Réalisation du jeu OXO avec Python3, Pygame Zero et Mu

CoderDojo Seneffe

Octobre 2020

Pendant cette session, nous allons découvrir comment créer un jeu OXO avec Python.

Retrouve les traductions anglais → français dans cette marge.

OXO (aussi appelé Tic-Tac-Toe ou Morpion) est considéré comme le premier jeu vidéo, créé en 1952.

Sur une grille de 3x3, deux joueurs s'affrontent. Ils doivent remplir chacun à leur tour une case de la grille avec le symbole qui leur est attribué : O ou X. Le gagnant est celui qui arrive à aligner trois symboles identiques, horizontalement, verticalement ou en diagonale. Le joueur jouant X commence la partie.

Pour créer notre jeu de OXO, nous allons utiliser Pygame Zero, une bibliothèque qui facilite la création de jeu avec Python, ainsi que Mu, un environnement de programmation (IDE) pour écrire et exécuter notre programme. Pygame Zero va nous permettre de gérer facilement les images, le clavier et les sons pour nous concentrer sur la programmation du jeu.

Attention, Pygame Zero est différent (et plus simple) que Pygame. Si tu cherches des exemples sur internet, il faut bien faire la différence.

1 Installation de Mu

Si Mu n'est pas déjà installé sur ton ordinateur, tu peux le télécharger sur `https://codewith.mu/en/download`.
Il n'est pas nécessaire d'installer Pygame Zero, cela est compris lorsque tu installes Mu.
Sur Windows et Mac, il n'est pas nécessaire non plus d'installer Python.

2 Hello World

1. Si tu ne l'as pas déjà fait, ouvre Mu, tu devrais voir un fichier vide. Si ce n'est pas le cas, tu peux créer un nouveau fichier en cliquant sur `New`.
2. Mu dispose de plusieurs modes en fonction de ce que tu veux faire comme type de programme. On va commencer par un exemple de Python "simple", sans Pygame Zero. Clique sur `Mode` et choisi `Python 3`.
3. Tape le code suivant dans le fichier :

   ```
   print("Hello World !")
   ```

4. Clique sur `Run` ou `Lancer` pour lancer ton programme, tu devras choisir un nom pour sauvegarder celui-ci.

Que se passe-t-il ?

Félicitations ! Tu viens d'écrire ton premier programme en Python. Tu peux modifier la ligne de code pour faire dire ce que tu veux à ton programme.

> Petite astuce : pour chaque étape de cette session, utilise un fichier différent. Ça te permettra de garder une trace de ce que tu as fait. C'est pratique pour se souvenir de comment tu as fait les choses !
>
> Dans Mu, tu peux double-cliquer sur le nom du fichier dans l'onglet pour lui donner un autre nom.

3 Les listes en Python

Dans notre programme, nous aurons besoin d'utiliser des listes.

Une liste en Python est un objet permettant de stocker un ensemble d'éléments, dans un ordre donné. On peut y placer des chiffres, des mots, ... et même une autre liste.

Voici un exemple de liste de mots

```
couleurs = ['rouge', 'vert', 'bleu']
```

On peut accéder à un élément de la liste en utilisant son index (sa position), en comptant à partir de 0. Affichons par exemple le premier élément de la liste

```
print(couleurs[0])
```

Si on veut parcourir l'ensemble de la liste, on peut utiliser une boucle comme ceci

```
for couleur in enumerate(couleurs):
    print(couleur)
```

for = pour
in = dans
enumerate =énumérer

Si tu exécutes ce programme, qu'affiche-t-il ?

On remarque que la variable couleur ne contient pas seulement l'élément de la liste mais aussi sa position. La fonction enumerate retourne en fait un tuple (un couple) de 2 éléments, la position et l'élément.

4 Les tuples

Un tuple est très similaire à une liste, c'est aussi une collection d'éléments stockés dans un ordre donné, mais à la différence des listes, une fois créé un tuple ne peut plus être modifié.

Les tuples sont souvent utilisés en Python pour permettre à une fonction de retourner plusieurs valeurs, comme la fonction enumerate ci-dessus.

Comme pour une liste, on peut accéder à un élément du tuple en utilisant son index.

Modifie le programme précédent pour n'afficher que la couleur.

On peut aussi assigner la valeur de chacun des différents éléments d'un tuple directement à des variables en une opération appelée "unpacking" (qu'on pourrait traduire par "déballage").

Dans l'exemple ci-dessus on aurait pu écrire la boucle
`for position, couleur in enumerate(couleurs):`
afin d'avoir directement accès au nom de la couleur.

5 Bonjour Pygame Zero

Maintenant, jouons avec Pygame Zero. Pour cela on va changer de mode dans Mu. Clique sur `Mode` et choisi `Pygame Zero`.

Nous allons commencer par afficher la fenêtre dans laquelle le jeux se déroulera et en colorier le fond en vert.

```
WIDTH = 200
HEIGHT = 200

def draw():
    screen.fill((0, 255, 0))
```

WIDTH = largeur
HEIGHT = hauteur

draw = dessiner
screen = écran
fill = remplir

1. Clique sur `New` ou `Nouveau` pour créer un nouveau fichier.
2. Recopie le programme ci-dessus dans l'éditeur.
3. Clique sur `Play` ou `Jouer` pour exécuter le programme. Tu devras donner un nom à ton fichier si ce n'est pas déjà fait, par exemple `OXO1`.
   Une fenêtre s'est-elle ouverte ? Le fond est-il vert ?
4. Peux-tu changer la taille de la fenêtre et la couleur du fond ?

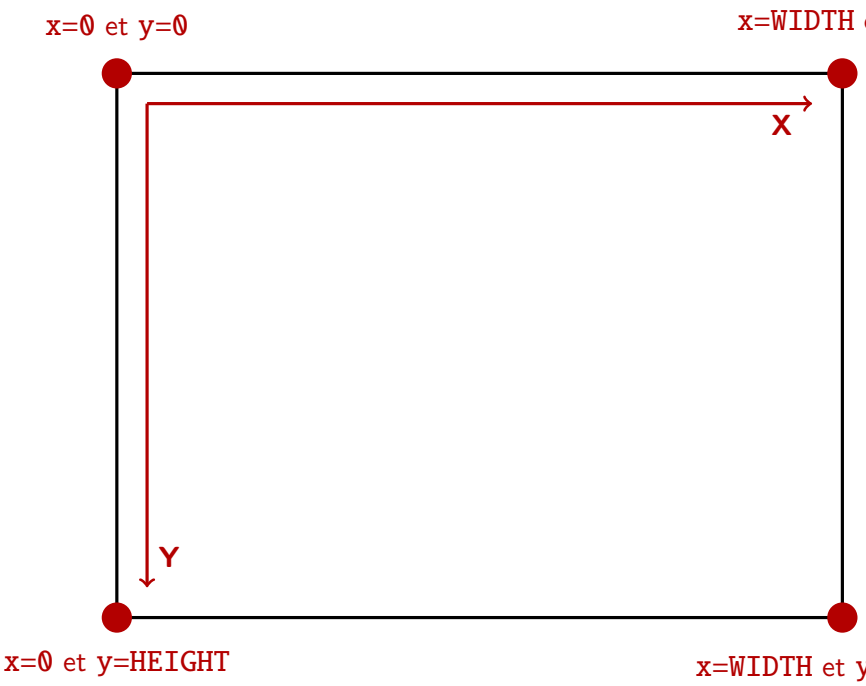
Quelques explications

Pygame Zero te simplifie la vie au maximum et a déjà programmé toute une série de choses pour toi, comme par exemple ouvrir une fenêtre pour ton jeu. Si tu ne précise rien, Pygame Zero décide de la taille de la fenêtre.
Mais il te donne la possibilité de choisir toi même la taille à travers des variables (`WIDTH` et `HEIGHT`).
En assignant une valeur à ces variables au début du programme, c'est toi qui choisi la taille de la fenêtre.

Pygame Zero sait aussi quand il est nécessaire de dessiner dans la fenêtre et quand c'est le cas, il te demande de le faire en appelant `draw()`.
Tu dois donc définir une fonction portant le nom de `draw` et y inclure le code qui va dessiner à l'écran.
C'est ce que le mot clé **`def`** indique (**`def`** pour define, qui veut dire définir en français).
Attention à ne pas oublier les parenthèses après le nom de ta fonction.
Les fonctions peuvent prendre des paramètres, qu'on défini entre les parenthèses, mais même quand il n'y a aucun paramètre, les parenthèses sont nécessaires.

6 Le positionnement des objects

Avant de dessiner la grille de jeu, il est important de savoir comment sont positionnés les éléments sur l'écran. La position de chaque élément est définie par des coordonnées x et y.



width = largeur

height = hauteur

L'axe horizontal (X) va de gauche à droite. Sur le bord gauche de la fenêtre, il a la valeur 0. Sur le bord droit de la fenêtre, il a la valeur `WIDTH`, qui est la largeur de la fenêtre.

L'axe vertical (Y) va de haut en bas. Sur le bord supérieur de la fenêtre, il a la valeur 0. Sur le bord inférieur de la fenêtre, il a la valeur `HEIGHT`, qui est la hauteur de la fenêtre.

1. Donne un nouveau nom à ton fichier, par exemple `OX02`.
2. Modifie ton fichier de la façon suivante :

```
WIDTH = 650
HEIGHT = 650

def draw():
    screen.clear()
    screen.draw.line((250, 100), (250, 550), (255, 255, 255))
```

clear = effacer

draw = dessiner

line =ligne

   L'instruction `screen.draw.line` permet de dessiner une ligne entre 2 points, donnés comme les 2 premiers paramètres de la fonction. Le 3ème paramètre indique la couleur de la ligne, exprimée comme un tuple indiquant l'intensité (entre 0 et 255) des 3 couleurs de base (rouge, vert et bleu) composant la couleur voulue.
3. Clique sur `Play` ou `Jouer` pour exécuter le programme.
4. Vois-tu une ligne à l'écran ? Peux-tu compléter le code pour dessiner la grille de jeu complète ?

7 Un peu de nettoyage

Si plus tard tu veux regarder un programme que tu as fait ou le donner à quelqu'un d'autre, c'est important que le programme soit lisible.

Il est donc intéressant de donner des noms parlants aux éléments du programme.

Modifie ton programme en utilisant des constantes pour les positions de la grille et pour la taille de chaque case.

En python, une constante est une variable à laquelle on assigne une valeur qu'une seule fois et qu'on écrit, par convention, en majuscule.

8 Afficher O et X

Nous pouvons maintenant dessiner le O ou le X sur la grille.

Pour ce faire, nous allons simplement afficher le texte 'O' ou 'X' à l'endroit approprié en utilisant la fonction

```
screen.draw.textbox(texte, Rect(x, y, largeur, hauteur))
```

qui affiche le texte en prenant toute la place dans le rectangle donné.

Marque à présent 2 cases de ton choix dans la grille, une avec un X, une avec un O.

9 Stocker la grille et l'afficher

Tu sais à présent marquer une case avec un X ou un O. Mais pour jouer, ton programme doit retenir où les joueurs ont placé des Xs et des Os.

Pour cela, nous allons stocker la grille dans une variable, et pour chaque case, indiquer si elle est vide, contient un X ou un O.

Ajoute une variable pour contenir la grille, initialise la avec des cases vides et dessine la grille à partir de cette variable.

Souviens-toi des listes dont nous avons parlé plus tôt.
Tu peux utiliser un espace pour indiquer le vide.
C'est la fonction `draw` *qui est utilisée pour l'affichage à l'écran.*

10 Utiliser la souris pour placer une marque

Maintenant que tu affiches la grille, on voudrait pouvoir placer un X ou un O dans une case, par exemple en cliquant dessus avec la souris.

PyGame Zero capture automatiquement les événements liés à la souris et appelle une fonction pour te prévenir. La fonction ci-dessous est appelée quand on appuie sur un bouton de la souris

```
def on_mouse_down(pos):
```

`pos` te donne la position où la souris se trouvait quand on a appuyé sur le bouton.
Il s'agit d'un tuple représentant les coordonnées : `pos[0]` te donne le premier élément, qui correspond à x et `pos[1]` te donne le second, qui correspond à y.

1. Capture la position de la souris et affiche la (avec un print).
2. En fonction de la position de la souris, calcule la case de la grille sur laquelle on a cliqué (indice : // permet de réaliser une division entière).
3. Place un X dans la case sélectionnée.

11 A chacun son tour

Pour l'instant, à chaque fois qu'on clique sur une case, on ajoute un X. On voudrait ajouter un X, puis un O, puis un X et ainsi de suite.

1. Modifie le programme pour qu'on change entre X et O à chaque tour.
2. Que se passe-t-il si tu cliques sur une case déjà occupée.

Tu peux utiliser une variable pour stocker le symbole à utiliser à chaque tour.
Attention : pour changer la valeur d'une variable globale dans une fonction, il faut l'indiquer, par exemple `global ma_variable`

12 Détecter si un joueur gagne

On voudrait maintenant que le programme détecte quand un joueur vient de créer une ligne gagnante.

1. Quelles sont les lignes gagnantes possibles ?
2. Ecrit une fonction qui teste si on vient de créer une de ces lignes dans la grille.
3. Si le coup qu'on vient de jouer est gagnant, afficher le nom du gagnant à l'écran et arrêter le partie.
4. Et si la grille est remplie et que personne n'a gagné ? C'est une égalité, affiche un message dans ce cas.

13 On rejoue ?

Une fois la partie terminée, on ne peut pour l'instant plus rien faire. Si on désire jouer une nouvelle partie, on doit quitter le programme et le relancer.

On pourrait par exemple recommencer une partie en appuyant sur une touche.

Comme c'était le cas pour la souris, PyGame Zero détecte automatiquement les événements liés au clavier et appelle la fonction `on_key_down(key)` pour te prévenir qu'on a appuyé sur une touche.

1. Détecte quand on appuie sur une touche (par exemple n, pour nouvelle partie) et recommence une partie.
2. As-tu bien pensé à tout ce qu'on doit faire quand on recommence une partie ?

14 Faire jouer l'ordinateur

N'as-tu pas envie de pouvoir jouer contre l'ordinateur ?
On va maintenant modifier le programme pour que l'ordinateur joue le tour des Os.

Dans un premier temps, l'ordinateur va choisir une case libre au hasard.

Choisir un nombre au hasard

Python offre une fonction pour choisir un nombre entier au hasard : `randint()`, celle-ci accepte 2 paramètres indiquant l'interval dans lequel les nombres sont générés (le plus petit et le plus grand nombre possible). Attention, cette fonction se trouve dans un module séparé, qu'il faut importer dans le programme, comme le montre le code ci-dessous.

```
import random

print(random.randint(0, 10))
```

1. Utilise la liste des cases vides et choisis-en une au hasard pour y placer un O.
2. L'ordinateur joue-t-il bien le tour des Os ? As-tu trop facile pour gagner ?
3. Modifie le programme pour qu'il vérifie d'abord si en jouant une case il peut créer une ligne gagnante.
4. Encore un peu facile ? Modifie le programme pour que l'ordinateur détecte que toi (les Xs) pourrait créer une ligne au tour suivant et bloque cette case.

15 Pour aller plus loin

Voici quelques idées pour aller encore plus loin dans ton jeu :

1. Compte les points et affiche les en haut de l'écran.
2. Ajoute un son quand on essaie de choisir une case occupée.
3. Colorie la ligne gagnante.
4. Améliore encore le jeu de l'ordinateur.

Pour cela, tu auras besoin des éléments suivants :

Afficher du texte à l'écran

Le morceau de code suivant permet d'afficher le texte 'Bonjour' à l'écran. Le deuxième paramètre représente la position du coin supérieur gauche du texte.

```
screen.draw.text('Bonjour', (0,0))
```

Convertir un nombre en texte

Les variables en Python ont un type, un nombre ce n'est pas la même chose que du texte (ou chaîne de caractères, string en anglais). Si tu as une variable qui contient un nombre et que tu veux la transformer en chaîne de caractères, utilise la fonction **`str()`** comme dans l'exemple ci-dessous.

```
nombre = 10
texte = str(nombre)
```

Jouer une note

Pygame Zero offre une fonction pour jouer une note de musique, telle que montré dans le code ci-dessous. Le premier paramètre est la fréquence de la note en Hertz (c'est sa hauteur, un plus petit nombre donne une note plus grave, un plus grand nombre une note plus aiguë), le second paramètre indique la durée de la note en seconde.

```
tone.play(100, 0.1)
```