Architecture des ordinateurs
Projet : jeu du snake en assembleur MIPS

Le jeu du snake est un jeu populaire qui consiste en la manipulation d'un snake a travers les touches directionnels pour manger des bonbons et le faire grandir sans avoir à toucher les bords de la grille ou il se trouve et les obstacles qui apparaissent au fil du temps. Ce projet a été élaboré par JBILOU Ghait et FARID Yasmina. On vous écrit ce document afin de vous présenter comment est ce qu'on a conçu les et construit les différentes fonctions demande et vous indiquez les différentes difficultés auxquelles ont a ete confrontes

➔ Fonction majDirection

cette fonction prend la nouvelle direction entrée par l'utilisateur et vérifie si cette dernière ne va pas créer de conflit, tel que le cannibalisme, par rapport à la direction actuelle

cette fonction sauvegarde dans la pile le registre $s0 et le registre $ra, ce qui nous permet d'utiliser le registre $s0 pour nos vérifications, qui ne font changer la valeur de snakeDir que si la valeur entrée par l'utilisateur n'est pas contraire à la valeur actuelle de snakeDir. Après avoir exécuté le corps de la fonction on restaure la valeur du registre $s0 et $ra et on désalloue l'espace mémoire utilisé dans la pile et on retourne à la fonction appelant avec la commande jr $ra

➔ Fonction updateGameStatus

Cette fonction est le cœur de ce programme, puisqu'elle permet de faire évoluer la snake et le faire bouger selon SnakeDir, le faire grandir lorsqu'il mange un bonbon, et permet de générer des bonbons et des obstacles dans des positions aléatoires.

Elle utilise 36 octets de la pile afin de sauvegarder tous les registres $s et le registre $ra. Ainsi pour mieux expliquer cette fonction je propose de la décomposer en 3 sous-fonctions :

- decalX : cette partie permet de décaler d'une case à droite le tableau des X du snake afin de pouvoir le bouger selon l'axe des X (en ignorant le dédoublement des cases)
- decalY : cette partie permet de décaler d'une case à droite le tableau des Y du snake afin de pouvoir le bouger selon l'axe des Y (en ignorant le dédoublement des cases)
- bougerTete : cette sous-fonction nous permet de faire modifier les coordonnées des la tête selon la valeur de la variable snakeDir
- verifCandy : après le mouvement du corps, on vérifie si les coordonnées de la tête sont similaires aux coordonnées du candy, si c'est le cas on

passe à la partie ajoutCandy, et si ce n'est pas le cas on continue le jeu sans interruption

- ajoutCandy : elle incrémente la taille du snake et on profite du dédoublement des cases effectué par decalX et decalY pour agrandir le corps du snake et puis on incrémente le score du jeu et on génère un nouveau candy dans la grille du jeu. Ajoutez à cela le fait qu'elle est responsable aussi de la génération de nouveaux obstacles et de la dégradation du retard et donc augmentation de la vitesse et la difficulté tant qu'une consommation de bonbon prend place durant la partie.

➔ Fonction conditionFinJeu

Cette fonction est la partie permettant de vérifier si les conditions de survies du snake ont été respectées.

Elle vérifie 3 conditions :

- si le snake n'a pas mangé une partie de son corps
- si le snake n'a pas dépassé le bord de la grille
- et si le snake n'a pas heurté d'obstacles

➔ Fonction affichageFinJeu

Cette fonction permet d'afficher à la sortie standard le message_fin_game qui est la chaîne de caractères suivante : « Game Over. Votre score est de : », ainsi que le score de la partie

Nous avons eu affaire à plusieurs difficultés et parmi ces dernières figurent notre mal-organisation au début du projet, parce qu'on pensait au-delà de la question demander dans le but de gérer tous les problèmes possibles qu'on aurait pu affronter, ce qui a retardé de manière considérable la conception des différentes fonctions demandées. Ajoutez à cela notre tendance à écrire beaucoup de code sans avoir à le vérifier ce qui nous a poussé à écrire à maintes reprises plusieurs bouts de fonctions.