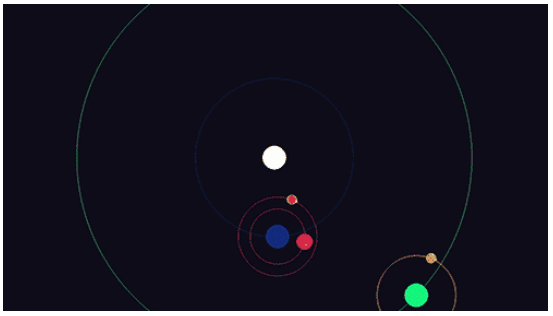
Simulation d'un système stellaire



Cette activité nécéssite le module `pygame`, l 'installer si nécéssaire:

- `pip install pygame` ou `python -m pip install pygame`
- Vérifier l'installation en important le module : `import pygame`

Les fichiers à compléter sont dans le même répertoire que ce document.

On modélise un corps céleste par un objet (rond pour simplifier) en orbite circulaire autour d'un point (pour simplifier encore). Cet objet contient, entre autres, les informations suivantes :

- `taille` : un nombre qui représente le rayon du corps céleste ;
- `rayon_orbite` : un nombre qui représente le rayon de son orbite circulaire ;
- `centre_orbite` : un tuple ou une liste contenant deux valeurs : l'abscisse et l'ordonnée de son centre ;
- `v_angulaire` : un nombre qui représente sa vitesse angulaire ;
- `couleur` : un tuple ou une liste contenant trois entiers compris entre 0 et 255 (RGB);
- ...

Le fichier `corpsceleste.py` contient la classe `CorpsCeleste` et le fichier `stellar_system.py` contient le programme principal de la simulation spatiale. Le fichier `vaisseau.py` contient la classe `Vaisseau` qui sera utile plus tard.

Pour créer et visualiser un système stellaire, il faut compléter la fonction `main()` de `stellar_system.py` : instancier un objet ou plusieurs objets de la classe `CorpsCeleste`, et placer chacun des corps célestes ainsi créés dans la liste `univers`.

Il suffit ensuite d'éxécuter le fichier `stellar_system.py`

PARTIE A : Créer des planètes

1. Ajouter une planète de couleur verte à l'univers, en orbite autour du coin en haut à gauche de la fenêtre.

Remarque : Le centre du repère est le coin en haut à gauche de la fenêtre et l'axe des ordonnées est orienté "vers le bas".

Réponse:

-
-

2. Ajouter une planète de couleur rouge à l'univers, en orbite autour du centre de la fenêtre. Cette planète sera 2 fois plus grosse que la verte, mais tournera 2 fois plus lentement.

Réponse:

-
-

3. Ajouter une planète de couleur fuschia à l'univers. Elle sera en orbite autour du coin en haut à droite de la fenêtre ; ajuster sa vitesse et sa phase de sorte que les planètes verte et fuschia soient visibles en même temps.

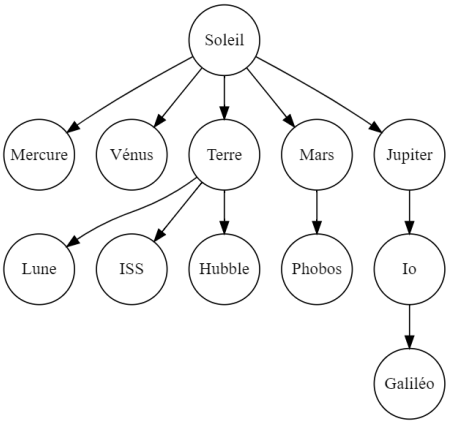
Réponse:

-
-

PARTIE B : Créer des satellites

Un exemple :

- Dans notre système solaire, le Soleil est au centre:
  - Des planètes tournent autour du Soleil.
  - Des satellites tournent autour de certaines planètes.
  - Des sondes peuvent même tourner autour de certains satellites.
- Notre système solaire a donc une structure d'arbre comme ci-contre (extrait).
- Le but de cette partie est de modifier la classe `CorpsCeleste` pour donner la possibilité à un objet créé d'avoir des satellites.



1. Pour cette partie, désactiver le mouvement des objets, en commentant la ligne qui provoque le déplacement dans la fonction `main()` .

Réponse :

-
-

2. Ajouter un attribut `satellites` à la classe `CorpsCeleste` qui représentera la liste des satellites d'une planète.Dans le constructeur, initialiser cet attribut à `[]` .

Réponse :

3. Compléter la méthode `add_satellite` ci-dessous dans la classe `CorpsCeleste` .

```
def add_satellite(self, taille, rayon_orbite, v_angulaire, couleur,trace=False, phase=0)
```

Indications :

- L'appel de cette méthode créé un nouveau corps céleste avec les paramètres indiqués, puis l'ajoute dans la liste `self.satellites` .
- Cette méthode renvoie le satellite créé.
- Le centre de rotation des satellites n'est pas donné, puisque ce sera la position de la planète dont ils sont satellites.

Réponse :

PARTIE C: Créer un système stellaire

1. Représenter sous forme d'arbre le système ci-dessous (on n'indiquera que les noms des objets).

```
etoile = CorpsCeleste(30, 0, (WINDOWWIDTH/2, WINDOWHEIGHT/2) , 0,(255, 255, 255))
planete = etoile.add_satellite(30, 200, 3.01, (22,45,123))
planete2 = etoile.add_satellite(30, 500, 1.51, (22, 245,123))
lune1 = planete.add_satellite(20, 70, -12, (223,45,76))
lune2 = planete.add_satellite(10, 100, 10, (223,45,76))
lune3 = planete2.add_satellite(10, 100, 10.01, (223,145,76), True)
sonde = lune2.add_satellite(3, 12, 50.01, (255, 255, 76), True)
sonde2 = lune3.add_satellite(3, 12, 50.01, (255, 255, 76), True)
```

Réponse :

2. Ajouter les lignes de la question précédente dans la fonction `main()` pour créer le système stellaire ci-dessus, puis modifier la méthode `dessine` de la classe `CorpsCeleste` pour que chaque planète dessine ses satellites. Vérifier que les satellites des satellites s'affichent.

Indication : On utilisera la récursivité.

Réponse :

3. Décommenter la ligne qui arrête le mouvement des planètes. Si les satellites ne tournent pas sur leur orbite, faire le nécessaire pour que :

- Ils tournent sur leur orbite.
- Ils suivent leur planète en tournant autour (pour ça, il faut que quand la planète se déplace,elle modifie le centre de rotation de ses satellites).

Réponse :

PARTIE D : Parcours avec un vaisseau

La classe Vaisseau, contenue dans le fichier `vaisseau.py` représente un vaisseau spatial. Son constructeur prend en argument une position initiale, une liste de planètes et une couleur. Le vaisseau part de sa position initiale, puis il visite chacune des planètes de sa liste de destinations dans l'ordre où elles sont inscrites.

1. Créer un vaisseau dans la fonction `main()` . Faire les modifications nécessaires pour que le vaisseau apparaisse et se déplace.

Indications :

- Créer la liste `destinations` qui contient les objets à visiter.
- Un vaisseau est un objet qu'il faut ajouter dans la liste `univers` .

Réponse :

2. Le vaisseau `faucon` souhaite parcourir le système stellaire créé précédemment en profondeur préfixe.

a. Que contiendra la liste `destinations` ?

Réponse :

b. Écrire la fonction récursive `parcours_prefixe` qui renvoie la liste des destinations pour le faucon.

Réponse :

c. Ajouter le faucon et visualiser le résultat.

Réponse :

3. Le vaisseau `enterprise` souhaite parcourir le même système stellaire créé précédemment en profondeur postfixe.

- Que contiendra la liste `destinations2` qui accueillera les noms des objets visités ?

Réponse :

- Écrire la fonction récursive `parcours_postfixe` qui renvoie une liste de destinations pour l'enterprise.

Réponse :

c. Ajouter l'enterprise et visualiser les parcours des deux vaisseaux simultanément.

Réponse :