Game Of Life

1 Avant-propos

Le but de ce TP est de vous faire découvrir la programmation orientée objet en C#.
Vous aborderez les notions d'objets et de classes qui vous ont été présentés pendant la conférence. Il est possible qu'elles vous paraissent un peu floues pour l'instant et c'est tout à fait normal. Ce tp vous permettra d'y voir plus clair et de vous apporter les bases de la POO.

N'hésitez pas à demander de l'aide aux assistants présents tout au long du tp et à regarder un peu par vous mêmes sur internet, notamment sur ce site qui vous sera très utile tout au long de l'année : `https://msdn.microsoft.com/en-IN/library/618ayhy6.aspx`.

2 Présentation

Le but de ce tp est de vous faire réaliser un "Jeu de la vie" ou "Game of Life". Il s'agit de simuler la vie de cellules : qui se dupliquent, survivent ou bien meurent. La version que vous allez implémenter aura la forme d'une grille (donc en 2 dimensions). Si vous souhaitez en savoir plus sur l'histoire et les détails intéressants du Game Of Life, visitez la page wikipedia sur Conway's Game of Life.

3 Règles du jeu

Le principe est très simple et les règles qui régissent le comportement des cellules le sont tout autant :

3.1 Cellule Vivante

Une cellule vivante meure si elle a moins (strictement) de 2 voisins vivants ou plus de 3 (toujours strictement).

Une cellule vivante survit si elle a 2 ou 3 voisins vivant.

3.2 Cellule Morte

Une cellule morte devient vivante si et seulement si elle a exactement 3 voisins vivants. Sinon, elle reste morte.

4 Rappels/Cours

4.1 Les Boucles

Contrairement au Caml que vous avez utilisé au début de l'année, le C# est un langage impératif et Orienté Objet (nous en reparleront un peu plus loin).
Vous allez donc pouvoir utiliser les boucles pour effectuer plusieurs fois des instructions (vous utilisiez la récursion en Caml).

While

```
while (condition) {
        //your instructions
}
```

```
int i = 0;
while (i < 10) {
        Console.WriteLine("Hello");
    i++;
}
```

For

```
for (var iterator; condition; iteration) {
        //your instructions
}
```

```
for (int i = 0; i < 10; i++) {
        Console.WriteLine("Hello");
}
```

Les deux boucles précédentes font la même chose. C'est à vous de choisir laquelle vous préférez utiliser et dans quel cas.

4.2 Les Tableaux

Les tableaux sont un moyen très pratique de stocker plusieurs valeurs d'un même type. En effet, il arrive très souvent que l'on veuille manipuler des données plus ou moins nombreuses et il serait très désagréable et désordonné de créer une variable pour chaque valeur. Avec les tableaux vous pouvez stocker dans une seule variable un nombre très grand (connu à l'avance) de données.

En C# vous pouvez créer des tableaux de différentes dimensions : nous verrons ici les tableaux de 1 et 2 dimensions (vous pouvez voir les tableaux à 2 dimensions comme des matrices).

```
//déclaration d'un tableau d'entiers de 2 manières
int[] mytab = {1, 2, 10, 42, 0, 0, 3};
int[42] tab2 = { 0 };

//déclaration d'un tableau à 2 dimensions
int[,] tab3 = new int[20,20];
```

Pour manipuler votre tableau, vous devrez les parcourir à l'aide des boucles. Pour accéder à une case de votre tableau, vous utiliser l'operateur "[]" :

```
int[,] matrix = new int[100, 400];
int i = matrix[42, 0];
```

4.3 Les Objets et les Classes

Les Objets Il est parfois assez difficile de comprendre le concept de l'objet en programation. L'une des définitions que l'on peut donner est la suivante :
Un objet est une représentation d'un concept.
Par exemple, vous pouvez créer des objets pour représenter une voiture, une personne (un personnage de jeu vidéo par exemple), ou encore pour représenter un environnement (la carte d'un jeu, le terrain,...)

Les Classes Les classes peuvent être vues comme la définition d'un objet. C'est dans les classes que l'on fixe les règles que vont suivre les objets que l'on souhaite créer.
Les classes permettent de définir les différents attributs de votre objet (dans le cadre d'un objet pour représenter une personne, il peut s'agir de sa taille, de la couleur de ses yeux...).
Vous pouvez également définir son comportement via ce que l'on appelle des méthodes (qui sont des "fonctions" propres à une classe).

```
//definition d'une classe voiture (Car)
public class Car
{
    public int MaxSpeed;
    public int Speed;
    public string Brand;
    public string Color;

    public Car(int speed, string brand, string color)
    {
        Speed = 0;
        MaxSpeed = speed;
        Brand = brand;
        Color = color;
    }
    public void Start(int speed)
    {
        Speed = speed;
    }
    public void Stop()
    {
        Speed = 0;
```

```
    }
}
```

```
//creation d'un objet de type Car
Car mycar = new Car(250, "Ferrari", "Red");
//appel à la methode start
mycar.Start(100);
//appel à la methode stop
mycar.Stop();
```

Attention ! Ce TP ne s'inscrit pas dans la chronologie des TP habituels (votre module d'IP). Certaines des notions abordées ici n'ont pas encore été abordés en cours et vous devrez donc attendre que vos assistants vous autorisent à les utiliser.

5 Let's Start Our Game

5.1 Les Classes

Pour cette implémentation, je vous propose de créer 2 classes simples pour gérer les différents éléments de notre jeu de la vie : Les Cellules, et La Grille (qui contient des cellules).

Cell Commençons par créer une classe pour définir ce que seront nos cellules : Comme expliqué dans les règles exposées en introduction, nos cellules peuvent se trouver dans 1 des 2 états suivants : Vivante ou Morte.
Egalement, les cellules peuvent changer d'états selon leurs voisines, il va donc falloir être capable de changer l'état d'une cellule, il va donc nous falloir une méthode pour cela :

```
public class Cell
{
    private bool state;
    //ajoutez ce que vous jugez nécessaire pour votre implémentation

    //public constructor
    public Cell(/* Les arguments pour votre constructeur */)
    {
        //initialisation des attributs de votre classe
    }
    //getter for State
    public bool GetState()
    {
        return this.state;
    }
    //setter for State
```

```
17      public void SetState(bool s)
18      {
19          state = s;
20      }
21  }
```

Grid Après avoir créé notre classe Cell, il nous faut les stocker dans une grille (qui est en fait l'environnement de notre jeu de la vie).
La classe Grid contiendra donc un tableau à deux dimensions de cellules et devra être capable (via une méthode bien entendu) de "mettre à jour" l'état de la grille. C'est à dire changer l'état des cellules selon si elle doivent mourir, vivre, ou survivre.
Faites bien attention à une chose : lorsque vous mettez à jour vos cellules, vous devez les mettre à jour toutes au même moment, en effet, l'état des cellules à l'instant t est déterminé par l'état de celles-ci à l'instant t - 1 (si vous changez vos cellules d'état au fur et à mesure et non pas toutes en même temps, l'état de votre grille sera faussé).

Déterminer l'état des Cellules à l'instant t + 1 Pour déterminer l'état des Cellules à l'instant t + 1, vous devez parcourir votre grille à l'instant t et pour chaque cellule vérifier l'état de ses voisins et en déduire le nouvel état de la cellule. Libre à vous de faire cela via une méthode dans votre "Grid" ou dans les "Cell".

```
1   public Grid
2   {
3       private Cell[,] grid;
4       //les attributs que vous voulez
5       public Grid(int width, int height)
6       {
7           this.grid = new Cell[width, height];
8           //Autre chose si besoin...
9       }
10
11      //vos methodes...
12
13      public void UpdateGrid()
14      {
15          //mise à jour de votre grille
16      }
17      public void PrintGrid()
18      {
19          //affichez votre grille de cellule à chaque instant
20      }
21  }
```

L'affichage de la grille se fera en console, vous aurez besoin de quelques instructions de la Classe systeme Console :

```
1 public static void Console.Write(char c);
2 public static void Console.Clear();
3 public static void Threading.Thread.Sleep(int ms);
```

6 Conclusion

Ce tp reste très simple, j'espère qu'il vous divertira, n'hésitez pas à implémenter d'autres fonctionnalités si vous le souhaitez.

Si vous êtes en difficulté ou que vous ne comprenez pas certains concepts ou certaines consignes, n'hésitez pas à solliciter les assistants !

The Code is The Law