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 :

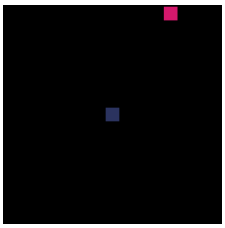
- Quel module Python contient les fonctions utilisant de l'aléatoire ? ……
- Qu'est-ce qui déclenchait l'ajout d'un ennemi dans la liste ? ……
- Comment étaient déterminées les coordonnées du nouvel ennemi ? ……
- Comment est représenté un ennemi dans la mémoire ? ……
- Quelle instruction faisait se déplacer l'ennemi ? ……
- Quelles sont les dimensions du vaisseau ? …… D'un ennemi ? …… D'un tir ? ……

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

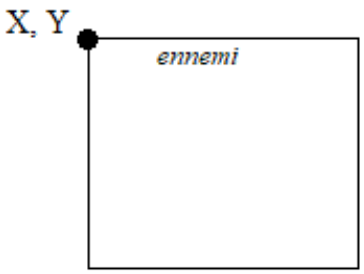


Les ennemis comme le vaisseau sont des rectangles stockés en mémoire par les coordonnées de leur coin supérieur gauche. On désigne ici les coordonnées d'un ennemi par (X, Y), et celles du vaisseau par (x, y)
Donner **4 conditions sur x et y où il ne peut PAS y avoir collision** avec l'ennemi :
(S'aider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

- Vaisseau trop à gauche : ……
- Vaisseau trop à droite : ……..
- Vaisseau trop haut : ……
- Vaisseau trop bas : ……

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

……



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.

Compléter le corps de la fonction $\boldsymbol{vaisseau\_suppression}(\boldsymbol{vies})$ 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 $\boldsymbol{update}()$, on mettra alors la variable $\boldsymbol{vies}$ à jour avec l'instruction suivante :

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

Et dans $\boldsymbol{draw}()$, 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***

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.

A intégrer dans une fonction $\boldsymbol{ennemis\_suppression}()$ qui sera appelée par $\boldsymbol{update}()$

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