Groupe : TP2

# Rapport Projet Assembleur Snake
# Jules Goy et Kilian Sakhi
# SAE 13.2

## Repartions du projet :

Kilian et Jules :

- Les touches zqsd changent la direction du Snake.
- Print du bonbon.

Jules :

- Incrémentation du score.
- Print des obstacles.
- Collision avec les obstacles et avec les bordures de l'écran.
- Affichage du score dans le terminal de Mars.

Kilian :

- La queue du Snake augmente quand un bonbon est mangé et elle suit les anciennes positions de la tête du Snake
- Le Snake avance en fonction de sa direction.
- Collision avec la queue.

## Répartition du travail :

La répartition du travail était égalitaire Jules s'est concentré sur les parties les plus "faciles" du code tandis que Kilian c'est lui concentré sur les parties les plus "difficiles" qui étaient en sous nombre par rapport aux parties "faciles".

## Temps de travail :

Nous avons tous les deux travaillés autant l'un que l'autre. Nous avons essentiellement travaillé lors des séances SAE prévues. Mais aussi chez nous où sur les temps d'ouverture de l'iut. Au total le projet Snake en assembleur nous a pris 15h.

## Explication du code (Partie déplacement):

Tout d'abord pour les changements de direction du Snake. Nous avons utilisé des conditions "beq" pour changer la valeur de SnakeDir en fonction de la touche. Ensuite pour ne pas que le Snake ne puisse se retourner en une action nous comparons la valeur de la soustraction de SnakeDir avec 2 ou –2. Car si la soustraction est égale à 2 ou –2 alors cela voudra dire que la touche entrée va dans la direction inverse de la direction du Snake.

Pour ce qui est de l'avancement du Snake sur la grille nous réalisons un calcul simple qui est : la valeur de la position X de la tête du Snake +- 1 et la valeur de la position Y de la tête du Snake +- 1.

Pour les déplacements de la queue du Snake, on a fait en sorte que les pixels après la tête deviennent noirs pour éviter une traînée de la couleur de la tête. Puis on a fait une boucle qui en fonction de la taille du Snake déplace les valeurs des registres des positions de 4

octets vers la droite en partant de la fin jusqu'au début de la liste des positions du serpent. Et enfin, on déplace la tête en fonction de snakedir.

## Explication du code (Partie incrémentation score, tailleSnake et apparition bonbon/obstacle) :

Pour savoir si le Snake a mangé un bonbon on compare ses coordonnées avec celles du bonbon et si oui alors on incrémente le score et la taille du Snake, pour se faire, on charge les 2 valeurs de ScoreJeu et de tailleSnake dans deux variables puis on les incrémente de un et enfin on recharge ces varaibles dans les deux words. Pour le print d'un bonbon on récupère des coordonnées avec newRandomObjectPosition puis on les charge dans le tableau Candy et on appelle la fonction PrintCandy. Pour ce qui est d'un obstacle on charge des coordonnées avec newRandomObjectPosition puis on charge ces coordonnées dans les tableaux obstaclePosX et obstaclePosY à l'indice numobstacle. Pour charger ces coordonnées à l'indice numobstacle on multiplie numobstacle par 4 qu'on ajoute à l'adresse des tableaux ce qui nous donne les adresses des indices numobstacle des de tableaux ux .

## Explication du code (Partie condition fin jeu) :

Pour les collisions entre le Snake et les bordures de l'écran, on compare les coordonnées X et Y du Snake avec 16(qui est la taille maximale en hauteur et largeur de l'écran). Si les coordonnées du Snake sont supérieures à 16 alors le Snake n'est plus visible à l'écran donc il meurt.

Pour les collisions avec les obstacles on créer une boucle qui va de 0 à numobstacle puis on compare chaque élément d'adresse |numobstacle + obstaclePosX| et d'adresse | numobstacle * 4 + obstaclePosY| s'il y a collision alors le jeu s'arrête sinon on ajoute 4 à numobstacle.

Pour les collisions avec la queue, on a utilisé la même technique qu'avec les obstacles on a juste remplacé numobstacle par tailleSnake et obstaclePos(X/Y) par les coordonnées de la queue.

## Ce que cette SAE nous a apportée :

En conclusion cette SAE nous a beaucoup aidée à comprendre comment fonctionne les registres et les pointeurs. De plus elle nous a permis de coder notre premier jeu vidéo qui plus est en assembleur c'est donc une grande fierté.