PROJET C++: Le Jeu de La Vie

Vincent Laurain

Le but du TP, est de réaliser une application console (nouveau projet console, vide si vous codez sous visual C++). Ce travail **doit être personnel**. Etant donné qu'il y a une assez grande composante créative, **il est impossible que deux personnes se retrouvent avec le même programme à la fin**. Comme aurait pu dire Spiderman, "de grandes similarités impliquent de grandes responsabilités...qui impliquent une division de la note finale". Vous pouvez également programmer sur votre propre machine.

La note de ce TP sera notée à la volée et donc par rapport à votre capacité à appliquer le cours.

Les codes ainsi que la solution et l'executable devront être déposés en ligne sur Arche.

Table des matières

1 But du Projet

Le but de ce TP est de programmer, en version console, du jeu de la vie. Si vous ne savez pas ce qu'est le jeu de la vie, vous pouvez allez voir cette excellent vidéo indiquant toutes les possibilités : *https ://www.youtube.com/watch ?v=S-W0NX97DB0.*

Ainsi, il s'agit simplement d'un tableau de cellules, dont l'évolution suit des règles simples, en produisant des effets très différents :

- **Une cellule vivante avec 2 ou 3 voisines vivantes survit. Dans les autres cas elle meurt (de solitude ou d'étouffement).**
- **Une cellule morte, vit si elle est entourée de trois voisines vivantes exatement.**

Vous disposerez de 9h pour réaliser ce TP (3 séances). Ici, en théorie, **vous n'aurez pas besoin de l"héritage**. Ainsi, ce TP n'évalue que la moitié des compétences dispensées dans ce cours, mais il consitue le minimum syndical dans le monde de la programmation objet.

Tout d'abord, pour les plus avancés, afin de vous placer dans des vraies conditions de production, je vous propose de réflechir par vous même à une structure d'un tel programme. Quels sont les classes nécessaires ? Comment s'imbriquent-elles ? Essayez de trouver une solution qui ne garde dans le main que le strict minimum. Bon courage.

Pour ceux qui ne se sentent pas capables de trouver la bonne structure, cherchez quand même, puis lisez la suite, une structure est proposée.

2 Les classes nécéssaires

Puisqu'un jeu de la vie n'est ni plus ni moins qu'un tableau de cellules, je vous propose de créer deux types d'objets : **des Cellules** et des **JeuDeLaVie**.

Les données membre

- **Les Cellules :** Une cellule représente une case. Elle contiendra 2 données membres :
  - `statut` : Indique si la cellule est morte ou vivante.
  - `statut_futur` : Indique si la cellule va survivre

- **Les JeuDeLaVie :** Il contiendra toutes les cellules. Et puisque c'est un tableau il doit avoir des dimensions
  - `Largeur` : Indique le nombre de cellules sur une ligne
  - `Hauteur` : Indique le nombre de cellules sur une colonne
  - `TableauCellules` : qui serait un tableau de cellules

.. et c'est tout.

Les fonctions membre

En ce qui concerne la classe **Cellule**, c'est très simple :

- Elle contient les `getters` et `setters`.
- `Affiche:` Une fonction qui permet de l'afficher : un "X" si elle est vivante un " " si elle est morte.
- `Evolve:` Une fonction qui lui permet à la cellule d'évoluer : Son `statut_futur` remplace son `statut`

En ce qui concerne la classe **JeuDeLaVie**, ce n'est pas beaucoup plus compliqué :

- Elle contient les `getters` et `setters`.
- `Compute_Next_State:` Une fonction qui permettra de définir, pour chaque Cellule son `statut_futur` en fonction de son environnement et des règles dévolution citées plus haut (c'est la fonction la plus compliquée à programmer).

— `Evolve:` Une fonction qui fera evoluer toutes les cellules du `TableauCellules`
— `Affiche:` Une fonction qui affichera le jeu de la vie
— `Initialise:` Une (ou deux) fonction qui permettra d'initialiser certaines cellules à "vivantes" au début (ici on pourra prendre un tableau de positions, ou alors faire une initialisation aléatoire).
— `get_Cell:` Une fonction qui sera très utile (surtout lorsqu'il s'agira de programmer l'état prochain du JeuDeLaVie) sera une fonction `get_Cell`, qui en fonction d'une position x et y, renverra la cellule correspondante. Cette fonction n'est pas obligatoire, mais elle sera très utile.

NB : Attention, la plus grosse difficulté du TP, consiste à récupérer, pour chaque cellule, quelles sont les cellules qui lui sont voisines pour savoir si elles vont vivre ou mourrir. Pour cela, il faudra faire attention de ne pas sortir des bords du tableau.

3 Main

En ce qui concerne le main, il devrait être assez simple. Utilisez les classes au maximum. Dans l'esprit ce **ressemblera** à cela :

```
int main(){

JeuDeLaVie j(200, 40);
j.Initialise();
j.Affiche();
while(int i<50000)
{
Sleep(0.01);
j.Compute_Next_State();
j.Evolve();
system("cls");// Vide l'écran de la console
j.Affiche()
}
return(0);
```

4 Annexe

Voici quelques fonctions dont vous aurez besoin pour réaliser le projet :
— `Sleep(t)` (t en secondes) sous windows et `usleep(t)` (t en microsecondes) sous Mac Os/Linux dans `<unistd.h>` : permet de faire faire une pause au programme du temps t
— `system("cls");` sous windows et `system("clear");` sour Mac OS dans `<stdlib.h>` : Permet de vider l'écran de la console afin de réafficher le jeu de la vie au même endroit.
— `rand()` qui permet de calculer un nombre aléatoire. Elle s'utilise comme suit en utilisant `<stdlib.h>` et `<time.h>` :

```
int rn;
srand (time(NULL));// initialise le random sur l'horloge de l'ordi
rn = rand() % 10 + 1; // genère un nombre aléatoire entre 1 et 10
```

— Vous pourrez utiliser les fonctions `max(a,b)` (qui est egal à a si a≥b, et b sinon) ou `min(a,b)`.

5 Conseils

Voici quelques conseils pour avancer au mieux :
— Tout d'abord, avant même de programmer, faites, **sur papier**, le header des classes, avec les bons types de fonctions et les bons paramètres. **Faites valider par l'enseignant.**

— Comme d'habitude, programmez et testez vos codes petit à petit. N'essayez pas de tout programmer puis de compiler. Ici je vous conseille de :
  — Créer la classe **Cellule**, puis de bien la tester.
  — Une fois que **Cellule** est validée, commencez **JeuDeLaVie**. Laissez pour la fin la fonction `Compute_Next_State` qui est la fonction la plus difficile à coder.
  — Enfin amusez vous avbec le main.

Bon courage !