Création d’une matrice capable de faire une simulation d'une épidémie

Noël Virginie & Ammar Fatma

Projet de Programmation en C

Master 1 Bio-Informatique et Biostatistique (BIBS)
Université Paris-Saclay
Paris, France
2020-2021

Table des matières

I/ Introduction

La Bio-Informatique est la mise en œuvre de méthodes, de concepts ou d'algorithmes éprouvés pour résoudre les problèmes posés par la biologie. Elle nous permet de simuler l'évolution d'un système où se propage un virus (coronavirus) ainsi que l'accélération de la production de la connaissance.

Notre projet consiste justement à construire une matrice d'un écosystème composé de bonhommes (lambdas et soignants) et de virus, de taille 80*30 qui va rendre cet affichage élégant. En effet, chaque case a une certaine probabilité d'accueillir l'une des 3 entités, dans le but de répondre à des besoins biologiques de la façon la plus efficace possible.

Notre programme repose sur le langage de programmation C, c'est un langage compilé qui contient différents types de fichiers distingués par leurs suffixes (.c, .h, .o).

Par conséquent, nous nous retrouvons avec un défi à relever, celui de faire un script qui est capable à la fois de gérer la complexité des informations et d'avoir une sortie de matrice efficace à la fois sur le plan fonctionnel et ergonomique.

II/ Expression du besoin

Le programme doit être capable de stocker des informations sur les bonhommes telles que leur statut malade ou non malade, soignant ou lambda, leur position et leur direction. Il faut aussi savoir pour chaque case de la matrice si elle contient un virus ou non, s'il y a un bonhomme qui l'occupe et la valeur de son gradient, signal émis par un malade dont l'intensité dépend de sa proximité par rapport à celui-ci.

Pour ce qui est de la simulation, les bonhommes doivent avoir la capacité de se déplacer et d'être infectés avec, dans ce cas, une certaine chance de tomber malade. Les malades doivent pouvoir être soignés par les soignants et fuis par les lambdas. De plus, les infectés et en particulier les malades qui meurent doivent pouvoir contaminer leur case, laissant ainsi la possibilité au virus de se propager.

Toutes ces informations doivent être affichées via le terminal selon des modalités données.

III/ Fonctionnalités

Dans cette partie, nous décrirons rapidement les différents fichiers du programme et leur contenu.

1- Utils

Le fichier utils.h contient les structures et constantes utilisées dans tout le programme (Coordonnees, Bonhomme et Case, taille de la matrice). Il s'agit aussi du fichier dans lequel on inclut les bibliothèques standards.

2- Initialisation

Le fichier initialisation ne sert qu'au début du programme. Il remplit les deux matrices de bonhommes (soignants et lambdas) avec set_bonhomme et initialise les cases de la matrice dans laquelle ils évoluent. Si une case contient un virus, son champ contamine_pour_x_tours vaut 4. Au début, il n'y a aucun malade ni infecté, donc tous les gradients valent 0.

3- Affichage

Ce fichier permet de réaliser l'affichage à chaque tour de jeu. La fonction simuler_tour appelle afficher_matrice, qui affiche les cases une par une avec afficher_case. Si la case est occupée, on affiche le bonhomme qui l'occupe. Sinon :

- Si la case est contaminée, on affiche un virus
- Si la case n'est pas contaminée et a un gradient positif (malade à proximité), on affiche celui-ci

Sinon, on n'affiche rien.
Affichage.h définit des constantes qui correspondent aux couleurs utilisées pour l'affichage.

4- Simulation

La première fonction utilisée est jouer_tour_lambdas. Elle parcourt le tableau de lambdas et pour chacun d'entre eux, appelle jouer_tour. Jouer_tour_soignant suit le même principe. A la fin de chaque tour, on diminue le temps de vie des virus présents sur la matrice.
La fonction jouer_tour appelle les fonctions restantes.

- changement_gradient_malade met à jour le gradient émis par un malade pour que les autres puissent le détecter
- changement_gradient_mort_ou_soin permet de mettre à 0 le gradient des malades qui sont en train d'être soignés (pour ne pas appeler plus de soignants) ou qui sont morts
- vers_malade est utile aux soignants pour se diriger vers un malade proche
- soigner_malade permet au soignant de guérir le malade dont il s'occupe en 2 tours
- fuir_malade permet aux lambdas de se diriger dans la direction opposée des malades proches
- deplacement appelle changer_case qui change la position du bonhomme s'il n'y a pas d'obstacle. Sinon, il a une certaine probabilité de se déplacer dans le sens inverse vers une case vide

Jouer_tour se charge de supprimer les malades de leur tableau s'ils meurent, de faire se déplacer les bonhommes non malades, de diminuer le temps d'infection des bonhommes

infectés et potentiellement d'infecter et rendre malade les bonhommes sur une case contaminée.
Simulation.h contient toutes les constantes utiles à la simulation (probabilités liées au virus et au déplacement).

IV/ Conclusion

L'élaboration d'une matrice qui propose autant d'informations n'était pas une tâche facile au début. Dans le sens où nous avions tellement d'idées et d'informations pour procéder mais pas de plan spécifique et bien structuré. Nous avons dû nous organiser, faire des fiches, filtrer nos idées et nous mettre à la place de l'utilisateur pour pouvoir mettre en place une matrice claire capable de nous faire visualiser la simulation de l'évolution d'un écosystème où se propage le virus mais dont l'utilisation reste agréable et simple à comprendre et à visualiser.

Ainsi, nous avons été capables de proposer un programme qui permet d'interagir avec l'utilisateur et de lui laisser la liberté de choisir le nombre de tours. Le côté ergonomique se trouve dans l'utilisation de différentes couleurs et différentes lettres, par exemple : M rouge pour les malades ; pour un soignant non infecté, le symbole sera S en vert quelque soit sa direction. S'il est infecté mais non malade, le S devient orange…

Notre groupe s'est organisé pour que chacune de nous fasse des fonctions séparément pour ensuite fusionner le tout dans un seul script séparé en différents fichiers. Cela nous a seulement permis de gagner en efficacité, et à obliger chacune de nous à comprendre l'intégralité du code afin qu'elle puisse faire fonctionner leur partie dans le script commun.

Bien sûr, nous avons rencontré plusieurs difficultés pendant l'élaboration de notre projet, parmi elles :

- Des problèmes d'initialisation dus à un manque de pointeurs
- Des erreurs d'affichage (par exemple aucun malade)
- Des lambdas immobiles alors que les soignants se déplacent
- Utilisation des différentes probabilités pour chaque cas

Nous avons aussi pensé à plusieurs idées qu'on aurait voulu ajouter au programme mais aussi quelques erreurs qu'on aurait voulu corriger mais malheureusement nous n'avons pas eu le temps de les ajouter et de les corriger. Comme par exemple:

- Le gradient a tendance à ne pas s'afficher/se remplir correctement
- Difficulté à définir comment un lambda doit fuir un malade ou un soignant se diriger vers lui (vers quelle case aller ?). Même problème si le bonhomme veut se déplacer sur une case occupée ou hors limites
- Réalisation de l'interface graphique avec l'utilisation de la librairie SDL et des images et laisser à l'utilisateur le choix d'afficher la simulation en console ou en mode graphique.

Nous avons dû effectuer des choix comme la suppression du tableau de virus qui ne nous semblait pas nécessaire (pour chaque case, on sait s'il y a un virus dessus et combien de temps il y reste).