Projet finale – Prog C Avancée : Rapport

Projet en langage C édité par : Massiouane Maibeche et Prashath Sivayanama – Groupe 3

Tables des matières :

I – Présentation et objectifs du projet

Le projet 'Stealth' est jeu programmé en langage C, vous prenez le contrôle d'un personnage qui a pour rôle de récupérer tous les reliques du jeu et de revenir à sa position initiale sans se faire repérer par les gardiens.
Le joueur principal possède du mana (tant qu'il en ramasse), et sa vitesse est graduelle tant qu'il se dirige vers la même direction.

Nous utilisons la bibliothèque graphique 'livMLV' pour représenter graphiquement les informations du jeu.

L'objectif de ce projet est donc de réalisé une interface graphique proposant un contrôle par des touches directionnelle en temps réelle, avec une utilisation de pouvoir spéciaux tant que le mana du joueur est suffisante.

II – Moyen mis en œuvre pour réponse aux exigences du proejt

Nous avons découpé notre projet en 10 modules en tout sans compter le main.

Le premier module est le module de déplacement, celui-ci permet au joueur d'avoir une des directions cardinale lorsqu'il se déplace dans une direction.
Ce module inclus également les coordonnées des objets du jeu (nous entendons reliques et tuiles) ainsi que des gardiens et du personnage.
Le gardien et le joueur possède des coordonnées de type 'double' pour permettre de gérer aisément leur vitesse pendant leur déplacement.
Tandis que les reliques et les tuiles ont une position fixe d'où l'utilisation du type 'int' afin de simplifier leur placement.

Les modules Gardien et Joueur sont assez similaire, ils ont en commun dans leur structure une direction (enum), une vitesse (double), une position (double).
Ils possèdent des fonctions tel qu'un leur permettant de changer leur vitesse graduellement,

Une autre de les configurer dans le but d'une initialisation.

Pour en venir au détail du type gardien, celui-ci a en plus dans sa structure un champs de compteur de panique (int), un décrivant son niveau d'alerte selon la disparition d'une relique (enum), un entier représentant leur rayon de detection (int), ainsi qu'un chrono (double) qui permet de savoir si celui est bien resté 30 secondes en mode panique.
Les gardiens possède une fonction lui permettant de changer de direction parmi 4 choix de direction possible dans prendre la même direction dans laquelle il allait.

Pour le joueur, celui-ci a en plus dans sa structure un champ pour de compteur de panique (int), un compteur pour sa mana ramassé (int), un mode permettant de savoir si le joueur se trouve dans un mode normal, ou si il utilise un mode acceleration ou d'invisibilité (enum).

Le module Relique possède une structure Relique possède un champ position (struct position) et un état décrivant si celui-ci a était récupérer ou non.

Le module Moteur de jeu contient une structure tuile, très similaire à la structure relique, cependant elle possède en plus un champ qui représente sa quantité de mana.
Ce module possède également la structure mère Plateau regroupant les éléments pour faire fonctionner une partie.
Elle se compose de deux compteurs de mana servant pour savoir ce qu'a consommé le joueur durant une partie (int), un tableau 2D contenant le plateau de jeu (int), un joueur (Joueur), 5 gardiens (Gardien), 3 reliques (Relique) et 20 tuiles (Tuile).

Le moteur de jeu se compose d'une fonction mère qui initialise tous ses éléments, génération de murs, placement des objets et des personnages du jeu etc.
Il se compose d'une autre fonction lui permettant de changer l'état des objets selon si ils ont était récupérer ou non.

Le module Collision n'a quant à lui aucune structure, mais il possède des fonctions tel que la détection proche d'un mur (collision), pour éviter que le joueur ou que les gardiens ne sortent du plateau.
Une fonction de détection du gardien, pour savoir si le joueur se trouve dans son rayon, ou si une relique a disparu.

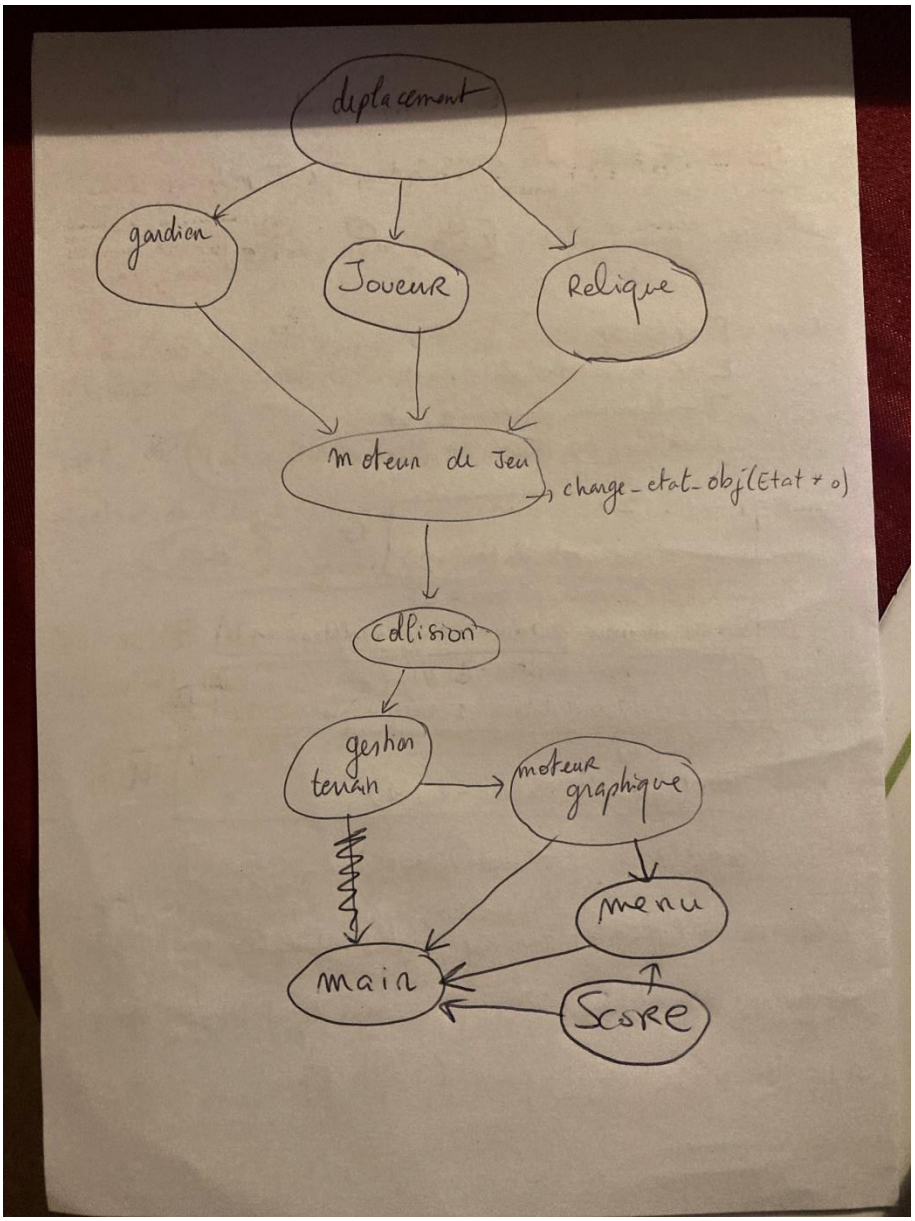
Le module Gestion terrain gère le terrain de jeu comme son nom l'indique, on y trouve les fonctions d'interaction joueur/relique, joueur/tuile, gardien/gardien, et d'autres fonctions tout aussi importantes comme le déplacement du joueur et du gardien, le retour du mana dépensé par le joueur vers les tuiles, la sortie du mode panique des gardiens et une fonction détectant si la fin de jeu a eu lieu.

Le module Moteur graphique, regroupe tous les évènements des précédents modules pour les représenter graphiquement avec la bibliothèque graphique livMLV.
Elle se compose d'une unique fonction (dessine_plateau_graphique) regroupant tous les éléments de la partie et les affichant.

Le module Menu regroupe la gestion d'un menu en début et en fin de partie, elle utilise des éléments de la bibliothèque graphique, pour gérer les évènements.

Le module Score, regroupe la gestion d'un fichier binaire pouvant contenir le score des meilleurs joueurs, celui-ci se compose d'une structure de données concernant le joueur, son temps de jeu et son mana dépensé.

Voici un graphe d'inclusion de notre projet :



Dans notre répartition des tâches, pendant que l'un procéder à la génération des murs dans le plateau pour le moteur de jeu, l'autre gerer les structure que le plateau contiendrai c-a-d joueur, gardien et relique.
Nous nous sommes ensuite repartie les modules collision, menu et gestion terrain et moteur graphique.

```
==10862==
==10862== HEAP SUMMARY:
==10862==     in use at exit: 132,271 bytes in 1,245 blocks
==10862==   total heap usage: 21,159 allocs, 19,914 frees, 7,963,320 bytes
 allocated
==10862==
==10862== LEAK SUMMARY:
==10862==    definitely lost: 8 bytes in 2 blocks
==10862==    indirectly lost: 0 bytes in 0 blocks
==10862==      possibly lost: 1,352 bytes in 18 blocks
==10862==    still reachable: 130,911 bytes in 1,225 blocks
==10862==                       of which reachable via heuristic:
==10862==                         newarray           : 1,536 bytes in 16 b
locks
==10862==         suppressed: 0 bytes in 0 blocks
==10862== Rerun with --leak-check=full to see details of leaked memory
==10862==
==10862== For lists of detected and suppressed errors, rerun with: -s
==10862== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Quelques bug sont à déclarer, très aléatoirement au lancement d'une partie celle-ci peut se voir terminer immédiatement, cela provient d'après valgrind d'un module SDL.

Nous avons également des erreurs causées par ce même module SDL qui survient de manière aléatoire, à certain moment aucune erreur peuvent apparaitre et à d'autre elle se manifeste.

Nous avons à certain temps changé notre structure de données pour l'adapter comme il le fallait, comme pour la position des personnages et le compteur de mana.

III – Conclusion

Comme dit plus haut en ce qui concerne la répartition des tâches pour ce projet nous avons fait un travail quasi-équivalent du début à la fin ce celui-ci.

Ce projet nous a permis de mieux maitriser certaine subtilité du langage C, et de mieux comprendre la bibliothèque MLV.

Les difficultés surmonté durant ce projet pour nous ont était les générations des murs, et la gestion d'un score par fichier binaire, bien que cela dissuade de la triche.

Nous sommes tous de même assez content du travail que nous avons fourni malgré le peu d'amélioration que nous y avons inclus.

IV – Annexe 1 : Instruction de compilation

Le projet se compile avec la commande suivante :

make

Une fois l'exécutable générer, celui-ci se trouve dans le dossier bin entrer la commande suivante :

bin/exe

Le joueur est représenter par la couleur rouge, en invisibilité il est de couleur blanche et dans son état d'accélération il est de couleur rose.
Le gardien est bleu, son rayon de même. En mode panique, il est vert.
Les tuiles sont de couleur orange, elles sont ensuite de couleur violette une fois récupérer par le joueur.
Les reliques sont de couleur cyan clair, et disparaissent une fois récupérer.