BARBARA NAJIB
AGOCS ARON

Groupe 08

COMPTE RENDU DE LA SAÉ 1.02

Presentation de l’application et des fonctionnalités incorporées

a)Presentation

Dans cette saé nous devions réalisé un jeu de chevalier et monstre sur terminal. Le système de duel entre un monstre et un joueur fonctionne avec un système de pierre, feuille, ciseau , considéré dans l’application comme un enum.

b)Fonctionnalités implémentées

Les fonctionnalités implémentées sont :

Un système de génération aléatoire de parties à partir d’un fichier déjà constitué de monstre, où chaque monstre est donc tiré au sort parmi les monstres du fichier.
Un système de création de partie manuelle qui demandent au joueur combien de monstres ils souhaitent dans le premier et second groupe de monstre.
Un système de “niveau” pour les joueurs a été aussi intégré, plus le meilleur score d’un joueur dépasse un certains paliers plus il gagne de la vie et des points de dégats.

c)Structure de donnée et interactions entre elles

Concernant les joueurs :

- les joueurs qui contiennent un nombre de dégat, un nombre de point de vie, et un nombre d’armes (qui est tout le temps 3 pour Pierre,Feuille, ciseau).
- un tableau de pointeurs sur joueur qui contient l’ensemble des joueurs qui ont déjà joué, car cela permet d’allouer un espace mémoire suffisant, tout

en pouvant déplacer les élèments avec une meilleur facilité qu'un tableau dynamique de joueur, étant donné les algorithmes de tri que nous devons utilisé dans le projet, et les ajouts de nouveaux joueurs
- un tableau dynamique de joueur est aussi utilisé temporairement pour contenir une copie du tableau des joueurs pour ensuite lui appliquer un trié échange par meilleurs, ce qui permet d'éviter

Concernant les monstres :
- les monstres qui contiennent un nombre de dégat, un nombre de point de vie, et un nombre d'armes (le niveau n'est pas stocké car il peut être calculé à partir de ces données). De plus quand un joueur choisit le niveau d'un monstre, un convertisseur est utilisé pour le transformer en points de vie, points de dégats et nombre d'armes.
- une pile de montstre utilisé pour le premier groupe, car il faut d'abord battre le premier monstre en haut de la pile pour pouvoir accéder au monstre suivant (donc dépiler la pile)
- une file circulaire utilisé pour le second groupe, car les monstres sont affronter de manière cycliques
- un tableau dynamique de monstre est utilisé pour générer une partie aléatoirement

d)Fichier de sauvegarde

Il y a en tout 3 types de fichiers dans l'application :
- le fichier joueur qui contient l'ensemble des joueurs et leurs scores associés
- le fichier monstre.txt qui commence par le nombre de monstre contenu dans le fichier, puis les monstres en question
- et les autres fichiers qui contiennent des parties à charger et à jouer. Ils commence par le nombre de monstre à chargé du premier groupe, puis le premier groupe de monstre, puis le nombre de monstre du second groupe, puis le second groupe de monstre.

e)Comparaison algorithmique

Concernant la recherche, j’ai utilisé une fonction de recherche dichotomique, étant donné que le tableau de joueurs est trié par ordre alphabétique à l’insertion des joueurs.
Il y aussi une autre fonction de recherche simple qui parcourt l’entièreté du tableau.

Concernant les tris il existe deux tris dans l’application, un tri échange par meilleur score qui utilise une copie du tableau, pour ainsi eviter de trier 2 fois le tableau (un tri par meilleur score, puis par odre alpabétique, pour revenir à l’ordre alphabétique).
Et un autre tri à l’insertion à l’aide d’une fonction de recherche dichotomique.