RAPPORT DE PROJET DU JEU CHIPS VS VIRUS

INTRODUCTION :

-Objectifs :

Le jeu du Chips Vs Virus est un jeu de tower défenses dont les combats seront discrétisés en tour par tour. L'objectif consistait donc à développer un Chips Virus en C avec la librairie MLV. Pour cela nous avions diviser le projet en 4 parties. La première tâche consistait à gérer la mémoire des structures du jeu. La deuxième tâche consistait à gérer les placements et les actions des Chips et des Virus dans les listes Chainés. La troisième tâche consistait à élaborer la fonction principale du jeu. Enfin, la dernière tâche consistait à élaborer la partie graphique.

-Manuel Utilisateurs :

Lors du lancement du lancement du jeu. On passe en paramètre notre choix d'affichage en ligne de Commande. Si, on souhaite jouer sur la partie ascii, alors on tape la commande « -a » lors de l'exécution du jeu. Si, on souhaite jouer sur la partie graphique, alors on tape la commande « -g » lors de l'exécution. Ensuite, lors du lancement du jeu, il est demandé de choisir le niveau de difficulté du jeu. Il existe trois niveau possible, « level1 », « level2 » et « level3 ». Sur la partie ascii, avant que la partie de visualiser commence il faut poser le nombre de tourelles que vous souhaitez tant jouer tant qu'il y a assez d'argent. En renseignant le type et les coordonnées de la tourelle. Pour la partie graphique, il faut sélectionner le type

de tourelles en cliquant sur la tourelle voulue, puis il faut cliquer sur la position ou l'on souhaite insérer la tourelle.

-Difficultés Rencontrées :

Durant notre projet, nous avions rencontrés plusieurs problèmes sur la partie ascii avec l'élaboration des actions des chips et des virus. En effet, notre fonction intitulé « moveVirus » qui gère les déplacements renvoie des positions et des distances non désirées. Le comportement de la fonction est imprévisible. En effet, lors des calculs de distance entre les chips et les virus. Le résultat de ses calculs sont anormaux. Ainsi, les résultats renvoyer par la fonction se répercute sur tous le programme. Ce qui fait buger le programme. Pour cela, nous avions réécrit la fonction entièrement en traitant chaque cas.

```c
void moveVirus(Virus* virus, Chips** OriginChips){
    Chips* current = *(OriginChips);
    int chip_in_front = 0;
    int gap_to_nearest;
    while (current != NULL){
        if (current->line == virus->line) {
            if (current->position <= virus->position){
                if (!chip_in_front){
                    chip_in_front = 1;
                    gap_to_nearest = current->position - virus->position;
                }
                else{
                    if (current->position - virus->position < gap_to_nearest) {
                        gap_to_nearest = current->position - virus->position;
                    }
                }
            }
        }
        current = current->next;
    }
    if (virus->prev_line == NULL) {
        if (chip_in_front && (virus->speed >= gap_to_nearest)){
            virus->position = virus->position - (gap_to_nearest - 1);
        }
        else virus->position = virus->position - virus->speed;
    else{
        int gap_to_virus = (virus->prev_line)->position - virus->position;
        if (chip_in_front && (gap_to_virus > gap_to_nearest)){
            if (chip_in_front && (virus->speed >= gap_to_nearest)){
                virus->position = virus->position - (gap_to_nearest - 1);
            }
            else virus->position = virus->position - virus->speed;
            return;
        }
        if (virus->speed >= gap_to_virus){
            virus->position = virus->position - (gap_to_virus - 1);
        }
        else virus->position = virus->position - virus->speed;
        return;
    }

}
```

D'autre part, nous avions aussi eu des problèmes sur les conditions de fin de partie. En effet, en parcourant le chainage de la structure Virus, on arrive dans une boucle infinie. Puis, nous avions eu certains problèmes sur les pointeurs des structures Game, Virus et Chips. En effet, nous n'avions pas réussi à faire marcher la chips de type « P » qui fait référence à la mine, car la mine avait pour mission d'exploser au contact des virus et mourir mais nous obtenons un mur incassable.

De plus, nous n'avions pas réussi à faire apparaître les points de vie pour les chips et les virus à l'affichage du plateau.

Enfin, nous avions plusieurs problèmes avec la partie graphique. En effet, lors de détection des clics sur la fenêtre ne renvoyer pas les bonnes coordonnées. Pour cela, nous avions donc mis à l'échelle les coordonnées des clics en parcourant dans une double boucle « for » imbriqué l'axes des ordonnées et l'axe des abscisses selon la position des points sur les lignes d'attaques. De plus, lors de l'importation des images sur le jeu. Nous n'avions pas trouvé de solutions pour retirer les images une à une. Ainsi, lors de l'avancement du jeu. Les virus aux anciennes positions étaient toujours dessinés. Pour cela, nous avions réfléchie à comment effacer ces images sans pour autant fermés la fenêtre. Pour remédier au problème, nous avions donc décider de mettre dans une fonction le traçage des lignes d'attaques, des positions et du positionnement de l'ordinateur centrale. Puis nous avions appelés la fonction après la création du plateau de jeu et juste avant le positionnement des virus et des chips sur le plateau de jeu à partir du deuxième tour.

- Structuration du Projet

Lors de l'élaboration de notre jeu, nous avions décidé de crée une main qui ne gère que les options passer en paramètres en ligne de commande. Puis, nous avions créé deux fonctions principales pour la partie ascii et la partie graphique. Qui sont appelés en fonction des arguments passer en paramètres lors de l'exécution. La fonction graphique ressemble énormément à la fonctions ascii. En effet, nous avions copier la fonction ascii pour nous avions rajouter la partie graphique par-dessus.

Ensuite, nous avions les fonctions transverse pour l'allocation de l'espace mémoire, l'insertion des chips et des virus dans les listes chainés, l'avancement des virus selon leurs vitesses dans les listes chainées.

- Avancement du Projet

Notre jeu « chips_vs_virus » répond à toutes les tâches du niveau 1 et la plupart des tâches du niveau 2. En effet, le jeu est fonctionnel en mode console. La lecture des fichiers standardisés et le chargement des niveaux se produisent sans faire d'erreur, les visualisations des vagues en ascii art s'effectue sans problème. Le menu et l'interaction clavier pour laisser le joueur construire ses défenses fonctions sans commettre d'erreurs, il y plus de 3 types de tourelles et de monstres et la visualisation de la partie est raisonnable.

Pour le niveau 2 sur l'interface graphique en libMLV, l'intégralité des points du niveau sont respectés. On peut voir l'avancement des vagues en graphique, l'interaction à la souris

s'effectue convenablement pour construire les défenses et l'argent de départ diminue lors de la pose d'une nouvelle chips. Il y au moins 4 types de tourelles et de monstres. Et pour finir, nous avions fait 3 niveaux de difficultés, un niveau facile, un niveau intermédiaire et un niveau difficile.

- Répartition du Travail

Dans ce projet nous nous sommes plutôt réparti le travail avec 50% du travail fait part Hervé et 50% du travail fait par Gaël. Sur la première partie dédier à l'allocation de l'espace mémoire, nous avions travaillé ensemble. Après les phases une et deux, ou nous avions travaillé ensemble, Hervé ses occuper de la partie ascii pendant que Gaël ses occuper de la partie graphique.

- Conclusion

Notre programme est fonctionnel et permet de réaliser les tâches demandées. Cependant, nous savions que notre programme peut être amélioré. Par exemple, avec les pistes d'améliorations que nous n'avions pas pu effectuer par manque de temps. Durant ce projet, nous avions appris à gérer correctement les insertions dans les listes chainées, ainsi que sur la récupération des informations présent dans les fichiers texte. Nous avions aussi appris à bien gérer nos pointeurs.