Rapport Projet C
Jeu de la vie

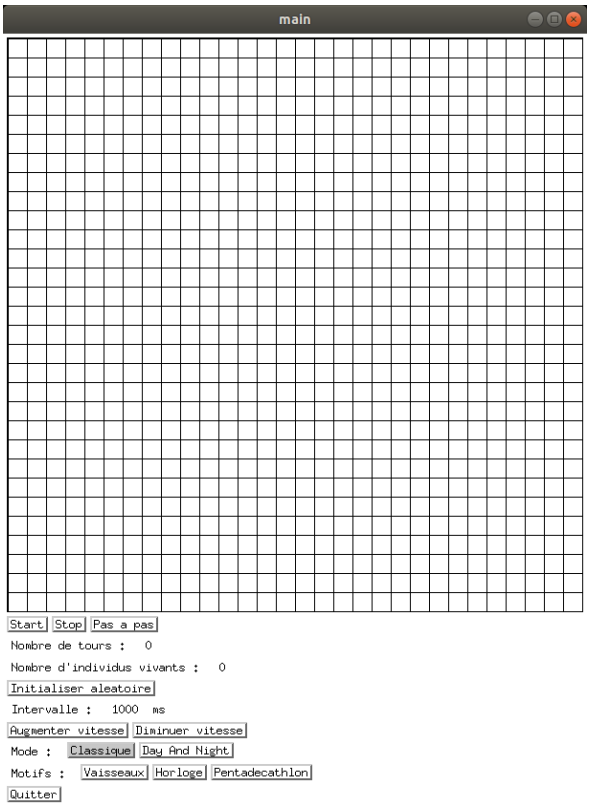
Enseignant : M. Vincent Granet

Année 2020 / 2021

Sommaire

Introduction

L'objectif de ce projet est de réaliser le jeu de la vie. Ce dernier est un jeu de simulation à zéro joueur à sens mathématique. En effet, nous avons au départ un nombre aléatoire d'individus sur une grille et selon certaines règles, ces individus apparaissent ou disparaissent.
Nous allons donc à travers ce projet, coder en langage C le jeu de la vie, et afficher ce dernier sur une interface graphique avec différents modes de jeu.



Voici l'interface du jeu. La grille est de dimensions 30x30.
L'utilisateur peut démarrer le jeu avec le bouton "start", et l'arrêter avec le bouton "stop". Il existe ensuite différentes fonctionnalités qu'il peut sélectionner pour faire varier aussi bien les règles que le motif de départ du jeu.

Explication code

Nous allons à présent vous expliquer notre codage, et comment nous avons pensé réaliser ce jeu de la vie pour que ce dernier soit le plus efficace possible.

Données (fichier .h)

Tout d'abord, nous allons expliquer comment nous avons choisi de structurer les données du programme.
Dans ce fichier se trouveront donc les données ainsi que les déclarations de fonctions du fichier modele.c car en effet, les données sont manipulées presque exclusivement par le modèle.
En premier, concernant les dimensions de la grille, on a choisi de les définir avec des #define car lors de l'exécution, elles resteront fixes.
Ensuite, nous avons déclaré trois types ETAT, RUN et MODE qui correspondront respectivement à l'état des cellules, si le jeu est fonctionnement ou non, et le mode de jeu (Classique ou Day and Night).
Enfin, nous avons créé une structure contenant toutes les données dynamiques du jeu, telles que la grille, le mode, les boutons devant être modifiés...

Modèle (fichier .c)

Nous allons dans ce fichier, mettre toutes les fonctions en lien avec la manipulation des données du jeu.

Ces fonctions possèdent toutes la même structure. En effet nous utilisons deux boucles for imbriquées pour parcourir notre grille de taille HAUTEUR * LARGEUR.

Dans un premier temps, la fonction *fillTabRand* permet de remplir aléatoirement un tableau de taille HAUTEUR * LARGEUR avec des 0 et des 1.

La fonction *initJeu* est ici pour initialiser notre jeu. En effet, les variables telles que le nombre de tours par exemple sont mises à 0. Elle permet aussi de remplir aléatoirement ce nouveau tableau avec notre fonction *fillTabRand*.

Les fonctions *afficheTab*, et *copieTab* permettent respectivement d'afficher, et copier notre tableau.

Les fonctions *nbVoisins*, et *nbVivants*, renvoient respectivement le nombre de voisins, ainsi que le nombre de vivants. Ceci nous est utile pour une meilleure compréhension du déroulement du jeu par l'utilisateur.

La fonction *compTab* compare deux tableaux, s'ils sont différents, alors celle-ci renvoie 0, sinon elle renvoie 1.

La fonction *fillMotif* permet de remplir notre tableau avec un motif dessiné auparavant dans un fichier. Nous utilisons donc ceci pour notre vaisseau, notre horloge, ou encore notre pentadecathlon.

Enfin, la fonction *tourSuivant* permet de calculer la grille à l'état suivant dans le jeu. Pour ce faire, nous vérifions le mode sélectionné, puis nous parcourons le tableau en y appliquant les règles correspondantes. Enfin, nous vérifions à la fin si les deux tableaux (l'ancien et le

nouveau) sont bien différents, si oui alors on laisse le jeu continuer. Si cela est non, alors cela veut dire que le jeu est bloqué et qu'il n'y a plus de possibilité de mouvement.

On notera que dans toutes les fonctions, les variables x et y permettant de parcourir le tableau seront déclarées de manière statique. Ainsi à chaque exécution de ces fonctions, on ne refera pas l'allocation en mémoire de ces variables. Comme par exemple sur la fonction *nbVoisins* qui est appelée 900 fois à chaque tour, cela peut avoir un certain intérêt.

Vue (fichier .c et .h)

Ces fichiers contiennent les éléments de notre code qui sont en lien avec la partie visuelle de notre interface. Nous y avons pour cela réalisé trois fonctions.

Dans un premier temps, nous avons réalisé la fonction *fillGrille.* Cette fonction permet de réaliser l'affichage visuel de notre grille sur notre interface graphique. Pour ce faire, nous utilisons les fonctions *DrawFilledBox*, ainsi que *DrawBox*. Nous décidons d'y afficher aussi le nombre de cellules vivantes, ainsi que le nombre de tours.

La fonction *doAnim* permet de faire fonctionner le jeu en mode automatique, sans action de l'utilisateur. Lorsque notre variable run passera à 1, c'est-à-dire que l'utilisateur lancera le jeu, alors nous afficherons la grille à l'aide de notre fonction *fillGrille*. Et lorsque la variable run repassera à 0, alors l'affichage ne se fera plus et le jeu sera en pause.
Concernant la programmation du mode automatique, nous avons choisi que cette fonction s'auto-appellera tout le temps, du lancement jusqu'à la fermeture du jeu. Les mises en marche et les arrêts ne seront donc que "visuels". Si on a voulu faire ainsi, c'est pour avoir un maximum de fluidité dans le déroulement du jeu, et éviter des "sauts" d'affichage liés au démarrage et à l'arrêt de la fonction.

Enfin, la fonction *initDisplay* est ici utilisée pour créer l'intégralité de la partie visuelle, c'est-à-dire nos boutons, nos labels...
Nous utilisons donc les fonctions *MakeButton* et *MakeLabel* pour créer la partie interactive. En effet, les boutons sont reliés à leurs callbacks respectifs. C'est-à-dire que lorsque l'utilisateur appuie sur le bouton, le programme exécutera la fonction correspondante.
Enfin, nous utiliserons la fonction *SetWidgetPos* pour positionner nos boutons pour que ces derniers soient visibles et agréables à utiliser.

Callbacks (fichier .c et .h)

Tout d'abord, nous écrirons notre code pour créer nos différentes fonctions dans notre fichier callbacks.c, qui sont déclarées dans le fichier callbacks.h. Ce fichier sera ensuite utilisé pour être appelé par différents fichiers utiles à la programmation du jeu. Nous pourrons alors grâce à cela, utiliser ces fonctions dans d'autres fichiers.

Nous allons dans ce fichier callbacks, créer les différentes fonctions qui sont en lien avec la manipulation du jeu par l'utilisateur. Il s'agit des actions effectuées lors des clics sur les boutons.

Tout d'abord, la fonction *drawGrille,* sert ici à créer la base de notre jeu. Nous dessinons un tableau de taille HAUTEUR*LARGEUR, grâce à la fonction *DrawBox*.

La fonction *quit* sert à quitter le jeu.

Les fonctions *start* et *stop* sont ici utilisées pour commencer ou mettre en pause le jeu. Pour ce faire, nous passons par une structure *data*, avec laquelle nous utilisons la variable run pour commencer ou arrêter le jeu.

La fonction *PasAPas* sert ici à continuer le jeu, seulement si l'utilisateur appuie sur le bouton "Pas a pas". Cependant, ce bouton ne relance pas le jeu mais applique un seul tour de jeu, ensuite le jeu s'arrête et attend un nouvel appui pour continuer.

La fonction *initGrilleRand* permet de remplir notre grille de manière aléatoire. Nous utilisons pour cela notre fonction *fillTabRand*.

Les fonctions *vaisseaux, horloge, pentadecathlon,* sont ici pour utiliser le jeu avec une structure de départ définie. En effet, il nous est demandé dans la partie 3 du sujet de tester notre jeu avec différentes structures. Nous avons donc créé ces structures dans des fichiers (avec des 1 et des 0), puis nous les appelons dans ces fonctions pour les afficher sur notre grille. Nous obtenons bien des vaisseaux comme demandés, ainsi qu'une structure périodique, ici une horloge ou un pentadecathlon.

Les fonctions *DayAndNight* et *Classique* permettent de choisir le mode de jeu "Day and Night", ou le mode de jeu "classique".

Enfin, les fonctions *SpeedUp* et *SpeedDown,* sont ici réalisées pour diminuer ou augmenter la fréquence du jeu. Pour faire ceci, on utilise l'entier *time*, ainsi que le widget *labelTime*. A chaque appui, la variable *time* sera changée, et on écrira la valeur de cette dernière grâce à la fonction SetLabel.

Main (fichier .c)

Dans la fonction *Main* nous initialisons tous composants du programme. En effet, on y initialise les données, ainsi que l'interface graphique pour pouvoir les utiliser. Enfin, nous démarrons l'interface avec *MainLoop*.

Makefile

Au niveau du Makefile :

CC = gcc
CFLAGS = -Wall
LDFLAGS = -lsx
SRC = main.c modele.c vue.c callbacks.c
PROG = main

On indique d'abord le compilateur utilisé, ensuite les options de compilation puis les options d'assemblage. Après, on met tous les fichiers source (.c) et enfin, le nom de l'exécutable que l'on veut.

Après ça, il n'est pas très utile de détailler les commandes exécutées en revanche, la dernière partie importante est le lien entre les fichiers .h et les fichiers objet :

main.o: donnees.h vue.h
modele.o: donnees.h
vue.o: donnees.h vue.h callbacks.h
callbacks.o: donnees.h vue.h callbacks.h

Pour chaque fichier .c, un fichier correspondant .o sera créé et pour chacun, il faut indiquer les fichiers .h que l'on a inséré dedans, à part les librairies du système. Il faut seulement mettre les fichiers .h que l'on a créés.

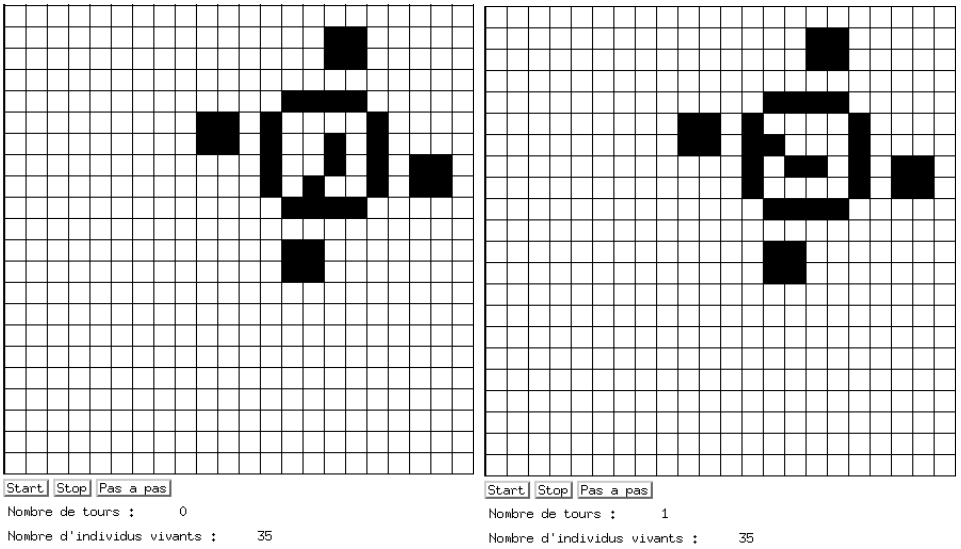
Résultats partie 3

Dans cette partie, il nous est demandé de réaliser plusieurs structures classiques (périodiques, vaisseaux ou stables), et de comparer l'évolution de la population en fonction des deux règles de génération du programme.

Structure périodique (Mode Classique) :

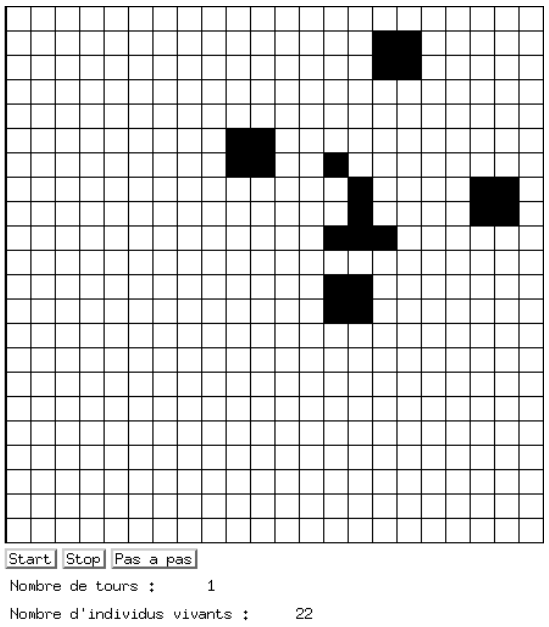
Pour illustrer les structures périodiques, nous avons deux motifs qui sont l'horloge et le pentadecathlon. Ici on a choisi de décrire l'horloge.

Au tour n=0, on voit le motif de base :



Au tour n+1, on voit que l'aiguille a tourné d'un quart de tour dans le sens horaire… On observe notamment que le nombre d'individus vivants à chaque tour reste fixe.

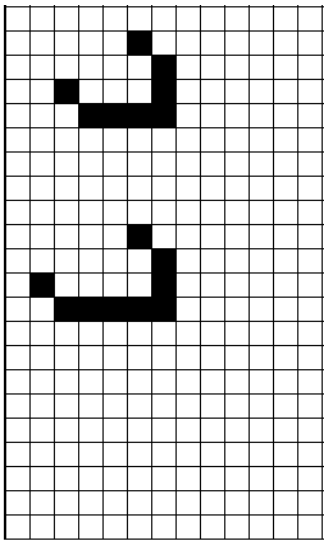
Structure périodique (Mode Day and Night) :

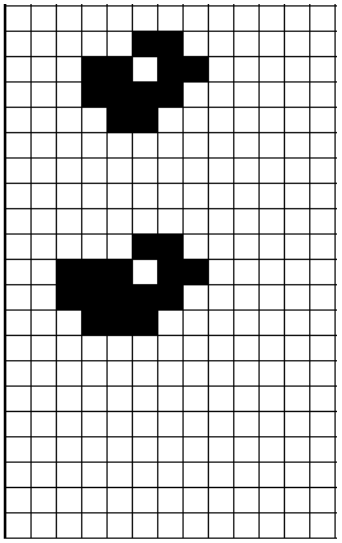


Maintenant si on exécute le motif de l'horloge en mode Day and Night, on voit qu'au tour n+1, la structure a été complètement modifiée et si on continue, on voit qu'au final on obtient une structure stable.

Structure vaisseaux (Mode Classique) :

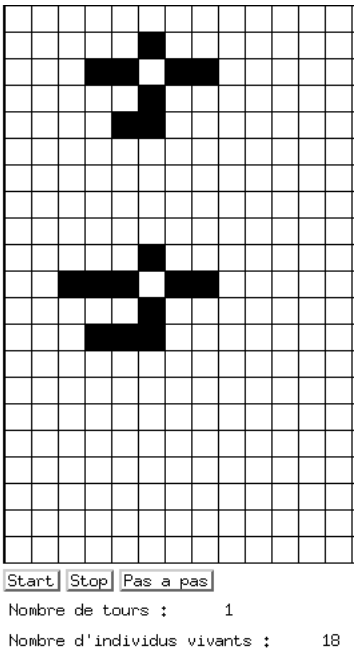
Les vaisseaux sont l'un des plus célèbres motifs du jeu de la vie car comme leur nom l'indique, ce sont comme des objets qui se déplacent et au fur et à mesure que les tours avancent, on observe une trajectoire assez linéaire.





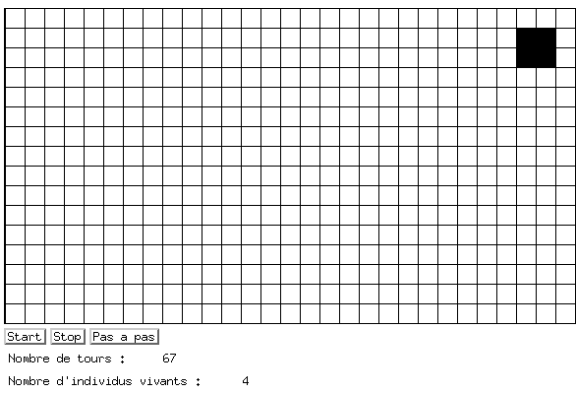
Structure vaisseaux (Mode Day and Night) :

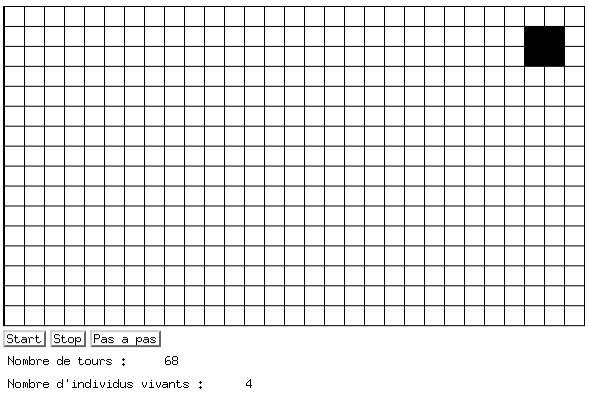
Si maintenant on exécute le motif des vaisseaux en mode Day and Night, on voit que dès le premier tour cela ne va pas créer un déplacement comme en mode classique mais on va finalement aboutir à une mini structure périodique.



A la fin des vaisseaux en mode classique, nous obtenons une structure stable, c'est-à-dire une structure qui ne change pas dans le temps :

Structure stable (Mode Classique) :





Une fois l'état stable obtenu, si l'on passe en mode Day and Night, la structure reste la même, nous avons toujours ce carré.

Explication choix graphique

Pour une bonne vision du jeu, nous avons décidé d'afficher la grille en partie haute de l'interface, et les boutons en dessous. Ceci permet une meilleure manipulation pour l'utilisateur. De plus, le mode de jeu choisi est légèrement grisé pour que l'utilisateur sache quel mode de jeu est sélectionné.

Conclusion

Ce projet nous a permis de comprendre comment réaliser une interface graphique en langage C, et de pouvoir travailler notre programmation sur un projet précis. De plus, le travail en binôme nous a permis de confronter des idées, et de voir une manière de programmation différente à celles dont on aurait pu penser.