| | |
|---|---|
| Module : **Génie Logiciel** | Nature : **TP-2** |
| Classe : $2^{eme}$ **année SIAD** & **RSI** | Date : **15 mars 2020** |
| Durrée : **1 semaine** | Code : |

# 1 Introduction

Le **Jeu de la vie** "*Game of Life*", également connu sous le nom de **"*Life*"**, est un automate cellulaire conçu par le mathématicien britannique *John Horton Conway* en 1970.

"*Life*"est un jeu sans joueur, ce qui signifie que son évolution est déterminée par son état initial, ne nécessitant aucune entrée supplémentaire. On interagit avec "*Life*"en créant une configuration initiale et en observant son évolution. C'est une machine de Turing complète et peut simuler toute autre machine de Turing.

L'univers de "*Life*"est une grille orthogonale bidimensionnelle infinie de cellules carrées, dont chacune est dans l'un des deux états possibles : $\{vivant, mort\}$, (ou peuplés et non peuplés, respectivement). Chaque cellule interagit avec ses huit voisins, qui sont les cellules adjacentes horizontalement, verticalement ou en diagonale. À chaque étape du temps, les transitions suivantes se produisent :

- Toute cellule vivante avec moins de deux voisins vivants meurt, à cause de l'ennui [1].
- Toute cellule vivante avec deux ou trois voisins vivants vit jusqu'à la prochaine génération.
- Toute cellule vivante avec plus de trois voisins vivants meurt, comme par surpopulation [2].
- Toute cellule morte avec exactement trois voisins vivants devient une cellule vivante, comme par reproduction [3].

Le modèle initial constitue la graine du système. La première génération est créée en appliquant les règles ci-dessus **simultanément** à chaque cellule de la graine ; les naissances et les décès surviennent **simultanément**, et le moment discret auquel cela se produit est parfois appelé **TIC**. Chaque génération est une fonction pure de la précédente. Les règles continuent d'être appliquées à plusieurs reprises pour créer de nouvelles générations.

# 2 Objectif du TP

"*Life*"est bien connu de presque tous ceux qui ont déjà programmé un ordinateur (surtout l'ancienne génération), mais beaucoup moins réalisent un niveau de sophistication dans la conception des modèles de vie.

En effet, "*Life*"a une riche structure mathématique qui ne peut être pleinement appréciée qu'en observant le comportement complexe qui découle d'un ensemble de règles aussi simples. Cependant, les **patch** qui présentent un tel comportement doivent être conçus avec soin, car la plupart des conceptions de ces **patchs** explosent simplement de manière chaotique pendant un certain temps, puis se stabilisent.

Dans ce TP, nous allons commencer l'implémentation d'un "*Life*"plus sophistiqué, que nous appellerons **"*GL-Life*"**. "*GL-Life*"devra respecter tous les principes *SOLID*, déjà vus en cours, pour permettre à notre logiciel de simuler de nouvelles formes de vie grâce à des extensions futures.

1. C'est des cellules très sociables
2. Finalement, pas trop sociables
3. !!

**Exercice 1** **( . . . . / 0 points)**

**Implémentation "*Dummy*" de "*Life*"**

Afin de vous faire apprécier "*Life*", une implémentation initiale (ne respectant pas *SOLID*) vous a été fournie. Commencez par télécharger le code à partir du dépôt ***git*** en utilisant la commande :

```
git clone https://github.com/sweeffon2/gl-life.git
```

Le code est constitué de trois projets :

— **Le modèle** : contient des interfaces pour l'abstraction des trois concepts (*IWorld*, *ICell*, *IPosition*).

— **L'implémentation pour les nulles** : contient une implémentation fonctionnelle de "*Life*"tels que définit dans l'introduction de ce TP (le IWorld est une matrice, les cellules sont carrées, et les règles sont basiques).

— **L'application lançable** : contient la fonction main qui nous permettra de lancer l'application en utilisant les implémentations .

(a) Compilez, exécutez et appréciez "*Life*".

(b) Générez des initialisations aléatoires et appréciez visuellement l'évolution du monde

(c) Ajoutez des motifs complexes (cherchez sur internet) et appréciez visuellement l'évolution du monde

**Exercice 2** **( . . . . / 0 points)**

**Implémentation sophistiquée de "*GL-Life*"**

Maintenant, nous souhaitons pouvoir ajouter de nouvelles extensions.

1. Les IHMs de visualisations doivent être plus sophistiqués et doivent permettre de :
   (a) contrôler l'écoulement du temps (play/pause/reinitialize)
   (b) ajouter des patchs
   (c) tuer ou ressusciter manuellement des cellules
2. Les cellules peuvent avoir des mutations (se comporter différemment)
3. Les mondes peuvent être plus complexes, au lieu de mondes matriciels comme dans l'exemple, pensez à des mondes hexagonaux, octogonaux, tridimensionnels, ...

(a) Proposez des conceptions permettant une dépendance faibles entre vos modules.

(b) Travaillez en groupes de 6 élèves, subdivisés en sous groupes de 2 élèves, de sorte à ce que chaque groupes ait une tâche indépendante des autres.

**Exercice 3** **( . . . . / 0 points)**

**Livrables**

(a) Remettez, en début de chaque séance de TP, un compte rendu par groupe contenant :
   — une introduction,
   — une partie conception (commune entre tous les sous-groupes),
   — un chapitre par sous groupe
   — une conclusion et l'état d'avancement.

(b) Remettez, en début de chaque séance de TP, les codes sources organisés par dossiers relatives à chaque sous groupe.

(c) Remettez, en début de chaque séance de TP, une version opérationnelle de votre logiciel.