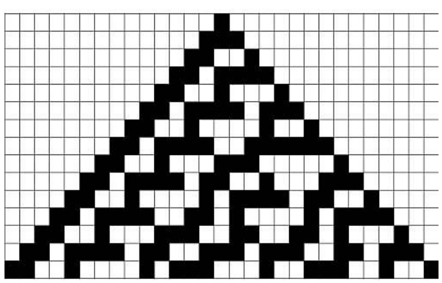
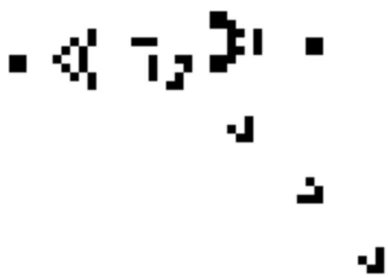
Institut national des sciences appliquées de Rouen

INSA de Rouen

Projet Info GM3 - Vague 1 - Sujet 2

Jeu de la vie





Auteurs :
Thibaut André-Gallis
thibaut.andregallis@insa-rouen.fr
Kévin Gatel
kevin.gatel@insa- rouen.fr

Enseignant :
Julien Saunier
julien.saunier@insa-rouen.fr

16 Octobre 2020

Table des matières

Table des figures

Introduction

Le jeu de la vie est un programme informatique imaginé par John Conway dans les années 70. Ce dernier est un mathématicien britannique née en 1937. Il vient de décéder il y a un mois à cause du Coronavirus. Ses travaux tournent autour des théories de groupes ainsi que les jeux combinatoires.

Depuis presque 50 ans ce jeu fascine les mathématiciens, les informaticiens mais aussi les biologistes et les philosophes. En effet il nous montre comment un système évoluant selon des règles simplistes peut engendrer des résultats incroyablement riches.

Il faut préciser que le jeu de la vie n'est pas vraiment un jeu au sens ludique, puisqu'il ne nécessite aucun joueur. Il s'agit simplement d'entrée une configuration initiale et de regarder son évolution.

C'est l'analogie entre ces règles et certains critères d'évolution de populations de bactéries qui a conduit à donner à cet automate le nom de jeu de la vie.

Dans ce projet, nous allons étudier le jeu de la vie à travers son implémentation en C [1]. Nous allons décrire le programme, montrer sa modélisation à partir de plusieurs dispositions initiales et fournir quelques analyses sur les résultats du jeu.

1. Le *main.c* est le programme principal, la grille initiale est définie dans les fichiers *fgrille*

1. Principe du jeu

Le jeu se déroule sur une grille à deux dimensions, théoriquement infinie (dans la pratique sa taille sera finie), dont les cases qu'on appelle des cellules (par analogie avec les cellules vivantes) peuvent prendre deux états distincts : vivantes ou mortes. On peut ainsi voir ce petit programme comme un automate cellulaire.

Le fonctionnement est très simple. L'état de l'automate à l'étape n est uniquement fonction de son état à l'étape $n-1$. L'évolution de l'état d'une cellule dépend de l'état de ses 8 cellules voisines.

Dans l'automate de Conway, les règles sont les suivantes :

- Une cellule morte à l'étape $n-1$ et ayant exactement 3 cellules voisines vivantes deviendra vivante à l'étape suivante. (naissance liée à un environnement optimal)
- Une cellule vivante à l'étape $n-1$ et ayant 2 ou 3 cellules voisines vivantes sera maintenue vivante à l'étape n sinon elle meurt. (destruction par désertification ou surpopulation)

Par exemple sur un cas très simple on aura :

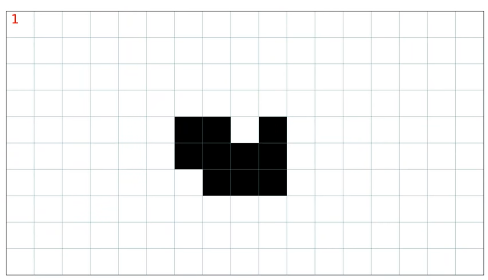


FIGURE 1.1 – Etape n_0

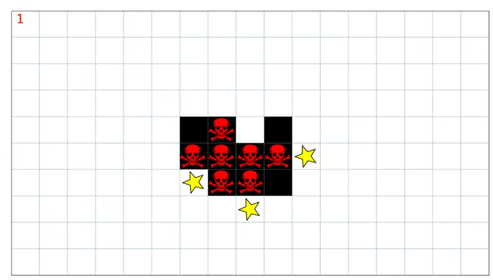


FIGURE 1.2 – Construction de l'étape $n_0 + 1$

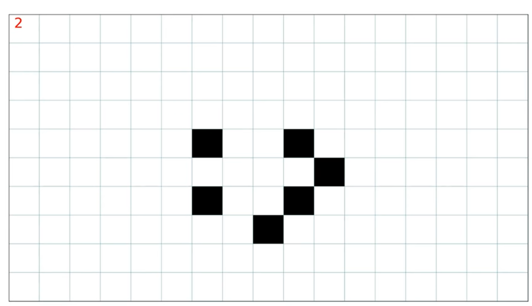


FIGURE 1.3 – Etape $n_0 + 1$

Ainsi, ce processus peut être répéter autant de fois que le souhaite l'utilisateur.

2. Généralités

Après plusieurs parties avec des configurations initiales différentes, on retrouve plusieurs situations et on s'aperçoit que certains motifs reviennent régulièrement.

On observe plusieurs cas récurrents :

- Soit les cellules finissent par mourir (ce cas n'est pas très intéressant)
- Soit les cellules convergent vers une ou plusieurs structures stables, ou cycliques (qui reviennent sur leur pas)
- Soit les cellules ont une disposition telle qu'elles progressent dans l'espace sans modifier leur structure

Lorsqu'on simule avec un état initial aléatoire, nous observons ces structures sous plusieurs formes différentes.

2.1 Structures Stables

On appelle figure stable un regroupement de cellules telle qu'à l'étape $n_0 + 1$ ce regroupement soit le même que l'étape n_0. Il existe plusieurs structures stables. Parmi elles, les plus récurrentes sont[1] :

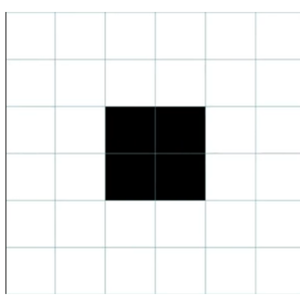


FIGURE 2.1 – Le bloc

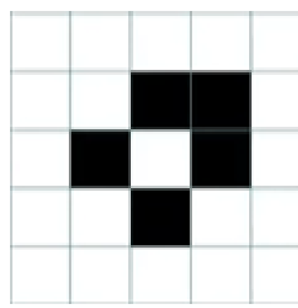


FIGURE 2.2 – Le bateau

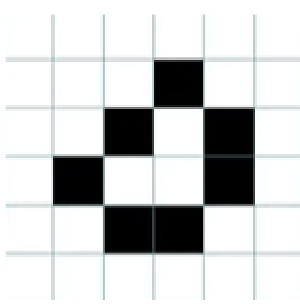


FIGURE 2.3 – Le pain

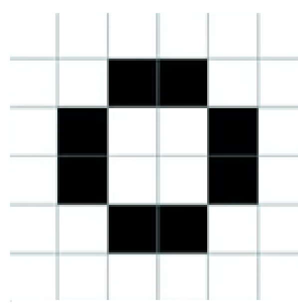


FIGURE 2.4 – La mare

1. Ces quatre cas de figure peuvent être observés en mettant le fichier *fgrille0* en état initial

2.2 Structures cycliques

On appelle structure cyclique ou oscillante un regroupement de cellules qui, au bout de n générations, retombe sur la même structure. On dira qu'elle est de période n. Il existe également plusieurs structures cycliques ou oscillantes. On observe les plus fréquentes ci-dessous [2] :

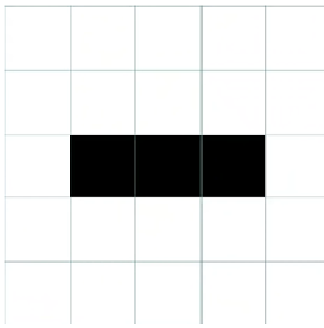


FIGURE 2.5 – Le clignotant n_0

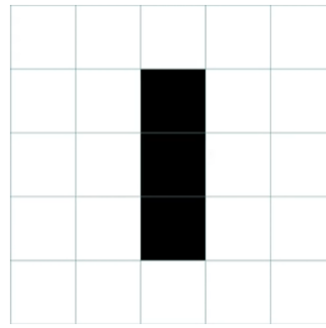


FIGURE 2.6 – Le clignotant $n_0 + 1$

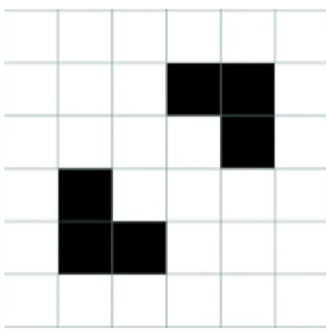


FIGURE 2.7 – La balise n_0

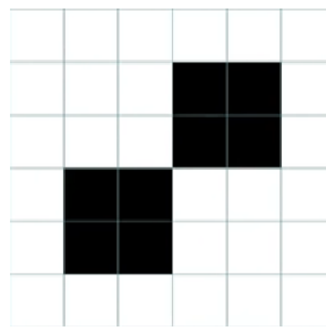


FIGURE 2.8 – La balise $n_0 + 1$

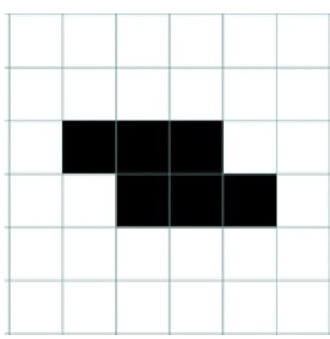


FIGURE 2.9 – Le crapeau n_0

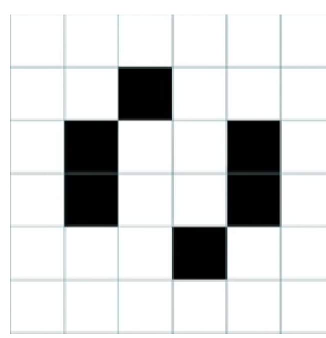


FIGURE 2.10 – Le crapeau $n_0 + 1$

2. Ces trois cas de figure peuvent être observés en mettant le fichier *fgrille1* en état initial

Celles-ci sont toutes de périodes 2. Mais il en existe plein d'autres de périodes plus élevées. Par exemple celle-ci se répétera toutes les 15 générations [3] :

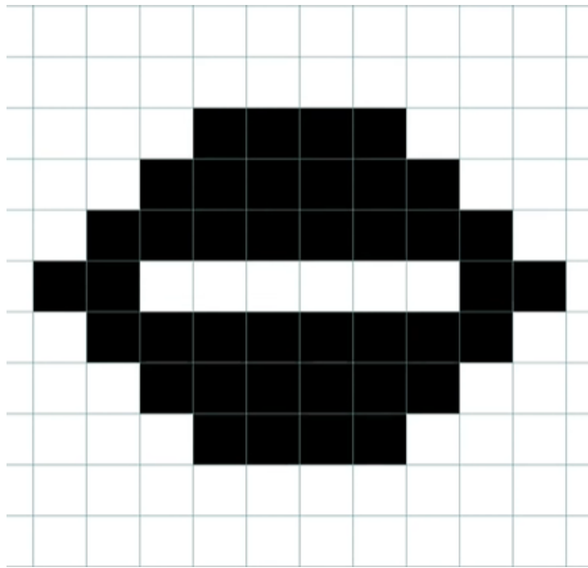


FIGURE 2.11 – Oscillateur de période 15

2.3 Planeurs

Un planeur est une structure qui retrouve sa configuration en s'étant déplacé d'une case voisine. Il existe des planeurs pouvant se déplacer par la droite (engendrant les planeurs se dirigeant la gauche, par le haut et par le bas à une rotation près) comme l'exemple ci-dessous [4] :

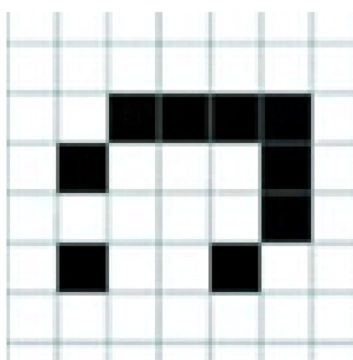


FIGURE 2.12 – Planeur de période 4

Et d'autres qui peuvent se déplacer en diagonale [5] :

3. Ce cas de figure peut être observé en mettant le fichier *fgrille2* en état initial
4. Cette structure peut être observée en mettant le fichier *fgrille3* en état initial
5. Cette structure peut être observée en mettant le fichier *fgrille4* en état initial

Figure 2.13 – L'oie du canada

2.4 Canons à planeurs

Les canons à planeurs sont les structures qui, lorsqu'elles retrouvent leur configuration initiale, émettent un ou plusieurs planeurs. [6] :

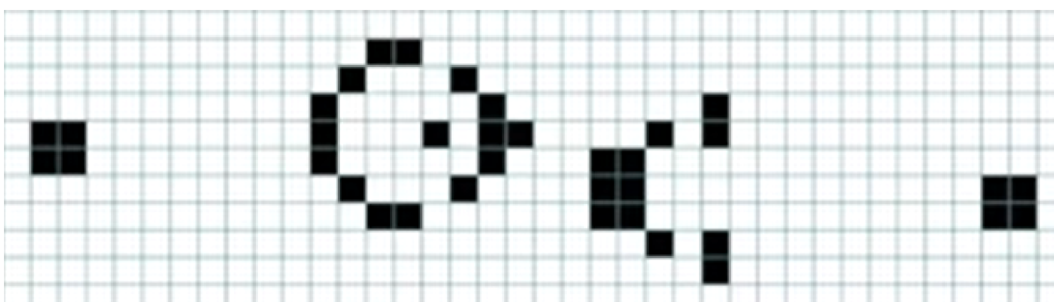


Figure 2.14 – Canon à planeurs de Gosper

La découverte d'une telle structure a provoqué une révolution dans le jeu de la vie. Celle-ci nous montre qu'avec un nombre fini de cellule, il est possible d'en créer avec intervalle régulier, soit une infinité.

6. Le canon peut être observé en mettant le fichier *fgrille5* en état initial

3. Programme

3.1 Indications d'utilisation

La taille d'un tableau doit être définie comme constante lors de la compilation d'un programme. Pour optimiser cette espace, nous avons décidé de la mettre en paramètre dans le programme qu'on décide arbitrairement selon la taille souhaitée.

Ainsi, il est important lors de la compilation du programme, de définir la variable N au niveau du #define dans le main.c puisqu'elle définit la taille du tableau à deux dimensions contenant toutes les cellules.

On aura alors pour l'appel des fichiers fgrille0, fgrille1, fgrille2 et fgrille3, un N devant être égale à 22. Pour les fichiers fgrille4 et fgrille5, N devra être égale à 45. Pour ce cas de figure, n'oubliez pas d'ouvrir le terminal en grand pour un meilleur affichage.

3.2 Pseudo-code

```
Programme Jeu de la vie

Constante : Morte ← 0
            Vivante ← 1
            color ← white
            vitesse ← 100
            N ← 45

Var :   Type Grille tableau de taille (N+2)*(N+2) d'entier
        i,tour,arret,c : entier
        cellule : grille

Procédure (E cellule : Grille)

var : i,j : entier

Début

        Pour i allant de 1 à N incrémenté de 1 Faire
        {
                Pour j allant de 1 à N incrémenté de 1 Faire
                {
                        Si cellule[i][j] = Morte
                        alors afficher("x " en rouge)
                        sinon afficher ("o " en color)
                }
                afficher (retour à la ligne)
        }

Fin

Procédure initfGrille(S cellule : Grille)

var :   fichier : fichier initialisé au fichier nul
        tab : tableau de taille N*N d'entier
        i,j,caractère : entier

Debut

        caractère ← 0
        fichier ← ouverture("fgrille4", lecture)

        Pour i allant de 0 à N+1 incrémenté de 1 Faire
        {
                Pour j allant de 0 à N+1 incrémenté de 1 Faire
                {
                        cellule[i][j] ← 0
                }
        }
        i ← 0
        Si (fichier ≠ fichier nul)
        alors faire
        {
```

```
        Tant que (caractère ≠ finfichier) Faire
        {
            caractère ← caractère(fichier)
            Si (caractère = 'x')
            alors Faire
            {
                tab[i] ← Morte
                i ← i+1
            }
            Si (caractère = 'o')
            alors Faire
            {
                tab[i] ← Vivante
                i ← i+1
            }
        }
        fermeture(fichier
        Pour i allant de 0 à (N^2)-1 incrémenté de 1 Faire
        {
            cellule[(i div N) +1][(i mod N) +1] ← tab[i]
        }
    }
    sinon afficher ("impossible d'ouvrir le fichier fgrille")

Fin

Fonction seraVivant(cellule : grille; x,y entier) : entier

var : i,j,result,compteur : entier

Début

    Pour i allant de x-1 à x+1 incrémenté de 1 Faire
    {
        Pour j allant de y-1 à y+1 incrémenté de 1 Faire
        {
            Si ((i ≠ x) et (j ≠ y))
            alors
                Si (cellule[i][j] = 1)
                alors
                    compteur ← compteur+1
        }
    }
    Si (cellule[x][y] = Vivante)
    alors
        Si((compteur ≠ 2) et (compteur ≠ 3)
        alors
            result ← Morte
        sinon
            result ← Vivante

    Si (cellule[x][y] = Morte)
    alors
```

```
                Si((compteur = 3)
                alors
                        result ← Vivante
                sinon
                        result ← morte
        retourner(result
Fin

Procédure Evolution(E/S cellule : grille)

Var :   i,j : entier
        cellule_temp : Grille
Début
        cellule_temp ← 0 //on initialise à 0 toutes les cases de cellule_temp
        Pour i allant de 1 à N incrémenté de 1 Faire
        {
                Pour j allant de 1 à N incrémenté de 1 Faire
                {
                        cellule_temp[i][j] ← seraVivant(cellule,i,j)
                }
        ]
        Pour i allant de 0 à N+1 incrémenté de 1 Faire
        {
                Pour j allant de 1 à N+1 incrémenté de 1 Faire
                        cellule[i][j] ← cellule_temp[i][j]
        }
Fin

Début

        tour ← -1
        arret ← 0
        initfgrille(cellule)
        afficher("tour n°0")
        affichage(cellule)
        afficheràlaligne("Combien Voulez-vous voir de génération ?")
        afficher("Nbre :")
        lire(tour)
        Pour i allant de 1 à tour incrémenté de 1 Faire
        {
                afficher("tour n°",i)
                Evolution(cellule)
                affichage(cellule)
                Si (i ≠ tour)
                alors
                        delai(vitesse*1000)
                sinon
                        arret ← 1
        }
        Si (tour<0)
        alors
                tour ← 0
Fin.
```

Conclusion

Tout d'abord ce projet a été enrichissant d'un point de vue programmation. En effet l'implémentation de ce programme relativement complexe nous a permis de nous familiariser avec un nouveau langage informatique qu'est le C. N'étant pas familier avec celui-ci, nous avons pu accroître nos compétences dans ce domaine.

D'autre part, nous ne connaissions pas le jeu de la vie avant cela. Nous avons donc découvert cette algorithme pourvu de deux règles très simplistes qui, en modélisant, est capable d'engendrer des cas de figure très complexes. Il semble ne pas y avoir de moyen de prédire l'évolution d'un système tiré aléatoirement

Cependant, la gestion de la taille du tableau contenant les cellules doit être en théorie infinie. Nous avons dû nous résoudre à travailler avec des dimensions finies à cause du stockage mémoire. Ainsi lorsqu'une structure touche un bord, son comportement ne peut plus être anticipé par le programme.

Enfin il a été très enrichissant de pouvoir comparer et compléter deux points de vue différents à l'origine pour arriver à la réalisation finale. Effectivement nous n'avions pas la même façon d'aborder le sujet lorsqu'il a fallu analyser le problème, mais la complémentarité de nos points de vue nous a permis d'arriver à analyser de façon complète le problème.

Ce projet nous a permis également à partir d'un problème mathématique, d'en faire un algorithme, pour enfin découvrir tout le potentiel une fois associé à l'implémentation informatique. Cet outil qu'est l'ordinateur se révèle indispensable ici dans la compréhension et l'utilisation d'un objet mathématique à l'origine.

La recherche d'animation engendrée par cet algorithme est sans fin, on pourrait s'amuser à en chercher à l'infini.

Annexes

Type grille = tableau[1..N,1..N] de booléen

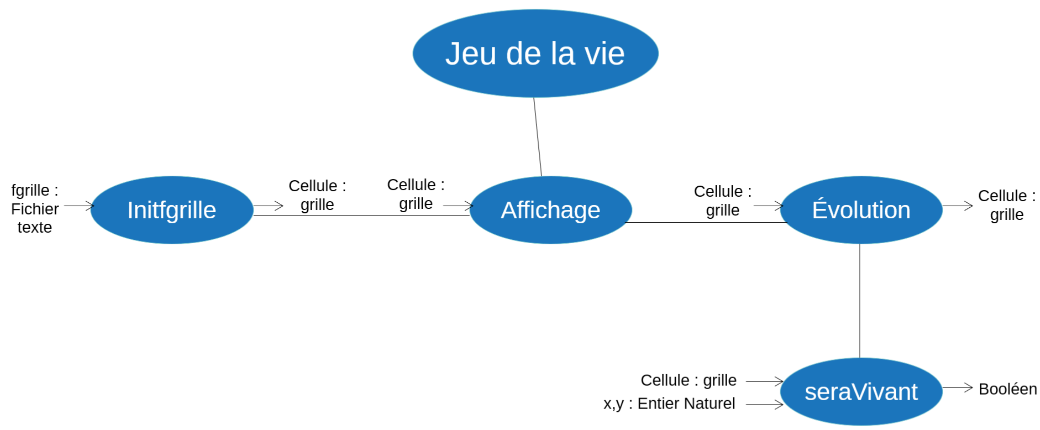


Figure 3.1 – Type(s) et Analyse descendante