RAPPORT DU PROJET

Le projet **CY-Fighters** est un jeu de combat tour par tour codé en C, dans lequel deux équipes de combattants s'affrontent à l'aide de techniques spéciales. L'objectif était de créer un jeu interactif avec un affichage lisible, des mécaniques de combat et de l'intelligence artificielle. Bien que peu important, le choix de l'univers duquel on aurait tiré nos combattants a été très difficile, on s'est donc décidé à piocher de plusieurs univers, chaque combattant inspiré d'une licence différente (Shadow pour Sonic, Shade pour Supercell, Joker pour DC, Goku pour Dragon Ball, Thanos pour Marvel et Asta pour Black Clover).

Il fallait d'abord créer un fichier textuel où stocker nos combattants et leurs caractéristiques. Pour faciliter la future lecture de ce fichier, Wael a décidé de décrire chaque combattant sur une ligne en gardant la même mise en forme. Wael s'est chargé de cette partie. Il a donc créé les fichiers combattant.c/.h et technique.c/.h en parallèle. Ces fichiers contiennent les structures décrivant un combattant et par la suite les structures décrivant une technique spéciale, également des procédures affichant et initialisant une technique spéciale, affichant tout simplement un combattant, gérant l'esquive, gérant les dégâts, etc. Il faut savoir que la paire de fichier utilitaire.c/.h a été créée au fur et à mesure du projet, contenant des fonctions utiles et réutilisables pour générer des nombres aléatoires, utiliser des entiers et seulement des entiers pour faire valider ses choix à un utilisateur, etc.

Hazem, lui, s'est chargé principalement des fichiers equipe.c/.h et affichage. Il fallait gérer des équipes, les initialiser, pouvoir les afficher, gérer des fonctions et procédures qui puissent aussi déterminer si une équipe est KO ou pas. Le plus difficile dans tout ça, vu que le projet s'appuyait essentiellement sur l'ergonomie de l'affichage, était d'avoir une interface claire avec laquelle le joueur pouvait comprendre tout ce qui se passait à chaque tour. On a donc décidé, et Hazem s'en est chargé, de rédiger des procédures qui affichaient la barre de PV, une "fiche" pour chaque combattant, les deux équipes côte à côte, etc.

C'est bien sympa tout ça, mais le fichier textuel du début, où on a stocké nos combattants, il fallait bien le lire ! Anes s'est donc chargé du fichier qui permettait de lire combattant.txt. Il s'est aussi chargé du déroulement du combat où il fallait combiner toutes les bases créées auparavant pour le bon déroulement et affichage des tours de la partie. Au même moment, on ajoutait un fichier IA pour ajouter l'ordinateur et son équipe ; l'IA dépendait des fonctions de combat, elle devait utiliser les mêmes mécaniques que le joueur, donc il fallait attendre que le combat fonctionne. On a rédigé le main.c tous ensemble à la toute fin afin d'appeler toute fonction/procédure indispensable et ajouter tout affichage annexe.

Comme tout amateur en langage C, on a bien sûr rencontré des problèmes : des erreurs de compilation liées à des bibliothèques manquantes, des point-virgule manquants, des accolades manquantes, etc. On avait cherché comment lire un fichier txt ligne par ligne, au début on avait trouvé strtok() mais elle était très compliquée, on s'est donc plutôt penché sur des boucles avec des index i et j qui lisent jusqu'à un certain caractère. Au tout début, l'affichage était trop condensé et on ne comprenait rien (alors ici c'est plus une limite qu'un problème, on a essayé des codes ANSI pour les couleurs mais sous Windows, c'étaient des caractères bizarres qui s'affichaient plutôt que des couleurs) on est donc resté sur du 100% ASCII avec un affichage des équipes horizontale. Parfois, le jeu crashait sans aucune raison, on a donc décidé de rendre notre code plus robuste en vérifiant chaque pointeur en paramètre, chaque malloc, l'ouverture du fichier avec des exit, etc. Les techniques spéciales étaient très compliquées à coder vu que chacune d'elle avait un effet complètement différent. On a bien sûr eu du mal à rédiger sans faute les fonctions et procédures très complexes telles que utiliserTechnique() et jouerTourJoueur() dans combat.c, chargerCombattant et lireChamp pour le fichier textuel, etc.

D'ailleurs, nous avons rencontré un problème que nous n'avons pas réussi à résoudre, malgré tous nos essais : la technique spéciale de goku permettait d'attaquer deux fois d'affilée et donc de sauter le tour de l'IA ; en fait dès qu'on essayait de modifier une partie du code dans ce but, le jeu créait d'autres multiples bugs, donc malheureusement on n'a pas pu régler ce problème.

Enfin, on voulait ajouter un bonus pour choisir la difficulté de l'IA mais on n'a pas eu le temps et on a vu que c'était un peu compliqué donc on a créé une IA cohérente qui varie les cibles en attaquant avec le premier perso vivant et dès qu'une technique spéciale est prête, a 50% de chances de l'utiliser. On a donc simplement ajouté une option pour rejouer.

N.B. : Durant la réalisation de notre projet, même si l'organisation avait l'air bien séparée, on s'est chacun aidé pour nos parties de code afin d'obtenir le meilleur code final possible tout en le maîtrisant. On s'est également permis de modifier quelques concepts dans le jeu : par exemple, on a l'agilité qui est un entier qui va ensuite nous permettre de générer une variable aléatoire entre 1 et cet entier, et si la variable aléatoire est 1, le perso esquive (donc plus l'agilité est grande, plus le perso a moins de chances d'esquiver), on a la vitesse qu'on a enlevé en supposant la vitesse de tous les persos à 1, et comme on considère les techniques spéciales très fortes par rapport à une attaque basique, on a décidé que si le joueur sélectionne une technique pas encore prête, et bien son tour compte et il ne peut pas attaquer et c'est directement à l'IA.

Avec CYFighters, un trio sans peur, un code en sueur et des combats de valeur…

Réalisé par MOKRANE Anes, FERJANI Hazem et HAJJI Wael

Preing1 SUPMECA 2024/2025