Mini projet Jeu du serpent

# Objectifs



L'objectif de ce mini projet est de vous familiariser avec la création de jeux vidéo 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](https://github.com/kitao/pyxel) prédéfinies qui réalisent automatiquement des actions pour le programmeur.

Pyxel est un module utilisable via Spyder en l’installant avec la commande :



0n 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 réapparait dans une case vide
* S'il quitte l'écran ou se mord, le jeu s’arrête.

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 

# Principes généraux des jeux vidéo

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

|  |  |  |
| --- | --- | --- |
| action |  | fonction Pyxel prédéfinie |
| Ecouter et mettre à jour l’état du jeu  | et  | **update()** |
| Dessiner les éléments à l’écran | Une image contenant Graphique, capture d’écran, cercle, logo  Description générée automatiquement | **draw()** |

Au début du programme, on crée la fenêtre du jeu : **pyxel.init(WIDTH, HEIGHT, title=TITLE)**

A la fin du programme, on lance l’exécution du jeu avec **pyxel.run(update, draw)** qui fait appel aux deux fonctions update() et draw() prédéfinies, qui seront appelées 30 fois par seconde.

Il y a de nombreuses méthodes permettant de dessiner, écrire du texte. Les couleurs sont numérotées de 0 à 15 :

|  |
| --- |
|  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

|  |  |
| --- | --- |
| Effacer l’écran et le remplir de noir(0 est la couleur noire) | pyxel.cls(0) |
| Détection d’interactions utilisateurs (ces fonctions renvoient un booléen) | Flèches clavierpyxel.btn(pyxel.KEY\_RIGHT) ou UP, LEFT, DOWNbarre d'espace : pyxel.btn(pyxel.KEY\_SPACE) |
| Ecrire du texte(7 est la couleur blanche) | pyxel.text(50, 64, 'GAME OVER', 7) |
| Dessiner un rectangle(1 est la couleur bleu foncé) | pyxel.rect(x, y, long, larg, 1)x et y : coordonnées du sommet haut gauche. Ensuite les dimensions. Dernier paramètre : la couleur |

En Pyxel, on utilise généralement des variables **globales** qui sont définies au début 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 située en dehors de la fonction, 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 1ère 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 …………… pixels.

import pyxel

#  Constantes du jeu

TITLE = "snake"

WIDTH = 200

HEIGHT = 160

CASE = 20

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

def draw():

    #  Ecran : à effacer puis remplir de noir.

    pyxel.cls(0)

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

Puis on peut créer la fenêtre avec pixel.init()

Coordonnées de la case :

en bas à gauche ……………………………….

et en haut à droite ………………………………………….

### Le serpent

Le serpent est représenté par une variable liste de listes : $snake = [[3, 3], [2, 3], [1, 3]]$, 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 ? …………………

### 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 $draw()$

*commentaires*

 #  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)

### Ecrire le score

#  Le score

#  7 est la couleur blanche.

pyxel.text(4, 4, f"SCORE : {score}", 7)

Au début, la variable globale score vaut 0 *(à définir au même endroit que la variable* snake*, au début 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().

**Jalon 1**

**Le serpent est dessiné**

def update():

    pass

pyxel.run(update, draw)

Enfin, on écrit une fonction update() pour l’instant vide, et on termine avec pyxel.run(update, draw)

En**registrer** cette première version du jeu sous le nom **serpent\_v1.py**

# Version 2 : animer le serpent

### Déplacer le serpent « tout droit » Copier serpent\_v1.py sous le nom serpent\_v2.py et fermer serpent\_v1.py

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

        #  La nouvelle tête est l'image de l'ancienne tête par une translation.

        #  Les coordonnées du vecteur de translation sont données par la

        #  liste direction.

        head = [snake[0][0] + direction[0], snake[0][1] + direction[1]]

        #  On insère la nouvelle tête au début de la liste de listes snake.

        snake.insert(0, head)

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

Que devient maintenant la variable $snake$ après un déplacement ? ……………………………………………….

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

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 ? …………………………………………………………………………………………………………………………..

### 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:

        head = [snake[0][0] + direction[0], snake[0][1] + direction[1]]

        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 fois 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 ? ………………………………………………………………..

### Version 3 : faire mourir le serpent Copier serpent\_v2.py sous le nom serpent\_v3.py et fermer serpent\_v2.py

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 : ……………………………………………………………..

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

 #  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()

* En abscisse : c’est dedans si ………… ET ………………..…

donc ça sort si …………………………… OU …………………..

* En ordonnée : c’est dedans si ………… ET ………………..

donc ça sort si ……………………………… OU ………………….

**Jalon 3**

**Le serpent peut mourir**

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 $pyxel.quit()$

### Version 4 : manger la pomme et exécuter des actions en conséquence Copier serpent\_v3.py sous le nom serpent\_v4.py et fermer serpent\_v3.py

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.

#  Dessiner 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)

Variable représentant la pomme :

$food = [8, 3]$

*(au début, on la place arbitrairement )*

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 …………………………………………….

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 $b$ inclus.

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

# Fait disparaitre la pomme et la fait réapparaitre ailleurs.

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 Copier serpent\_v4.py sous le nom serpent\_v5.py et fermer serpent\_v4.py

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…

Résumé de la structure du programme :

import pyxel

#  Constantes du jeu

TITLE = "snake"

WIDTH = 200

HEIGHT = 160

CASE = 2

FRAME\_REFRESH = 15

#  Initialisation des variables

score = 0

snake = [[3, 3], [2, 3], [1, 3]]

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

def draw():

    global food

    #  L'écran : à effacer puis remplir de noir.

    pyxel.cls(0)

    #  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)

...

    #  Le score

    #  7 est la couleur blanche.

    pyxel.text(4, 4, f"SCORE : {score}", 7)

    ...

def update():

    global direction

    #  On écoute les interactions du joueur (30 fois 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]

    ...

    #  Section avec un rafraichissement plus faible pour le déplacement du serpent

    if pyxel.frame\_count % FRAME\_REFRESH == 0:

        #  La nouvelle tête est l'image de l'ancienne tête par une translation.

        #  Les coordonnées du vecteur de translation sont données par la

        #  liste direction.

        head = [snake[0][0] + direction[0], snake[0][1] + direction[1]]

pyxel.run(update, draw)