HAMIDOU Elarif
FREZOULS Guillaume
MOUSSET Anthony

Rapport de projet IN505

"Réalisation d'un jeu de bataille en forêt convivial via interface graphique"

INTRODUCTION :

Dans le cadre du module de Langages Avancées inclus dans la troisième année de la licence d'informatique de l'UVSQ, il nous a été proposé de tester nos connaissances à travers un projet délimité par un cahier des charges fixé par le sujet imposé par les responsables du module (Béatrice Finance, Leila Kloul, …). Notre équipe se compose de HAMIDOU Elarif, et de FREZOULS Guillaume, et nous avons été rejoints plus tard par MOUSSET Anthony.

Notre vision du projet et de l'énoncé est que deux personnages sont lâchés aléatoirement dans une forêt, ils peuvent se déplacer librement dans la forêt et pilotent des hélicoptères télécommandés pouvant, ou non, larguer des bombes qui détruisent les différents obstacles de la forêt. Leur objectif est de tuer l'adversaire en le touchant avec une bombe.

Le projet a été réalisé entièrement en C++, et nous avons utilisé l'outil d'interface graphique SFML pour créer notre interface graphique conviviale.

Nous allons développer en détail chaque étape du projet selon l'énoncé, en précisant les éventuelles difficultés rencontrées durant le projet et l'approche algorithmique des différentes fonctionnalités du jeu.

Règles du Jeu:

- Les flèches servent à déplacer le personnage, et la souris permet de larguer les bombes.
- Il y a deux modes d'édition : Éditer une nouvelle forêt et Éditer une forêt existante. Ces modes sont accessibles selon le clic du bouton de l'interface principale (droit pour éditer une forêt existante, gauche pour éditer une nouvelle forêt)
- Il faut entrer le nom des fichiers dans le terminal.
- La vainqueur est également affiché dans le terminal.

HAMIDOU Elarif
FREZOULS Guillaume
MOUSSET Anthony

ETAPE 1 : Réalisation du monde virtuel

La première étape était de créer le catalogue d'objets. Nous avons créé une classe abstraite Obstacle, et quatre classes (Arbre, Buisson, Rocher, Lac) qui héritent de la classe Obstacle. *Il était initialement prévu de donner une taille différente à chaque obstacle, mais nous avons finalement décidé de donner une taille de 1 pixel à chaque objet (un pixel étant une case du tableau).*

Chaque obstacle est défini par une hauteur aléatoire, des Points de Vie (PV) et des coordonnées selon la case du tableau.

La forêt est représentée par un tableau en deux dimensions définies par une taille fixe. La taille du pixel étant également fixée, on obtient la taille du tableau en nombre de pixels. Il contient également une liste d'obstacles, qui va contenir les objets créées dans le niveau.

Les Points de Vie sont déterminées selon la nature de l'objet :

- Les Buissons sont deux fois moins résistants que les Arbres;
- Les Rochers sont deux fois plus résistants que les Arbres;
- Les Lacs sont implémentées comme indestructibles (théoriquement).

Les obstacles perdent de la vie selon la portée de la bombe larguée (voir Étape 2). Et les obstacles sont supprimées de la liste et du tableau une fois qu'ils n'ont plus de vie.

Les obstacles sont représentés dans une liste, et sont ajoutés dans l'éditeur graphique une fois que l'utilisateur décide de quitter l'éditeur, ce qui permet, si l'utilisateur décide de modifier ultérieurement un niveau créé, de le modifier sans altérer la liste, car elle est vidée à chaque chargement, et reconstruite à partir des éléments que l'utilisateur place sur le terrain.

Le terrain se gère à partir d'un tableau d'entiers, qui contient différentes valeurs, allant de 1 à 4 :

- 1 représente un Arbre
- 2 représente un Buisson
- 3 représente un Rocher
- 4 représente un Lac

C'est à partir des données du tableau que l'on construit la liste et que l'on gère l'affichage des niveaux. On va charger le tableau, dans l'éditeur, et permettre la modification du tableau.

On gère également un vecteur de RectangleShape (rectangle) permettant l'affichage des différents obstacles, chaque rectangle étant indépendant. La hauteur était initialement codée et implémentée mais dans l'avancée du programme, nous avons décidé de la rendre fixe.

La sauvegarde est gérée et le chargement sont gérées par la librairie Boost. Il consiste en une sérialisation de la classe Forêt. Dans l'éditeur, on ne peut sauvegarder un terrain vide, donc si la liste est vide, aucun fichier n'est sauvegardé.

ETAPE 2 : Les joueurs

Une fois que la forêt a été implémentée, que l'édition de nouveaux niveaux et de niveaux déjà créées était affichable dans une interface graphique et sans interface graphique, il a fallu implémenter les joueurs.

Les joueurs sont dans une classe Personnage. et chaque Personnage créé est caractérisé par un nombre de PV, et une position dans le tableau. Le personnage peut se déplacer librement dans la carte et peut tirer n'importe où dans la carte. (*L'orientation des personnages était initialement incluse mais nous n'en avions plus d'utilité une fois qu'il a été décidé que les joueurs contrôlent des hélicoptères*)

Le déplacement du joueur dépend de la flèche sur laquelle le joueur appuie. Les joueurs ne peuvent pas traverser les obstacles, on vérifie donc si le déplacement désiré est faisable (donc s'il y a un obstacle). On change l'affichage du personnage selon la flèche.

L'arme d'attaque des joueurs est un hélicoptère qui largue des missiles, il est situé en hauteur, et permet de lâcher différentes bombes sur le terrain, les bombes ont une portée allant de 1 à 5.

Le choix de tirer ou non revient à cliquer sur une zone ou non. Donc si le personnage clique, on effectue le tir et on efface les objets de la portée du tir, définie par l'utilisateur.

Le tir est obligatoire pour changer de personnage.

HAMIDOU Elarif
FREZOULS Guillaume
MOUSSET Anthony

Etape 3 : Une vraie Partie

Maintenant que le tir est gérée, que le déplacement est gérée et que l'IA, ainsi que les terrains sont faits, nous allons effectuer une vraie partie.

Le déroulement d'une partie est simple:

- Deux personnages sont lâchés aléatoirement sur le terrain, sur une case vide.
- Le joueur 1 commence à jouer.
- Il se déplace autant qu'il veut et son tour s'arrête dès qu'il tire
- Quand le tour d'un joueur est terminé, la main passe, il redevient invisible, et le joueur 2 devient visible, et peut effectuer les mêmes actions que le joueur 1.

Les joueurs ne sont pas directement sur le terrain mais sur une couche au-dessus. Ils subissent les obstacles car ils se déplacent dessus, mais chaque personnage est indépendant, ce qui fait que le joueur 1 peut se trouver sur la même case que le joueur 2.

Si un joueur tue l'autre joueur, la partie se termine. Le gagnant est affiché sur le terminal, et la fenêtre de jeu se ferme.

Il est important de noter que même s'il n'y a aucun obstacle sur le terrain, les joueurs ne peuvent pas se voir, on considère que les joueurs peuvent se cacher sous le sol une fois le tour terminé

Le choix du monde virtuel dans lequel la partie va se dérouler se fait dans le terminal, l'utilisateur rentre le nom d'un niveau et le niveau se charge, juste avant les personnages.

HAMIDOU Elarif
FREZOULS Guillaume
MOUSSET Anthony

Conclusion

Ce projet de bataille en forêt nous a permis, à chacun d'entre nous, d'enrichir et de renforcer nos connaissances du langage C++, ainsi que de la gestion de la bibliothèque graphique SFML. Le travail en équipe est quelque chose dans laquelle il faut être rigoureux, et continuellement observer les modifications faites par les différentes personnes, sous peine de ne plus pouvoir apporter de modifications.

Bien évidemment, comme chaque projet, nous avons rencontrés beaucoup de difficultés :

- La gestion du tir a été modifié : pendant un court instant, il était prévu de faire le tir selon une formule de distance, en considérant que l'arme était au sol, en utilisant la vitesse (qui aurait été notre force) un angle qui aurait pu être modifié par l'utilisateur, ainsi qu'une portée de tir. La formule était $d=v^2*\sin(2*angle)/g$. Nous devions utiliser cette formule, couplée à la position de la souris, aurait dû nous permettre de calculer la distance, mais gérer la perspective fut un peu compliquée, et la gestion de la physique du déplacement de l'objet nous a fait changer d'avis, et l'idée de l'hélicoptère nous paraissait plus simple et amusante.
- Suite à notre vision du projet, beaucoup de données ne nous ont pas été utiles et donc n'ont pas été implémentées, ou bien l'ont été puis ont été supprimés (la hauteur des objets par exemple)
- Le déplacement des objets dans l'interface de création, qui fut abandonnée par faute de temps, mais qui fut remplacée par une création d'objets plus simple à ajouter et à supprimer.
- Les coordonnées relatives par rapport au centre de la forêt, qui fut remplacé par des coordonnées par case, fixes, et ajoutées à chaque objet après sa création.
- L'IA, la gestion des PV des obstacles et la limitation des déplacements des personnages n'a pas pu être gérée par faute de temps.
- L'affichage n'est pas très bien gérée, car l'entrée des niveaux se fait par le terminal

Au final, toutes les difficultés rencontrées auraient pu être des fonctionnalités supplémentaires implémentées dans le programme principal. Mais ce projet fut une véritable expérience pour chacun.

Sources utilisées :

- Le site developpez.com et son forum

- Le site officiel de la bibliothèque graphique SFML
- Youtube et les vidéos d'apprentissage SFML
- StackOverflow
- Le cours pour la partie C++, complété avec des recherches diverses sur internet