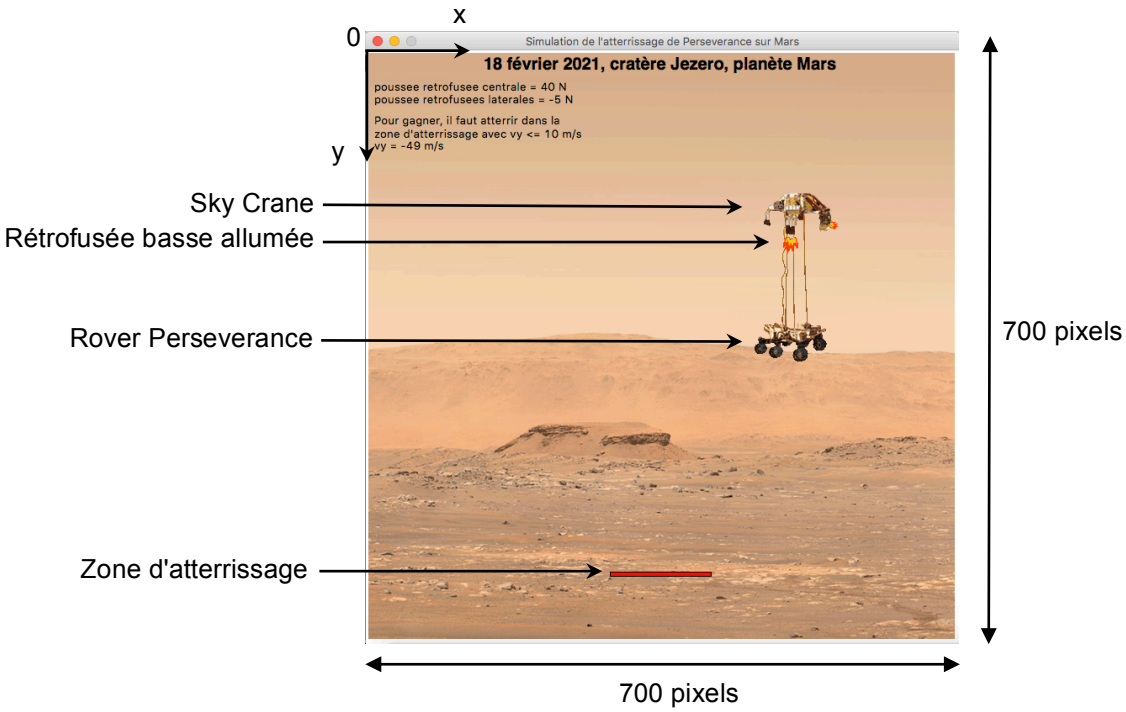
Arriverez-vous à poser le rover Perseverance sur Mars ?

Le rover Perseverance s'est posé avec succès sur la planète Mars le 18 février 2021 au fond du cratère Jezero. Vous allez écrire un programme Python permettant de simuler cet atterrissage. Ce programme devra permettre de piloter le "Sky Crane" (grue du ciel), l'étage de descente qui soutenait par des câbles le rover Perseverance pour le déposer à la surface de Mars. Le pilotage se fera en allumant ou en éteignant les rétrofusées du Sky Crane.
Pour réussir l'atterrissage, il faudra parvenir à atterrir dans le rectangle rouge avec une vitesse verticale inférieure à 10 m/s.

Ce programme fera appel à la notion de Programmation Orientée Objet. Les différents éléments graphiques du jeu (Sky Crane, flammes, zone d'atterrissage...) seront représentés par des objets. Ces objets seront des instances de classes. Vous écrirez le programme en langage Python.



1. Création de la fenêtre du jeu

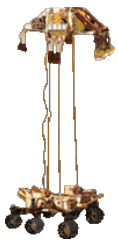
La partie graphique du jeu est gérée grâce à la bibliothèque Tkinter. Vous disposez dans le dossier d'un document PDF *Fiche_simplifiee_tkinter.pdf* avec les méthodes nécessaires.
La classe `Game` est le contrôleur principal du programme.

1. Compléter la méthode constructeur de la classe `Game` pour créer la fenêtre graphique du jeu avec les spécificités suivantes :
   - Titre : "Simulation de l'atterrissage de Perseverance sur Mars"
   - Redimensionnement impossible ;
   - Taille de la zone de dessin indiquée dans l'énoncé ;
   - Image de fond conforme à l'énoncé.

2. Créer un objet `jeu` de la classe `Game` pour afficher la fenêtre graphique.

2. Ajout du système {Sky Crane+rover}

Le rover suspendu au Sky Crane est représenté par une seule image (ci-contre). On nommera atterrisseur (*lander* en anglais) le système {Sky Crane+rover}.



1. Créer une classe `Lander` et compléter sa méthode constructeur pour pouvoir ajouter l'image d'un atterrisseur à la position de coordonnées (`x` ; `y`) ayant pour attributs `x`, `y`, `vx` et `vy`. Ces deux derniers attributs désignent la vitesse selon x et selon y de l'atterrisseur.
   Indice : vous pouvez ajouter d'autres arguments à la méthode constructeur que ceux cités ci-dessus.

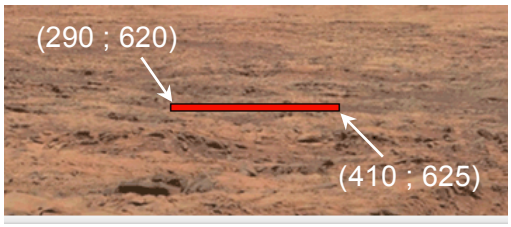
Thomas Appéré - Lycée Saint-Paul, Vannes

2. Créer un objet `perseverance` de la classe `Lander` pour afficher l'image `lander.gif` aux coordonnées (500 ; 0) et avec une vitesse initiale `vx = 0` et `vy = 100`.

3. Ajout de la zone d'atterrissage

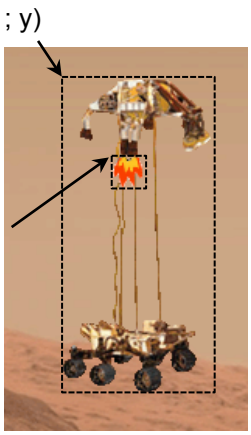
1. Créer une classe `Rectangle` et compléter sa méthode constructeur pour pouvoir ajouter un rectangle d'une couleur donnée en argument et dont les coins supérieur gauche et inférieur droit ont pour coordonnées respectivement (`x1 ; y1`) et (`x2 ; y2`).

2. Créer un objet `landing_site` de la classe `Rectangle` pour afficher un rectangle rouge correspondant à la zone d'atterrissage. Les coins supérieur gauche et inférieur droit du rectangle doivent avoir pour coordonnées respectives (290 ; 620) et (410 ; 625).



4. Ajout des rétrofusées

L'atterrisseur dispose de 3 rétrofusées : une en bas, une à gauche et une à droite.



1. Créer une classe `Retrorocket` et compléter sa méthode constructeur pour pouvoir ajouter l'image d'une rétrofusée placée aux coordonnées (`x_rocket ; y_rocket`) **par rapport** à l'atterrisseur. En effet, la rétrofusée doit se déplacer en même temps que l'atterrisseur.

2. Créer un objet `central_rocket` de la classe `Retrorocket` pour afficher l'image `pousseeCentrale.gif` aux coordonnées (22 ; 48) par rapport à l'objet `perseverance`.

3. Ajouter l'attribut `allume` à la classe `Retrorocket` qui prend `False` comme valeur par défaut.

4. Rendre l'image des rétrofusées invisible par défaut. Ainsi, les rétrofusées suivent le déplacement de l'atterrisseur lors de son déplacement tout en restant invisibles tant qu'elles ne sont pas activées.

5. Ajouter la méthode `allumage` prenant comme argument `evt` (événement) dans la classe `Retrorocket`. Cette méthode exécute les actions suivantes :
   - Si l'attribut `allume` a la valeur `True`, alors il passe à `False` et l'image de la rétrofusée est supprimée.
   - Si l'attribut `allume` a la valeur `False`, alors il passe à `True` et l'image de la rétrofusée est créée.

6. Compléter la méthode constructeur pour que la méthode `allumage` soit exécutée lorsque la touche associée à la rétrofusée est pressée. Tester votre programme.
   Exemple : lorsqu'on presse une fois la touche Bas du pavé numérique, la rétrofusée centrale s'affiche. Lorsqu'on appuie de nouveau une fois sur la touche Bas, elle ne s'affiche plus.

7. Créer un objet `left_rocket` de la classe `Retrorocket` pour afficher l'image `pousseeLateraleGauche.gif` aux coordonnées (-26 ; 15) par rapport à l'objet `perseverance`.

8. Créer un objet `right_rocket` de la classe `Retrorocket` pour afficher l'image `pousseeLateraleDroite.gif` aux coordonnées (-26 ; 15) par rapport à l'objet `perseverance`.

5. Déplacement de l'atterrisseur

Il va s'agir ici de déplacer l'image de l'atterrisseur en respectant les équations du mouvement de l'atterrisseur dans le champ de pesanteur martien.

1. Ajouter les attributs `delta_x` et `delta_y` de valeurs égales à 0 à la classe `Lander`. Ces deux attributs correspondent au déplacement de l'atterrisseur respectivement selon x et y en pixels par unité de temps.

2. Ajouter les attributs `central_thrust`, `left_thrust` et `right_thrust` de valeurs égales à 0 à la classe `Lander`. Ces trois attributs correspondent à la force de poussée exercée respectivement par les rétrofusées centrale, gauche et droite.

3. Ajouter la bibliothèque `time` au programme.

4. Ajouter au début du programme la valeur de la pesanteur martienne, `gravity = 3.8` (en $m.s^{-2}$).

5. Ajouter une méthode `deplacer` à la classe `Lander`. Cette méthode va permettre de déplacer l'image de l'atterrisseur. Elle fonctionne selon le principe suivant, que vous devrez coder :

   i. Détermination du temps écoulé `time_elapsed` depuis le dernier déplacement effectué. Pour y accéder, vous aurez besoin de la variable `time.time()` qui renvoie la date en secondes sous la forme d'un réel.

   ii. Calcul de la nouvelle valeur de la vitesse selon x et selon y de l'atterrisseur. Les équations sont les suivantes :

   $$vx = time\_elapsed \times (left\_thrust - right\_thrust)$$
   $$vy = time\_elapsed \times (gravity - central\_thrust)$$

   iii. Calcul du déplacement selon x et selon y de l'atterrisseur. Les équations sont les suivantes :

   $$delta\_x = time\_elapsed \times vx + 0.5 \times (left\_thrust - right\_thrust) \times time\_elapsed^2$$
   $$delta\_y = time\_elapsed \times vy + 0.5 \times (gravity - central\_thrust) \times time\_elapsed^2$$

   iv. Déplacement de l'image de l'atterrisseur.

   v. Détermination de la nouvelle date.

6. Ajouter la méthode `boucle_principale` à la classe `Game`. Coder cette méthode pour qu'elle réponde aux spécifications suivantes :
   - Cette méthode contient une boucle infinie qui ne s'arrête que lorsqu'on ferme la fenêtre. On l'implémente en tapant *Tant que 1*.
   - Dans cette boucle, la méthode `deplacer` s'applique à l'objet `perseverance`.
   - Puis la fenêtre est rafraîchit.
   - Enfin, le programme est mis en pause pendant 50 ms à l'aide la commande `time.sleep(0.05)`.

7. Modifier votre programme pour tenir compte de cette méthode `boucle_principale` et vérifier que l'atterrisseur se déplace à l'écran à l'exécution du programme.

6. Déplacement des rétrofusées

Dans le domaine des jeux vidéo, un élément graphique qui peut se déplacer sur l'écran est appelé un lutin (*sprite* en anglais). L'atterrisseur et les rétrofusées sont des lutins.

1. Ajouter l'attribut `sprites` égal à une liste vide à la classe `Game`.

2. Ajouter les objets `perseverance`, `central_rocket`, `left_rocket` et `right_rocket` à la liste `sprites`.

3. Modifier la méthode `boucle_principale` de la classe Game pour exécuter la méthode `deplacer` sur tous les lutins.

4. Ajouter une méthode `deplacer` à la classe `Retrorocket`. La compléter pour que les images des rétrofusées se déplacent en même temps que l'image de l'atterrisseur. Vous pourrez être amené à mettre à jour les coordonnées x et y de l'image de l'atterrisseur.

7. Prise en compte de l'action des rétrofusées

Il s'agit maintenant de prendre en compte l'action des rétrofusées sur le déplacement de l'atterrisseur.

1. Ajouter au début du programme la valeur de la poussée maximale de la rétrofusée centrale, `central_thrust_max = 40` (en N) et des rétrofusées latérales, `lateral_thrust_max = 5` (en N).

2. Modifier la méthode `deplacer` de la classe `Lander` pour que l'appui sur la touche Bas entraine le passage de l'attribut `central_thrust` à la valeur `central_thrust_max` ou à la valeur `0` suivant la situation. Procéder de même pour les touches Gauche et Droite.

8. Zone d'atterrissage atteinte ?

Le rover Perseverance atterrit avec succès sur Mars s'il atteint la zone d'atterrissage avec une vitesse verticale inférieure à 10 m/s. S'il dépasse cette vitesse, alors il se crashe. Dans tous les cas, il faut faire en sorte que le lander arrête de se déplacer lorsqu'il atteint l'altitude de la zone d'atterrissage.

1. Ajouter l'attribut `enfonction` à la classe `Game` et fixer sa valeur par défaut à `True`.

2. Modifier la méthode `boucle_principale` de la classe `Game` pour que les lutins ne soient pas déplacés lorsque l'attribut `enfonction` vaut `False`.

3. Modifier la méthode `deplacer` de la classe `Lander` pour que l'attribut `enfonction` passe à `False` lorsque le lander a atteint l'ordonnée de la zone d'atterrissage.

4. Modifier les méthodes de la classe `Retrorocket` pour qu'on ne puisse plus allumer les rétrofusées une fois que le lander a atteint l'ordonnée de la zone d'atterrissage.

5. Modifier la méthode `deplacer` de la classe `Lander` pour afficher dans la fenêtre une zone de texte contenant le message "Zone d'atterrissage ratée !" lorsque les conditions sont requises. Vous déterminerez ces conditions.

9. Atterrissage réussi ?

1. Ajouter au début du programme la valeur de la vitesse maximale à l'atterrissage du lander, `vmax = 10` (en $m.s^{-1}$)

2. Modifier la méthode `deplacer` de la classe `Lander` pour afficher dans la fenêtre une zone de texte contenant le message "BOOM !" lorsque les conditions sont requises. Vous déterminerez ces conditions.

3. Modifier les méthodes de la classe `Lander` pour afficher une zone de texte contenant en temps réel la vitesse verticale du lander.

4. Modifier la méthode `deplacer` de la classe `Lander` pour afficher dans la fenêtre une zone de texte contenant le message "We're safe on Mars!" lorsque l'atterrissage est réussi.

5. Ajouter une classe `Debris`. Cette classe doit notamment comporter une méthode permettant d'afficher l'image des débris lorsque le lander s'écrase à la surface de Mars.

10. Informations pour l'utilisateur

Compléter le programme pour afficher en permanence les informations présentes sur la copie d'écran en première page de ce polycopié : date et lieu de l'atterrissage, poussée des rétrofusées actualisée en temps réel, consigne pour réussir le jeu.
Pour rendre le jeu plus dynamique, vous pouvez utiliser ces paramètres : gravity = 38, central_thrust_max = 200, v_max = 50.

Thomas Appéré - Lycée Saint-Paul, Vannes