Le Snake

Pyxel est un module Python pour créer des jeux vidéos rétros.
Grâce à ses spécifications simples inspirées par les consoles rétro, comme le fait que seulement 16 couleurs peuvent être affichées et que seulement 4 sons peuvent être lus en même temps, vous pouvez vous sentir libre de créer des jeux vidéo dans le style pixel art.
Les spécifications et les API de Pyxel sont inspirées de PICO-8 et TIC-80.
Pyxel est libre et open source.

Pour avancer dans ce tutoriel, il vous faudra aller chercher certaines informations dans la documentation de ce module.
Cette documentation est accessible ici :
https://github.com/kitao/pyxel/blob/main//docs/README.fr.md#documentation-de-lapi

Pour chaque étape de se tutoriel de découverte, retrouvez sur studserver ce que vous devez obtenir.

Vous êtes bien sur libres de prendre des libertés avec ce tutoriel.
N'hésitez pas à tester, à modifier, à améliorer le jeu que vous allez créer en suivant les étapes de ce tutoriel.
Faîtes preuve d'imagination et de créativité !

Pour travailler vous avez trois possibilités :

Utiliser votre IDE préféré (spyder) en important le module `pyxel` au début de votre programme.
Cette méthode est à privilégier si `pyxel` est installé sur la machine sur laquelle vous travaillez.

Utiliser l'app en ligne Pyxel Studio : https://www.pyxelstudio.net/.

Utiliser le cahier numérique associé à ce tutoriel : https://cahiernum.net/YC7UF8

Copiez le lien indiqué en haut de la page du nouveau projet.
Sauvegardez le en lieu sur (clef USB + mail + agenda ....).
Il vous sera nécessaire pour retrouver votre travail et ne pas recommencer à zéro à chaque fois.

Étape 1 : Créer une application avec Pyxel

Un jeu vidéo peut se résumer de la façon suivante :

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 attend quelques millisecondes.

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
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(200, 160, 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.

```
1  import pyxel as P
2
3  P.init(200, 160) #les dimensions de la fenetre de jeu
4
5  x = 60
6  y = 60
7
8  def update():
9       if P.btnp(P.KEY_Q):
10              P.quit()
11
12 def draw():
13      P.cls(0) #efface la fenetre
14
15 P.run(update, draw)
```

A vous de jouer

Trouver dans la documentation de pyxel la fonction permettant de dessiner un rectangle :
https://github.com/kitao/pyxel/blob/main//docs/README.fr.md#documentation-de-lapi
Sur la ligne 14, dessinez un rectangle vert dont le coin gauche est positionné en `(x, y)` et de dimensions 8x8.

Dans ce premier exemple, le rectangle est immobile car la fonction `draw()` effectue toujours la même suite d'instructions.
La fonction `update`, permet d'écouter les interactions de l'utilisateur. Ici elle fermera l'application lorsque la touche `Q` sera appuyée.

Pour que le carré se déplace vers la droite, il suffit de modifier la fonction `update` :

```
def update() :
        global x, y
        x = x + 8
        if pyxel.btnp(pyxel.KEY_Q):
                pyxel.quit()
```

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 `x` : on écrira `global x`.

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 5 frames.

On ajoute une variable `frame = 5` à la racine du script. Puis on modifie la fonction `update()` :

```
def update() :
        global x, y, frame
        if P.frame_count % frame == 0 :
                x = x + 8
        if P.btnp(pyxel.KEY_Q):
                P.quit()
```

Pour que le carré reviennent à gauche de l'écran lorsqu'il en sort par la droite, il suffira d'utiliser le reste de la division euclidienne de `x` par la largeur de la fenêtre :

```
def update() :
        global x, y, frame
        if P.frame_count % frame == 0 :
                x = (x + 8)%200
        if P.btnp(P.KEY_Q):
                P.quit()
```

On peut également définir la direction à la racine du script : `direction = (1, 0)` et mettre à jour la valeur de `x` en fonction de cette direction :

```
def update() :
        global x, y, frame, direction
        if P.frame_count % frame == 0 :
                x = (x + direction[0]*8)%200
        if P.btnp(P.KEY_Q):
                P.quit()
```

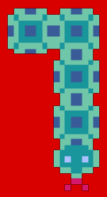
A vous de jouer

1. Lorsque la flèche gauche est activée, le carré doit changer de direction en `(-1, 0)`.
2. Le carré doit pouvoir se déplacer dans les quatres directions.
   S'il sort du cadre par le haut, il doit revenir par le bas et réciproquement.

le carré ne doit pas pouvoir passer d'une direction horizontale (resp verticale) à l'autre.
Il faudra passer par la direction intermédiaire verticale (resp. horizontale) avant

Un exemple de ce qu'y attendu est disponible sur studserver.

Étape 2 : Le serpent



Dans notre jeu, le serpent sera une suite de carrés de côtés 8 pixels.
On implémentera donc cet animal par une liste de couples, où chaque couple correspond au coin haut gauche du carré.
Par exemple on initialisera la variable `serpent` à la racine du programme par

```
serpent = [(32, 64), (40, 64), (48, 64), (56, 64), (64, 64)]
```

Le dernier élément correspond à la tête `(64, 64)`.

On écrira alors les instructions pour dessiner le serpent dans la fonction `draw` :

```
def draw ():
    P.cls (0)
    for anneau in serpent :
        x, y = anneau[0], anneau[1]
        P.rect(x, y, 8, 8, 11)
```

Évidemment, pour l'instant le serpent ne se déplace pas puisque l'on ne met pas à jour les coordonnées des anneaux.

A vous de jouer

Modifiez la fonction `update` pour que le serpent se déplace. Il suffit pour cela :

- d'ajouter à la liste serpent la nouvelle position de la tête en fin de liste ;
- de supprimer le premier élément de la liste.

Un exemple de ce qu'y attendu est disponible sur studserver

Étape 3 : La pomme

On veut placer une pomme au hasard dans la fenêtre.
On implémente cette pomme par la variable `pomme` initialisée à la racine du programme :

```
pomme = (randint(0, 23)*8, randint(0, 19)*8)
while pomme in serpent : #il ne faut pas que la pomme soit placee sur le serpent
    pomme = (randint(0, 23)*8, randint(0, 19)*8)
```

A vous de jouer

- Compléter la fonction `draw()` pour dessiner la pomme (un carré rouge) ;
- Lorsque la `pomme` est égale à la tête du serpent :
  - on replace la pomme au hasard dans la fenêtre ;
  - on ajoute un carré à la queue du serpent.

Il faut maintenant ajouter une variable `score`, initialisée à 0 à la racine du programme.

A vous de jouer

Explorez la document pour afficher la valeur de `score` dans la fenêtre de jeu.
Lorsque le serpent mange la pomme, il faut évidemment que le score augmente.

Un exemple de ce qu'y attendu est disponible sur studserver

Étape 4 : Un jeu est toujours plus intéressant lorsque qu'il se termine.

A vous de jouer

Pour corser un peu le jeu, on peut augmenter la vitesse de déplacement
- soit toutes les 1000 frames par exemple ;
- soit à chaque fois que le serpent a croqué 10 pommes ;
- soit les deux.

Pour augmenter la vitesse de déplacement du serpent, il suffit de diminuer la variable `frame`, en prenant garde qu'elle soit au minimum égale à 1.

On peut initialiser une variable `perdu = False` à la racine du programme.
- La mise à jour des variables ne sera effective que si la variable `perdu` vaut `False` ;
- si la tête du serpent appartient à la liste `serpent` (sans la tête bien sur), on peut considérer que le serpent se mort la queue. La variable `perdu` passe à `True` et le jeu s'arrête.

On peut également imaginer que le joueur perd si la variable `frame` est égale à 1.
Dans ce cas chaque pomme mangée permet de diminuer un peu la vitesse.
Vous l'avez compris toute variation des règles du jeu est intéressante. Soyez imaginatifs

Un exemple de ce qu'y attendu est disponible sur studserver

Étape 5 : Un peu de pixel art

Il ne nous reste plus qu'à afficher un vrai serpent et une vraie pomme à la place de nos carrés.

- Téléchargez l'environnement de jeu `snake.pyxres` prévu pour cet exemple sur studserver
- Si vous utilisez l'app Pyxel Studio :
  - Menu Burger : 
  - uploader le fichier snake.pyxres : 
  - Pour éditer le fichier : 

- Si vous utilisz un IDE avec Pyxel installé sur votre machine :
  Dans un terminal, la commande `pyxel edit snake.pyxres` permet d'ouvrir le fichier dans l'éditeur de ressources.

Pour utiliser cette ressource, il faut placer à la racine du code la ligne : `P.load('snake.pyxres')` juste après la ligne `P.init (200 , 160)`.

A vous de jouer

- Étudiez la documentation pour placer à la position `pomme[0], pomme[1]` la région dont le coin supérieur gauche est en `(0, 0)` dans la banque d'images `0` et de dimensions `8, 8`.
  Évidemment, on ne devra plus afficher le rectangle rouge pour la pomme.
- Modifier le code pour afficher le serpent.

Le dessin de la tête du serpent dépendra de sa direction !

Vous avez maintenant un jeu tout à fait fonctionnel.
Néanmoins rien ne vous interdit de faire preuve d'imagination pour l'améliorer :

- changer la couleur du serpent lorsqu'il croque une pomme ;
- ajouter des fruits qui permettent de diminuer la taille du serpent ;
- ajouter des fruits qui permettent de diminuer la vitesse ;
- ... Bref toute amélioration qui vous passe par la tête sera la bienvenue !

Ce tutoriel est inspiré de https://nuit-du-code.forge.apps.education.fr/DOCUMENTATION/PYTHON/TUTORIELS/autres-tutoriels/