Trinome_MEF 1-D
Tomas Carre
Farès Kassar
Numance Saint-Clair

Rapport projet informatique

Pour ce projet d'informatique, nous nous sommes d'abord réunis pour choisir ensemble un sujet parmi ceux proposés. Après discussion, nous avons opté pour "CY fighters", car c'était selon nous le sujet offrant le plus de liberté et nous permettant de faire pleinement usage de notre imagination.

Nous avons ensuite réfléchi au type de jeu de combat à réaliser et avons immédiatement pensé à Pokémon et ses jeux iconiques sur DS, auxquels presque tous les enfants de notre génération et de la précédente ont joué.

De là, nous avons pu commencer à nous organiser et structurer notre projet. Nous l'avons conçu partie par partie avant de tout assembler et optimiser. Nous avons débuté par l'interface du menu, où les joueurs pourraient choisir leur mode de jeu. Le début a été un peu difficile, car nous manquions de connaissances pour certaines fonctionnalités (comme la fonction getch() qui lit un caractère sans l'afficher ni nécessiter de validation, utile pour naviguer dans un menu, ou des fonctions d'affichage comme system("clear") pour vider le terminal ou sleep() pour créer des pauses). Après avoir acquis ces connaissances et maîtrisé les fonctions essentielles au projet, nous avons pu avancer et finaliser le menu, basé sur une navigation avec les flèches haut/bas et une sélection par la touche Entrée. Nous y avons ajouté des éléments esthétiques en ASCII générés via un site.

Ensuite, nous sommes passés à la sélection des Pokémon, ce qui nous a obligés à créer des structures et leurs attributs (nom, type, représentation graphique, attaques, etc.). Une fois les structures définies, nous avons réutilisé le concept des menus précédents pour créer celui-ci, toujours avec des affichages ASCII.

Puis est venue la phase de l'interface de combat. Nous avons commencé par le 1vs1, puis le 2vs2, et enfin le 3vs3. Nous nous sommes rendu compte que le 3vs3 couvrait tous les cas de figure en utilisant des structures "nulles" (avec 0 PV) pour les emplacements vides. Pour concevoir cette interface, nous avons d'abord affiché les Pokémon et leurs PV sous forme de barres, puis implémenté le choix des attaques (reposant sur le même principe que les menus), ainsi que les fonctions de changement de Pokémon. La phase de combat a été particulièrement complexe et longue en raison du grand nombre de conditions à gérer : comparaison des vitesses

pour déterminer l'ordre d'attaque, changements de Pokémon, différents types d'attaques (soin, dégâts, modifications de stats), vérification des PV pour détecter les KO, etc.

Une fois cette partie terminée, nous avons effectué un premier test (basique), qui semblait fonctionner sur l'ordinateur de Numance (un Mac avec un terminal similaire à ceux de l'école sous Linux). Nous avons alors constaté que notre code était très répétitif et qu'il nous manquait des fichiers .h. Nous avons donc réorganisé le projet en créant ces fichiers, optimisé les fonctions, et réduit la duplication (par exemple, le menu de démarrage est passé d'une douzaine de fichiers à un seul).

Avec toutes les fonctions nécessaires en place, nous avons créé un main qui appelle chronologiquement :

Le menu (dont le choix détermine la suite),

La sélection des Pokémon,

La phase de combat (à laquelle nous avons ajouté un mode "contre l'IA", avec des actions aléatoires mais des résistances/faiblesses selon la difficulté choisie les Pokémons sont sélectionnés en fonction de ceux du joueur).

Notre professeur de TD durant la deuxième séance nous a ensuite fait remarquer qu'il manquait un élément obligatoire du sujet ainsi qu’un deuxième élément primordial pour l’exécution du programme :

Les attaques spéciales : Nous les avons ajoutées dans les structures des Pokémon et dans la phase de combat (affichage des statuts, barre de chargement, etc.).

Le Makefile : Nous avons adapté un exemple fourni par le professeur pour compiler notre projet, l’ordinateur de Numance lui permettait d’exécuter sans makefile grâce à Visual studio qui compilait pour chaque programme donc ne nous nous étions pas encore souciés de cela auparavant.

Lors du dernier TD, nous avons rencontré un problème majeur : le code ne compilait pas sur les ordinateurs de l'école (alors qu'il fonctionnait sur le Mac de Numance). Après investigation, nous avons découvert qu'une fonction appelée trop fréquemment allouait une quantité excessive de mémoire. Ce bug nous a retardés, mais nous l'avons résolu juste à temps pour faire tous les tests nécessaire le vendredi et corriger les dernières erreurs avant la deadline.