

Architecture des Ordinateurs

Projet : Jeu du Snake

Introduction

Le but de ce projet est d'implémenter un petit jeu populaire : le snake. A titre de rappel, le principe du jeu repose sur le contrôle d'une longue ligne semblable à un serpent en se basant sur un système de scoring : il s'agit d'obtenir le meilleur score possible à chaque partie. Pour cela, il est nécessaire de faire grandir son serpent le plus possible en mangeant des pastilles de nourriture disséminées dans l'environnement. Le jeu prend fin lorsque le serpent heurte un obstacle, un bord de l'écran ou son propre corps. Le code sera réalisé en assembleur MIPS, et exécuté grâce à l'émulateur Mars. Le projet est à réaliser en binômes, et à rendre avant **05/12/2021 à 23h59** en respectant l'ensemble des spécifications données dans la section 1. L'archive projet.tar.gz contient :

- le présent document
- le code assembleur de départ à compléter

1 Spécifications

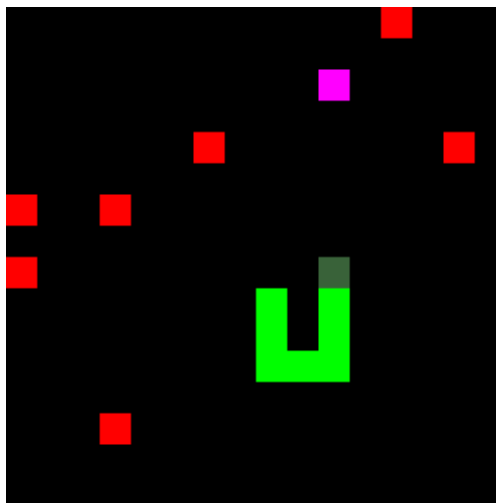


FIGURE 1 – Exemple d'affichage

Votre code prendra la forme d'un fichier `.asm`, exécutable grâce à l'émulateur `Mars4_5`. Lors de son exécution, il demandera à l'utilisateur de contrôler les mouvements du serpent grâce à un ensemble d'entrée clavier. Le jeu prendra fin lorsqu'une des conditions d'arrêt est rencontré. La Figure 1 donne un exemple d'affichage obtenu grâce au simulateur MIPS : le serpent est représenté en vert avec sa tête en vert foncé, les obstacles sont en rouge et la nourriture permettant de faire grandir votre serpent est colorée en rose. La visualisation du jeu correspondra à une représentation graphique d'un segment de mémoire statique, fournie par les outils de Mars.

Le rendu prendra la forme d'une archive nommée `Nom1_Nom2.tar.gz`, qui contiendra : le fichier `.s` et un rapport au format `.pdf` de maximum 2 pages expliquant les choix effectués durant la conception de votre projet, vos structures de données, qui a fait quoi, les difficultés rencontrées, les fonctionnalités ajoutées, et autres observations. Cette archive sera déposée avant le **05/12/2021 à 23h59** sur la plateforme Moodle. La suite de ce document présente quelques aides sur l'utilisation de l'émulateur Mars (section 2), un résumé exhaustif de l'exécution attendue (section 3) ainsi que les fonctionnalités à implémenter (section 4).

2 Mars et appels système

L'émulateur Mars se lance à partir de l'archive java *Mars4_5.jar* grâce à la commande `java -jar Mars4_5.jar`. Pour lancer votre programme, vous devez charger votre fichier *snake.asm*. Ensuite, vous devez activer l'affichage et la capture des entrées claviers en allant dans l'onglet **Tools** pour lancer **Bitmap Display & Keyboard** ainsi que **Display MMIO Simulator**. Dans l'outil **BitMap Display**, vous devez sélectionner une fenêtre carrée 256x256 et une taille de pixel 16x16. Ne pas oublier de connecter les deux widgets à votre code MIPS. Finalement, il ne vous reste plus qu'à lancer l'exécution votre programme et interagir grâce aux entrées claviers. On peut effectuer des appels système grâce au mot clé *syscall*. La liste complète des appels système disponibles ainsi que leur utilisation est décrite à l'adresse : <http://courses.missouristate.edu/kenvollmar/mars/help/syscallhelp.html>. On notera l'appel système 9 qui permet de réserver de la mémoire dans le tas du processus. Le code assembleur MIPS suivant alloue 4 octets de mémoire dans le tas : la taille mémoire, en octet est passé dans le registre \$a0, et après l'appel système, l'adresse de la mémoire allouée se retrouve dans le registre \$v0.

```
1 li $a0 4 # on demande 4 octets
2 li $v0 9 # 9 pour appel système sbrk
3 syscall
4 sw $v0 4($sp) # $v0 contient l'adresse de la mémoire réservée dans le tas
```

3 Déroulement du jeu

Décrivons le déroulement d'une partie de snake en analysant la boucle principale de jeu :

1. Capture d'une touche directionnelle ('z' (haut), 'q' (gauche), 's' (bas), 'd' (droite)). Par défaut la direction précédente du serpent est stocké et réutilisé si aucune touche directionnelle n'est capturé dans le délai entre deux mouvements (entre chaque tour de boucle principal). Au départ le serpent est positionné dans la case en haut à gauche avec une taille initialisé à 1 et la direction du serpent pointe vers la droite.
2. Si une touche directionnelle est capturée, la direction du serpent est mise à jour.
3. Mise à jour de la structure de données contenant les positions de votre serpent. Les positions du serpent sont stockés dans deux tableaux statiques (le premier pour les positions en X du serpent , un second pour les positions en Y). La tête du serpent est toujours stocké en première position et la queue en dernière position. Un décalage des positions dans les deux tableaux est effectué et la tête du serpent est mise à jour en considérant la direction du serpent.
4. Mise à jour du serpent et de l'environnement si une pastille de nourriture était présente sur la tête du serpent après le décalage. Dans ce cas la taille du serpent augmente de un et la position suivante de la prochaine pastille de nourriture est calculée aléatoirement en vérifiant quelques contraintes (position valide dans la grille et position non occupée). De la même manière le tableau statique contenant la liste des obstacles est lui aussi mis à jour en ajoutant un obstacle supplémentaire dès qu'une pastille de nourriture est consommée. L'ordre des obstacles dans le tableau statique n'a pas d'importance. Ne pas oublier de faire évoluer le score dès qu'une pastille est mangé.
5. Dans le cas où le jeu continue, il est désormais possible de rafraîchir l'affichage pour apercevoir les changements.
6. Puis le jeu est mis en sommeil (pendant 0.5 sec par défaut) pour laisser le temps à l'utilisateur d'appuyer sur une touche directionnelle.
7. L'étape suivante consiste à vérifier qu'aucune règle de fin de jeu n'a été atteinte. Il y a trois contraintes à surveiller pour déterminer si le jeu doit s'arrêter ou non :
 - La tête du serpent ne doit pas dépasser la bordure de la grille
 - La tête du serpent ne doit pas rentrer en contact avec un obstacle
 - Le serpent ne doit pas rencontrer une partie de son propre corpsSi l'une de ces règles a été enfreinte, le jeu se termine (Game Over) et il faut afficher dans la console le score obtenu.
8. La boucle principale retourne à l'étape 1.

4 Travail à fournir

Les fonctions d’affichage graphique et de rafraîchissement sont déjà implémentés, aucune modification n’est à apporter dans cette partie du code source.

Vous devez programmer les fonctionnalités suivantes :

- Mise à jour de la direction
- Mise à jour des structures de données (serpent, obstacle, nourriture)
- Détecter et programmer les conditions de fin de jeu
- Affichage de fin de partie avec le score

De plus, le code doit être commenté de façon à pouvoir comprendre l’utilisation des registres temporaires.

Le projet est considéré comme fini lorsque ce cahier des charges est respecté. Néanmoins, pour avoir la note maximale, des fonctionnalités supplémentaires peuvent être ajoutées :

- Afficher le score graphiquement sur l’écran de jeu en fin de partie
- "Rainbow snake" ou chaque partie du serpent possède une couleur différente
- Système de niveau prédéfini en fonction du score obtenu

Enfin, il est interdit d’utiliser le mot-clé *.macro*.