Projet Snake

1 Présentation du jeu

— Le joueur contrôle un serpent qui se déplace sur un plateau.
— L'objectif est de manger des pommes qui apparaissent aléatoirement sur un plateau.
— A chaque pomme mangée, le serpent grandit.
— La partie est perdue lorsque le serpent se cogne à un bord du plateau ou à lui même.

2 Projet

Coder une version du jeu sur une carte `micro:bit`.

2.1 Carte Microbit

La carte `micro:bit` est un microcontrôleur programmable en Python. Pour coder, nous utiliserons l'éditeur `Mu` et nous importerons les fonctions spécifiques par : `from microbit import *`.
L'affichage sera réalisé sur la matrice de leds 5*5.
Le contrôle du serpent pourra se faire avec l'accéléromètre de la carte en inclinant la carte dans la direction souhaitée.

Débugger un programme sur Mu n'est pas pratique : les messages d'erreurs s'affichent sur la carte. Les outils possibles sont :

— le bouton `vérifier`,
— l'éditeur Spyder : les méthodes du module `microbit` n'étant pas disponibles sur Spyder, un pseudo-module `microbit` est fourni pour simuler les entrées (boutons, accéléromètre) et les sorties (matrice de leds, affichage de texte).

L'utilisation de la récursivité semble régulièrement poser des problèmes sur microbit : privilégier des versions itératives des méthodes dans le module `File`.
Il est également possible qu'il y ait des erreurs lors du transfert des fichiers sur la carte : il faut alors réduire au maximum le volume de code (supprimer les codes et les méthodes inutiles comme `__str__` dans `File` qui n'est utile que pour la mise au point du programme).

2.2 Matrice de leds

— `display.set_pixel(x, y, intensite)` : allume la led de coordonnées (x, y). L'origine du repère est la led en haut à gauche.
— `display.clear()` : efface l'écran.
— `display.scroll(texte)` : fait défiler une chaîne de caractères.

2.3 Accéléromètre

Mesure l'inclinaison gauche/droite et devant/derrière de la carte. La lecture se fait instantanément. Il faudra quand même prévoir une temporisation dans le tour de jeu pour que le jeu ne soit pas trop rapide.

— `accelerometer.get_x()` : renvoie l'inclinaison gauche/droite sous la forme d'un entier compris entre -1000 et 1000.
— `accelerometer.get_y()` : renvoie l'inclinaison devant/derrière sous la forme d'un entier compris entre -1000 et 1000.

Il faudra pouvoir une sensibilité de l'ordre de 100 pour la prise en compte de l'inclinaison dans le mouvement : x ou y supérieurs à 100 ou inférieurs à -100.

2.4 Fichiers

On pourra découper le programme en 4 fichiers :
— un module pour une classe `File`,
— un module pour la classe `Serpent`, héritant de `File`,
— un module pour les classes `Pomme`, `Amanger` et `Digestion`,
— le programme principal pour le déroulement du jeu.

Les fichiers annexes doivent être importés dans la carte par copier-coller en utilisant le bouton `fichiers` de Mu, à partir du dossier `mu_code`.
Le programme principal est transféré avec le bouton `flasher`.

3 Pommes

3.1 Pomme

Une pomme est un point aléatoire dont les coordonnées sont choisies en dehors du serpent.
On pourra tirer aléatoirement des points jusqu'à obtenir un point qui n'appartienne pas au serpent.

3.2 Amanger

`Amanger` est une `File` stockant les pommes disponibles à manger (a priori, une seule pomme dans cette file, mais on pourrait imaginer une version du jeu avec plusieurs pommes disponibles).

3.3 Digestion

`Digestion` est une **file** de `Pommes` :
— lorsque le serpent mange une pomme de `Amanger`,
  — la pomme passe dans la file `Digestion`,
  — une nouvelle pomme est créée et enfilée dans `Amanger`
— le serpent avance jusqu'à ce que la pomme de tête arrive au niveau de la queue (fin de la digestion de la pomme).
— lorsque la queue du serpent dépasse la pomme de tête, le serpent grandit :
  — la pomme est défilée de `Digestion`
  — la pomme est ajoutée à la queue du serpent.

4 Classe Serpent

Le `corps` du serpent est une file : il est constitué de points. Nous parlerons ici de la tête du serpent (entrée de la file) et de la queue du serpent (sortie de la file). Le serpent a également une `direction` (sous la forme d'un vecteur) dans laquelle se déplace sa tête.
— Lorsqu'il `avance`, on enfile un nouveau point en tête et on défile côté queue.
— Lorsqu'il `tourne`, on modifie sa direction.
— Lorsqu'il `grandit`, on ajoute un point à la queue.
— Lorsque le serpent avance, il faut vérifier les `colisions` :
  — il ne sort pas de la matrice de leds,
  — il ne se cogne pas à son propre corps.

Il faut donc prévoir des méthodes pour accéder ou modifier les valeurs à la tête et à la queue.

5 Module Snake

Programme principal : comme tout programme tournant sur un micro-contrôleur, le programme tourne en boucle tant que l'appareil est alimenté. On utilise donc une boule `while True`.
A chaque tour :
— prévoir un temps de pause d'une seconde environ : on utilisera une variable `vitesse` que l'on pourra faire varier si besoin et la fonction `sleep` (fonction du module `microbit` en millisecondes).
— le serpent tourne (éventuellement, en fonction de l'accéléromètre),

— le serpent avance (toujours),
— vérifier les collisions,
— le serpent grandit (éventuellement),
— le serpent mange (éventuellement),
— l'affichage est actualisé.

6 Oral

En fin de projet, oral de 4 à 5 minutes, en binôme. Vous pourrez développer :
— Organisation du travail dans le groupe,
— Évolution des versions,
— Perspectives d'évolution,
— Point du programme réinvestis,
— Ce que vous avez appris,
— Difficultés rencontrées,
— ...