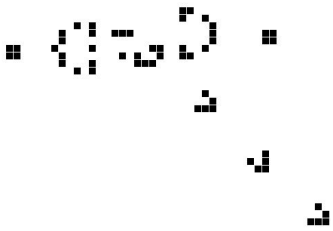
Autour du jeu de la vie

14 mai 2007



Le jeu de la vie est un automate cellulaire imaginé par John Horton Conway en 1970. Il peut être vu comme une modélisation d'une population de cellule. Le principe est assez simple : on se place sur une grille infinie où à chaque case représente une cellule qui peut être soit vivante, soit morte.

La population évolue par étape selon des règles simples. Chaque cellule a 8 voisins. Toutes les règles sont appliquées de façon simultanée.

- Si au temps t, une cellule est vivante et a strictement moins de 2 voisins vivants, elle meurt d'isolement.
- Si une cellule est vivante et a strictement plus de 3 voisins vivants, elle meurt d'étouffement.
- Si une cellule est morte et a exactement 3 voisins vivants, elle reprend vie.

Par exemple,
```
####        ##
#     devient ###
            # #
```

Exercice 1. Pour commencer

Sur une feuille de papier, calculer deux étapes d'évolution de ces trois motifs :

```
##          ###          ##
##                       # #
                         #
```

Modélisation

Dans la suite du TP, on se propose de programmer une simulation du jeu de la vie.

Restriction : un jeu fini

Représenter un plan infini peut poser quelques problèmes, on se limitera donc à modéliser un carré de taille $N \times N$ (on prendra par exemple $N = 20$). Afin de régler les problèmes qui peuvent se poser aux bords, on suppose que l'espace est torique, c'est à dire que la collone de gauche est voisine de celle de droite, celle du haut est voisine de celle du bas. Ainsi chaque cellule a bien 8 voisins.

Celà est équivalent à considérer un jeu infini periodique où la colonne N est la même que la colonne 0.

Structure de donnée

Dans toute la suite, on utilisera un tableau de N sur N contenant un 1 si la case est vivante et un 0 si elle est morte. La configuration vide peut par exemple être construite par

```
jeu :=[[0$N]$N] ;
```

Ainsi la cellule (i, j) peut-être accéder par `jeu[i,j]`.

Afin de pouvoir modifier N, on fera attention à utiliser `N` et non un nombre dans les programmes.

Exercice 2. Affichage et densité

a. Écrire une fonction `densite` calculant la densité d'une configuration (*ie* le nombre de cellule vivantes sur le nombre totale de cellule).

Pour tracer une configuration, on utilisera la commande `plot` qui permet de tracer des nuages de points avec une commande du genre :

```
plot([[7, 8], [1, 3], [1,4],[0,0]], x=0..10,y=0..10,style=point)
```

b. Écrire une fonction `affiche` qui prend une configuration et trace le nuage de point correspondant.

Exercice 3. Évolution

Parmis les voisins de la cellule $(1,1)$ se trouve les cellules $(1,2),(2,2),(2,1)$ mais aussi $(N,N),(1,N),\ldots$, ce qui peut poser des problèmes d'indices.

a. Écrire une fonction `f` qui prend en argument un entier i et rend un entier compris entre 1 et N, telle que $f(0)=N, f(N+1)=1,\ldots$

b. Écrire une procédure `voisins` qui prend en entrée une configuration et rend un tableau de $N\times N$ nombres où la case (i,j) est le nombre de voisins de la case (i,j).

c. Écrire une procédure `suivante` qui prend en entrée une configuration et rend la configuration suivante. La tester sur les trois exemples précédants.

Exercice 4. Tracé de l'évolution

Maple permet de tracer des annimations, en utilisant la commande `display`, avec le paramètre `insequence=true` qui se trouve dans la bibliothèque `plots` (que l'on charge avec la commande `with(plots)`. On se reportera à l'aide pour plus d'informations.

a. À l'aide des procédure précédantes, écrire une procédure `evol` prenant en entrée une configuration initiale et un entier n et rendant l'animation correspondant aux n premières étapes de l'évolutions.

b. la tester sur les exemples donnés au début. La tester aussi sur des lignes de 5, 7 et 8 cellules : configurations :

```
#####
```

Exercice 5. Configuration aléatoire, évolution du nombre de cellule

a. À l'aide de la commande `rand(n)()`, écrire une fonction prenant un nombre x et qui renvoie une configuration aléatoire où $x\%$ de cases sont remplies.

b. Écrire une fonction qui trace l'évolution de la densité en partant d'une configuration donnée.

c. Tester ces deux fonctions.

Exercice 6. Allons plus loin

a. Justifier le fait que dans un espace torique, tout jeu de la vie est ultimement periodique. Écrire un algorithme pour déterminer sa periode. Attention à l'occupation mémoire de votre algorithme.

b. Proposer des idées pour gérer un jeu de la vie infini (mais où le nombre de cellules vivantes serait fini).