SNAKE AVEC LE MODULE PYXEL

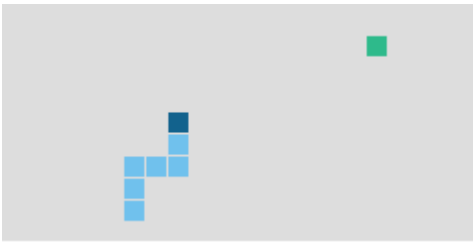
1 - INTRODUCTION

La Nuit du Code (https://www.nuitducode.net/) est un concours organisé chaque année dont l'objectif est de réaliser en équipe un jeu en quelques heures.

Afin de se concentrer sur le fond, le module **pyxel** est utilisé.

La documentation de ce module :
https://github.com/kitao/pyxel/blob/main/docs/README.fr.md

Avant de se lancer dans un projet personnel, vous allez découvrir son utilisation avec ce mini projet, un SNAKE. Dans un premier temps, il faudra programmer ce jeu de manière classique (en mode impératif). Dans un deuxième temps, il faudra le re-programmer en mode Objets en appliquant les nouveautés de Terminale avec la Programmation Orientée Objets.



2 - CAHIER DES CHARGES

Avant de coder un jeu, il faut en connaître le comportement précis donc maîtriser les règles du jeu.

Pour le SNAKE :

- une pomme est placée aléatoirement sur l'écran ;
- le serpent se déplace automatiquement, on peut changer sa direction avec les flèches du clavier ;
- s'il mange la pomme, il grandit et celle-ci réapparait dans une case vide ;
- s'il quitte l'écran ou se mord, perdu et le jeu s’arrête.

De manière générale, un jeu vidéo se résume à une boucle infinie qui le fait progresser suivant une sucession de tours.

A chaque tour :

1. On écoute les interactions du joueur.
2. On met à jour l'état du jeu.
3. On dessine les éléments à l'écran.
4. On attend quelques millisecondes.

Le module Pyxel gère la boucle infinie et le délai entre chaque tour. Les points 2. et 3. sont à programmer dans deux fonctions prédéfinies :

- point 2 : update()
- point 3 : draw()

Au début du programme, on crée la fenêtre du jeu : *pyxel.init(4 00, 400, title ="snake")*

A la fin du programme, on lance l'exécution du jeu avec *pyxel.run(update, draw)* qui fait appel aux deux fonctions prédéfinies, qui seront appelées 20 fois par seconde.

Il existe de nombreuses méthodes toutes faites permettant de dessiner, écrire du texte…

Les couleurs sont désignées par des entiers de 0 à 15 (0 désignant le noir).

Quelques instructions que l'on va utiliser :

pyxel.cls(0) : efface l'écran et le remplit de la couleur 0 (noir)

pyxel.btn(pyxel.KEY_RIGHT) ou UP, LEFT, DOWN, *pyxel.btn(pyxel.KEY_SPACE)* : récupère les actions clavier des flèches et espace

pyxel.text(50,64, 'GAME OVER', 7) : place du texte en coordonnées (50,64) avec la couleur 7

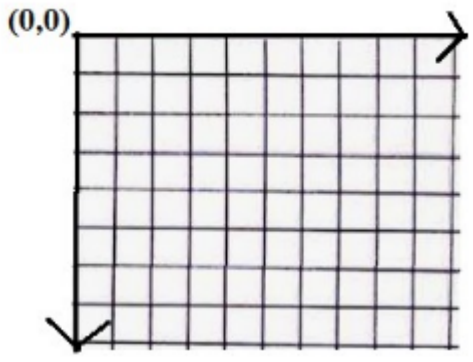
pyxel.rect(x, y, long , larg , 1) : crée un rectangle en coordonnées (x,y) de dimensions long sur larg de couleur 1

3 - SNAKE version 1

a - La grille

On va dessiner le serpent en utilisant les cases d'une grille.

Les cases seront représentées par des coordonnées. L'origine est en haut à gauche. On commence à zéro, la 1 ère coordonnée est l'abscisse (numéro de colonne) et la seconde l'ordonnée (numéro de ligne)



Question 1 : ici, la grille a pour dimensions 200x160 pixels, et 10 cases par 8. Chaque case est carrée de côté ..... pixels.

Coordonnées de la case en bas à gauche : ..... ; et en haut à droite : .....

SNAKE V 0 : le programme peut débuter de cette manière avec les quatre premières variables explicites (en majuscule), ce sont des variables globales utilisées en plusieurs endroits.

```
import pyxel

TITLE = "snake"
WIDTH = 200
HEIGHT = 160
CASE = 20

pyxel.init(WIDTH,HEIGHT,title=TITLE)

def draw():
  pyxel.cls(0)
```

b - Le serpent

Le serpent est représenté par une liste :

snake = [[3, 3], [2, 3], [1, 3]] , place ce code après pyxel.init.

Le premier élément est sa tête placée sur la grille en [3,3] ensuite vient son corps.

Question 2 : Quel est son nombre d'anneaux initial ? : .....

c - Dessiner le serpent

Pour dessiner sur l'écran les cases du serpent, on utilise la méthode *pyxel.rect(x,y,L,l,color)*.

Place les instructions suivantes dans la fonction draw() :

```
for anneau in snake[1:]:
  x,y=anneau[0],anneau[1]
  pyxel.rect(x*Case,...,...,...,11) # à compléter, 11 couleur verte
x_head, y_head = snake[0]
pyxel.rect(..., ..., ..., ..., 9) # à compléter, 9 couleur orange
```

Question 3 : complète les deux lignes correctement.

Pour tester ajouter le code suivant :

```
def update():
  pass

pyxel.run(update,draw)
```

d - Le score

Au début, la variable globale *score* vaut 0 (à définir au même endroit que la variable snake, au niveau principal du programme, à l'extérieur de toute fonction).

On la mettra à jour plus tard dans la fonction *update()*, mais on peut déjà écrire le score initial sur la fenêtre, par une instruction dans la fonction *draw()*, y ajouter :

```
pyxel.text(4,4,f"SCORE : {score}",7) # 7 couleur blanche
```

Tester.

4 - SNAKE version 2

On va animer le serpent. On va ajouter une variable globale *direction = [1,0]* donnant le sens de déplacement au début du pprogralle.

Question 4 : Avec cette initialisation, quel est le sens de déplacement initial du serpent ?

a - Déplacer le serpent tout droit

Le code permettant de mettre à jour la tête du serpent :

```
head = [snake[0][0]+direction[0],snake[0][1]+direction[1]]
snake.insert(0,head)
```

Avant le premier déplacement, *snake = [[3, 3], [2, 3], [1, 3]]*.

Question 5 : Que devient la variable *snake* après le premier déplacement ?
.....

Que dire de la taille du serpent ? .....

On efface le dernier élément de *snake* pour terminer le mouvement : *snake.pop()*

Question 6 : Intégrer ces instructions dans la fonction *update()* et lancer le programme. Que se passe-t-il ? .....

b - Ralentir le jeu

La fonction *update()* est lancé 30 fois par seconde ce qui donne une bonne fluidité mais cela est trop rapide pour le déplacement du serpent. Pour ralentir, on utilise le compteur de frames intégré à Pyxel et on imposera un mouvement par exemple toutes les 15 frames.

Ajoutez la variable globale *FRAME_REFRESH = 15* au début du programme puis ajouter le test suivant dans la fonction *update()* et tester :

```
global direction,FRAME_REFRESH,score
if pyxel.frame_count % FRAME_REFRESH == 0:
        head = [snake[0][0]+direction[0],snake[0][1]+direction[1]]
        snake.insert(0,head)
        snake.pop()
```

c - Changer de direction

Cela va se faire dans la fonction *update()* en « écoutant » les interactions du joueur (quand il tape sur une touche du clavier) avec *pyxel.btn*. On modifiera la variable *direction* en prenant en compte que le changement de direction se fait à gauche ou à droite de la direction actuelle (pas de retour en arrière).

Question 7 : Ajouter le code suivant en complétant les 4 zones.

```
if pyxel.btn(pyxel.KEY_ESCAPE):
        exit()
elif pyxel.btn(pyxel.KEY_RIGHT) and direction in ([0,1],[0,-1]):
        direction=[1,0]
elif pyxel.btn(pyxel.KEY_LEFT) and direction in ([0,1],[0,-1]):
        direction=[-1,0]
elif pyxel.btn(pyxel.KEY_UP) and direction in ([1,0],[-1,0]):
        direction=[... , ...]
elif pyxel.btn(pyxel.KEY_DOWN) and direction in ([1,0],[-1,0]):
        direction=[... , ...]
```

d - Cas d'arrêt

Dans le jeu, le serpent perd s'il se mord la queue ou s'il quitte l'écran : le jeu doit s'arrêter et on quitte la fenêtre.

Pour savoir si la tête touche son corps : on teste si les coordonnées de la tête correspond à un anneau existant du serpent.

On doit vérifier plusieurs conditions pour savoir si la tête sort de l'écran.

On ajoute alors le test dans *update()* :

```
if head in snake[1:] or head[0]<0 or head[0]>WIDTH/CASE-1 or head[1]
<0 or head[1]>HEIGHT/CASE-1:
    exit()
```

5 - SNAKE version 3

On ajoute une pomme (food) sachant que si la tête du serpent l'atteint le serpent s'agrandit d'un anneau (la queue n'est pas effacée) et le score augmente de 1.

On utilisera le module *random* afin de placer aléatoirement la pomme *food* (encore une variable globale):

```
from random import randint

food = [randint(0,WIDTH/CASE-1),randint(0,HEIGHT/CASE-1)]
```

Le problème est qu'aléatoirement, la pomme peut se retrouver sur le serpent. On doit donc ajouter ce code :

```
while food in snake:
    food = [randint(0,WIDTH/CASE-1),randint(0,HEIGHT/CASE-1)]
```

Ce code devra être ajouté aussi à *update()* par la suite.

Dans la fonction *draw()* :

```
x_food, y_food = food
pyxel.rect(x_food*CASE,y_food*CASE,CASE,CASE,8) # 8 couleur rose
```

Question 8 : finaliser votre code afin que le jeu s'éxécute conformément au cahier des charges.

Si vous avez le temps, vous pouvez améliorer la dernière version (ajouter un message à la fin, recommencer automatiquement le jeu, mémoriser un score max, ...)