

Snake très simple avec Pyxel, pas à pas

Objectifs

L'objectif de ce petit projet initial est de vous familiariser avec la création de jeux vidéos à l'aide du moteur de jeux « rétro » Pyxel, logiciel libre et open-source.

Le fonctionnement de Pyxel repose sur des [constantes et des fonctions](#) prédéfinies qui réalisent automatiquement des actions pour le programmeur.



Pyxel est un module utilisable via Pyzo en l'installant avec la commande >> ***pip install pyxel***
On l'utilise ensuite en important le module au début du script : ***import pyxel***

Cahier des charges :

- le serpent se meut automatiquement, on peut le déplacer avec les flèches du clavier.
- s'il mange la pomme, il grandit et celle-ci réapparaît dans une case vide
- s'il quitte l'écran ou se mord, il meurt, et le jeu s'arrête.s

Principes généraux des jeux vidéos

Un jeu vidéo peut être résumé ainsi : 

Dans Pyxel, la boucle infinie est implicite, et l'attente des quelques millisecondes déjà prise en charge => pas besoin de s'en occuper.

Des fonctions prédéfinies gèrent les actions 2 et 3

- Une **boucle infinie** fait progresser le jeu :
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

action

fonction Pyxel prédéfinie

Mettre à jour l'état du jeu

update()

Dessiner les éléments à l'écran

draw()

Au début du programme, on crée la fenêtre du jeu : `pyxel.init(400, 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.)

Effacer l'écran et le remplir de noir	<code>pyxel.cls(0)</code>
Détection d'interactions utilisateurs	Flèches clavier <code>pyxel.btn(pyxel.KEY_RIGHT)</code> ou UP, LEFT, DOWN barre espace : <code>pyxel.btn(pyxel.KEY_SPACE)</code>
Ecrire du texte	<code>pyxel.text(50,64, 'GAME OVER', 7)</code>
Dessiner un rectangle	<code>pyxel.rect(x, y, long, larg, 1)</code> x et y : coords du sommet haut gauche. Ensuite les dimensions. Dernier paramètre : la couleur

En Pyxel, on utilise généralement des variables **globales** qui sont définies à la racine du script et sont mises à jour dans `update()`. (Ce n'est pas une bonne pratique... mais c'est facile) Pour préciser que la fonction a le droit de modifier une variable globale, par exemple le score : on écrira `global score`.

Version 1 : dessiner le serpent

La 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*)

Exemple : ici, la grille a pour dimensions 200x160 pixels, et **10 cases par 8**. Chaque case est carrée de côté ...**20**... pixels.

On définit alors les variables HEIGHT, WIDTH, CASE (en majuscules car ce sont des constantes : convention !)

Puis on peut créer la fenêtre avec `pyxel.init`

Coordonnées de la case en bas à gauche ...**(0 ; 7)**..... ; et en haut à droite ...**(9 ; 0)**.....

Le serpent

Le serpent est représenté par une variable double liste : `snake = [[3,3],[2,3],[1,3]]`, définie au début du programme (après `pyxel.init`)

Le premier élément est sa tête, elle est en [3,3] ensuite vient son corps.

Représenter le serpent initial sur la grille ci-dessus. Quel est son nombre d'anneaux initial ? ...**3**.....

Dessiner le serpent

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

- `x` et `y` sont les coordonnées du coin supérieur gauche, `L` et `l` les dimensions du rectangle.
- `color` est un indice entre 0 et 15 désignant une couleur de la palette prédéfinie Pyxel.

Les instructions suivantes seront placées dans la fonction `draw()`

```
# dessiner le corps en vert
for anneau in snake[1:]:
    x, y = anneau[0], anneau[1]
    #11 est une couleur un peu verte
    pyxel.rect(x * CASE, y * CASE,
              CASE, CASE, 11)

# dessiner la tête en orange
x_head, y_head = snake[0]
#9 est la couleur orange
pyxel.rect(x_head * CASE, y_head * CASE,
          CASE, CASE, 9)
```

Commentaires

Snake[1 :] est la liste sauf le 1^{er} élément

On reconnaît une affectation par tuple

*X*CASE correspond à un nombre de pixels alors que x est un numéro de colonne*

On traite à part le 1^{er} élément de la liste snake car il n'est pas de la même couleur

Ecrire 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()`

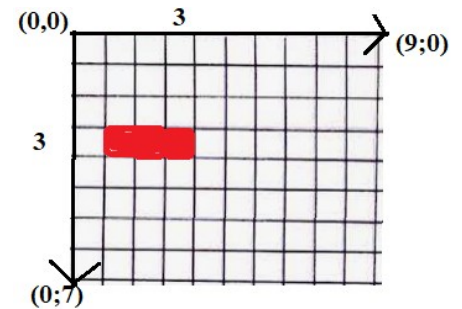
Enfin, on écrit une fonction `update()` pour l'instant vide, et on lance le jeu avec `pyxel.run`

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

```
def update():
    pass

pyxel.run(update, draw)
```

Jalon 1
Le serpent
est dessiné



Version 2 : animer le serpent

Jusqu'ici, le serpent ne bougeait pas. On va l'animer un peu.

Déplacer le serpent « tout droit »

Pour commencer, on va supposer que la direction de déplacement du serpent est $direction = [1,0]$ c'est-à-dire que le serpent va ...**vers la droite**.... On ajoute la variable globale $direction$ au début.

```
# la nouvelle tête est l'ancienne, déplacée dans la direction
head = [snake[0][0] + direction[0], snake[0][1] + direction[1]]

# on l'insère au début
snake.insert(0, head)
```

Exemple : au début, on a $snake = [[3, 3], [2, 3], [1, 3]]$

Que devient maintenant $snake$ après un déplacement ? $snake = [[4, 3], [3, 3], [2, 3], [1, 3]]$.

Que dire de la taille du serpent ? ...**elle est augmentée de 1**.....

On efface le dernier élément de $snake$ pour terminer le mouvement : $snake.pop(-1)$

A FAIRE : Intégrer ces instructions dans la fonction $update()$ qui est appelée automatiquement par Pyxel 30 fois par seconde, et lancer le programme.

Que se passe-t-il ?**ça va beaucoup trop vite** !.....

Ralentir le jeu

30 images par secondes (ou *Frames Per Second FPS*), ça donne une bonne fluidité d'affichage, mais ça fait quand même trop rapide pour le mouvement du serpent. Pour ralentir, on va utiliser le compteur de frames intégré à Pyxel, en effectuant le mouvement par exemple uniquement tous les 15 frames.

On rajoute la constante $FRAME_REFRESH = 15$ au début avec les constantes, puis dans la fonction $update$ on met le mouvement au sein d'un test. Vérifiez : le mouvement est beaucoup plus lent !

```
def update():
    if pyxel.frame_count % FRAME_REFRESH==0:
        # la nouvelle tête est l'ancienne, déplacée dans la direction
        head = [snake[0][0] + direction[0], snake[0][1] + direction[1]]
        # on l'insère au début
        snake.insert(0, head)
        snake.pop(-1)
```

Changer la direction du serpent

Cela va se faire dans la fonction $update()$ en « écoutant » les interactions du joueur (quand il tape sur une touche du clavier) avec $pyxel.btn$

NB : pour avoir le droit de modifier la variable $direction$ au sein de la fonction $update$, elle doit être bien déclarée comme globale dans cette fonction.

```
#on écoute les interactions du joueur (30 x par seconde)
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]
```

Jalon 2
Le serpent
doit tourner

Question : à quoi sert la première ligne dans le if ? ...**détecte si on a appuyé sur Escape et quitte** !.....

Version 3 : faire mourir le serpent

Dans notre version du jeu : le serpent meurt lorsqu'il se mord la queue, ou lorsqu'il quitte l'écran. Dans ce cas, le jeu s'arrête, et on quitte la fenêtre.

Pour savoir si la tête du serpent a touché son corps : on teste si les coordonnées de la tête correspondent à un anneau déjà existant du serpent :**head in snake[1:]**.....

Pour savoir si la tête du serpent « sort » de la fenêtre : on doit vérifier plusieurs conditions :

- En abscisse : c'est dedans si ...**x>=0** ET **x<=9**.....
donc ça sort si **x<0**..... OU **x>9**.
- En ordonnée : c'est dedans si ... **y>=0**... ET ... **y<=7**.....
donc ça sort si **y<0**..... OU ... **y>7**.....

```
# mort du serpent ?
# s'il touche son corps ou s'il quitte l'écran
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()
```

D'où la condition multiple dans la fonction *update()*

Vérifiez son fonctionnement : pour le cas où il se mord la queue, vous aurez besoin de définir au début un serpent plus long. On pourra remplacer *exit()* par *pyxel.quit()*

Jalon 3
Le serpent
peut mourir

Version 4 : manger la pomme... et réagir !

On place une pomme (matérialisée par une case rose) au hasard dans la fenêtre. Lorsque le serpent mange la pomme, il grandit d'un anneau (sa queue n'est pas effacée), et le score augmente de 1.

Variable représentant la pomme :

```
food = [8,3]
```

(au début, on la place arbitrairement)

```
#la nourriture :
#food est une variable globale
x_food, y_food = food
#8 est la couleur rose
pyxel.rect(x_food * CASE, y_food * CASE, CASE, CASE, 8)
```

Comment tester si le serpent a mangé la pomme ? On teste si les coordonnées de la tête coïncident avec celles de la pomme, donc si**head == food (ou snake[0]==food)**.....

Pour replacer une nouvelle pomme, on tire au hasard des coordonnées dans la grille. Pour cela, on a besoin de la fonction *randint*.

randint doit être importée depuis le module *random*, au tout début du programme

```
import pyxel
from random import randint
```

randint(a, b) renvoie un entier aléatoire entre a et b.

On recommence jusqu'à ce que ces coordonnées soient OK (pas « dans le corps du serpent »)

```
while food in snake:
    # nécessaire de tirer plusieurs fois si on n'a pas de chance !
    food = [randint(0, WIDTH/CASE - 1),
            randint(0, HEIGHT/CASE - 1)]
#sortie du while : on a trouvé une nouvelle case pour la pomme
```

Jalon 4
Le serpent grandit, la
pomme réapparaît

Extensions possibles

A ce stade, le jeu est terminé ! Plusieurs améliorations sont possibles : au lieu de quitter si le serpent meurt, relancer instantanément une nouvelle partie ; conserver un high score (tant qu'on ne quitte pas le jeu puis de manière persistante en l'écrivant dans un fichier), améliorer les graphismes, ajouter du son...