Projet 2 : Asteroids

1 Objectifs

Il s'agit de réaliser une implémentation de base du jeu Asteroids `https://fr.wikipedia.org/wiki/Asteroids` dans le but d'illustrer les idées de la programmation objet vues en cours et exercices

On peut voir sur Youtube une courte vidéo du jeu de l'époque (1979)
`https://www.youtube.com/watch?v=i-Gs01omJyI`

Il existe différents GUI (Graphical User Interface) permettant d'implémenter ce jeu en Python :

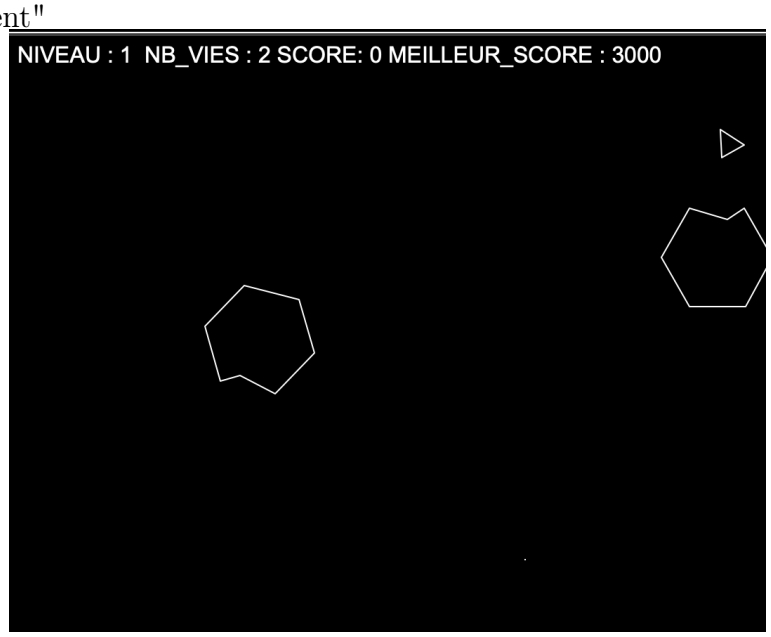
1. Le module `tkinter`
2. Le module `pygame`
3. Un module plus récent `arcade`

On a choisi d'utiliser le module `turtle` qui utilise le module `tkinter`, dans un but de simplification

2 Règles du jeu

Une version simplifiée

Au début du jeu un vaisseau spatial de forme triangulaire apparaît au centre de l'écran, immobile ainsi que trois "gros" astéroïdes se déplaçant en ligne droite "lentement"



Les astéroïdes sont de trois tailles : gros, moyen et petit

Le joueur peut faire tourner le vaisseau sur lui même avec les touches 'Right' et 'Left', le faire accélérer avec la touche 'Up'

Si un astéroïde de n'importe quelle taille heurte le vaisseau le joueur perd une vie et le joueur n'a que trois vies

Si le joueur tire des missiles sur un gros astéroïde et si ce dernier est touché alors il est cassé en deux astéroïdes moyens plus rapides qu'un gros astéroïde

De même si une missile touche un astéroïde moyen par contre si une missile touche un petit astéroïde alors ce dernier disparait et le joueur gagne 100 points

Le joueur change de niveau lorsqu'il n'y a plus d'astéroïdes

D'un niveau au suivant le nombre de gros astéroïdes à l'état initial augmente d'une unité

Le score du joueur ainsi que son niveau sont affichés à l'écran

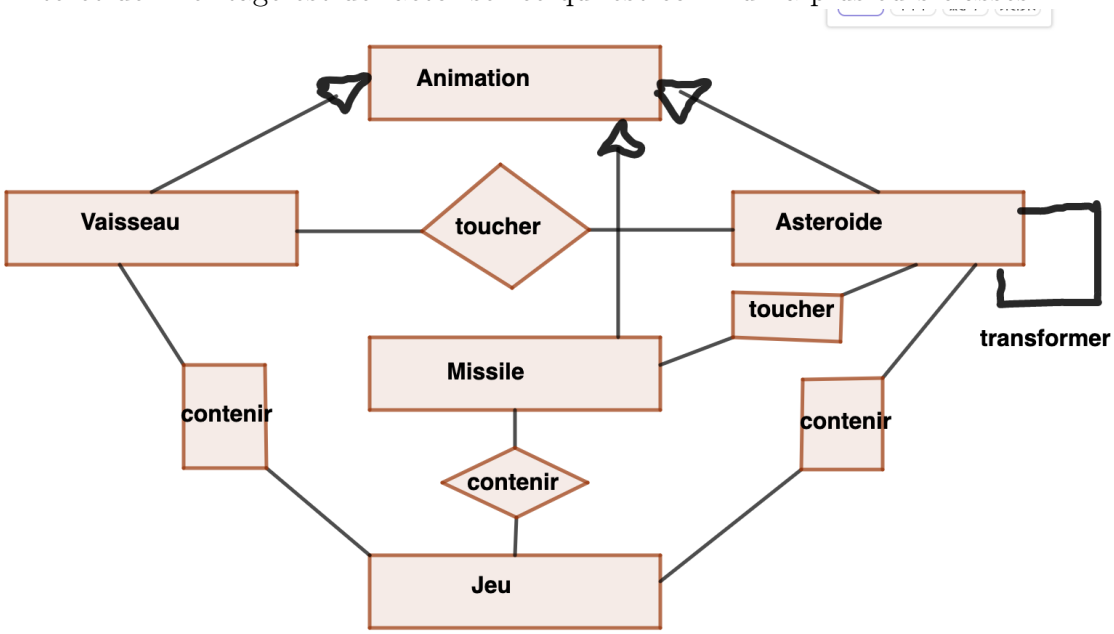
A la fin d'une partie lorsque le joueur n'a plus de vies on lui propose d'enregistrer son score si ce dernier fait partie des 10 derniers meilleurs scores

3 Modélisation

A la lecture des règles du Jeu on isole les classes suivantes :

1. Les classes Vaisseau, Asteroide et Missile qui **héritent** d'une classe Animation qui hérite de la classe Turtle du module turtle
2. Une classe Jeu

L'intérêt de l'héritage est de factoriser ce qui est commun à plusieurs classes



4 Mise au point

4.1 Matériel

Récupérer le dossier compressé projet_asteroids contenant :

1. le squelette asteroids_squelette.py
2. deux fichiers tir.wav et explosion.wav

Renommer le dossier avec le nom de famille des élèves du groupe
On va compléter le fichier .py

4.2 Boucle principale du jeu

La partie principale consiste à instancier la classe Jeu et d'appliquer sur l'objet ainsi créé la méthode `boucle_principale()`

```
#------------MAIN------------
if __name__ == "__main__":
    jeu = Jeu()
    jeu.boucle_principale()
```

Pour la plupart des jeux la méthode `boucle_principale()` est définie par :

1. Une boucle "infinie" qui est interrompue si il n'y a plus de joueur voulant jouer une partie

   ```
   while True
   ```

2. Toute partie commence par un écran d'accueil que l'on quitte en appuyant sur la touche "Entrée". Sur cet écran peut se trouver des informations comme le score le plus élevé du jeu pour l'instant
3. Ensuite il y a une boucle pour une partie qui se termine lorsque le joueur n'a plus de vies
4. A la sortie de cette boucle il y a un bilan de la partie qui est fait et on propose de jouer une nouvelle partie ou d'arrêter le jeu

Ce qui donne :

```
def boucle_principale(self):
    while True:
        #écran d'accueil
        while not self.commencer_une_partie:
            self.ecran_accueil()
        #une partie en cours
        while self.partie_en_cours():
            turtle.update()
            self.jouer()
            self.afficher()
        #on efface les dessins des tortues
        self.ecran.getcanvas().delete('all')
        #bilan de la partie
        self.bilan_partie()
        #arrêt du jeu ou nouvelle partie pour un nouveau joueur
        choix = self.ecran.textinput("Asteroids", "Une nouvelle part
        if choix == 'N':
            break
        elif choix == 'O':
            #on retourne vers l'écran d'accueil pour une nouvelle
            #partie
            self.commencer_une_partie = False
            self.vaisseau.nb_vies = 3
```

```
        turtle.onkeypress(self.commencer, "Return")
        turtle.listen()
```

Il vous faut compléter toutes les fonctions pour que le jeu fonctionne mais la structure générale du jeu est faite

A vous de faire : Exécutez le fichier dés à présent et vous observerez un écran noir avec un message

Si vous appuyez sur Entrée une fenêtre apparaît et remplissez la zone de texte avec N

La suite du projet consiste à définir principalement la méthode `jouer()` et par conséquence les méthodes et attributs des classes `Vaisseau`, `Asteroide` et `Missile`

4.3 Compréhension des animations

Le module `tkinter` est une interface graphique permettant de programmer des jeux en Python

Le module `turtle` utilise le module `tkinter` :

En effet, lorsqu'on fait tracer avec le module `turtle` un carré on observe le **déplacement de la tortue à l'écran** :

Régulièrement, toutes les T ms, la forme de la tortue, en général une flèche, est affichée en un certain endroit de l'écran puis effacée puis réaffichée ailleurs, ce qui donne cette illusion de déplacement (comme les dessins animés)

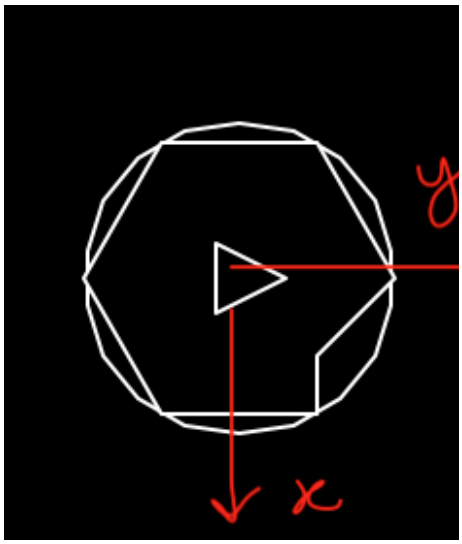
On va utiliser cette technique pour donner l'illusion du déplacement du vaisseau, des astéroïdes et des missiles, car la tortue a des formes prédéfinies (arrow, circle, triangle,

turtle) mais on peut créer d'autres formes à condition de les enregistrer.

Ainsi pour représenter le vaisseau et les missiles on va plutôt utiliser la forme triangle, par contre on va créer une forme pour représenter un astéroïde qui sera défini par le tuple

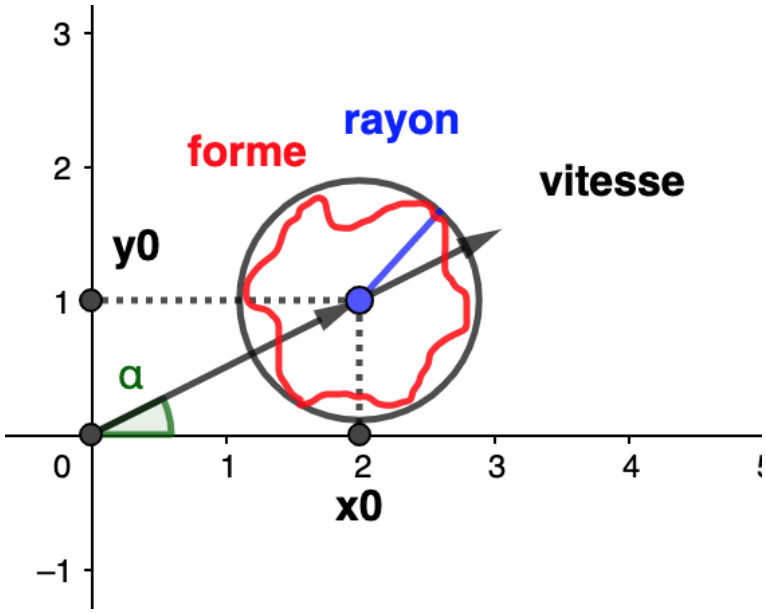
asteroide =((-34.6,20), (-34.6,-20), (0,-40),(34.6,-20),(34.6,20),(20,20),(0,40))

A l'écran lorsque la tortue "regarde vers l'est" (direction = 0) voici la forme de l'astéroïde circonscrite dans un cercle de rayon 40 pixels



Pour généraliser on va créer une classe Animation qui hérite de Turtle ainsi un objet de type Animation a tous les attributs d'un objet de type Turtle ce qui nous permet de le repérer (position et direction) et le faire avancer mais pour gérer les interactions entre les animations on ajoute les attributs

1. `rayon` :le rayon du cercle qui circonscrit la forme de l'animation
2. `vitesse` : l'animation va avancer **dans sa direction** α tous les T ms de la quantité de pixels `vitesse`



4.4 Asteroide sans mouvement

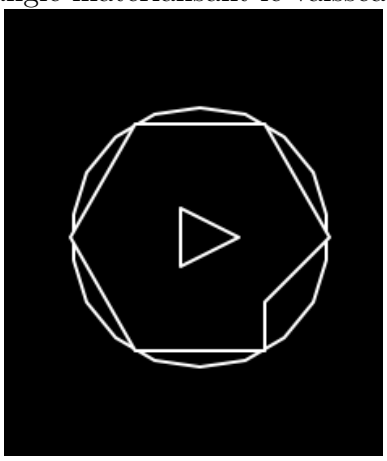
A vous de faire :

Dans le constructeur de la classe `Jouer`

1. Implémenter la méthode `partie_est_en_cours()` (une partie est en cours lorsque l'attribut `nb_vies` de l'attribut `vaisseau` de la classe `Jouer` est différent de 0)
2. Ajouter l' attribut `niveau` initialisé à 1 à la classe `Vaisseau`
3. **Pour quitter le jeu tant qu'on est en train de mettre au point le jeu on va exceptionnellement implémenter la méthode `tourner_a_gauche()` de la classe `Vaisseau` par l'instruction `self.nb_vies -= 1` de telle sorte qu'à chaque fois qu'on appuie sur la touche "Left" le nombre de vies diminue de 1 et finisse par être égal à 0 ce qui arrête la partie en cours proprement**
4. En dessous du commentaire #`un attribut asteroides de type list` dans la méthode `commencer` de la de la classe `Jouer` insérer

```
a = Asteroide(1,0,0,1)
t1 = turtle.Turtle()
t1.color("white")
t1.penup()
t1.ht()
t1.goto(0,-40)
t1.pendown()
t1.circle(40)
```

Vous devez observez un "gros" astéroide circonscrit par un cercle de rayon 40 et un triangle matérialisant le vaisseau



A vous de faire :

1. Circonscrire un 'moyen' asteroïde de taille 0.5 par un cercle de centre (0,0) et de rayon 20
2. Circonscrire un 'petit' asteroïde de taille 0.25 par un cercle de centre (0,0) et de rayon 10

Observer que chaque objet a une orientation différente

4.5 Vaisseau sans mouvement

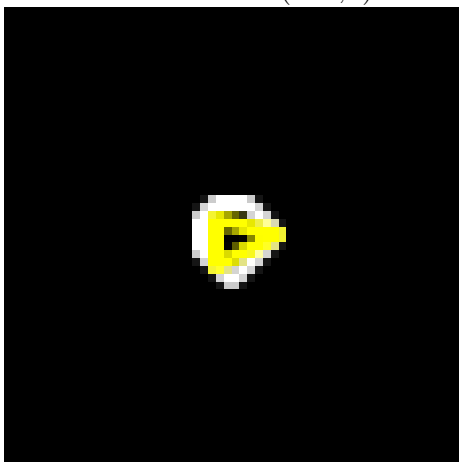
A vous de faire :

1. Désactiver l'affichage d'un astéroide
2. Circonscrire le vaisseau par un cercle de centre (0,0) et de rayon 12

4.6 Missile sans mouvement

A vous de faire :

Circonscrire une missile placée en (100,0) (pour ne pas être cachée par le vaisseau) par un cercle de centre (100,0) et de rayon 2.5



Ce n'est pas terrible ! mais on ne verra rien dans le mouvement !

4.7 Asteroide avec mouvement

A vous de faire :

1. En dessous du commentaire #`un attribut asteroides de type list` dans la méthode `commencer` de la de la classe `Jouer` insérer un attribut **momentané**
   `self.asteroide = Asteroide(1,0,0,1)`
   (Effacer toutes les instructions pour circonscrire les animations)
2. En dessous du commentaire #faire bouger les animations dans la méthode `jouer()` de la Classe `Jouer` écrire
   `self.asteroide.avancer()`
3. Exécuter et observer. Etes vous satisfait du déplacement de l'astéroïde, cela correspond-il à un déplacement sur un tore ?
   `https://en.wikipedia.org/wiki/Wraparound_(video_games)`

4. Puisque la classe `Asteroide` hérite de la classe `Turtle` on peut dans le constructeur de la classe `Asteroide` régler la direction de l'astéroïde aléatoirement entre 0 et 360 degrés.

   Tester

5. Initialiser définitivement l'attribut `self.asteroides` par une liste **en compréhension** de 3 objets de type `Asteroide` placé aléatoirement dans l'écran entre -300 et 300 sur les x et les y

6. En dessous du commentaire #faire bouger les animations dans la méthode `jouer()` de la Classe `Jouer` modifier

   `self.asteroide.avancer()`

   (En français pour chaque astéroide de la liste asteroides faire avancer....)

4.8 Contact entre animations

A vous de faire :

1. Implémenter la méthode `toucher()` de la classe Animation qui retourne vrai si deux animations entre en contact au niveau de leur cercle circonscrit
2. Faire avancer le vaisseau ainsi que les trois astéroïdes et **dès que le vaisseau est touché par un astéroïde mettre le nombre de vies du vaisseau à 0 ce qui arrêtera la partie**
3. Si un astéroide touche le vaisseau ce dernier est dévié de sa trajectoire en prenant la direction de l'asteroide

   Mettre au point la méthode `devier(direction)` de la classe `Vaisseau`

4.9 Explosion des astéroides

Il s'agit maintenant de mettre au point la méthode `exploser()` de la classe `Asteroide`

Un objet de type `Jeu` a un attribut `asteroides` qui est une liste d'objets de type `Asteroide`

A vous de faire :

1. Implémenter la méthode `exploser(asteroides,vaisseau)` de la classe `Asteroide` en tenant compte de la taille de l'astéroïde

   (a) Si l'astéroide est de taille 1 ou 0.5 , l'objet en question (self) est réduit de moitié et "à côté de lui" (à vous de voir) apparaît un autre astéroide qu'il faudra ajouter à la liste `self.asteroides` de la classe `Jeu`. En plus il faudra augmenter le score du vaisseau de 50

   (b) Si l'astéroïde est de taille 0.25 il disparaît dans le trou noir de la galaxie défini à l'extérieur de l'écran par `TROU_NOIR = (500,500)` et il faut l'enlever de la liste `self.asteroides` En plus il faudra augmenter le score du vaisseau de 100

4.10 Gestion des évènements

Maintenant on va faire en sorte que le vaisseau

1. tourne à droite si on appuie sur la touche "Right "du clavier
2. accélère si on appuie sur la toupe "Up"

Pour l'instant on laisse la touche "Left" pour quitter proprement le Jeu

A vous de faire :

Pour cela implémenter la méthode `tourner_a_droite()` en utilisant la méthode `right()` de la classe `Turtle` et l'attribut de classe `angle` et implémenter la méthode `accelerer()`

Ne pas oublier de mettre à jour les écouteurs dans la méthode `commencer()` de la classe `Jeu` (procéder par **analogie** avec l'écouteur déjà en place)

4.11 Tir des missiles

A vous de faire :

1. Mettre au point la méthode `tirer()` pour la classe `Vaisseau` : à chaque fois que le joueur appuie sur la touche "Espace" un objet de type `Missile` est créé et ajouté à l'attribut `tirs` de type `list` de la classe `Vaisseau`
2. Utiliser `os.system()` pour jouer le son du fichier "tir.wav"
3. Tenir compte du système d'exploitation du joueur pour activer la bonne commande `os.system()`
4. Dans la méthode `jouer()` de la classe `Jeu` compléter la partie en dessous du commentaire # tenir compte des interactions entre les animations
5. Utiliser le fichier son "explosion.wav"
6. Attention ! un missile ne peut pas détruire un astéroïde **en dehors de l'écran**

4.12 Changer de niveau

Lorsque le joueur a détruit tous les astéroïdes il passe au niveau supérieur.

Au niveau supérieur le nombre d'astéroïdes augmente d'une unité (on commence avec 3 astéroïdes au niveau 1)

A vous de faire :

1. Mettre au point la méthode `passer_au_niveau_superieur(niveau)` de la classe `Jeu`
2. Utiliser cette méthode dans la méthode `jouer()` de la classe `Jeu`

4.13 Bilan d'une partie

Si le score du joueur est strictement plus grand que le dernier des 10 meilleurs scores enregistrés dans le fichier scores.txt alors le score et le nom du joueur sont enregistrés dans le fichier scores.txt

A vous de faire :

1. Compléter la méthode `bilan_partie()` de la classe `Jeu`
   Ecrire dans le fichier scores.txt le nom et le score du joueur si c'est nécessaire
2. Ici se termine la partie commune du projet

5 Créativité

Aller chercher sur le Web ce que faisait à l'origine les soucoupes volantes dans le Jeu pour inspirer votre propre scénario

Ensuite vous devez écrire une classe `Soucoupe` implémentant votre scénario et en vous inspirant de tout ce qui a été fait précédemment (Ne pas oublier d'ajouter un fichier .wav associé au déplacement de la soucoupe volante)