Mini-projet: Bataille Navale

Clément Roger, Clément Leboulenger

Janvier 2020

Table des matières

1 Sujet

Le sujet était présenté ainsi :

> Réalisation d'un jeu de bataille navale pour un joueur. Dans ce projet, l'utilisateur joue contre la machine et doit torpiller sa flotte en un minimum de coup(`http://fr.wikipedia.org/wiki/Bataille_navale(jeu)`). Le programme placera aléatoirement sa flotte. On implémentera deux niveaux de jeu : *débutant* qui donnera une indication quand le missile tombe dans l'eau sur une case adjacente d'un navire ; *expert* sans indication.

2 Analyse du travail à faire

Notre programme suit les règles originelles (trouvées sur Wikipédia) à savoir de placer cinq bateaux aléatoirement sur une grille 10X10 (un de cinq cases, un de quatre cases, deux de trois cases et un de deux cases). L'utilisateur saisit ensuite une position horizontale et une position verticale (x et y) et le programme indique s'il a touché un bateau ou non et s'il a coulé un bateau. Lorsqu'il n'y a plus de bateau, le jeu s'arrête. Pour réaliser ce travail de la manière la plus efficace possible nous avons tout d'abord décidé de créer un dépôt github afin de pouvoir travailler chacun de notre côté tout en mettant notre travail en commun. Nous nous sommes ensuite attaqués à la réalisation du projet en C afin d'avoir une base de travail concrète. Cela nous a notamment permis d'avoir une liste non exhaustive de fonctions à réaliser. Nous avons ensuite pu ensuite nous atteler à la réalisation du projet en MIPS. Nous avons ensuite dressé la liste des fonctions à réaliser (plutôt similaire à celle réalisée précédemment en C) et nous nous la sommes partagée. Chaque fonction importante était testée (dans un nouveau fichier test) une fois finie pour vérifier son bon fonctionnement et faciliter le débogage. Une fois le projet initial terminé nous y avons rajouté des fonctionnalités pour augmenter l'ergonomie du jeu et le plaisir de l'utilisateur :

- Un score pour que l'utilisateur sache le nombre de coups dont il a eu besoin pour l'emporter
- Un enregistrement des trois meilleurs scores (un tableau de scores)
- Une sauvegarde de la partie, l'utilisateur pouvant reprendre ultérieurement une partie non terminée

3 Fonctions principales

Les fonctions principales que nous avons créées sont les suivantes :

Affiche grille : Cette fonction prend en argument l'adresse de la grille et affiche cette dernière.

Debug affiche grille : Cette fonction a également besoin de l'adresse de la grille et affiche la "grille cachée", c'est à dire la grille avec tous les bateaux de visibles.

Pose bateaux : Cette fonction nécessite trois arguments : l'adresse de la grille, le nombre et la taille des bateaux à poser. Elle pose un nombre de bateaux de même taille aléatoirement sur la grille.

Encore bateau : Cette fonction teste s'il reste des bateaux de taille donnée en argument sur la grille passée également en argument.

Remplit grille : Cette fonction remplit la grille passée en argument en faisant appel à Pose bateaux. Elle renvoie le nombre de bateaux disposés.

Tableau score : A partir d'un score, cette fonction met à jour (si besoin ie s'il y a un record) le fichier de scores.

Tire : Cette fonction tire une torpille sur la case de coordonnées demandées à l'utilisateur. Elle nécessite l'adresse de la grille ainsi que l'adresse du nombre de bateaux restants en arguments.

Tire débutant : Fonction jumelle de la précédente à la différence près que cette fonction affiche un message à l'utilisateur lorsqu'il torpille une case adjacente à un bateau.

Sauvegarde : Fonction qui sauvegarde une partie pour qu'elle soit reprise ultérieurement si nécessaire. Elle nécessite l'adresse de la grille, la difficulté de la partie, le nombre de coups ainsi que le nombre de bateaux.

4 Difficultés rencontrées

La principale difficulté rencontrée fut le temps. Ayant tous les deux un emploi du temps chargé il fut assez difficile de mener à bien tout ce que nous voulions faire sur ce projet (notamment améliorer le "menu" d'entrée ou bien agrandir le tableau de scores). Bien que nous avons eu à réaliser des choses non vues en cours (ouvrir, lire et écrire dans un fichier par exemple), la documentation MIPS (*https ://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html*) fut d'une grande aide, elle comporte notamment des exemples concrets fort utiles.

5 Remarques sur la programmation en assembleur

La programmation en assembleur fut une expérience plutôt enrichissante, la conception d'un jeu nous permettant de nous favoriser avec de nombreux domaines (que ce soit la manipulation de fichiers, de chaînes de caractères, de tableaux ou encore de fonctions récursives). Le langage est évidemment moins intuitif et le code plus imposant que dans un langage de haut-niveau, néanmoins on y retrouve le même plaisir qu'à coder en C ou autre et à découvrir les rouages de ces langages.