PROJET IAP2: JEU DE LA VIE

1. **Présentation**

Le Jeu de la Vie (ou *Game of Life*) est, en réalité, un automate cellulaire défini en 1970 par le mathématicien anglais John Conway afin de tenter de résoudre un problème soulevé par le mathématicien, père de linformatique, *John Von Neumann*.

Ce jeu n'est pas á proprement parler un jeu: il ne nécessite pas l'intervention d'un joueur humain, hormis pour fixer les conditions initiales du jeu. Il se compose d'un univers dans lequel évoluent des cellules vivantes suivant des règles d'évolution précises.

Dans la version originale decrite par John Conway, l'univers est défini sur une grille á deux dimensions, de taille variable, où chaque élément de cette grille est une cellule pouvant prendre deux états différents: *morte* ou *vivante*. Le passage dun état à l'autre est guidé par les règles d'évolution suivantes:

(1) Une cellule morte au temps t devient vivante au temps $t+1$ si et seulement si, elle a exactement 3 cellules vivantes dans son voisinage.
(2) Une cellule vivante au temps t reste vivante au temps $t+1$ si et seulement si, elle a exactement 2 ou 3 cellules vivantes dans son voisinage, sinon elle meurt.

Le voisinage utilisé est le **8-voisinage**: pour une cellule donnée, ses voisines sont les 8 cellules qui lentourent, comme illustré sur la Figure 1.

Á partir de ces règles de base beaucoup d'extensions ont été proposées mais le jeu garde le même principe: il se déroule suivant un nombre de pas de temps théoriquement infini. À chaque pas de temps t, on évalue, pour chaque cellule, l'état de la cellule au pas de temps $t+1$ en fonction des règles d'évolution des cellules.

2. **Implementation**

A. Règles de base

Implémenter la version originale du jeu. La taille de l'univers est définie par l'utilisateur (i.e. la taille de la grille de cellules). Pour initialiser le jeu, un ensemble de cellules vivantes est positionné aléatoirement sur cette grille. Le nombre de

	$v1$	$v1$	$v1$		
	$v1$	$c1$	$v1$		
	$v1$	$v1$	$v1$	$v2$	$v2$
				$v2$	$c2$

FIG 1. Exemple de voisinage pour les cellules $c1$ et $c2$. Leurs voisines sont, respectivement, les cellules $v1$ et $v2$.

ces cellules est donné par l'utilisateur au début du jeu.

B. Déroulement du programme

Le programme devra s'exécuter de la façon suivante:

(i) Affichage d'un menu avec les différentes options du jeu, dont (a) une pour lancer le jeu avec les règles de base, (b) une permettant de quitter le programme, et (c) une pour mettre en pause la simulation pour observer plus en detail les evolutions.

(ii) Demande à l'utilisateur de la taille de l'univers (nombre de cellules sur la largeur et sur la hauteur) et vitesse du jeu.

(iv) Nombre de cellules vivantes au départ.

(v) Déroulement du jeu, à chaque étape :

(a) Afficher l'univers à l'écran (le caractère O symbolise la présence d'une cellule vivante):

0	
	0

(b) Calculer le nouvel état en fonction des règles d'évolution décrites par Conway.

(c) Permettre l'utilisateur de faire une excution pas pas. Dans ce cas, il faut attendre que l'utilisateur appuie sur la touche 'entrée' pour passer á l'étape suivante ou 'q' pour revenir au menu.

(d) Si le jeu n'a plus de cellules vivantes, retourne au menu.

2.1. **Extensions.**

Implémenter les extensions suivantes. Toutes ces extensions doivent être proposées dans le menu d'accueil du programme.

A. Âge d'une cellule

Développer un mécanisme qui, pour chaque cellule vivante, compte son âge. L'âge d'une cellule représente le nombre de pas de temps consécutifs durant lesquels la cellule est vivante. En utilisant l'âge d'une cellule, on ajoute la règle suivante: une cellule meurt quand son âge dépasse une valeur fixée par lutilisateur. L'âge des cellules vivantes doit être affiché lors de l'affichage de l'univers.

B. Cellules non viables

Développer un mécanisme qui permet de définir, dans l'univers, des cellules où la vie ne peut pas se développer. À aucun moment du déroulement du jeu, ces cellules ne pourront être déclarées comme vivantes. Ces cellules devront être marquées comme non viables et visibles lorsque lunivers est affiché à l'écran (via un 'X' par exemple). Lors de l'initialisation de l'univers, l'utilisateur choisit le nombre de ces cellules non viables. Elles sont positionnées aléatoirement lors de l'initialisation du jeu.

C. Voisinage cyclique d'une cellule

Le voisinage considéré d'une cellule est le **8-voisinage**. Sur les bords de l'univers, ce voisinage est réduit aux seules cellules voisines réellement existantes : par exemple, seulement 3 cellules pour la cellule de coordonnées $(0, 0)$. Afin d'étendre ce voisinage sur les bords, on propose de rendre cyclique le voisinage:

(1) la première cellule (respectivement la dernière) d'une ligne a pour voisine de gauche (droite) la dernière (première) cellule de la ligne.
(2) idem pour les colonnes, comme le montre la Figure 2.

c	v			v
v	v			v
v	v			v

FIG 2. Exemple de voisinage cyclique pour la cellule c. Ses voisines sont les cellules v.

D. Chargement d'un univers depuis un fichier de configuration

Les univers sont définis de façon aléatoire, développer un mécanisme pour charger un univers depuis un fichier de configuration. Utiliser un fichier texte dont le format est le suivant pour représenter un univers:

(1) hauteur et largeur de l'univers sur la première ligne.
(2) nombre de cellules vivantes lors du début du jeu sur la seconde.
(3) liste des coordonnées (i, j) de ces cellules (une cellule par ligne).
(4) nombre de cellules non viables de l'univers.
(5) liste des coordonnées (i, j) de ces cellules (une cellule par ligne).

La Figure 3 présente le format de fichier à utiliser ainsi qu'un exemple de fichier.

hauteurUnivers largeurUnivers nbCellules Vivantes i j ... nbCellules NonViables i j ...	5 10 3 1 1 2 4 0 9 2 3 1 2 0

FIG 3. **Sur la gauche**: Format des fichiers decrivant un univers.
Sur la droite: Exemple de fichier d'ecrivant un univers.

Une fois l'univers chargé, il devra être possible de l'utiliser sur tous les types de jeu implémentés.

E. Probabilites d'apparition des cellules

Développer un mécanisme de probabilites d'apparition des cellules(i.e, chances de survie ou de vie (en %) en fonction du nombre de voisins). Une cellule pourrait ainsi par exemple avoir 50% de chances de naître si elle avait 5 voisins vivants. Pour suivre strictement les règles du jeu de la vie, il suffit de regler la probabilite d'apparition à 100% ou 0% suivant les cas. Ce reglage se fait simplement en editant le fichier de configuration.

F. Le jeu torique

L'univers est un tore de dimensions k×k. Cela signifie que l'univers est un carré de dimensions k×k où les cases du bord supérieur sont voisines de celles du bord inférieur et où les cases du bord droit sont voisines de celles du bord gauche. Développer un mécanisme permettant de faire du plateau de jeu un tore. Ce reglage se fait simplement en editant le fichier de configuration: activation (1) ou non (0) de loption.

G. Garbage collector

Développer un mécanisme de "*garbage collector*". La problematique est de reduire le nombre de malloc et de free pour rendre le programme moins gourmand en ressources système. L'interet est d'optimiser un minimum la gestion de la memoire et l'utilisation des ressources système. Les avantages de cette fonction se voient très facilement grace aux statistiques affichees en fin de parties. Le nombre d'allocations memoire effectif est largement inferieur au nombre de creation de grappes (qui correspond au nombre theorique d'allocation qu'il aurait fallu faire sans le garbage collector).

3. **Structure du projet**

Pour plus de transparance et une meilleure repartition des taches, le projet est constitue de plusieurs fichiers:

(1) Les fichiers *.h: ce sont les fichiers d'en-tete, qui contiennent les macro, les variables de preprocesseur et les descriptions importantes.
(2) Les fichiers *.c: fichiers sources, c'est le contenu fonctionnel du jeu. Exemples de fichiers: (*i*) fichier sources pour constituer la boucle principale, (*ii*) une fichier *a* qui est le moteur du jeu, (*iii*) une fichier pour permettre laffichage graphique.
(3) Un Makefile pour la compilation croisee des fichiers .c et .h, ainsi que pour les flags de compilation.
(4) D'autres fichiers classiques pour des projets de programmation: README (contenant les lignes de commande pour compiler et lancer le programme), CHANGELOG, AUTHORS
(5) Un fichier de configuration.

En plus, utilisez un logiciel permettant de générer de la documentation à partir du code source (e.g., Doxygen).

Il est demandé de programmer de façon modulaire.

4. **Notation**

Le projet est à rendre sous forme électronique pour le mardi **6 janvier 2009** et fera l'objet d'une soutenance dans la semaine suivante.

Ce projet sera évalué sur **la base du travail personnel** réellement fourni pour le réaliser. ***Le travail est à réaliser seul***.

Il devra comporter au minimum:

(1) Un rapport présentant obligatoirement des précisions éventuelles du sujet, vos choix de conception et vos algorithmes les plus significatifs en pseudo-code. Il doit d'ecrire la façon dont vous avez implémenté le programme.
(2) L'ensemble des sources commentées.
(3) Tout programme doit obligatoirement compiler sans warning et s'exécuter sans plantage et sans procédure complexe dans les salles de TP.
(4) Vos jeux d'essais.

Remark

La qualité du rapport et du code, de la conception et la clarté du code compteront pour une bonne partie de la notation du projet.