Projet Mario : un jeu de plates-formes

# Introduction

Un **jeu de plates-formes** (ou *platformer*) est un [genre](https://fr.wikipedia.org/wiki/Genre_de_jeux_vidéo) de [jeu vidéo](https://fr.wikipedia.org/wiki/Jeu_vidéo), sous-genre du [jeu d'action](https://fr.wikipedia.org/wiki/Jeu_d%27action). Dans les jeux de plates-formes, le joueur contrôle un [avatar](https://fr.wikipedia.org/wiki/Avatar_(informatique)) qui doit sauter sur des plates-formes suspendues dans les airs et éviter des obstacles.

[*Donkey Kong*](https://fr.wikipedia.org/wiki/Donkey_Kong_(jeu_vidéo,_1981)), [jeu d'arcade](https://fr.wikipedia.org/wiki/Jeu_d%27arcade) créé par [Nintendo](https://fr.wikipedia.org/wiki/Nintendo) et sorti en juillet 1981, est le premier jeu qui a permis au joueur de sauter par-dessus des obstacles et de franchir des précipices, ce qui en fait le premier jeu de plates-formes à proprement parler. Le jeu introduit le personnage de [Mario](https://fr.wikipedia.org/wiki/Mario_(personnage)) qui devient une véritable icône du genre.

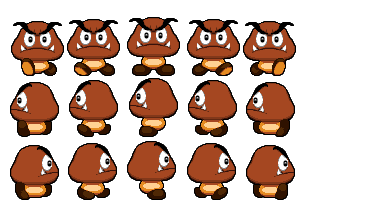
Source : <https://fr.wikipedia.org/wiki/Jeu_de_plates-formes>

Un sprite, ou lutin, est dans le [jeu vidéo](https://fr.wikipedia.org/wiki/Jeu_vidéo) un élément graphique qui peut se déplacer sur l'[écran](https://fr.wikipedia.org/wiki/Moniteur_d%27ordinateur). En principe, un sprite est en partie transparent, et il peut être animé (en étant formé de plusieurs [images matricielles](https://fr.wikipedia.org/wiki/Image_matricielle) qui s'affichent les unes après les autres). Le fond de l'écran constitue généralement le [décor](https://fr.wikipedia.org/wiki/Décor) et les sprites sont les personnages et les objets qui se superposent au fond d'écran et qui se déplacent. Un sprite peut parfois aussi passer derrière un élément du fond d'écran.

Source : <https://fr.wikipedia.org/wiki/Sprite_(jeu_vidéo)>

# Présentation des fichiers de départ

Le point de départ est constitué de 2 fichiers :

* **B\_mario.py** : il contient le déroulement du jeu 2D, importe le fichier *sprites.py* et utilise la bibliothèque *pygame* développées pour réaliser des jeux (traitement d’image, prise en compte du clavier, de la souris, etc)
* **B\_sprites.py** : permet de réaliser une liste ordonnée d’images d’un personnage en mouvement à partir d’un sprite sheet.

La liste d’un sprite contient :

* 6 images correspondant à un déplacement du sprite à droite,
* Puis 6 images correspondant à un déplacement du sprite à gauche,
* Puis l’image du personnage à l’arrêt tourné vers la droite,
* Puis l’image du personnage à l’arrêt tourné vers la gauche,
* D’autres images peuvent être insérées (shoot, jump, …).

Le déroulement du programme consiste à afficher les bonnes images au bon endroit en fonction des déplacements.

# Préliminaire : comprendre le code proposé.

Dans cette partie, vous devez analyser soigneusement le fonctionnement des deux fichiers **B\_mario.py** et **B\_sprites.py**

**Répondez aux questions suivantes dans votre compte rendu** :

Vous pouvez vous aider du tutoriel suivant <https://zestedesavoir.com/tutoriels/846/pygame-pour-les-zesteurs/>

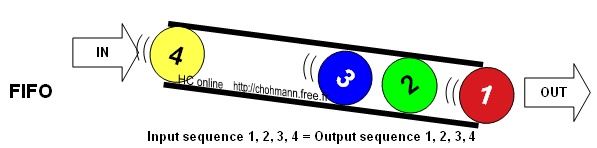
ou de la documentation du site de pygame <https://www.pygame.org/docs/>

**1°) Création de la fenêtre :**

1. Dans quel fichier est créée la fenêtre (variable **fenetre**) où se déroulera le jeu ?
2. Quelle instruction permet de réaliser le lien entre les deux fichiers ?
3. Que permet de faire la fonction **pygame.display.set\_mode() ?** *(fichier B\_sprites ligne 8 )*
4. Que permet de faire la fonction **pygame.image.load() ?** *(ligne 10)*
5. Dans cette même ligne 10, quel est l’effet de rajouter **.convert()** à la fin ?
6. Expliquer la différence avec le **.convert\_alpha()** *(ligne 16)*
7. Quelle instruction permet d’afficher une image sur l’écran ? *(fichier B\_mario)*
8. Quels sont les formats d’images supportés par pygame ?
9. Que permet de faire la fonction **pygame.display.flip() ?** *(ligne 47)*
10. Que permet de faire la fonction **pygame.time.Clock().tick(10) ?** *(ligne 43)*

**2°) Gestion des événements :**

Un événement peut être une entrée clavier (soit l’appui, soit le relâchement d’une touche), le déplacement de votre souris, un clic… Il faut savoir que chaque événement créé est envoyé sur une **file** (ou *queue*), en attendant d’être traité, comme lorsque l’on fait la queue au supermarché !



La ligne 24 crée une boucle qui va parcourir cette file d’événements, puis tester chaque événement enregistré en interrogeant son type (ligne 25)



1. A quel événement correspond le type **pygame.QUIT** ? *(ligne 25)*
2. A quel événement correspond le type **pygame.KEYDOWN** ?*(ligne 27)*
3. Quel est le type d’événement correspondant à l’appuie sur la flèche droite ?
4. Quel est le type d’événement correspondant au relâchement d’une touche ?
5. A quoi sert la variable **continuer** définie ligne 17 ?

## Travail préliminaire à réaliser

1. Modifier le programme pour qu’il permette le déplacement de Mario vers la gauche (de façon « naturelle », avec animation des images)
2. Modifier le programme pour que Mario puisse s’élever de 10px vers le haut à chaque appui sur la flèche haut. Ajouter une possibilité de le faire redescendre de façon similaire par appui sur la flèche du bas.

# Création d’une classe Personnage

On se rend vite compte que si on multiplie le nombre de personnages l’écriture du programme va devenir très lourde.

Comme tous les personnages ont des comportements semblables, la solution est de créer une classe Personnage que l’on instanciera pour chaque nouveau personnage.

Propriétés de la classe Personnage :

* **sprites** : liste des images réalisant le mouvement du personnage (voir *sprite.py*),
* **xp** : abscisse du coin en haut à gauche du personnage dans la fenêtre d’affichage,
* **yp** : ordonnée du coin en haut à gauche du personnage dans la fenêtre d’affichage,
* **deplt** : déplacement élémentaire du personnage en pixels.

Méthode de la classe Personnage :

* **le constructeur**,
* **deplacer\_droite() -> image:** qui déplace le personnage à droite et renvoie l’image en cours du personnage,
* **deplacer\_gauche() -> image:** qui déplace le personnage à gauche et renvoie l’image en cours du personnage,
* **stop() -> image:** qui renvoie l’image du personnage à l’arrêt, soit tourné vers la droite, soit vers la gauche.

## Travail à réaliser

1. Enregistrer B\_*mario.py* sous C\_*mario\_classe.py*.
2. Créer un module C\_*classes.py* dans lequel vous écrirez la classe Personnage
3. Dans C\_*mario\_classe.py*, instancier la classe Personnage pour créer l’objet mario.
4. Modifier C\_*mario\_classe.py* pour n’utiliser que les propriétés et méthodes de l’objet mario.
5. De même, créer l’objet gooma et faire en sorte qu’il aille toujours en direction de Mario.
6. BONUS : créer 2 aigles qui se déplacent dans le ciel en boucle.

On voit qu’il devient alors très simple de créer autant de personnages que l’on veut.

# Création d’une classe Mario

Le personnage de Mario a des comportements plus complexes que les autres personnages. Le saut ou la détection de collisions par exemple. Il peut être intéressant de créer une classe spécifique Mario qui hérite de la classe Personnage. De cette façon la classe Mario bénéficie des propriétés et des méthodes de la classe Personnage.

**class** **Mario(**Personnage**):** # la classe Mario hérite de la classe Personnage

# la déclaration du constructeur de Mario est identique

# à celle du constructeur de Personnage

**def** \_\_init\_\_**(**self**,** sprite**,** xp **=** 50**,** yp **=** 296**,** deplt **=** 10**):**

# Appel explicite au constructeur de la classe Personnage

Personnage**.**\_\_init\_\_**(**self**,** sprite**,** xp **=** 50**,** yp **=** 296**,** deplt **=** 10**)**

# D’autres propriétés peuvent être rajoutées ici

## Travail à réaliser

1. Enregistrer C\_*mario\_classe.py* sous D\_*mario\_classe.py* et *C\_classes.py* sous *D\_classes.py*
2. Créer la classe Mario dans *D\_classes.py* et modifier l’instanciation de mario en conséquence.
3. Écrire la méthode Mario.saute()-> image: qui fait sauter Mario à l’aide de la flèche du haut.
4. BONUS : définir l’attribut **self.rectangle = sprite[0].get\_rect().inflate(-5, -5)** dans la classe Personnage. Cette propriété sert à la détection de collisions en utilisant par exemple la méthode **mario.rectangle.colliderect(gooma.rectangle)-> booleen.**