Le jeu de la vie (automate de Conway)

2 séances de TP: 6e et 7e

Le TP peut être préparé seul ou en binôme. Le rendu doit consister en un fichier archive `<NOM1>_<NOM2>.tar.gz` ou `<NOM1>_<NOM2>.zip` contenant :

- votre code C++, commenté et facilement compilable (commande `make`)
- si la commande `make` ne suffit pas à compiler votre code, ajoutez un fichier README expliquant précisément comment compiler votre code.
- (optionnel) un compte-rendu de TP répondant aux questions qui ne demandent pas un programme ainsi que toute explication complémentaire.

Le TP doit être déposé sur moodle.
La date limite de rendu est la veille de la 8e séance de TP, à 23h59.

Le jeu de la vie est un automate cellulaire, proposé par Conway en 1970. 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 bidemensionnelle dont chaque case, peut être soit occupée par une cellule soit inoccupée. A chaque pas de temps, l'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 cases 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.

Exercice 1. Création du plateau

a. Écrire une classe `Grille` qui représentera la grille où les cellules évolueront. Elle devra contenir

- des nombres m de lignes et n de colonnes en champs privés.
- une `grille` représentant une matrice de valeurs booléenne, de dimension $m \times n$. Vous êtes libre de choisir comme l'implémenter (tableau de tableaux, tableau unidimensionnel, décrivant la matrice ligne par ligne, coefficients `bool, int` ou autre...)
- des méthodes `AjouteCellule`, `SupprimeCellule`
- une methode `Affiche` affichant la grille sur la sortie standard, en marquant les cellules par un caractère `'O'`.
- une méthode `EstOccupee(int i, int j)` indiquant si la case (i, j) est occupée.
- une méthode `NbVoisins(int i, int j)` retournant le nombre de cellules voisines de la cellule courante.

Exercice 2. Création du Jeu

On souhaite maintenant gérer le déroulement du jeu au fil du temps. Il faut pour cela écrire une classe `Jeu` qui maintient l'état courant d'une grille et dispose d'une méthode `Avance` qui effectue l'évolution d'un pas de temps.

a. Réfléchissez à comment effectuer cette mise à jour de la grille en limitant au maximum le nombre d'allocation en mémoire. Écrire cette classe `Jeu`. Elle devra disposer en outre, de méthodes permettant d'ajouter et de supprimer une cellule à un endroit donné, et une methode affichant la grille courante sur la sortie standard (`std::cout`).

b. Testez votre jeu en créant dans un `main` une boucle infinie effectuant successivement un appel à `Avance`, `Affiche`, et une attente d'un temps fixé par un champ de votre classe Jeu.

Exercice 3. Interface graphique

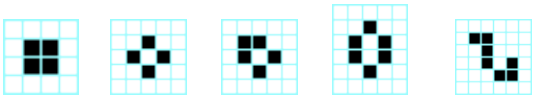
a. Proposer une interface graphique pour votre programme composée d'une fenêtre contenant
— Un panel de dessin où les cellules seront représentées par des disques
— Des boutons `Start`, `Stop` et `Clear`.
— Une possibilité d'ajouter/supprimer une cellule par un clic sur la zone de dessin, et ceci même pendant qu'un automate s'exécute.

b. Ajouter la possibilité de choisir les configurations initiales fournies en section 4 (dans un premier temps par des boutons, ou directement par une liste de selection, voir en ouvrant des fichiers).

Exercice 4. Quelques exemples d'automates cellulaires

Structures stables

Les configuration suivantes sont stables : elles n'évoluent pas au fil du temps :

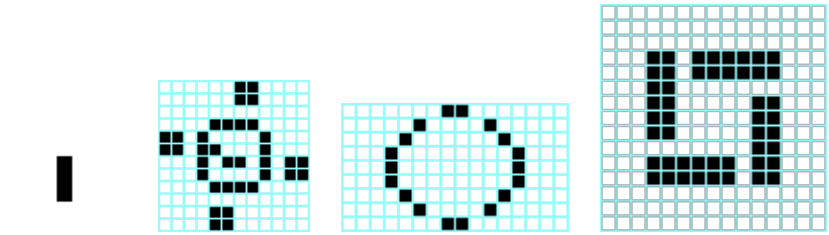


Bloc Tube Bateau Ruche Hameçon

Structures périodiques

La première question qui s'est posée est de savoir s'il existe des configurations qui restent indéfiniment instables.

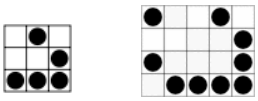
La réponse est oui. Voici une liste quelques un des plus célèbres qui sont périodiques, c'est à dire qui évoluent et reviennent à leur situation initiale :



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

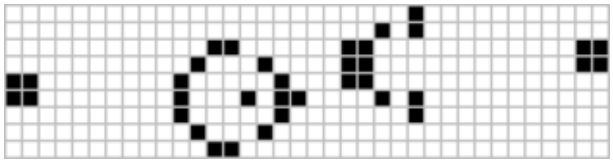
Autres structures

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 stabiiliser vers des structures périodiques ou vaisseau.

Le U est une structure également extrêmement simple, qui évolue des configuration beaucoup plus sophistiquée, en passant par une tête de clown.

Structures infines

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'existance 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`