Un morpion pour sauver le monde

Pygame

version 3.0





1. Introduction

Au début de la décennie, en 1980, le monde vit avec la peur constante d’un conflit nucléaire. Pour être sûr d’être prêt à riposter, une puissance occidentale a développé un ordinateur capable de simuler le conflit en continu ; en cas d’attaque, il serait alors capable de riposter de la meilleure façon. Le problème, c’est que cet ordinateur est devenu aussi paranoïaque qu’un être humain, et a donc décidé de déclencher sans provocation un conflit nucléaire. Il a toutefois donné une condition pour qu’il arrête la procédure avant qu’il ne soit trop tard : il faut lui démontrer que les êtres humains sont plus qu’une créature qui agit selon son instinct. Pour ce faire, il vous défie au morpion.

Votre mission, si vous décidez de l’accepter, est de créer un jeu du morpion en Python.



Jeu du Morpion

1. Consignes

* En cas de question, pensez à demander de l’aide à votre voisin de droite.  
  Puis de gauche. Ou inversement. Puis demandez enfin à un Cobra si vous êtes toujours bloqué(e).
* Suivez bien la documentation d’installation pour python “Python et ses outils”.
* Si l’installation d’un logiciel se déroule mal, recommencez depuis le début.
* Pensez à faire valider votre projet par un Cobra lorsque vous aurez terminé.
* Le nom de repository pour ce projet est : cc\_morpion\_pygame

1. La logique avant tout
   1. Un maxi outil qui sera utile !

Pygame est une bibliothèque libre multiplateforme qui a pour but de faciliter le développement de Jeux vidéo, infographie, gestion audio, avec le langage de programmation [Python](https://fr.wikipedia.org/wiki/Python_(langage)). Nous allons l’utiliser aujourd’hui pour sauver le monde d’un hiver nucléaire ! Cette bibliothèque a commencé son développement en 2000, de nombreux logiciels amateurs l’utilisent encore aujourd’hui. La librairie est un des piliers des jeux homebrew, indépendants open source !



Pygame le petit serpent des jeux

Pour bien débuter le projet, il va falloir importer certaines libraires à notre programme. Ces lignes sont à placer au plus haut du fichier

import pygame, sys, random

from pygame.locals import \*

* 1. Il faut ouvrir la fenêtre

La première étape vous devez ouvrir une fenêtre [Pygame](https://www.pygame.org/docs/) qui devra :

* Avoir une taille de 600 par 600
* Un titre de “Morpion”

. 

Rendu de la fenêtre

* 1. Mais où sont les images ?

Il va falloir charger les images qui sont fournis avec ce sujet. Les importations doivent être stocké dans des variables pour être réutiliser plus tard.

image = pygame.image.load("background.jpg").convert\_alpha()

image2 = pygame.image.load("croix.png").convert\_alpha()

image3 = pygame.image.load("rond.png").convert\_alpha()

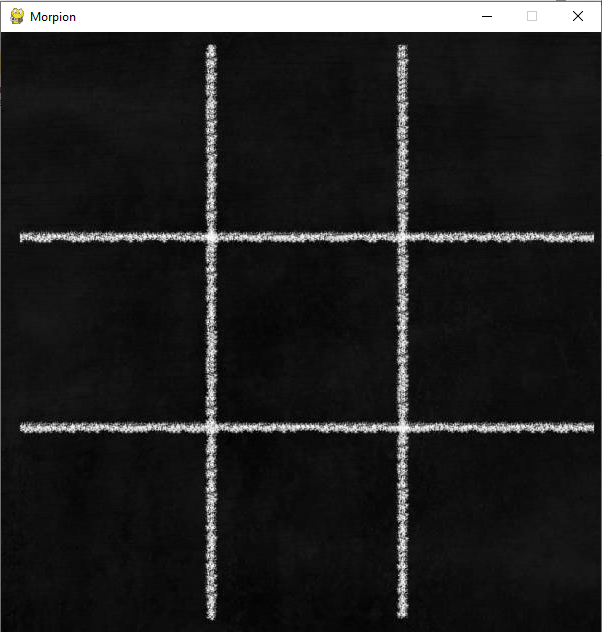
Pour afficher l’image de fond, il faudra interagir avec la fenêtre pygame, avec la variable "window", pour placer la variable "image". Elle devra être dans une boucle afin que le programme ne se termine pas directement. Pour que cela soit plus simple, il faudra placer cette boucle dans une fonction nommé "main", puis appeler celle-ci.

def main():

while True :

window.blit(image, (0, 0))

main()



*L’affichage du morpion dans la fenêtre pygame*

1. Jouons tous ensemble à un petit jeu

Quelques ajouts et modifications s’imposent :

Pour connaitre le joueur qui doit jouer, vous allez devoir vous servir de 2 variables dont la valeur sera soit 0, soit 1. Il faudra placer ceci au plus haut du fichier, après les importations :

joueur1 = 0

joueur2 = 1

Pour définir le joueur qui devra jouer, il va falloir créer une variable au début de la fonction "main", en dehors de la boucle, qui contiendra le joueur actif.

joueur\_actif = joueur1

A la suite de cela, il faudra définir la map de notre jeu afin de savoir si une position est vide, ou qu’un des joueur à placé un point. Il faudra placer cette variable à la suite de la variable "joueur\_actif".

map = [['\_', '\_', '\_'], ['\_', '\_', '\_'], ['\_', '\_', '\_']]

Revenons-en à notre boucle. Dans celle-ci il va falloir ajouter quelques petites choses notamment lorsque l’un des joueur interagit avec la fenêtre pygame, par exemple pour fermer la fenêtre ou placer un point dans la map :

while True:

for event in pygame.event.get():

if event.type == QUIT: # Si l’utilisateur ferme la fenètre

pygame.quit()

sys.exit()

if event.type == pygame.MOUSEBUTTONDOWN: # Si il clique

if pygame.mouse.get\_pressed() == (1,0,0): # Clique gauche

x,y = pygame.mouse.get\_pos() # On récupère la position

i = ... # Regarder dans quelle case horizontal est le click

j = ... # Regarder dans quelle case vertical est le click

if (...): # Si la case [i, j] est vide

map = ... # Placer le jeton du joueur sur la case

joueur\_actif = ... # Changer de joueur

if (...): # Si un joueur à gagner ou si la map est pleine

pygame.quit()

sys.exit()

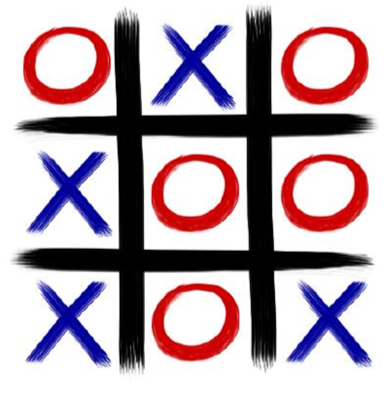
... # Afficher la map et les jetons des joueurs

pygame.display.flip()

Il est fortement recommandé de créer des fonctions pour remplacer les ... dans le code ci-dessus. Voici un exemple de création de fonction :

def nom\_de\_la\_fonction(arguments\_donné\_à\_la\_fonction):

...



Un jeu de morpion fini

🌟 À vous de jouer maintenant ! 🌟

1. Conclusion

Bravo, vous avez sauvé le monde d’une catastrophe nucléaire ! Mais la menace est toujours là : vous n’avez pas réussi à éteindre l’ordinateur qui vous menaçait. Pour ce faire, vous pouvez essayer de le faire effectivement jouer au morpion que vous avez fini de programmer.

Bonus:

* Ajouter un algorithme qui a pour but de gagner la partie
* Ajouter un menu pour sélectionner le mode de jeu : joueur contre joueur ou intelligence artificielle contre joueur
* Ajouter un mode intelligence artificielle contre intelligence artificielle