Le jeu de la vie (automate de Conway)

Le jeu de la vie est un automate cellulaire, proposé par Conway. Il ne s'agit pas à proprement parler d'un jeu, mais d'une modélisation simplissime du comportement d'une population de cellules sur une grille évoluant au fil du temps via des règles de disparition et de reproduction. Le jeu consiste en une grille bi-dimensionnelle dont chaque case, peut être soit occupée par une cellule soit inoccupée. À chaque pas de temps, le prochain état d'une case de la grille est déterminé de façon unique par l'état des huit cases voisines de la façon suivante :

reproduction : une case inoccupée ayant exactement trois voisines occupées par des cellules devient vivante ;

isolement : une cellule vivante n'ayant pas plus qu'une case voisine occupée disparaît ;

étouffement : une cellule entourée par au moins quatre cellules meurt ;

maintient : une cellule entourée de deux ou trois case occupées se maintient en vie.

Le but de ce TP est de mettre en place une implémentation de cet automate cellulaire permettant de créer une configuration initiale de cellule sur une grille, puis de suivre l'évolution au cours du temps de cette colonie obéissant aux règles d'évolution ci-dessus. L'implémentation a pour but de manipuler les éléments de base d'une interface graphique (fenêtres, boutons, écouteurs d'événements, etc.)

Exercice 1. Affichage d'une grille

Dans cette première partie, nous nous intéressons uniquement à l'affichage d'une grille de cases. Nous implémenterons le comportement du jeu par la suite.

a. Écrire une classe `Case` qui représentera une case de la grille. Comme nous voudrons plus tard pouvoir cliquer sur une case pour choisir la configuration initiale, cette classe devra hériter de la classe `JButton`. La classe doit contenir :

— un booléen `occupied` pour indiquer si la case est occupée par une cellule ou non
— un attribut statique `size` qui contient la taille (en pixels) d'une case sur la grille

b. Écrire une classe `Grid` qui représente la grille. Cette classe doit gérer l'affichage de la grille, et donc doit hériter de `JPanel`. Elle devra contenir des attributs pour la taille de la grille et un tableau de `Case`.

Note : Le layout permettant de facilement agencer les éléments selon une grille est le `GridLayout`. Il peut être construit à partir d'un nombre de lignes et de colonnes.

Note : On peut donner une taille particulière à un `JPanel` de la façon suivante :

```
setPreferredSize(new Dimension(largeur, hauteur));
```

c. Écrire une classe `GameWindow` qui représente la fenêtre de jeu. Elle doit hériter de `JFrame` et doit instancier et contenir la grille de jeu.

d. Écrire une classe `Game` implémentant une méthode `main` qui lance l'affichage de la `GameWindow`.

Exercice 2. Saisie de la configuration initiale

Implémentez un `ActionListener` pour les cases. Un clic doit avoir deux effets :

1. mettre à jour la valeur du booléen `occupied`
2. changer la couleur de la case

Note : On peut changer la couleur d'un `JPanel` grâce à la méthode `setBackground`.

Exercice 3. Implémentation du jeu

a. Ajoutez une méthode `step` à la classe `Grid`, qui met à jour chaque case en fonction des règles du jeu décrites en introduction.

b. Ajoutez un écouteur des événements du clavier pour que le jeu se lance (ou se remette en pause) lorsqu'on appuie sur 'entrée'.

c. Faites en sorte que le jeu s'accélère lorsqu'on appuie sur '+', et qu'il décélère lorsqu'on appuie sur '-'.

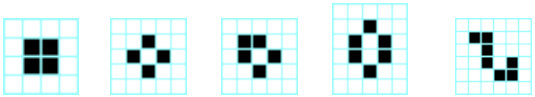
Exercice 4. Pour aller plus loin

a. L'interface du jeu est pour l'instant un peu austère : elle ne contient que la grille ! Plutôt que de gérer la mise en pause, l'accélération et la décélération du jeu via des touches du clavier, gérez les via des éléments graphiques (boutons, sliders).

b. Ajoutez une `JComboBox` pour pouvoir choisir une configuration initiale parmi plusieurs pré-enregistrées.

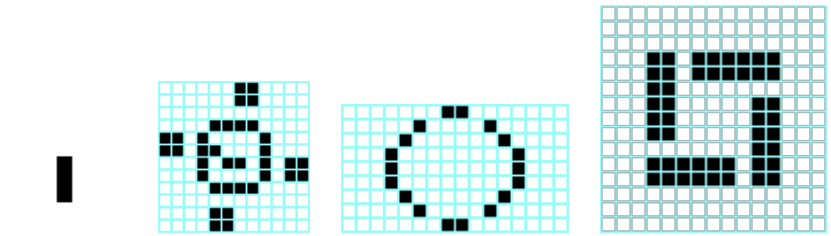
A Quelques exemples d'automates cellulaires

Structures stables. Les configurations suivantes sont stables : elles n'évoluent pas au fil du temps :



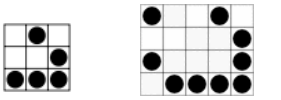
Bloc Tube Bateau Ruche Hameçon

Structures périodiques. La première question qui s'est posée est de savoir s'il existe des configurations qui restent indéfiniment instables. La réponse est oui. Voici une liste quelques un des plus célèbres qui sont périodiques, c'est à dire qui évoluent et reviennent à leur situation initiale :



Clignotant Horloge pentadecathlon Galaxie

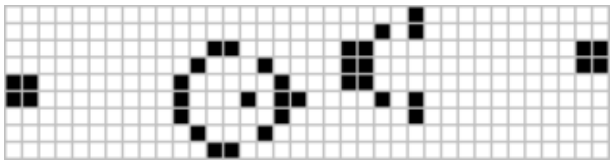
Structure vaisseau. Une structure est en vaisseau lorsqu'elle *se déplace* dans la grille. Il s'agit en général d'une structure cyclique mais dont la position après un cycle est translatée de la position précédente. Le plus petit vaisseau est le planeur (ou glisseur).



Glisseur Vaisseau léger

Autres structures. Le Pentamino R est un automate cellulaire de type Mathusalem, c'est à dire qu'il évolue pendant une longue période (ici 1103 générations) avant de se stabiliser vers des structures périodiques ou vaisseau.

Structures infinies. Enfin, la question suivante est de savoir s'il existe des configurations qui vont engendrer un nombre croissant de cellules. La réponse est oui, et une preuve est apportée par l'existence d'une configuration, le canon à glisseur de Gosper, dont une partie est périodique et qui à chaque cycle va générer un glisseur qui partira glisser vers l'infini.



Référence et Licence

Les images de cet énoncé proviennent des pages concernant les automates cellulaires et du jeu de la vie de wikipedia.fr et sont protégées par les licences CC-by-SA. On trouvera plus d'information sur le sujet sur ces pages : `https://fr.wikipedia.org/wiki/Jeu_de_la_vie`