Projet de programmation en C
simulation d'une épidémie !

Stéphanie Chevalier (schevalier@lri.fr)

Contexte et règles de simulation

Votre objectif va être de simuler l'évolution d'un système où se propage un virus.
Cet "écosystème" sera composé de:

bonhomme "lambda" : 

bonhomme "soignant" : 

virus : 

Ils évolueront sur une matrice, de taille 80×30 si la résolution de votre écran rend cet affichage élégant (sinon adaptez la taille pour que l'affichage rende bien - affichage en console ou en fenêtre graphique pour les plus avancés).

Lors de l'initialisation de cette matrice, chaque case a une certaine probabilité d'accueillir l'une des 3 entités présentées ci-dessus. Par exemple: 20% d'accueillir un bonhomme lambda (que je vais dorénavant simplement appelé "lambda"), 10% un soignant, 5% un virus. Vous testerez d'autres jeux de probabilité afin de réaliser des cas où la simulation termine sur une éradication du virus, et des cas où c'est la population de bonhommes qui succombe (pauvres bonhommes).

Au départ, chaque case ne contiendra donc qu'une seule entité. Ensuite lors de la simulation, il continuera de ne pouvoir y avoir qu'un seul bonhomme par case. Par contre un bonhomme pourra se déplacer sur une case contenant un virus.

Virus

Déplacement Une fois déposé sur une case, il ne peut pas en bouger "par sa propre volonté". Si un bonhomme arrive sur sa case, il l'infecte : le virus se déplacera désormais avec lui, et on dira que le bonhomme est *infecté* (qu'il soit lambda ou soignant).

Durée de vie Le virus survie 4 tours sur une case "contaminée". Passé ce délai, s'il n'a infecté personne, il disparaît.

Affichage Pour l'affichage en console, la présence d'un virus sur une case est symbolisé par V.

Bonhomme (lambda ou soignant)

Le bonhomme est capable de se déplacer au sein de la matrice 2D, dans les 8 directions *(N, NE, E, SE, S, SO, O, NO)*.

Infection S'il arrive sur une case virus, le bonhomme se retrouve infecté et transporte désormais le virus avec lui. Deux cas de figure sont alors possibles.

Cas n°1 (2 chance sur 3) : le bonhomme n'a aucun symptôme, il ne sait pas qu'il est malade. Il se comporte alors comme tout bonhomme non infecté. L'affichage le distingue simplement par une coloration en orange de la marque ou de l'image qui le représente.

Cas n°2 (1 chance sur 3) : le bonhomme est malade et les autres bonhommes le fuient (ils se dirigent dans la direction opposée). Si le malade est un soignant, il perd temporairement ses capacités de soin. Il ne se déplace plus et à chaque tour de la simulation il a 1 chance sur 10 de mourir. Il est sauvé si un soignant non malade reste pendant 2 tours à ses côtés. Qu'il soit lambda ou soignant, un bonhomme malade est représenté en rouge.

À chaque tour de la simulation, un bonhomme infecté mais non malade a une certaine probabilité (1 chance sur 4 pour commencer) de contaminer une case (sa case ou l'une de celles à son contact) en y faisant apparaître un virus. Si cette case contient un lambda, alors il devient aussitôt infecté. Si c'est un soignant, il est mieux protégé : il n'est pas infecté mais la case sera tout de même porteuse du virus - le soignant sera donc infecté s'il reste sur la case au tour suivant, ou s'il y retourne avant que le virus n'ait disparu.

Un bonhomme malade répand un "signe" de sa contamination sur les cases alentours par des valeurs d'intensité décroissante. Illustration de cette coloration à 2 niveaux d'intensité:

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0
0	0	1	2	2	2	1	0	0
0	0	1	2	**M**	2	1	0	0
0	0	1	2	2	2	1	0	0
0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Déplacement Tant qu'un bonhomme ne rencontre pas d'obstacle ou n'est pas à proximité d'un malade, il a une probabilité de 70% de se déplacer (dans ce cas, il privilégiera avec une chance de 3 sur 4 la direction de déplacement qu'il avait au tour précédent, et sinon la nouvelle direction sera choisie aléatoirement).
S'il rencontre un obstacle (bord de la matrice ou autre bonhomme), il aura autant de chance de rester sur place que de changer de direction.
Lorsqu'un bonhomme se retrouve à proximité d'un malade, sa réaction dépend de son type:
– lambda : il s'éloigne du malade, il se dirige donc dans le sens inverse du gradient.
– soignant : il s'approche du malade, il se dirige donc dans le sens du gradient. Une fois au contact du malade, le gradient disparaît afin d'éviter d'attirer inutilement d'autres soignants. Le soignant collé au malade ne bouge plus jusqu'à ce que le malade soit soigné (grâce à 2 tours de contact) ou mort (disparition du bonhomme et du gradient).

Lorsqu'un bonhomme meurt, il a "contaminé" la case sur laquelle il était. Le bonhomme disparaît donc, mais laisse la place au virus pour une durée de 4 tours. Si passé ce délai il n'a infecté personne, il disparaît également.

Affichage Pour l'affichage en console, le symbole d'un lambda sur une case va dépendre de sa direction :
• vers la gauche $<$
• vers la droite $>$
• vers le haut $\wedge$
• vers le bas $\vee$
• en diagonale NO ou SE $\backslash$
• en diagonale NE ou SO $/$
Si le bonhomme est infecté, les symboles sont identiques mais colorés en rouge au lieu de la coloration par défaut.

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.

Quelque soit le type de bonhomme, s'il est malade, il devient symbolisé par un M.

Implémentation

Déroulement d'un tour de simulation Lors d'un tour de simulation, c'est d'abord l'ensemble des bonhommes lambda qui devront être mis à jour (déplacement, infection, survie ou non des malades), avant que ce soit l'ensemble des soignants qui le soient à leur tour. Enfin, vous vérifierez les virus (décrémentation du temps de vie ou disparition).

Le tour d'un lambda se décompose ainsi:
Évaluation du déplacement.
Évaluation du risque d'être infecté selon son voisinage. S'il est infecté, il a une chance sur 3 d'être malade.

Le tour d'un soignant est très similaire, à la différence que s'il repère un malade (grâce au gradient de valeurs autour de lui) il se dirige vers lui et reste à sa proximité immédiate pendant 2 tours (il ne se déplace alors plus pendant ces 2 tours). Si le malade est toujours en vie (donc présent sur la carte) 2 tours plus tard, alors il est soigné, et le soignant reprend sa liberté de mouvement.

Le tour d'un bonhomme malade se décompose ainsi:
Évaluation de sa survie (1 chance sur 10). S'il meurt, alors il disparaît de la carte. Si un soignant est à sa proximité immédiate, il est soigné au bout de 2 tours: il redevient un bonhomme non infecté.

Un tour de simulation correspond à la mise à jour de tous les bonhommes lambda puis tous les soignants. Attention: ne parcourez pas la matrice de taille $N * M$ pour connaître à chaque tour la position des uns et des autres ! Stockez astucieusement ces informations en créant un tableau de bonhommes lambda et un tableau de soignants (de tailles N*M pour être certain de ne jamais réaliser de débordement).

Gestion des probabilités d'actions À chaque fois que vous avez à traiter une probabilité, définissez-la en constante.
Par exemple:
• `PROBA_LAMBDA` correspondra à la probabilité de faire apparaître un bonhomme lambda dans la case de la matrice lors de l'initialisation,
• `PROBA_MALADE_SI_INFECTE` correspondra à la probabilité qu'un bonhomme infecté tombe malade,
• `PROBA_DISPERSION_VIRUS` correspondra à la probabilité qu'un bonhomme infecté mais non malade contamine une case à son contact,
• `VIRULENCE` correspondra à la probabilité pour un malade de mourir (1 chance sur 10 par défaut), etc ...
Ainsi vous pourrez facilement tester différente combinaison de probabilité afin de réussir un cas où le virus tue l'ensemble de la population, et un cas où c'est le virus qui fini par être éradiqué.

Aide à la modélisation

Étape 1

Vous aurez besoin de 3 structures: Coordonnées, Bonhomme et Case. En effet, en réfléchissant à la modélisation du virus, vous vous apercevrez que cette "entité" ne nécessite pas la définition d'une structure.

Étape 2

Afin de pouvoir mettre à jour l'ensemble des lambdas avant l'ensemble des soignants, et vérifier que les virus ne sont pas en fin de vie, créez trois tableaux : un stockant les lambdas, un pour les soignants, un pour les virus. À chaque tour, vous parcourrez une seule fois chacun de ces tableaux, et non 3 fois l'écosystème entier.

Si un virus a infecté un bonhomme, ou s'il a disparu à l'issue de son espérance de vie sur une case, il devra être retiré du tableau. Lorsqu'un virus est créé et n'est pas associé à un bonhomme, il devra être ajouté au tableau. Soyez astucieux: lors de la suppression d'un virus dans le tableau, ne translatez pas l'ensemble des éléments suivants pour combler le vide : comblez-le en allant chercher le dernier. Il en va de même lorsqu'un bonhomme meurt.

Modularité Veuillez à découper judicieusement votre projet.
Vous devriez avoir des fichiers séparés pour la gestion:
- de l'initialisation
- de la simulation

- des cases
- de l'affichage en console
- de l'affichage en fenêtre graphique si vous allez jusque là

Bonus

- si vous "supprimez" les bords, au sens que le bord gauche est lié au bord droit, et celui supérieur à l'inférieur. Exemple dans ce cas de figure: un bonhomme situé contre le bord gauche qui se dirigerait sur sa gauche, rentrerait sur le bord droit de la carte.
- si vous réalisez une interface graphique. Pour cela, utilisez la librairie SDL, et les images que je vous mets à disposition pour symboliser les entités. Laissez à l'utilisateur le choix d'afficher la simulation en console ou en mode graphique.

Évaluation

Vous m'enverrez par mail une archive contenant votre projet (les fichiers .c, .h et le makefile permettant de compiler) ainsi qu'un rapport pour expliciter jusqu'où vous avez traité le sujet, les potentiels choix réalisés, les difficultés rencontrées...
Ce projet est un travail en binôme. La date limite du rendu du projet est fixée au samedi 16 janvier 2020 23h59.
Votre projet doit pouvoir être compilé et exécuté sous Unix.

Pour ceux qui connaissent ou souhaitent se former aux gestionnaires de version permettant un travail collaboratif, vous pouvez utiliser GitHub ou GitLab et m'adresser pour rendu le lien vers celui-ci (et non l'archive).

En plus du fonctionnement du programme, votre code sera évalué selon les critères suivants:

- Le code doit être lisible (indentation, espacement, longueur des lignes).
- Le nom des variables, fonctions et structures doit être pertinent et respecter les règles de formatage.
- Utilisation pertinente de constantes.
- Chaque fonction doit se charger d'une seule tâche bien définie et spécifiée sans ambiguïté dans le commentaire qui accompagne la fonction.
- Les passages importants du code doivent être commentés de sorte à ce qu'on puisse comprendre vos raisonnements.
- Le découpage modulaire doit être pertinent : il regroupe les fonctions par thématique.
- Votre Makefile devra invoquer l'option -Wall pour le compilateur gcc. Tous les avertissements seront pris en compte dans l'évaluation.