TP1

Exercice 1) Jeux de la vie

Définition extraite de Wikipédia : *En préambule, il faut préciser que le jeu de la vie n'est pas vraiment un jeu au sens ludique, puisqu'il ne nécessite aucun joueur ; il s'agit d'un automate cellulaire, un modèle où chaque état conduit mécaniquement à l'état suivant à partir de règles pré-établies.*

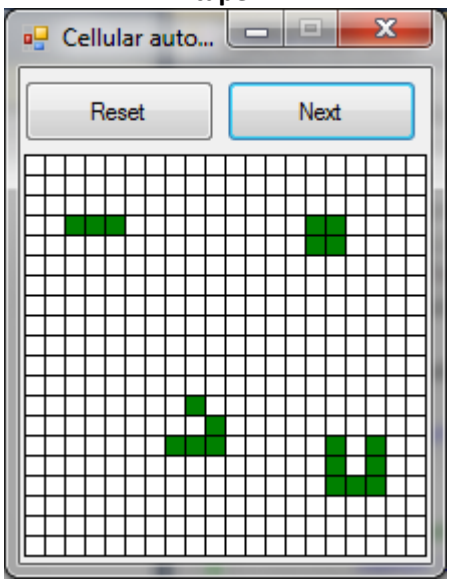
Le jeu se déroule sur une grille à deux dimensions, dont les cases — appelées « cellules », par analogie avec les cellules vivantes — peuvent prendre deux états distincts : « vivantes » ou « mortes ».

À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines de la façon suivante :

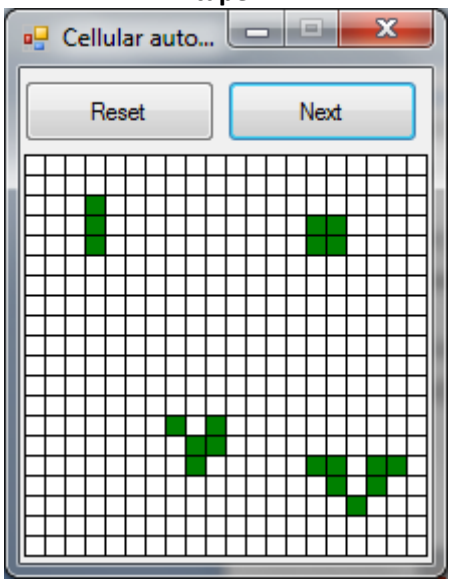
- *Une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît).*
- *Une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.*

Exemple:

Etape 1



Etape 2



En théorie la grille de jeu est infinie mais en pratique on se limitera à une grille de taille finie. On pourra considérer, au choix, que:

- soit toutes les cellules en dehors de la grille sont mortes;
- soit le monde est cyclique, lorsque l'on va à droite du bord droit on retombe sur le bord gauche (et inversement), et lorsque l'on va en bas du bord bas, on retombe en haut (et inversement).

Dans cet exercice vous allez devoir coder le jeu de la vie à partir du squelette téléchargeable à cette adresse : http://www.esiee.fr/~perretb/I3FM/AlgoProg/JeuxDeLaVie.zip . Ce squelette contient le

code nécessaire à l'interface graphique (composants, définition des évènements) et le prototype des différentes fonctions à utiliser. Votre travail consiste à compléter le code des prototypes de fonctions.

Le squelette définit quatre champs que vous allez devoir utiliser:

- `const int worldHeight = 50;` : hauteur du monde (en nombre de cases)
- `const int worldWidth = 50;` : largeur du monde (en nombre de cases)
- `bool[,] world;` : le tableau qui représente le monde. Chaque case du tableau représente une cellule, si la valeur est true, la cellule est vivante, sinon elle est morte.
- `const int cellWidth = 10;` : la largeur d'une case du monde en pixel (pour la représentation graphique)

Commencer par regardez comment est construit le squelette, puis implémenter chacune des fonctions décrite ci-dessus. **Tester votre code dès que possible, n'attendez pas d'avoir écrit toutes les fonctions pour réaliser vos tests.**

- `private void DrawGrid(Graphics g)` : dessine la grille représentant le monde dans l'objet graphics passé en paramètre
- `private void DrawCells(Graphics g)` : dessine les cellules vivantes dans l'objet graphics passé en paramètre
- `private void ResetWorld()` : remet le jeu à zero, tue toutes les cellules
- `private Point ConvertPixelToWorldCoordinate(Point p)` : convertit une position dans l'espace pixel de la représentation du monde en une position sur la grille. Par exemple, le point de coordonnées (123,78) en pixels correspond à la case (12,7) en supposant que chaque cellule mesure 10 pixels de côté.
- `private void CreateKillCell(Point p)` : change l'état de la cellule indiquée (la rend vivante si elle été morte et la tue si elle était vivante).
- `private int CountNeighbours(Point p)` : compte le nombre de voisins vivants autour de la position donnée.
- `private bool ComputeNewCelluleState(Point p)` : calcule si la cellule à la position donnée doit être morte ou vivante au prochain tour de jeu.
- `private void computeNextGameState()` : calcule le nouvel état de toutes les cellules au prochain tour de jeu.