TP INFORMATIQUE

Création d'un petit jeu

Le jeu que vous allez créer consiste à éviter avec un vaisseau des astéroïdes en mouvement.
Le score correspond au nombre d'astéroïdes évités ; chaque collision remet le score à zéro !

Préambule technique

L'interface du jeu nécessite le module `pygame`. Au lycée il devrait être déjà installé. S'il ne l'est pas (ou si vous voulez tester chez vous), après avoir lancé Pyzo, il suffit de taper dans le shell la commande `pip install pygame`. Une fois validée, l'installation devrait se lancer.

Par ailleurs, ce tp utilise deux scripts python et deux images 'png'. Vous devez placer les fichiers correspondants dans un *même dossier* :

- Vous pouvez ouvrir les images `ufo.png` et `stone.png`. Ce sont tout simplement celles des objets du jeu.
- Le fichier `game.py` est un fichier utilitaire qui gère la partie graphique et les réponses aux commandes. Vous pouvez l'ouvrir par curiosité mais vous n'êtes pas censé comprendre ce qu'il contient et **vous n'avez pas à le modifier**.
- Le fichier `tp_jeu.py` est celui que vous devez ouvrir dans Pyzo et **est le seul que vous modifierez.**

Premier lancement

> AVERTISSEMENT
> Pour exécuter le fichier `tp_jeu.py` sous Pyzo, vous devrez cliquer sur `Run/Run file as script` ou, alternativement, vous pouvez appuyer sur `Ctrl+F5`.

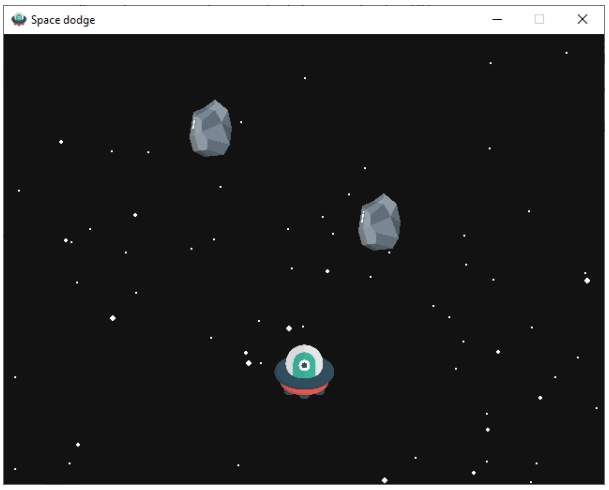
Lorsque vous lancez le script pour la première fois une petite fenêtre doit s'afficher. Elle comporte notamment deux astéroïdes et une soucoupe volante.

Rem : les étoiles sont générées aléatoirement donc pourront être placées différemment, cela n'importe pas.

Pour l'instant vous pouvez uniquement déplacer la soucoupe vers la droite avec la flèche droite (à côté du pavé numérique).

Vous pouvez quitter la fenêtre en appuyant sur Echap (ou en cliquant sur la croix en haut à droite...).

Si vous avez cette fenêtre, tout a l'air de fonctionner correctement. Dans le cas contraire, demandez de l'aide au professeur.



La première fenêtre

Les objets du jeu sont repérés par leurs coordonnées dans la fenêtre de jeu.
(par exemple dans le script vous pouvez trouver les coordonnées `(ship_x, ship_y)` du vaisseau)

L'origine du repère est le pixel tout en haut à gauche de la fenêtre.
Le pixel en bas à droite est de coordonnées $(640, 480)$.
NB. Les lignes (ordonnée) sont numérotées de haut en bas.

↪ Vous pouvez tester de modifier les coordonnées de départ du vaisseau.

Développement du jeu

Dans cette section vous allez écrire les fonctions python requises pour animer un peu nos objets.

Pour expliquer le principe, le jeu est actualisé un certain nombre de fois par secondes (30 fois ici). À chacune des ces actualisations sont mises à jour : les positions des différents objets du jeu (partie « logique ») et l'affichage ensuite (partie « graphique »).

Ici je vous propose d'intervenir sur la partie logique.

Déplacement de la soucoupe volante

1. Initialement vous avez une fonction de déplacement à droite déjà écrite.

↪ Écrire similairement les fonctions `deplacement_gauche`, `deplacement_haut`, `deplacement_bas`.

Rappelez-vous la numérotation des lignes de haut en bas...

Ces fonctions sont "rattachées" aux flèches directionnelles du clavier.

NB. Respectez précisément ces noms (sans accent) sinon cela ne fonctionnera pas.

2. Tel quel le vaisseau peut sortir de la fenêtre de jeu, ce faciliterait un peu trop l'évitement des astéroïdes.

↪ Modifiez les fonctions de déplacement de sorte qu'elles retournent la position actuelle si la nouvelle position est hors de la fenêtre (pour rappel les colonnes sont numérotées de 0 à 639 ; les lignes de 0 à 479)

Test. Vérifiez que le vaisseau est bloqué sur les quatre bords.

Plus précisément le vaisseau peut déborder hors de la fenêtre mais son centre doit rester dans la fenêtre.

Collision avec les astéroïdes

Pour l'instant le vaisseau peut traverser librement les astéroïdes. On va mettre en place une détection de collision.

On choisit de considérer le vaisseau et les astéroïdes comme des cercles. Une collision se produira lorsque les centres respectifs sont trop proches (ici $<$60 px).

1. ↪ Compléter la fonction auxilliaire `distance(x,y,a,b)` qui retourne la distance entre les points de coordonnées (x, y) et (a, b).

Vous pourrez avoir besoin d'importer la fonction `sqrt` du module `math`.

2. Les astéroïdes sont modélisés par la *liste* de leurs coordonnées (voir l'exemple dans le script).

↪ Compléter la fonction `collision` qui retourne un booléen déterminant si le vaisseau est à distance < 60 d'au moins un des astéroïdes.

(attention votre fonction doit s'adapter à toute liste, le nombre d'astéroïdes est amené à changer)

Test. Une fois la fonction collision complétée, le mot "Collision" doit s'afficher en bas de la fenêtre de jeu si le vaisseau rencontre un astéroïde et le vaisseau retourne au départ.

Cette détection de collision est assez simpliste (on est loin du "pixel perfect") mais ça fait un bon début.

Déplacement et création des astéroïdes

Pour l'instant, les astéroïdes ne bougent pas et il n'y en a que deux. On peut faire plus intéressant...

1. On va pour l'instant déplacer tous les astéroïdes vers le bas de 7 pixels.

↪ Compléter la fonction `deplacement_asteroides` qui retourne la liste des nouvelles positions des astéroides (chaque couple (a, b) devient $(a, b + 7)$).

Une fois cette fonction définie, les astéroïdes doivent se mettre à bouger.

2. Tel quel on continue à stocker les nouvelles positions des astéroïdes même s'ils sont sortis de la fenêtre de jeu. Remédions à ceci.

↪ Modifier `deplacement_asteroides` de sorte qu'on n'ajoute dans la liste `new_asteroides` une nouvelle position $(a, b + 7)$ que si $b < 500$.

3. Il s'agit maintenant de renouveler les astéroïdes. On veut faire en sorte d'avoir toujours 10 astéroïdes.

↪ Compléter la fonction `ajout_asteroides` afin qu'elle adjoigne à la liste `asteroides` un nouveau couple (cf ci-dessous) tant que la liste est de longueur < 10.

> Les nouveaux couples seront des couples aléatoires (a, b) avec
> $$0 \leq a \leq 640 \text{ et } -150 \leq b \leq 0$$
> (les astéroïdes sont donc créés "au dessus" de la fenêtre de jeu).

On utilisera la fonction `randint(min,max)` *du module random qui donne un nombre aléatoire entre* `min` *et* `max`

Attention : cette fonction modifiera directement la liste `asteroides` sans rien retourner (fonction à effet de bord).

Score et autres améliorations

1. Le module `game` comporte deux fonctions (déjà écrites) `reset_score()` et `up_score()` (sans paramètre) qui, respectivement, réinitialise le score et augmente le score d'un point.

↪ Modifier judicieusement certaines des fonctions que vous avez écrites pour actualiser le score à chaque disparition d'un astéroide ou à chaque collision (rappelez-vous comment utiliser des fonctions d'un autre module)

Une fois ces fonctions placées, le score s'affichera sur la fenêtre de jeu (à partir du premier point ou de la première collision).

2. Par la magie de l'importation, la variable `game.score` est accessible depuis votre script.

↪ Modifiez votre script pour que le nombre d'astéroïdes (et/ou leur vitesse, à votre guise) augmente en fonction du score.

(par exemple, 2 astéroïdes qui se déplacent de 2px au début et tous les 5 points, le nombre d'astéroïdes augmente d'un et leur vitesse de 1px ; ou toute autre).

3. A vous d'améliorer encore votre jeu. Quelques pistes accessibles :

- Optimiser le placement des astéroïdes générés. Pour l'instant ils peuvent se superposer...
- Au lieu d'un déplacement fixe pour les astéroïdes, avoir une part d'aléatoire.
- Plus compliqué : faire aussi apparaître des astéroïdes en bas (qui devront remonter).
  Attention : la fonction `deplacement_asteroides` *doit toujours ne retourner qu'une seule liste et la fonction* `ajout_asteroides` *modifiera toujours la liste* `asteroides`.