Programmer un jeu vidéo avec Pyxel : 3/6

Ajouter des ennemis

Nous repartirons dans ce tutoriel du script réalisé précédemment : $tuto\_pyxel\_2.py$
Faire Enregistrer Sous pour renommer à présent ce script en $tuto\_pyxel\_3.py$

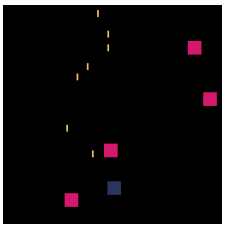
1. Pour démarrer…

Rappels vus lors du tutoriel précédent :

- Comment définir une liste $\boldsymbol{tirs\_liste}$ vide ? …… **tirs_liste = []**……
- Comment parcourir une liste en balayant chaque élément ? …… **for element in liste :**…….
- Par quoi était représenté un tir dans la mémoire de l'ordinateur ? ……**par une liste [x,y] représentant les coordonnées du coin supérieur gauche**……
- Comment ajouter un élément à la fin de $\boldsymbol{tirs\_liste}$? ……**tirs_list.append([x,y])**……
- Qu'est-ce qui déclenchait l'ajout d'un tir dans la liste ? **appui et relâchement sur Espace**……..
- Comment étaient déterminées les coordonnées du nouveau tir ? ……**avec les coordonnées du vaisseau : vaisseau_x+4, vaisseau_y-4 (pour être bien centré en haut du vaisseau)**…….
- Quelle instruction faisait se déplacer le tir ? …… **tir[1] - = 1**……..
- Quel module Python contient les fonctions utilisant de l'aléatoire ? ……**random**……

Cette étape ressemblera beaucoup à la précédente, avec deux différences cependant : les ennemis apparaîtront à un **endroit aléatoire**, et à **intervalles de temps régulier** (sans action de l'utilisateur)

2. Ajouter des ennemis



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

Chaque ennemi sera représenté par un carré rose de côté 8 pixels. Au début du jeu, la liste des ennemis sera vide. A intervalles de temps régulier, un nouvel ennemi sera généré et ajouté à la liste. Comme pour les tirs, le carré sera stocké en mémoire par les coordonnées de son coin supérieur gauche.

Dessin

Dans cette partie, on suppose que la liste des ennemis est déjà créée.

Pour les dessiner à l'écran, on va balayer $\boldsymbol{liste\_ennemis}$ en appelant $\boldsymbol{ennemi}$ l'élément rencontré au fur et à mesure, et on utilisera $\boldsymbol{pyxel.rect}$ en lui passant les coordonnées de cet ennemi
Dans quelle fonction se trouvera cette boucle dessinant les tirs ? ……**draw()**…….
Ecrire ci-contre le corps de cette boucle :

```
# ennemis
for ennemi in ennemis_liste:
    pyxel.rect(ennemi[0], ennemi[1], 8, 8, 8)
```

Création de la liste d'ennemis

Dans le cas des ennemis, on veut les faire apparaître **en haut** de l'écran : comment traduire cela sur les coordonnées de l'ennemi ? ……**y=0**……

On veut aussi que l'ennemi puisse apparaitre n'importe où en haut de l'écran. On va donc choisir une valeur aléatoire (entière) pour *x* qui soit comprise entre …**0**… et ……**120**…….. Pour cela, on utilisera la fonction Python $\boldsymbol{randint(a, b)}$ qui renvoie un nombre entier aléatoire entre a et b inclus. Cette fonction se trouve dans le module random, qu'il faut importer tout comme pyxel.

Instruction qui ajoute un ennemi à la liste : **ennemis_liste.append([random.randint(0, 120), 0])**

On veut qu'un nouvel ennemi apparaisse chaque seconde. Rappelons que la fonction $\boldsymbol{update}()$ est appelée 30 fois par seconde, et chacun de ces appels constitue ce qu'on appelle un « frame ». Pyxel donne accès au nombre de frames écoulé depuis le début du jeu grâce à la variable $\boldsymbol{pyxel.frame\_count}$

Au bout de combien de frames doit-on créer le 1er ennemi ? …**30**… Le 2nd ? ……**60**.. Le 3ème ……**90**
Il s'est écoulé 1680 frames : doit-on créer un ennemi ? …**oui car 30*56**…….. Même question pour 3745 frames ……**non car pas multiple de 30**…….
Comment tester dans le cas général s'il est temps de créer un nouvel ennemi ? ……**on regarde si nb_frames modulo 30 ==0**…….

Compléter alors la fonction ci-dessous qui sera appelée par $\boldsymbol{update}()$

```
def ennemis_creation(ennemis_liste):
    """création aléatoire des ennemis"""

    # un ennemi par seconde
    if (pyxel.frame_count % 30 == 0):
        ennemis_liste.append([random.randint(0, 120), 0])
```

Jalon 1 : quand on lance le script, on doit voir le vaisseau au centre de l'écran avec des ennemis qui apparaissent chaque seconde en haut. Pour l'instant, ces ennemis sont fixes.

b. Déplacement des ennemis

On voudrait maintenant que les différents ennemis se déplacent vers le bas au fur et à mesure, pour créer l'animation. Pour cela, on va compléter la fonction $\boldsymbol{update}()$ en rajoutant un appel à une fonction $\boldsymbol{ennemis\_deplacement}(\boldsymbol{ennemis\_liste})$

Voici une ébauche cette fonction $\boldsymbol{ennemis\_deplacement}$

```
def ennemis_deplacement(ennemis):
    """déplacement des ennemis vers le bas
   et suppression s'ils sortent du cadre"""
    for ennemi in ennemis:
        ennemi[1] += 1
        if  ennemi[1]>128:
            ennemis.remove(ennemi)
```

- Quel est le type du ***paramètre formel*** de cette fonction *(celui qui figure dans la parenthèse)* ? ……**une liste (contenant elle-même des listes à 2 éléments)**…….
- 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)* ……**liste_ennemis**…
- L'élément courant de la liste $\boldsymbol{ennemis}$ est la variable $\boldsymbol{ennemi}$ (au singulier) : quel est son type ? ……..**une liste [x,y] des coordonnées du CSG de l'ennemi**……..
- Comment tester si le carré matérialisant l'ennemi est entièrement sorti du cadre par le bas ? ……**on regarde si l'ordonnée est > 128**……..
- Compléter les lignes du script, et intégrer ce code dans votre script de jeu.

Jalon final : Les ennemis apparaissent et se déplacent. Pour le moment, les tirs n'ont pas d'effet…