Programmer un jeu vidéo avec Pyxel : 4/6

Ajouter des collisions

Nous repartirons dans ce tutoriel du script réalisé précédemment : $tuto\_pyxel\_3.py$
Faire Enregistrer Sous pour renommer à présent ce script en $tuto\_pyxel\_4.py$

1. Pour démarrer…

Rappels vus lors du tutoriel précédent :

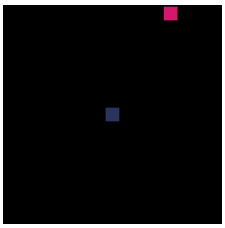
- Quel module Python contient les fonctions utilisant de l'aléatoire ? ……**random**……
- Qu'est-ce qui déclenchait l'ajout d'un ennemi dans la liste ? ……**30 frames écoulées**……
- Comment étaient déterminées les coordonnées du nouvel ennemi ? ……**y=0 et x aléatoire**……
- Comment est représenté un ennemi dans la mémoire ? ……**par coordonnées du coin sup gauche**
- Quelle instruction faisait se déplacer l'ennemi ? ……**ennemi[1] += 1**……..
- Quelles sont les dimensions du vaisseau ? …**8*8**……. D'un ennemi ? ……**8*8**……. D'un tir ? …**1*4**…

On voudrait maintenant qu'il se produise quelque chose quand un tir « rencontre » un ennemi, ou quand un ennemi « rencontre » le vaisseau.

Pour cela, on va créer deux nouvelles fonctions : $\boldsymbol{ennemis\_suppression}()$ et $\boldsymbol{vaisseau\_suppression}(\boldsymbol{vies})$ qui seront toutes les deux appelées par $\boldsymbol{update}()$ dans la boucle principale du jeu, et s'occuperont de détecter d'éventuelles collisions.

2. Ajouter des collisions

a. Collision ennemi / vaisseau



Les ennemis comme le vaisseau sont des rectangles stockés en mémoire par les coordonnées de leur coin supérieur gauche. On désigne ici les coordonnées d'un ennemi par (X, Y), et celles du vaisseau par (x, y)
Donner **4 conditions sur x et y où il ne peut PAS y avoir collision** avec l'ennemi :
(S'aider du schéma)

- Vaisseau trop à gauche : …**x+8 < X**……
- Vaisseau trop à droite : …… **x > X+8** …..
- Vaisseau trop haut : …… **y+8 < Y** …….
- Vaisseau trop bas : … **y > Y+8** …….

Finalement, quelle condition globale doit être vérifiée pour être sûr qu'il y A BIEN une collision :

… **x+8 >= X and x <= X+8 and y+8 >= Y and y <= Y+8** cad $x \in [X-8; X+8]$et $y \in [Y-8; Y+8]$ (ou le contraire en raisonnant sur X et Y), ou encore avec valeur absolue : $|x-X| \leq 8$ et $|y-Y| \leq 8$

vaisseau_x+8 >= ennemi[0] and vaisseau_x <= ennemi[0]+8 and vaisseau_y+8 >= ennemi[1] and vaisseau_y <= ennemi[1]+8

A transposer dans le programme en utilisant les variables qui sont réellement dans le programme pour désigner les coordonnées du vaisseau et de l'ennemi.

La variable $\boldsymbol{vies}$ est un entier qui désigne le nombre de vies du vaisseau. Elle est initialisée à 3 au niveau principal du programme.

```
def vaisseau_suppression(vies):
    """disparition du vaisseau et d'un ennemi si contact"""

    for ennemi in ennemis_liste:
        if ennemi[0] <= vaisseau_x+8 and ennemi[1] <= vaisseau_y+8 \
        and ennemi[0]+8 >= vaisseau_x and ennemi[1]+8 >= vaisseau_y:
            ennemis_liste.remove(ennemi)
            vies = vies-1

    return vies
```

Compléter le corps de la fonction ***vaisseau_suppression(vies)*** pour qu'en cas de collision, l'ennemi soit supprimé de la liste des ennemis. La fonction devra **renvoyer** (avec l'instruction ***return***) le nombre de vies diminué de 1.

Dans ***update()***, on mettra alors la variable ***vies*** à jour avec l'instruction suivante :

```
# suppression du vaisseau et ennemi si contact
vies = vaisseau_suppression(vies)
```

Et dans ***draw()***, on rajoutera un test pour savoir si le jeu continue. On effacera toujours l'écran, mais on ne dessinera le vaisseau etc. QUE s'il y a toujours au moins une vie. Dans le cas contraire, on dessinera le texte GAME OVER en utilisant la méthode ***pyxel.text***

```
def draw():
    """création des objets (30 fois par seconde)"""

    # vide la fenetre
    pyxel.cls(0)

    # si le vaisseau possede des vies le jeu continue
    if vies > 0:

        # vaisseau (carre 8x8)
        pyxel.rect(vaisseau_x, vaisseau_y, 8, 8, 1)

        # tirs
        for tir in tirs_liste:
            pyxel.rect(tir[0], tir[1], 1, 4, 10)

        # ennemis
        for ennemi in ennemis_liste:
            pyxel.rect(ennemi[0], ennemi[1], 8, 8, 8)
    # sinon: GAME OVER
    else:

        pyxel.text(50,64, 'GAME OVER', 7)
```

Jalon 1 : les collisions entre ennemis et vaisseau sont détectées.

b. Collisions ennemis / tirs

Refaire un dessin et raisonner de la même façon pour trouver une condition caractérisant une collision entre un tir et un ennemi.

```
def ennemis_suppression():
    """disparition d'un ennemi et d'un tir si contact"""

    for ennemi in ennemis_liste:
        for tir in tirs_liste:
            if ennemi[0] <= tir[0]+1 and ennemi[0]+8 >= tir[0]\
            and ennemi[1]+8 >= tir[1] and ennemi[1]<= tir[1]+4:

                ennemis_liste.remove(ennemi)
                tirs_liste.remove(tir)
```

A intégrer dans une fonction ***ennemis_suppression()*** qui sera appelée par ***update()***

dans update()

```
# suppression du vaisseau et ennemi si contact
vies = vaisseau_suppression(vies)

# suppression des ennemis et tirs si contact
ennemis_suppression()
```

Jalon final : on peut faire disparaître les ennemis en leur tirant dessus