CHIPS VS VIRUS

(Version Mario)

Cousson Sophie | Naudin Maxime

Objectif :

Réaliser un jeu de tower défense en langage C .

Notre projet :

Notre projet contient un jeu Chips VS Virus avec une version graphique et un mode console. Nous avons au total 5 niveaux de jeu, progressif en termes de difficulté et qui intègrent à chaque fois un nouveau virus.

Les fonctions et éléments qui nous ont aidés :

Pour réaliser le mode console et nos fonctions de jeu tels que la modification, suppression et initialisation de nos Chips et Virus, nous avons utilisé des fonctions d'affichages qui ne sont plus dans le code final :

```c
void affiche_listeVirus(Game * game){
    Virus * lst = game->virus;
    int cpt = 0;
    printf("----------------------------\n");
    while(lst){
        cpt++;
        printf("type d'ennemi %d : \n", lst->type);
        printf("vie de l'ennemi %d\n", lst->life);
        printf("ligne de l'ennemi %d\n", lst->line);
        printf("position de l'ennemi %d\n", lst->position);
        printf("vitesse de l'ennemi %d\n", lst->speed);
        printf("Tour de l'ennemi %d\n",lst->turn);
        printf("Can move : %d\n", lst->can_move);
        lst = lst->next;
    }
    printf("cpt %d\n", cpt);
}
```

Cela nous a permis de nous rendre compte des erreurs et du bon fonctionnement de notre jeu. Dans la même optique nous avons fait en sorte que certaines de nos fonctions fasse des return qui ne sont plus utilisé maintenant mais qui permettaient de contrôler les erreurs.

Modifications des structures imposées :

Nous avons ajouté quelques éléments aux structures imposées. Par exemple pour que chaque virus et chaque chip soit plus complet nous avons ajouté un champ int dgt qui indique les dégâts que font ces derniers par tour, mais aussi `int can_move;` dans la structure du virus. Cette variable peut prendre la valeur 1 ou 0 : 1 si le virus peut avancer, sinon 0. Cela nous permet de gérer les mouvements des virus. On ne regarde que les can_move des virus qui sont en tête de

leur ligne. Les virus qui se trouvent derrière celui-ci seront bloqués s'ils sont amenés à le dépasser (Cf move_virus).

Nous avons également modifié game en y ajoutant le dernier virus et la dernière chip de la liste chainé pour simplifier certaines boucles.

Nos difficultés et nos solutions :

Nous avons rencontré dans un premier temps une difficulté dans la suppression des virus, nous avions un segmentation fault qui nous a bloqué un certain temps.

Nous avons résolu le problème en recommençant complétement la suppression sans vouloir reprendre les fonctions précédentes et nous avons résolu le problème.

Nous avons rencontré une difficulté majeure lors de la conception de notre programme. Après avoir quasiment terminé la partie graphique, qui était basée principalement sur des constantes, nous nous sommes rendu compte que notre jeu graphique était uniquement jouable si l'utilisateur possède un écran ayant pour résolution 1920x1020 pixels. Nous avions un fond de jeu avec des lignes tracées grâce à Paint et des constantes avec des positions initiales :

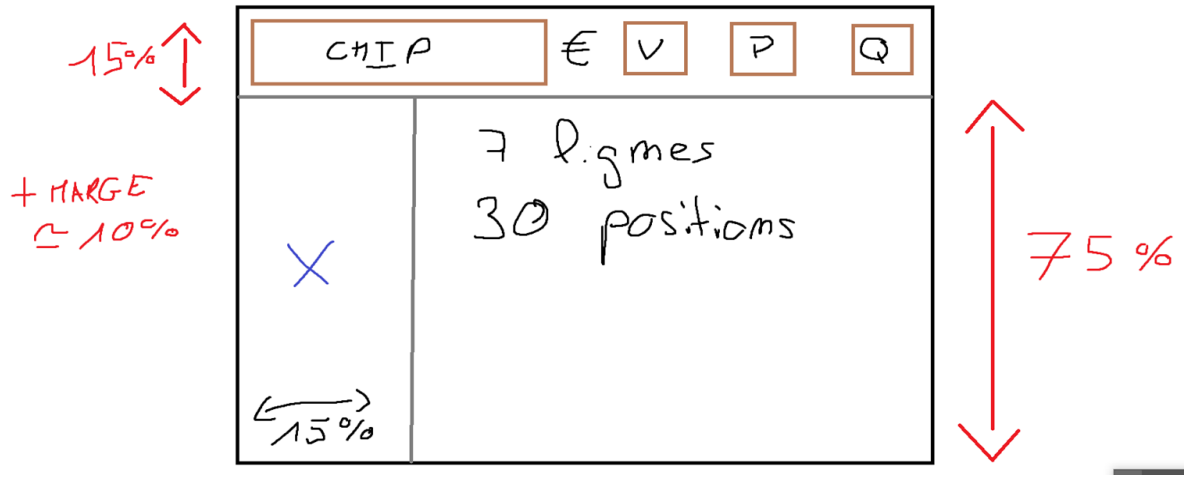
```
16    #define POS_INIT_X 395
17    #define POS_INIT_Y 250
```

```
22    #define ESPACE_POS 45
23    #define ESPACE_LIGNE 105
```

Si nous voulions jouer sur un écran avec une résolution différente, nous ne pouvions pas… C'est pourquoi nous avons repris toute la version graphique pour l'adapter sur un écran un peu plus petit, ce qui a mené au menu de la version graphique.

Nous avons donc choisi d'utiliser un tableau de type int à une dimension pour stocker la largeur et la hauteur de l'écran.

Voici la représentation en termes de proportions du jeu :



Nous avons alors choisi d'ajouter dans notre menu, un bouton option, permettant à l'utilisateur de choisir entre 3 résolutions d'écran. Par défaut, le jeu se lance en plein écran.

Concernant le thème de notre jeu, nous trouvions le monde de Mario amusant. Les différentes fleurs jouent le rôle des Chips et les ennemis de Mario sont les Virus. Nous avons ajouté de la musique dans notre jeu, elles sont toutes issues de super Mario Bros. Il y en a pour chaque phase différente du jeu.

Organisation de notre travail :

Nous avons travaillé via un répertoire github que nous avons mis à jour après chacune de nos sessions de travail, souvent l'un travaillait dans la matinée et l'autre pouvait reprendre le code pour continuer dans la soirée et ainsi de suite. Cela nous a permis de voir rapidement quelles modifications ont été faites sur le programme, d'avoir une trace de chacune de nos versions et de pouvoir partager notre travail plus facilement. Pour chaque session de travail, on faisait :

- un récapitulatif de ce que nous avions codé.

- une liste d'objectifs courte et précise de ce que nous devions faire lors de la prochaine session.

Nous sommes tous les deux content de notre rendu final et du travail fourni ensemble.