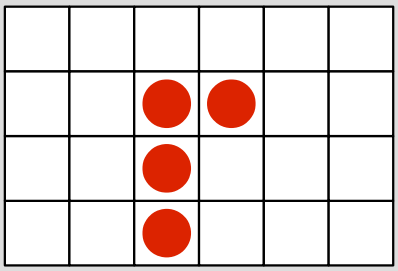
Laboratorio di informatica – classe 4ª – data 3/11/2016

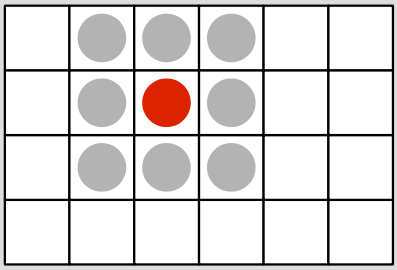
Life game

Si desidera implementare il gioco Life. Questo simula un mondo che obbedisce a un semplice sistema di regole e che può essere rappresentato mediante una griglia bidimensionale di celle; ogni cella può trovarsi negli stati **viva** o **morta**. (In figura ci sono 4 celle vive e 20 morte.)



Evoluzione nel mondo di *Life*

L'evoluzione è determinata dalla nascita e dalla morte delle celle. Lo stato futuro di una cella viene determinato da quante celle vive le sono vicine. (Nota bene: ogni cella è adiacente ad altre 8, eccetto quelle situate sui bordi della griglia.)



Le regole che determinano l'evoluzione di una cella sono:

1. se le celle adiacenti vive sono due, la cella permane nel suo stato (viva o morta che sia);
2. se le celle adiacenti vive sono tre, la cella nasce (o resta viva);
3. in tutti gli altri casi la cella muore (o resta morta).

Evoluzione "simultanea" delle celle

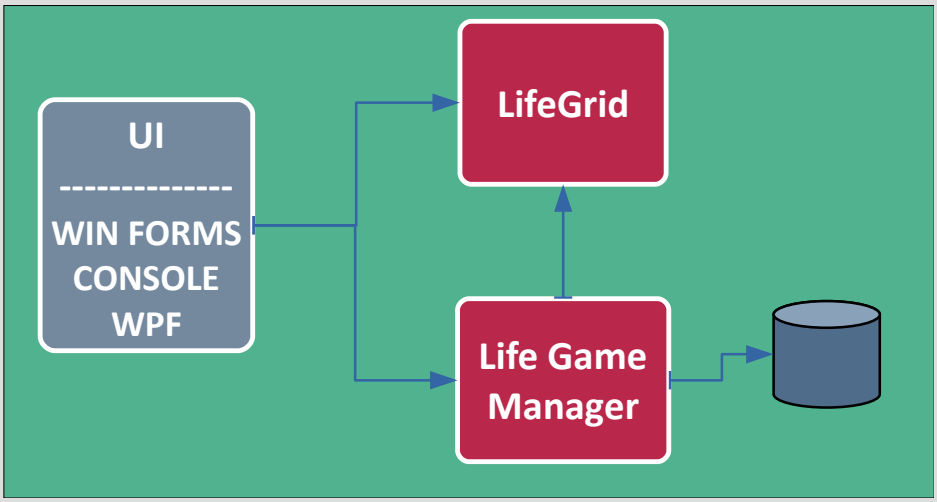
Nel determinate lo stato futuro di una cella, si considera lo stato presente delle celle adiacenti, prima che si siano evolute. In sostanza: l'evoluzione del mondo non viene influenzata dall'ordine con il quale sono esaminate le celle.

Stato iniziale di Life ed evoluzione

In *Life* l'evoluzione si misura in numero di **generazioni**. Lo stato delle celle della "generazione zero" viene stabilito dall'esterno (casualmente e/o mediante scelta dell'utente). Dopodiché, a ogni generazione lo stato delle celle cambia in accordo alle regole descritte in precedenza.

Achitettura del programma

Sotto è rappresentata l'architettura che si desidera realizzare.



Si vuole separare la gestione del mondo virtuale dalla sua visualizzazione, che può essere realizzata mediante interfaccia Console, WinForms o WPF (o mobile).

LifeGrid: "motore" del gioco

La classe implementa la griglia che rappresenta il mondo virtuale del gioco. Si possono supporre le seguenti funzionalità:

1. Creazione della griglia (creazione dell'oggetto): indicazione delle sue dimensioni.
2. Impostazione della generazione zero.
   a) Consentire al codice esterno di stabilire le celle vive all'inizio.
   b) Generare una configurazione casuale di celle.
3. Evoluzione alla successiva generazione.
4. Comunicazione delle "celle mutate" nel passaggio da una generazione alla successiva. (E/o comunicazione dello stato dell'intera griglia.). Questo punto può essere realizzato in concomitanza con il punto 3, oppure separatamente.
5. Accesso allo stato delle singole celle e/o accesso allo stato di tutte le celle e/o accesso allo stato delle celle vive.

LifeManager

Gestisce il salvataggio e il caricamento su disco di tutte le informazioni sul mondo virtuale, in modo che l'utilizzatore possa salvare una "fotografia" del mondo e ricaricarla successivamente. (Nota bene: per fare questo è necessario che **LifeGrid** definisca un meccanismo per conoscere lo stato di tutte le celle vive.

Note sull'implementazione

Nella sua versione minimale, LifeGrid potrebbe avere la seguente interfaccia pubblica:

```
public class LifeGrid
{
    public  LifeGrid(int rowCount, int colCount) {}
    public void SetZeroGeneration(int[,] cellList) {}
    public CellInfo[] GoNextGeneration() {}
    public CellInfo[] LiveCells() {}
}
```

Il metodo **SetZeroGeneration()** resetta la griglia e imposta su "viva" le celle posizionate alle coordinate specificate nella matrice (generazione zero).

Il tipo **CellInfo** definisce le informazioni su una cella (coordinate e stato).

Il metodo **GoNextGeneration()** fa "evolvere" il mondo e restituisce l'elenco di celle che sono mutate. Questo elenco può essere utilizzato per aggiornare la UI.

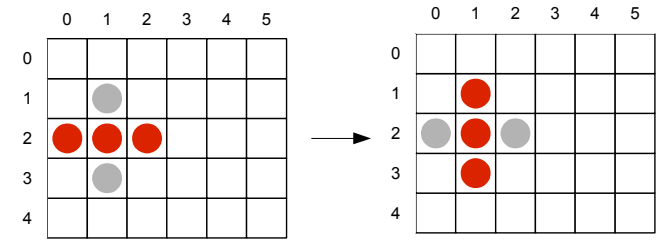
Il metodo **LiveCells()** restituisce tutte le celle vive. Questo elenco può essere utilizzato per aggiornare la UI e da LifeManager per salvare lo stato della griglia.

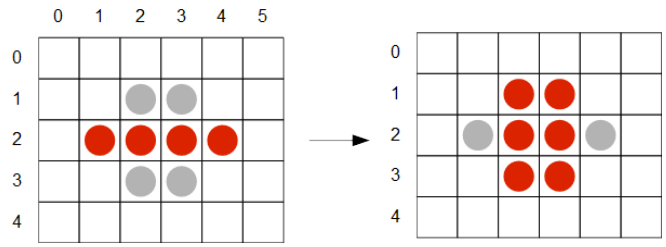
La suddetta proposta è soltanto un'ipotesi. La classe può essere anche diversa; l'importante è che implementi i requisiti elencati nei punti 1 – 5. Anche l'utilizzo del "record" **CellInfo** è soltanto un'ipotesi.

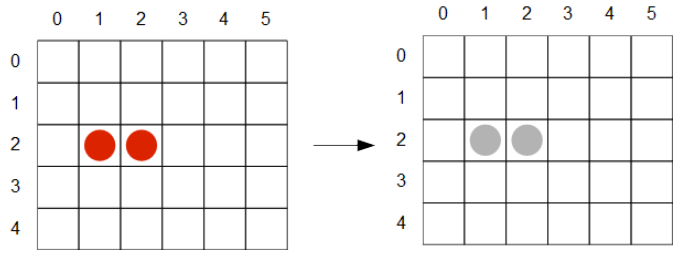
Testare il funzionamento della griglia.

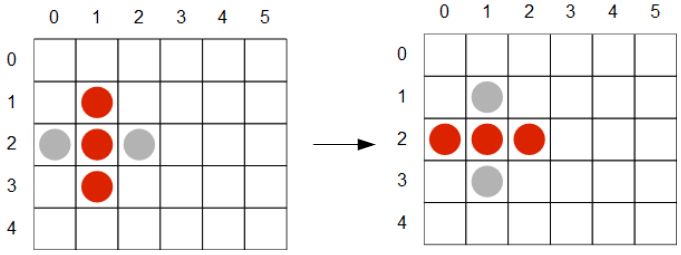
Anche senza UI è possibile testare la classe LifeGrid (e LifeManager). Seguono alcuni scenari che è possibile testare.

In grigio: a sinistra sono rappresentante le celle che nasceranno, a destre le celle che sono morte nel passaggio da una generazione alla successiva.









Un tipico test, dunque, si riduce a impostare la generazione zero riportata nella griglia a sinistra e verificare che dopo una generazione la griglia corrisponda a quella di destra.