Programmer un jeu vidéo avec Pyxel : 2/6

Doc officielle de Pyxel : https://github.com/kitao/pyxel

Nous repartirons dans ce tutoriel du script réalisé précédemment : $tuto\_pyxel\_1.py$
Faire Enregistrer Sous pour renommer à présent ce script en $tuto\_pyxel\_2.py$

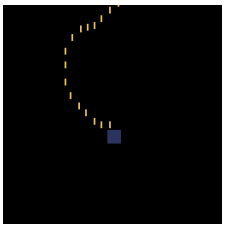
1. Pour démarrer…

Rappels vus lors du Serpent et du tutoriel précédent :

- Quelles sont les deux fonctions prédéfinies appelée automatiquement par Pyxel dans une boucle infinie ? ……
- Comment s'appellent les variables définies au niveau principal du script ? (en-dehors des fonctions) …… Quelle instruction doit-on écrire dans le corps d'une fonction pour que celle-ci ait le droit de modifier la valeur de ces variables ? …… Quelle fonction a le rôle de mettre à jour ces variables ? ……
- Comment savoir si la touche espace est appuyée ? ……
- Comment définir une liste ***tirs_liste*** vide ? ……
- Comment parcourir une liste en balayant chaque élément ? ……

2. Ajouter des tirs

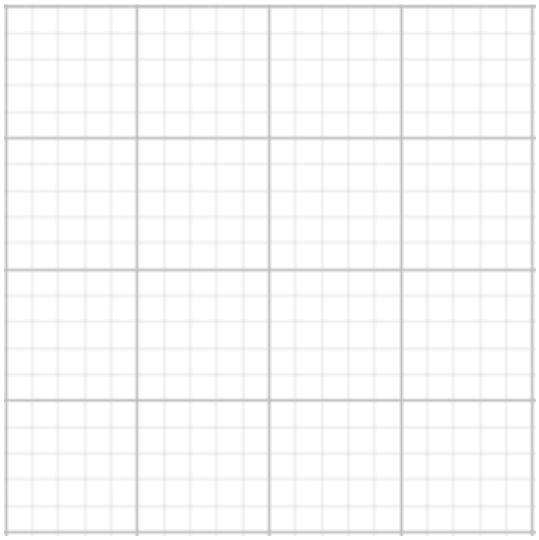
a. Créer la liste des tirs, et dessiner les tirs



Dans cette partie, nous allons représenter chaque tir par un rectangle jaune, de largeur 1 pixel et de hauteur 4 pixels. Au début du jeu, la liste des tirs sera vide. A chaque fois que l'utilisateur appuiera puis relâchera la touche Espace, cela créera un nouveau tir qui sera ajouté à la liste.

Concrètement, le rectangle sera stocké en mémoire par les coordonnées de son coin supérieur gauche.

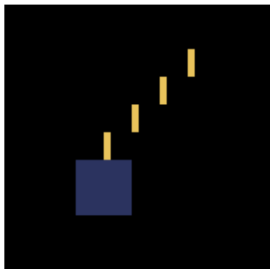
Exemple : dessiner sur la fenêtre ci-dessous (20x20 pixels) les tirs représentés en mémoire par $liste\_tirs = [[2,16],[6,12],[10,8],[14,4]]$



Pour les dessiner à l'ordinateur, on va balayer ***liste_tirs*** en appelant ***tir*** l'élément rencontré au fur et à mesure, et on utilisera ***pyxel.rect*** en lui passant en paramètres les coordonnées de ce tir.

```
for tir in tirs_liste:
    pyxel.rect(.......... , ........ , 1, 4, 10)
```

Dans quelle fonction se trouvera cette boucle dessinant les tirs ? ……



Jalon 1 : quand on lance le script, on doit voir le vaisseau au centre de l'écran avec une salve de tirs comme ci-dessus. Pour l'instant, ces tirs sont fixes.

b. Créer le tir à l'endroit (variable !) où se trouve vaisseau

Dans cette partie, nous allons compléter la fonction update() qui est appelée 30 fois par seconde par Pyxel au sein de la boucle infinie du jeu

```
def update():
    """mise à jour des variables (30 fois par seconde)"""

    global vaisseau_x, vaisseau_y
    # mise à jour de la position du vaisseau
    vaisseau_x, vaisseau_y = vaisseau_deplacement(vaisseau_x, vaisseau_y)

    # creation des tirs en fonction de la position du vaisseau
    tirs_creation(vaisseau_x, vaisseau_y, tirs_liste)
```

- A quoi sert la ligne ***global vaisseau_x, vaisseau_y*** ? ……
  ……

La variable ***tirs_liste*** n'a pas besoin d'être déclarée globale : en effet, c'est une liste, donc un objet MUTABLE, c'est-à-dire qu'il peut être modifié directement dans la mémoire. La dernière ligne appelle une fonction ***tirs_creation*** qui va effectuer cette modification dans la mémoire. Voici ci-dessous l'ébauche de son code :

```
def tirs_creation(x, y, tirs_liste):
    """création d'un tir avec la barre d'espace"""
    # btnr pour eviter les tirs multiples
    if pyxel.btnr(pyxel.KEY_SPACE):

        #modification en mémoire de tirs_liste (mutable)
        tirs_liste.append(..............)
```

Remarque : pour l'interaction utilisateur, on utilise ici ***pyxel.btnr*** qui teste si la touche vient d'être relâchée. Cela évite des tirs multiples au cas où l'utilisateur laisse trop longtemps la touche Espace appuyée.

La méthode ***append*** ajoute un élément à la fin de la liste. On veut ajouter un tir au « centre haut » du vaisseau, comme sur le dessin. Le vaisseau est supposé se trouver aux coordonnées $[x, y]$, qui correspondent à son coin supérieur gauche.

Jalon 2 : quand on lance le script, on doit voir le vaisseau au centre de l'écran. Quand on bouge le vaisseau et qu'on appuie sur Espace, un unique tir apparaît, qui se situe à son centre haut. Les tirs restent en place et ne bougent pas, même quand on déplace le vaisseau
Faire afficher avec print la variable ***liste_tirs*** à chaque fois qu'elle est modifiée dans ***update()***

c. Déplacement des tirs

On voudrait maintenant que les différents tirs se déplacent vers le haut au fur et à mesure, pour créer l'animation. Pour cela, on va compléter la fonction ***update***()

```
def update():
    """mise à jour des variables (30 fois par seconde)"""

    global vaisseau_x, vaisseau_y
    # mise à jour de la position du vaisseau
    vaisseau_x, vaisseau_y = vaisseau_deplacement(vaisseau_x, vaisseau_y)

    # creation des tirs en fonction de la position du vaisseau
    tirs_creation(vaisseau_x, vaisseau_y, tirs_liste)

    # mise a jour des positions des tirs
    tirs_deplacement(tirs_liste)
```

Voici une ébauche de la fonction ***tirs_deplacement*** qui est appelée dans ***update***()

```
def tirs_deplacement(tirs):
    """déplacement des tirs vers le haut
     et suppression s'ils sortent du cadre"""
    #on balaie la liste des tirs
    for ..... :
        #on diminue l'ordonnée d'un pixel pour monter
        tir[1] = .............
        #on regarde si le tir est sorti du cadre
        if  ..... :
            #dans ce cas, on le supprime de la liste
            tirs.remove(tir)
```

- Quel est le type du ***paramètre formel*** de cette fonction *(celui qui figure dans la parenthèse)* ? ……
- Lors de son utilisation, quel ***paramètre effectif*** lui sera passé ? *(celui avec lequel on fait réellement « fonctionner » la fonction lors de l'exécution du script*) ……
- L'élément courant de la liste ***tirs*** est la variable ***tir*** (au singulier) : quel est son type ? ……..
- Comment tester si le rectangle matérialisant le tir est entièrement sorti du cadre par le haut ? ……..
- Compléter les lignes du script, et intégrer ce code dans votre script de jeu.

Jalon final : tirs mobiles suivant le vaisseau dans les 4 directions.