Laurent Maëva
Menlet Enzo
L2 Math-info
Programmation C
S3 2021-2022

Documentation Technique

Vérifications des arguments pour l'éxécution :

- int verif_arguments :

Cette fonction prend le nombre d'arguments entrés **int argc** et un tableau de caractères à deux dimensions contenant les caractères des arguments entrés **char *argv[]**.
Elle compte le nombre d'arguments pour vérifier qu'il est bien égal à 2 (le nom du programme et le type de jeu (ASCII ou graphique)). Sinon le programme est interrompu.
if(argc>2){printf("too many arguments were given\n"); exit(1);}
if(argc<2){printf("too less arguments were given\n"); exit(1);}
Elle vérifie ensuite que le deuxième argument entré est bien **-a** ou **-g**, et renvoie **1** si c'est **-a**, et **2** si c'est **-g**. Sinon le programme est interrompu.
if (argv[1][0] == '-'){ if (argv[1][1] == 'a') return 1; else if (argv[1][1] == 'g') return 2;}

Recherche de maximums :

- int max_line :

Cette fonction prend en paramètre l'adresse de la liste chaînée des virus **Virus* v**.
Elle initialise un entier **max** à 0 .
Elle parcours ensuite la liste des virus avec une boucle :
for (; v != NULL; v = v→next)
et attribue la valeur de la ligne du virus actuel à **max** si elle est supérieure au **max** actuel avec :
if(v->line > max) max = v→line;
Elle retournera enfin max après avoir parcouru toute la liste.
max_line a donc pour effet de trouver le nombre de lignes sur lesquelles apparaîtront des virus.

- int max_turn :

Même fonctionnement que **max_line** mais avec **v->turn** à la place de **v->line**.
max_turn a donc pour effet de trouver à quel tour apparaîtra le dernier virus.

Définition des spécialités des types :

- void spe_virus :

Cette fonction prend en paramètre l'adresse d'un virus **Virus* v**

et son type **char type_virus**.
A l'aide de différents **if**, comme par exemple **if (type_virus == 'E')**,
elle attribue vie et vitesse au virus pointé en fonction de son type :
tmp->life = 4; tmp->speed = 2; par exemple.
Elle servira dans une future fonction de création d'un virus (**alloue_cellule_virus**).

- int capacite_virus :

Cette fonction prend en paramètre le type d'un virus **char type_virus**.
En fonction de différents **if**, comme par exemple **else if (type_virus == 'S')**
elle renvoie un entier différent.
Ce renvoi servira dans une future fonction (**action_virus**).

- void spe_chips :

Même fonctionnement que **spe_virus** avec les chips, mais attribue un prix à la place d'une vitesse.
Elle servira dans une future fonction de création d'un chips (**alloue_cellule_chips**).

- int capacite_chips :

Même fonctionnement que **capacite_virus** avec les chips.
Ce renvoi servira dans une future fonction (**action_chips**).

Insertion dans les liste chaînées :

- void insertion_virus :

Cette fonction prend en paramètre l'adresse de la liste des virus **Virus* v**
et l'adresse d'un virus **Virus* creation_virus**.
Elle parcourt la liste des virus si elle n'est pas vide (**if(v != NULL)**)
avec une boucle **for(; v ->next != NULL; v = v->next)**
Et chaîne le virus à la fin avec : **v->next = creation_virus**.
Sinon, si elle est vide, elle fait pointer la liste sur le virus : **chips = creation_chips**.
Elle servira dans une fonction d'initialisation de la liste des virus (**init_liste_virus**).

- void insertion_chips :

Même fonctionnement que **insertion_virus** avec les chips.
Elle servira dans une future fonction de création de chips (**Pose_tourelles**).

Allocation de mémoire :

- Virus * alloue_cellule_virus :

Cette fonction prend en paramètre un numéro de tour **int num_tour**, un numéro de ligne **int num_ligne** et un type de virus **char type_virus**.
Elle initialise un pointeur vers un nouveau virus en allouant la place nécessaire à sa création avec :
Virus * tmp = (Virus *)malloc(sizeof(Virus))

et si l'allocation s'est bien faite **if(tmp != NULL)**, elle initialise tous ses attributs : **tmp->type = type_virus; tmp->next = NULL;** par exemple.
C'est ici que **spe_virus** intervient.
Enfin, elle renvoie **tmp**.
Elle servira dans une fonction d'initialisation de la liste des virus (**init_liste_virus**).

- Virus * alloue_cellule_chips :

Même fonctionnement que **alloue_cellule_virus** avec les chips.
Elle servira dans une future fonction de création de chips (**Pose_tourelle**).

Recherche de tourelle :

- int recherche :

Cette fonction prend en paramètre l'adresse de la liste des chips **Chips* c**, un numéro de ligne **int nb_ligne** et numéro de colonne **int nb_colonne**.
Elle parcours ensuite la liste des chips avec une boucle :
for (; c != NULL; c = c→next)
Si un chips est trouvé aux positions entrées en paramètre :
if(c->line == nb_ligne && c->position == nb_colonne)
Elle renvoie 1; 0 sinon.
Elle servira dans diverses fonctions.

Création des listes chaînées :

- void init_liste_virus :

Cette fonction prend en paramètre un pointeur vers un fichier **FILE* fichier**, un pointeur vers la structure jeu **Game* jeu**, et un pointeur vers une liste de virus **Virus* virus**.
Elle initialise un numéro de tour **num_tour**, un numéro de ligne **num_ligne**, et un type de virus **type_virus**.
Tant que le fichier comporte une ligne, on la lit, on met ses informations dans les variables précédemment initialisées, et on alloue de la mémoire pour crée un virus en fonction de ces dernières variables. C'est ici que **alloue_cellule_virus** intervient.
Si la mémoire n'a pas pu s'allouer, on arrête le programme, sinon on l'insert dans la liste des virus **virus**. C'est ici que **insertion_virus** intervient.
Une fois que la fonction a fini de lire le fichier, on fait pointer l'adresse de la liste des virus du jeu à celle de la liste tout juste créée, en ignorant le premier élément possédant des attributs aléatoires donnés au moment de la création de la structure **virus** : **jeu->virus = virus→next.**
La structure **jeu** contient maintenant la liste des virus du fichier.

- void init_position_virus :

Cette fonction prend en paramètre l'adresse de la liste des virus **Virus* v** du jeu et le nombre de colonnes du jeu **int nb_colonnes**.
Elle parcours la liste des virus avec **for(; v != NULL; v = v→next)**

et initialise la position de chaque virus à la valeur maximale de position :
v->position = nb_colonnes.

- void Pose_tourelles :

Cette fonction prend en paramètre un pointeur vers la structure jeu **Game* jeu**, un pointeur vers une liste de virus **Chips* chips**, le nombre de lignes du jeu **int nb_lignes** et le nombres de colonnes du jeu **int nb_colonnes**.
Elle initialise un type de chips **char type_virus**, un numéro de lignes à 0 **int num_l** et un numéro de colonne à 0 **int num_c**.
A l'aide de différents **scanf**, elle demandera de rentrer un type de virus, et sa position (ligne colonne), en vérifiant à chaque fois que les entrées sont bien correctes (type existant, sur le plateau, pas de superposition).
C'est ici que **alloue_cellule_chips** intervient.
La fonction vérifie ensuite qu'on a bien assez d'argent.
Si tout est bon, on l'insert dans la liste des chips **chips**. C'est ici que **insertion_chips** intervient.
Une fois que le joueur décide d'arrêter de poser des tourelles, on fait pointer l'adresse de la liste des chips du jeu à celle de la liste tout juste créée, en ignorant le premier élément possédant des attributs aléatoires donnés au moment de la création de la structure **chips** : **jeu->chips = chips→next.**
La structure **jeu** contient maintenant la liste des chips du fichier.

- void selec_type_chips :

Cette fonction est propre à la version graphique du jeu.
Elle prend en argument une coordonnée **int x** et une coordonnée **int y**.
En fonction des coordonnées reçues, c'est à dire si on a bien cliqué sur un chips dans la boutique du jeu, elle renverra un caractère différent,correspondant donc à un type de chips. Par exemple :**if(x>=30 && x<=80 && y>=35 && y<=75) return 'A';**

- void Pose_tourelles_graph :

Cette fonction est propre à la version graphique du jeu.
Elle initialise une hauteur de case **int hauteur_case** et une largeur de case **int largeur_case** qui correspondent respectivement à un proportion de la largeur et hauteur de la fenêtre, divisé par le nombre de lignes et de colonnes du jeu :
int hauteur_case = 600/nb_lignes;
int largeur_case = 1050/nb_colonnes;
Elle initialise également deux entiers, qui transforment des coordonnées en numéros de ligne et colonne :
num_ligne = round((num_ligne-110)/(float)hauteur_case);
num_colonne = round((num_colonne-110)/(float)largeur_case);
Ensuite, elle alloue de la place pour créer un chips avec comme caractéristiques le numéro de ligne et de colonne tout juste calculés, et le type de chips renvoyé par **selec_type_chips**.
Tout comme **Pose_tourelles**, elle vérifie ensuite que la mémoire s'est bien alloué, que le joueur a assez d'argent, et si une tourelle n'est pas déjà présente à l'endroit pointé. Si tout est bon, le chips est maintenant inséré dans la liste de chips du jeu.

Chaînage des virus sur une ligne :

- void chainage_ligne :

Cette fonction prend en paramètre l'adresse de la liste des virus **Virus* v** du jeu et le nombre de lignes du jeu int **nb_lignes**.
Elle initialise un tableau **tab** d'adresses de virus de taille **nb_lignes** à **0**.
Ensuite, elle parcours la liste des virus avec **for(; v != NULL; v = v->next)** et :

- Si le virus courant est le premier de sa ligne, alors son **prev_line** pointe la case du tableau correspondant à son numéro de ligne, donc **NULL** (vu que le tableau est initialisé avec des **0**, qui agissent comme des **NULL**), puis cette case pointe l'adresse du virus (**tab[(v->line)-1] = v**).

- Sinon, si le virus courant n'est pas le premier de sa ligne, donc si la case du tableau correspondant à sa ligne pointe déjà un virus (**if (tab[(v->line)-1] != NULL)**), alors il devient le **next_ligne** du virus pointé par cette case du tableau (**tab[(v->line)-1]->next_line = v**), puis ce dernier virus devient le **prev_line** du virus courant (**v->prev_line = tab[(v→line)-1])** et le virus courant prends sa place dans le tableau.

Cette fonction a donc pour effet de chaîner les virus par lignes.

Affichage de la vague du plateau (graphique) :

- void affiche plateau graph :

Cette fonction est propre à la version graphique du jeu.
Elle prend en paramètre le nombre de lignes du jeu **int nb_lignes** et de colonnes du jeu **int nb_colonnes**.
Elle initialise, comme dans **Pose_tourelles_graph**, une hauteur **int hauteur_case** et une largeur **int largeur_case** de case.
Elle initialise ensuite deux entiers **l** et **c** qui serviront grâce à une double boucle, de dessiner des cercles correspondant à toutes les positions possibles du plateau.
Elle dessine aussi un fond, des rectangles correspondants aux différents boutons du jeu, et les noms des boutons grâce aux fonctions **MLV_draw_filled_rectangle**, **MLV_draw_rectangle** et **MLV_draw_text**.

- void affiche_boutique :

Cette fonction est propre à la version graphique du jeu.
Elle prend en paramètre l'argent du jouer **int money**.
Elle affiche dans un rectangle en haut à droite de la fenêtre toutes les tours disponibles pour défendre. Pour cela il faut charger l'image, puis l'afficher :

```
mousse = MLV_load_image("images/mousse.png");
MLV_draw_image(mousse, 30, 35);
```

par exemple.
Elle affiche aussi leur coût, et la monnaie du joueur en temps réel.

Affichage de la vague :

- void affiche_vague :

Cette fonction prend en paramètre l'adresse de la liste des virus **Virus* virus** du jeu, le nombre de lignes int **nb_lignes** et le nombre de colonnes du jeu **nb_colonnes**.
Elle initialise deux entier i et j, un pointeur sur la liste des virus **Virus * v**, et un tableau de char à deux dimensions **char plateau[nb_lignes][nb_colonnes*2]**.
Dans un premier temps elle remplit le tableau d'espaces en le parcourant avec **i** et **j**, sauf aux deux premières cases de chaque ligne réservées pour le numéro de la ligne, et à la fin de chaque sous tableau avec des '**\0**' pour marquer la fin de la ligne.
Elle remplit ensuite le tableau avec les types de virus en parcourant la liste, en fonction de leur position : **plateau[v->line - 1][v->turn * 2 + 1] = v→type.**
Enfin elle affiche le tableau avec des **printf** pour chaque sous tableau :
for (i = 0; i < nb_lignes; i++) printf("%s\n", plateau[i]).

Cette fonction a donc pour effet d'afficher la vague du niveau.

- void affiche_vague_graphique :

Cette fonction est propre à la version graphique du jeu.
Elle prends en paramètre l'adresse de la liste des virus **Virus* virus** du jeu, le nombre de lignes int **nb_lignes** et le nombre de colonnes du jeu **nb_colonnes**.
Elle initialise, comme dans **Pose_tourelles_graph**, une hauteur **int hauteur_case** et une largeur **int largeur_case** de case.
Elle charge les images de tous les virus avec des **MLV_load_image**, parcourt la liste des virus, puis les affiche à leur bonne position sur la fenêtre, en fonction de leur tour d'apparition. Par exemple :

```
if(v->type == 'E')
  MLV_draw_image(sidney, v->turn*largeur_case+85 , v->line*hauteur_case+85);
```

Cette fonction donne un aperçu de la vague du niveau.

Action des virus et des chips :

- void action_virus :

Cette fonction prend en paramètre la liste des virus du jeu **Virus* v**, la liste des chips du jeu **Chips* c**, et le numéro de tour du jeu **int tour**.
Elle initialise deux pointeurs sur la liste des virus, **tv** et **tb** (pour le virus Booster), et un pointeur sur la liste des chips **tc**.
Pour chaque virus, elle comparera sa position avec celle de chaque chips (ici la fonction **recherche** interviendra, et le fera agir sur un chips si il se situe devant.

C'est ici que la fonction **action_virus** intervient.

- void action_chips :

Même fonctionnement que **action_virus** mais avec les chips, qui (pour la plupart) agissent à distance.

- void supprime_virus :

Cette fonction prend en paramètre l'adresse de la liste des virus **Virus* v**.
Si la liste de virus est vide, elle renvoie **NULL**.
Sinon, elle initialise un pointeur **prev** sur cette liste, et tant que la vie du virus pointée est nulle (ou négative) et que le virus chaîné à ce virus est différent de **NULL**,
while(prev->life <= 0 && prev->next != NULL)
on supprime l'élément courant, et on pointe vers le suivant, tout en actualisant le chaînage **next_line** et **prev_line** :
v = v->next;
if(prev->next_line != NULL) prev->next_line->prev_line = NULL;
free(prev); prev = v;
Ensuite, on initialise un nouveau pointeur **tmp** qui pointe le virus qui suit **prev**.
Si ce virus n'existe pas, on regarde la vie de **prev**, et on le supprime si elle est nulle.
Sinon, on parcours la liste tant que **tmp** est différent de **NULL**, et on répète la même procédure qu'avec **prev**, sans vérifier que le virus suivant est différent de **NULL**.
Si la vie du virus pointé n'est pas nulle, on décale **prev** et **tmp** de 1.
On a besoin de deux pointeurs sur la liste, car on ne peut pas revenir en arrière pour chaîner le virus précédent au suivant quand on en supprime un. C'est le rôle de **prev**.
Enfin, cette fonction renverra donc un pointeur vers la nouvelle liste **v**, vidée de tous les virus morts.

- void supprime_chips :

Même fonctionnement que **supprime_virus** mais avec les chips, et sans chaînage **next_line** et **prev_line**.
Elle renverra donc un pointeur vers la nouvelle liste **c**, vidée de tous les chips morts.

Affichage du tour :

- void affichage_tour :

Même fonctionnement que **affiche_vague**, mais affiche également les chips en plus des virus, sur 2 cases au lieu d'une (pour vie et type), et dépend de la position des virus (qui change dans la partie) et des chips, et non pas de leur tour d'apparition.
Elle affiche aussi, pour indicatif, le numéro des colonnes.
Elle affichera l'état courant du plateau et s'actualisera à chaque tour du jeu.

- void affiche_chips_graphique :

Cette fonction est propre à la version graphique du jeu.

Elle fonctionne comme **affiche_vague_graphique**, mais pour les chips, et sera rappelée à chaque nouveau chips posé, et à chaque tour du jeu.

- void affiche_tour_graphique :

Cette fonction est propre à la version graphique du jeu.
Elle fonctionne comme **affiche_vague_graphique**, mais dépend de la position des virus, et non pas de leur tour d'apparition, affiche aussi les chips avec **affiche_chips_graphique**, et s'actualise à chaque tour.

Fonction de délai :

- void delay :

Cette fonction prend en paramètre un nombre de millisecondes **int nb_ms**.
Elle initialise un entier **nb** égal à 1000 fois **nb_ms**, démarre un chronomètre avec **time_t start = clock()** et boucle tant que ce chronomètre n'a pas atteint le temps de millisecondes entré en paramètre : **while(clock() < start+nb)**.
Elle servira à faire avancer le jeu à une vitesse raisonnable.

Conditions de victoires pour virus :

- void condition_victoire :

Cette fonction prend en paramètre l'adresse de la liste des virus **Virus* v**.
Elle parcours cette liste avec **for(; v!= NULL; v = v→next)** et regarde si un virus à atteint le bout du plateau (**if (v->position == 1)**).
Si oui, elle renvoie 1, sinon 0.

Suppression des listes :

- void supprime_listes :

Cette fonction prend en paramètre la l'adresse de la liste des virus **Virus* v** et l'adresse de la liste de chips **Chips* c**.
Elle fonctionne de la même manière pour les deux listes :
Si la liste n'est pas vide, elle initialise un pointeur dessus, et tant que ce pointeur n'est pas nul **for(; tmpc != NULL;)**, on pointe le next de la liste (**c = c->next),** on libère l'élément courant (**free(tmpc)**), et on pointe vers le suivant (**tmpc = c**).
Cette fonction permet de libérer la place allouée pour les listes en fin de partie pour éviter les fuites de mémoire.