Laurent Maëva
Menlet Enzo

L2 Math-info
Programmation C
S3 2021-2022

Rapport de projet
CHIPS VS VIRUS

Commentaires sur le code :

Nous n'avons pas créé de structures complémentaires, ni modifié les structures imposées dans le sujet. En effet, pour faire agir les chips et virus (attaquer, ralentir, soigner, accélérer) qui ne possèdent pas d'attributs concernant leurs actions dans leur structure, nous avons préféré utiliser des fonctions, une pour les chips (**capacite_chips**), une pour les virus (**capacite_virus**), renvoyant des entiers différents en fonction du virus ou chips détecté, qui déterminera quelle(s) action(s) devront être réalisée(s) dans deux autres fonctions répertoriant les différentes actions possibles (**action_chips** et **action_virus**).

Pour exécuter le programme il y a deux options : ./projet -a ou /projet -g, pour un mode ASCII ou graphique.

Difficultés, problèmes rencontrés et solutions trouvées :

La première difficulté que nous avons rencontré était due à un oubli. Nous n'arrivions pas à manipuler la liste des virus et des chips pour y accéder et en créer de nouveaux. Nous avions cherché plusieurs solutions mais elle était en réalité très simple : nous avions oublié dans le main, au moment de la déclaration des structures, d'initialiser leur next à NULL :

```
Virus virus;
virus.next = NULL;
Chips chips;
chips.next = NULL;
```

Cela nous empêchait d'accéder aux structures et nous a bloqué pendant un petit moment sur le programme.

La deuxième difficulté à laquelle nous avons fait face était le chaînage par lignes. Nous avons commencé en essayant de parcourir la liste avec 2 pointeurs et un compteur, et en essayant de trouver le premier virus de chaque ligne en parcourant le fichier. Cette fonction était beaucoup trop longue et pas très optimale en terme de mémoire, et ne marchait pas dans certains cas, comme par exemple si il n'y avait qu'un virus sur une ligne. On a donc opté pour une version totalement différente, en utilisant un tableau (voir doc).

Ensuite, nous avons eu des problèmes avec l'affichage des tours du jeu.
Dans un premier temps, nous avions essayé de faire un affichage par lignes. C'est à dire qu'on cherchait le premier virus de chaque ligne une par une dans le fichier, et on affichait tous les virus qui lui étaient chaînés sur la ligne. Pour cela on initialisait une variable **int prev_line égale** à 0 et on cherchait le premier virus du fichier qui avait une ligne supérieur à **prev_line + 1**, puis **prev_line** prenait la valeur de sa ligne, puis on refaisait une recherche, etc pour chaque ligne.

Premièrement, cette méthode était catastrophique en terme de mémoire, car la liste était parcourue plusieurs fois. Ensuite la fonction était très longue composée de plein de boucles et conditions qui la rendaient incompréhensible. Enfin, elle ne marchait tout simplement pas pour les chips. Si on ne les rentrait pas dans un ordre croissant de lignes et colonnes, certains ne s'affichaient pas.
On a donc abandonné cette idée et sommes partis sur une toute autre façon de procéder, en utilisant un tableau, comme recommandé dans le sujet (voir doc).

Nous avons aussi eu des problèmes concernant la détection des chips pour les actions des virus. Un virus est censé s'arrêter et attaquer un chips si il se retrouve face à lui, mais il passaient au-dessus. Les conditions que nous essayions ne marchaient pas, par exemple :
if(capacite_virus(tv->type) == 1 && tv->position == tc->position + 1 &&tv->line == tc->line)
(tv étant un pointeur parcourant la liste des virus, et tc parcourant la liste des chips)
Cela a donc donné naissance à la fonction **recherche**, sans laquelle nous n'arrivions pas à détecter la présence de chips.

Sont venus ensuite les problèmes de fuite de mémoires. En lançant notre programme avec Valgrind, de nouvelles erreurs sont apparues : "**Conditional jump or move depends on uninitialised value(s)**", "**Use of uninitialised value of size 8**", "**Invalid read of size 8**", etc...
Cela était dû à de mauvaises utilisations de **->next** pour initialiser des pointeurs sur les listes (qui pouvaient être vides ou composées d'un seul élément) ou pour les parcourir.
Cela a été contraignant car il a fallu, déjà, comprendre ce que signifiaient toutes ces erreurs, puis ensuite repasser sur un grand nombre de fonctions de notre programmes pour les remanier de sorte à ce que les fuites de mémoires disparaissent. Cela posait surtout problème dans nos fonctions de suppression des chips et virus morts. Notre manière de parcourir les listes n'était pas correcte et engendrait plusieurs erreurs, et nous avions de problèmes avec la suppression des premiers éléments de chaque liste.
C'est grâce à cela aussi que nous avons remarqué que nous avions oublié de libérer la mémoire occupée par les listes en fin de partie, d'où la création de **supprime_listes**.

Nous n'avons pas eu de grandes difficultés concernant l'implémentation du mode graphique, le plus embêtant ayant été le placement des chips, mais qui n'a pas été si compliqué que cela au final.

Répartition du travail :

Il n'y a pas de commentaires spéciaux à faire concernant la répartition du travail. Nous avons fait la quasi intégralité du projet ensembles : si l'un codait, soit l'autre l'aidait pour réfléchir à deux, soit il s'occupait de coder autre chose à côté, ou rédigeait la doc ou le rapport. Les seuls moment ou l'on a travaillé seuls était si l'autre n'était pas disponible pendant une grande partie de la journée, mais cela a été rare.
Maëva a été plus impliquée dans l'initialisation du jeu, dans les actions et déplacements des virus, dans le placement des chips en graphique, et Enzo dans les différents affichages, le chainage par lignes, et les actions des chips, et les déplacements des virus en graphiques. Le reste à été plus ou moins réfléchi et codé à deux.