Projet informatique

TP 5 Bonus : Jeu de la vie

Dans ce court dossier, je vais vous présenter une implémentation en langage C du jeu de la vie. Deux fichiers sont disponibles : game_life_L.c pour une executions sous linux et game_life_W.c pour une exécution sous windows. Vous pourrez, en naviguant à travers le menu du programme découvrir différents types d'objets que nous allons détailler plus tard. Prenez le temps de vous "amuser" à l'exécuter et à tester les fonctions.

Dossier et codes réalisés par :

CORCOS Ludovic
L2 Informatique
Université Clermont-Auvergne

Table des matières

I - Introduction

Un mathématicien américain d'origine hongroise nommé **John Von Neumann** est un vrai génie dont les travaux ont, entre autres, jeté les bases de l'informatique. Il se pose beaucoup de questions dans les années 1950, notamment sur **les automates autoreproducteurs** (pouvant reproduire une exacte copie d'eux-mêmes)

Un autre mathématicien, **Stanislas Ulam** (polonais), lui suggère la méthode décisive. Il veut créer sur des ordinateurs des jeux capables d'inventer des formes géométriques.

L'univers de Von Neumann commence ainsi par un damier où les cases, baptisées **cellules**, peuvent revêtir 2 états : éteint ou allumé.

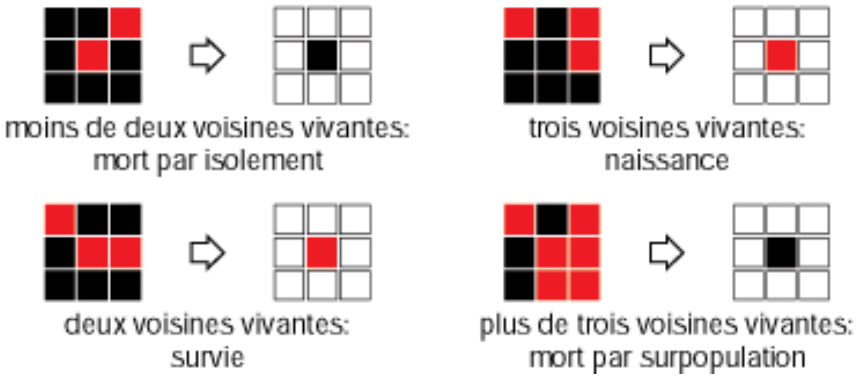
C'est John Conway, sur une idée de Von Neumann, l'inventeur du célèbre jeu de la vie (1970) : jeu à un joueur dont l'objectif est la survie et la croissance d'une population représentée par des jetons sur un quadrillage dont les cases sont des cellules tant au sens biologique que topographique...

II - Rappel des règles

Le monde du « Jeu de la vie » est un plan infini quadrillé, dont **chaque case est, soit occupée par une cellule, soit vide** (une cellule peut être morte ou vivante). Avant de commencer le jeu, on fournit un état initial.

Le jeu est maintenant prêt à commencer, ce qui implique d'avancer dans le temps une étape à la fois. **Le sort d'une cellule dépend de l'état de ses 8 voisins** les plus proches (la grille utilise un habillage, ce qui signifie qu'une cellule à l'extrême gauche est considérée comme le voisin d'une cellule à l'extrême droite, et le même principe s'applique en haut et en bas), **c'est une grille toroïdale.**

Si une cellule est vivante et que 2 ou 3 de ses voisins sont également vivants, la cellule reste vivante.
Si une cellule est vivante et qu'elle a plus de 3 voisins vivants, elle meurt de surpopulation.
Si une cellule est vivante et qu'elle a moins de 2 voisins vivants, elle meurt de solitude.
Si une cellule est morte et qu'elle a exactement 3 voisins, elle redevient vivante.

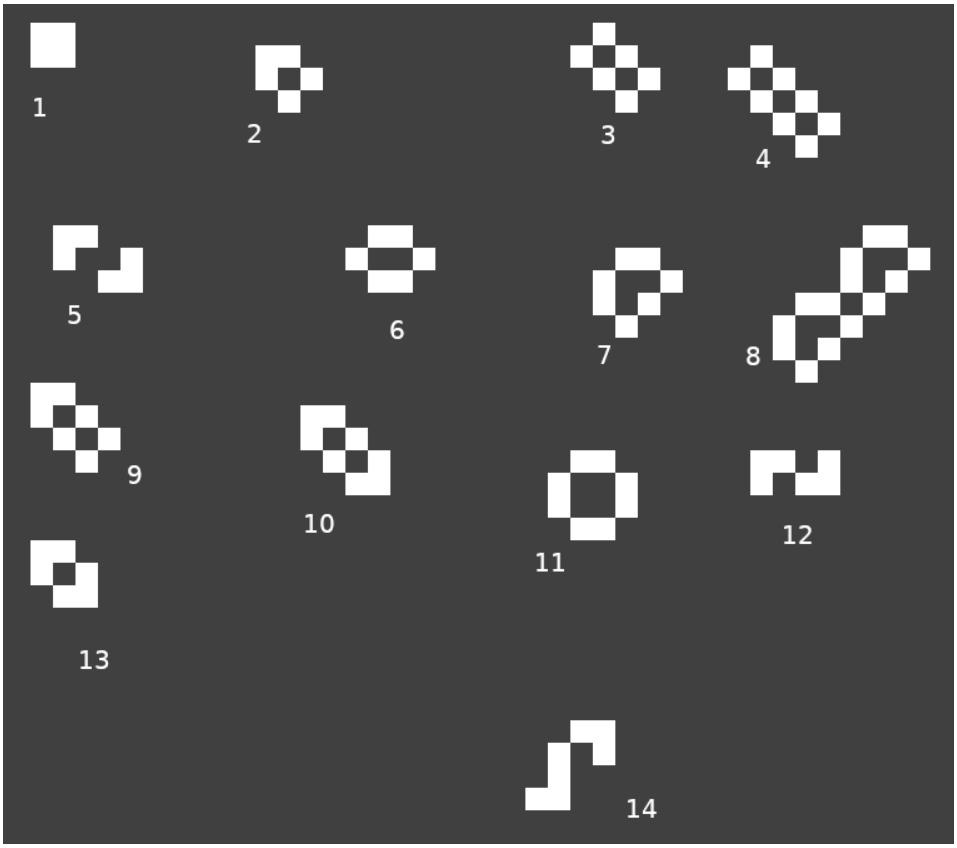


Ces 4 règles apparemment simples peuvent entraîner des séquences différentes et folles. **Parfois, un état initial crée une séquence chaotique imprévisible. D'autres fois, il créera une séquence répétitive** (comme le planeur, le pulsar et le vaisseau spatial ou d'autres que nous verrons plus tard). **Et d'autres fois, toutes les cellules mourront rapidement ou se stabilisent en une formation statique**, connue sous le nom de nature morte, comme un carré 2x2 (bloc).

III - Présentations des formes implémentées

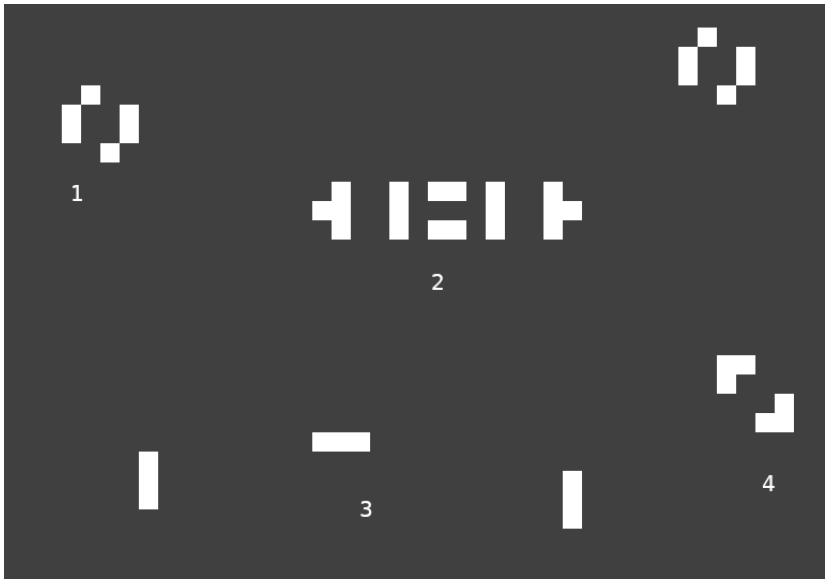
Le programme fonctionne grâce aux fichiers *.lif qui sont en réalité des tableaux pré-créé contenant soit des 0 soit des 1 séparés par des tabulations. Un "1" représente une cellule vivante, tandis qu'un "0" représente une cellule morte.
Je vous présente dans ce qui suit les différentes formes de bases du jeu de la vie.

- **Les formes stables :**



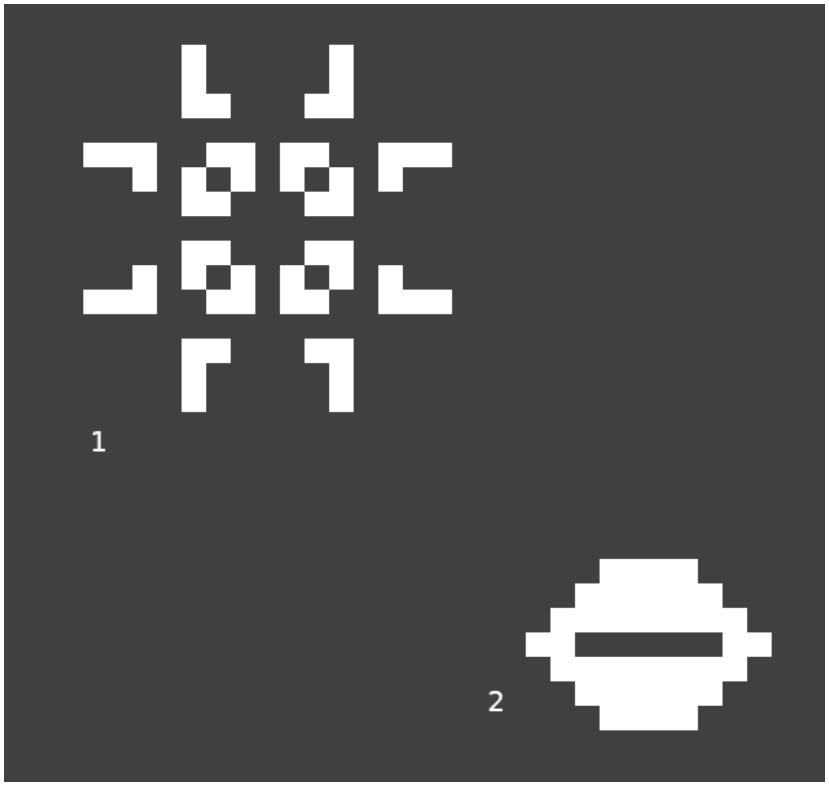
1) Bloc
2) Bateau
3) Péniche
4) Longue péniche
5) Porte-avion
6) Ruche
7) Flâneur
8) Bi-flâneur
9) Bateau long
10) Navire long
11) Etang / mare
12) Serpent
13) Navire
14) Mangeur

- **Formes oscillantes :**



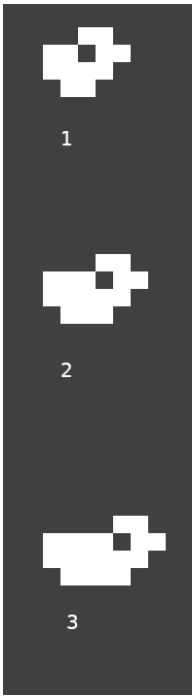
1) Crapaud
2) Pentadecathlon
3) Clignotant
4) Phare / balise

- **Les pulsars :**



Ce sont de véritables "géants", oscillant entre 3 phases de 48, 56 et 72 cellules. Le plus remarquable est que cette forme gigantesque et durable est le fruit de la croissance de l'une des 3 graines de pulsar définies dans le fichier “Puls.lif”.

- **Les vaisseaux**



1) Vaisseau léger

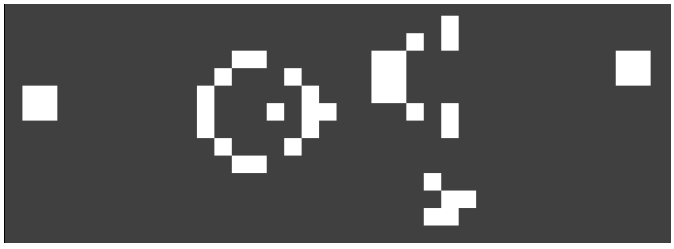
2) Vaisseau moyen

3) Vaisseau lourd

- **Mangeur et canon à planeur de Gosper :**

Un canon (gun dans la littérature anglophone) est un motif fini dont la partie principale se répète périodiquement, comme un oscillateur, et qui é**met des vaisseaux de type planeur à intervalles réguliers.**

Bill Gosper a découvert le premier coup de canon à planeur en 1970, gagnant 50 $ de Conway. La découverte du pistolet de planeur a finalement conduit à la preuve que le jeu de la vie de Conway **pourrait fonctionner comme une machine de Turing.**



Le mangeur de planeur est une petite forme **permettant de faire disparaître, de manger les planeurs sans changer de forme.**

- **Les autres options présentes dans le menu**

- Evolution ruche et escalier

Dans ce cas, il est intéressant de voir comment des petites structures peuvent “dégénérer” pour au final **arriver à des structures stables** presque de manière imprévisible.

- Croissance infinie

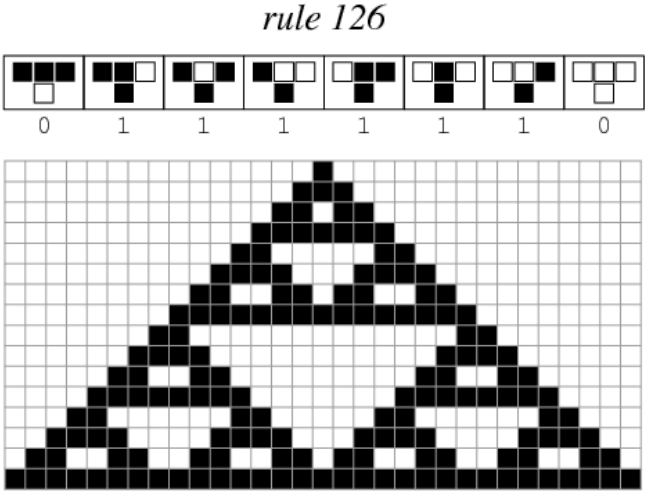
Une graine simple au départ, permet de construire un motif qui laisse **des débris stable** derrière lui, et ainsi peut **grandir de manière presque infinie !**

- Configuration aléatoire - personnalisable

Le fichier pour lequel est issu cette simulation a été généré grâce au générateur aléatoire de type **Mersenne Twister implémenté par Makoto Matsumoto.** Le fichier est composé de “1” et de “0” dispersés de manière aléatoire pouvant provoquer une situation avec une **évolution peu prévisible.**

IV - Automates cellulaire - règle 126

Les automates cellulaires élémentaires ont deux valeurs possibles pour chaque cellule (0 ou 1), et des règles qui dépendent uniquement des valeurs de voisin le plus proche. En conséquence, **l'évolution d'un automate cellulaire élémentaire peut être complètement décrite par un tableau spécifiant l'état d'une cellule donnée dans la prochaine génération en fonction de la valeur de la cellule à sa gauche, de la valeur de la cellule elle-même et de la valeur de la cellule à sa droite.** Puisqu'il existe $2 \times 2 \times 2 = 2^3 = 8$ des états binaires possibles pour les trois cellules voisines d'une cellule donnée, il existe un total d' 2^8 = **256 automates cellulaires élémentaires**, dont chacun peut être indexé avec un nombre binaire à 8 bits (selon Wolfram 1983, 2002).



La règle 126 est l'une des règles élémentaires des automates cellulaires introduites par Stephen Wolfram en 1983. Il spécifie la couleur suivante dans une cellule, en fonction de sa

couleur et de ses voisins immédiats. Ses résultats de règles sont encodés dans la **représentation binaire du nombre 126 = 01111110_2** . Cette règle est illustrée ci-dessus avec l'évolution d'une seule cellule noire qu'elle produit après 15 étapes. **C'est le triangle de Sierpinski (structure fractale).**

Je vous conseille vivement, pour tester cette fonction avec le programme de **mettre votre terminal en mode plein écran** (si possible avec une résolution 1920 * 1080) pour garantir un affichage optimal.

Voici ce que devrait afficher votre terminal :