Flappy Bird IA

DIVERSITY by EPITECH

I - Introduction

Le python est un langage de programmation qui est apparu au début des années 90, il se caractérise par sa structure orienté objet comme le Java mais il est beaucoup plus abordable dans sa syntaxe et il est plus facile d'utilisation pour faire de l'algorithmie !

Il est particulièrement utilisé comme langage de script pour automatiser des tâches simples, comme un script qui récupérerait la météo sur Internet, ou bien du traitement de données.

Concernant PyGame c'est une bibliothèque Python qui aide dans la création de jeux vidéo en temps réel.
Toutefois il n'est pas seulement utilisé pour la création de jeux vidéo il est également utilisé pour toutes applications nécessitant une interface graphique tel qu'un logiciel ou encore une simulation.

II – Consignes

1. Le nom du repo est : flappy_bird_cc

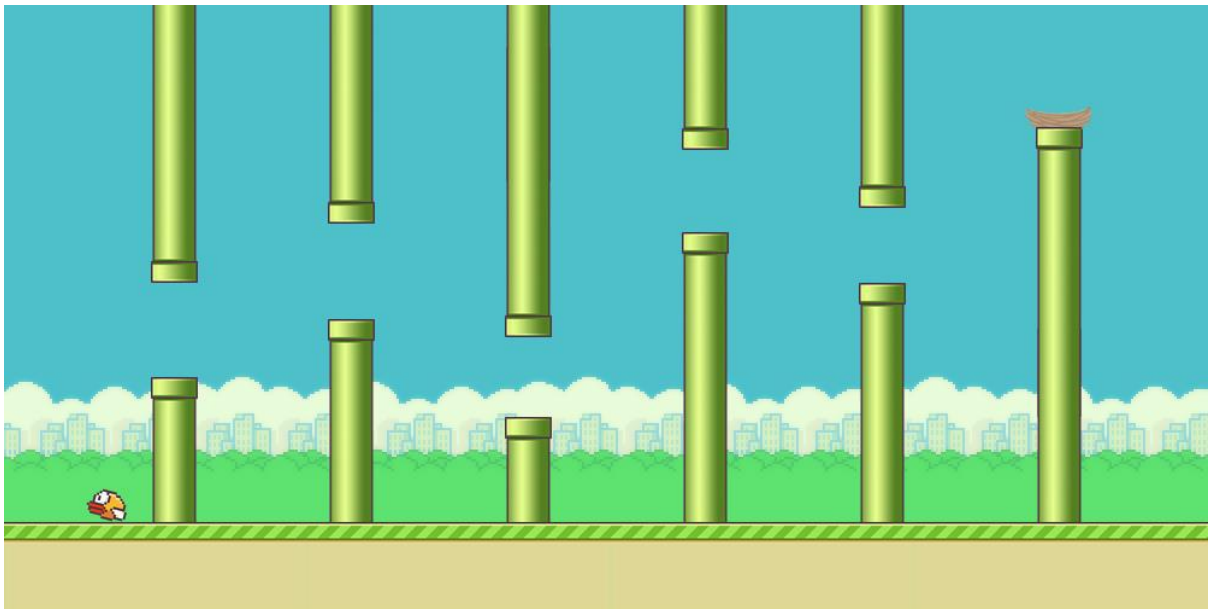
2. En cas de questions, n'hésitez pas à vous entraider et si vous êtes toujours bloqué, un cobra se fera une joie de vous aider.

3. Si l'installation ne se déroule pas bien demandez de l'aide à un cobra.

4. Pour l'installation veuillez suivre le tutoriel « Installation Python et ses outils ».

III – Regarde maman un bébé oiseau !

Vous venez de trouver un oisillon tombé du nid, le pauvre il ne sait même pas voler. Mais vous apercevez au loin un nid formé sur le haut d'un tuyau.
Le seul problème c'est que ce tuyau est trop long et que pour y accéder il faut passer entre plusieurs tuyaux.
Il va donc falloir apprendre à cet oisillon à voler !



IV – Avant de savoir voler il faut savoir marcher

Pour commencer on va s'occuper de la base du jeu pour cela on va mettre les imports. Les imports vont nous permettre d'importer des modules qui nous donnes accès à des fonctions selon nos besoins.

Pour recréer Flappy Bird on va avoir besoin des modules :
- pygame
- random
- math
- os
- time
- pygame.font.init()

On va ensuite donner une taille à la fenêtre, on veut que la taille soit de 500x800 afin d'avoir un affichage de type portrait. Ce qui va donner :

```
WIN_WIDTH = 500
WIN_HEIGHT = 800
```

Après on va afficher une image grâce au module pygame importé plus tôt de cette manière :

```
nom_image = pygame.transform.scale2x(pygame.image.load(os.path.join(« dossier », « image.extension »)))
```

Et enfin pour conclure on va créer un fonction pour afficher la fenêtre, cette fonction servira pour y mettre toutes les différentes images en faisant attention à l'ordre. Ensuite il nous faudra une fonction main qui va exécuter le jeu, initialiser la fenêtre, mettre une clock et on va mettre aussi la possibilité de pouvoir fermer la fenêtre.

Normalement votre code devrait ressembler à cela :

```python
#!/bin/usr/env python3

import pygame
import random
import math             # Permet d'utiliser des fonctions système.
import os
import time
pygame.font.init()

WIN_WIDTH = 500         # Largeur de la fenêtre
WIN_HEIGHT = 800        # Hauteur de la fenêtre

BACKGROUND = pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bg.png")))

def draw_window(win):               # Permet d'afficher les éléments souhaités.
    win.blit(BACKGROUND, (0, 0))    # L'ordre des draw est importante !
    pygame.display.update()         # Permet d'actualiser la fenêtre


def main():
    run = True # Une variable qui nous permettra de savoir quand stoper le jeu
    win = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT)) # Initialisation de la fenêtre
    clock = pygame.time.Clock() # Permet de calculer le temps entre chaque boucle

    while (run):
        clock.tick(30)
        for event in pygame.event.get():
            if event.type == pygame.QUIT: # Ferme la fenêtre si on clique sur la croix
                run = False               # en haut à droite
                pygame.quit()
                quit()
        draw_window(win)

if __name__ == "__main__":
    main()
```

V – L'oiseau est tombé du nid

Maintenant que l'oiseau est tombé du nid, tu vas devoir l'ajouter à ton code. Pour cela, assure-toi de lui faire un bilan médical complet afin de récupérer un maximum de données. Ainsi, tu pourras initialiser les variables de ta class Bird sans soucis.

Tu devrais avoir quelque chose qui ressemble à ça.

```
BIRDS = [pygame.transform.scale2x(pygame.image.load(os.path.join("imgs/", "bird1.png"))),
pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bird2.png"))),
pygame.transform.scale2x(pygame.image.load(os.path.join("imgs", "bird3.png")))]
                                # Création d'un tableau qui contient toutes les images
                                # de notre oiseau
class Bird:
    IMGS = BIRDS
    MAX_ROTATION = 25           # Initialisation des variables utiles
    ROT_VEL = 20
    ANIMATION_TIME = 5

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.tilt = 0
        self.tick_count = 0     # Initialisation des variables de notre class Bird
        self.velocity = 0
        self.height = self.y
        self.img_count = 0
        self.img = self.IMGS[0]

    def jump(self):
        self.velocity = -10.5   # La fonction qui nous permettra de sauter
        self.tick_count = 0
        self.height = self.y

  def move(self):     # Cette nous permet de faire bouger l'oiseau, c'est-à-dire le faire tomber si rien ne se passe
      self.tick_count = self.tick_count + 1
      d = self.velocity * self.tick_count + 0.5 * 3 * self.tick_count**2

      if (d >= 16):
          d = ((d / abs(d)) * 16)
      if (d <= 0):
          d = d - 2
      self.y = self.y + d
      if (d < 0 or self.y < self.height + 50):
          if (self.tilt < self.MAX_ROTATION):
              self.tilt = self.MAX_ROTATION
      else:
          if (self.tilt > -90):
              self.tilt = self.tilt - self.ROT_VEL

  def draw(self, win):    # Cette fonction permet d'animer notre oiseau, le voir battre des ailes
      self.img_count += 1

      if (self.img_count < self.ANIMATION_TIME):              # On commence à la 1er image
          self.img = self.IMGS[0]                             # A chaque tour de boucle on change d'image
      elif (self.img_count < self.ANIMATION_TIME * 2):        # Ici nous sommes à la deuxième image
          self.img = self.IMGS[1]
      elif (self.img_count < self.ANIMATION_TIME * 3):        # 3ème etc...
          self.img = self.IMGS[2]
      elif (self.img_count < self.ANIMATION_TIME * 4):        # Attention, une fois que nois avons atteint notre 3ème
          self.img = self.IMGS[1]                             # image, il faut retourner au début car notre oiseau
      elif (self.img_count == self.ANIMATION_TIME * 4 + 1):   # n'a que 3 images dans son tableau.
          self.img = self.IMGS[0]
          self.img_count = 0

      if (self.tilt <= -80):
          self.img = self.IMGS[1]
          self.img_count = self.ANIMATION_TIME * 2

      rotated_image = pygame.transform.rotate(self.img, self.tilt)
      new_rect = rotated_image.get_rect(center=self.img.get_rect(topleft = (self.x, self.y)).center)
      win.blit(rotated_image, new_rect.topleft)

  def get_mask(self):
      return pygame.mask.from_surface(self.img) # Permettra d'afficher l'oiseau
```

Rappel : Une fois que tu as créé une class, il faut l'ajouter à la fonction main et à la fonction draw_window().

```python
def draw_window(win, bird):          # Permet d'afficher les éléments souhaités.
    win.blit(BACKGROUND, (0, 0))     # L'ordre des draw est importante !
    bird.draw(win)                   # Afficher l'oiseau
    pygame.display.update()          # Permet d'actualiser la fenêtre


def main():
    run = True # Une variable qui nous permettra de savoir quand stoper le jeu
    win = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT)) # Initialisation de la fenêtre
    clock = pygame.time.Clock() # Permet de calculer le temps entre chaque boucle
    bird = Bird(230, 350)      # Création de notre oiseau

    while (run):
        clock.tick(30)
        for event in pygame.event.get():
            if event.type == pygame.QUIT: # Ferme la fenêtre si on clique sur la croix
                run = False               # en haut à droite
                pygame.quit()
                quit()
        bird.move()                       # Déplacement de l'oiseau
        draw_window(win, bird)

if __name__ == "__main__":
    main()
```

VI – L'oiseau qui aimait le vol en rase-motte.

Il ne faut pas oublier que notre oiseau est jeune et qu'il vient de tomber du nid. Il ne volera pas à haute altitude c'est pourquoi, tu vas devoir ajouter la class Base qui représentera le sol de ton jeu.

Tu devrais avoir un code qui ressemble à ça.

```python
class Base:
    VEL = 5
    WIDTH = BASE.get_width()
    IMG = BASE

    def __init__(self, y):
        self.y = y              #Initialisation des variables de notre class base
        self.x1 = 0
        self.x2 = self.WIDTH


    def move(self):
        self.x1 = self.x1 - self.VEL
        self.x2 = self.x2 - self.VEL
                                            # Comme nous n'avons qu'une image base
        if ((self.x1 + self.WIDTH) < 0):    # et que la taille en largeur de la fenêtre
            self.x1 = self.x2 + self.WIDTH  # représente une base. Il faut faire en sorte
                                            # de déplacer une fois l'image vers la gauche
        if ((self.x2 + self.WIDTH) < 0):    # et ensuite réafficher la même image juste après
            self.x2 = self.x1 + self.WIDTH  # pour qu'on ait un effet de déplacement.

    def draw(self, win):
        win.blit(self.IMG, (self.x1, self.y)) # Afficher la base du jeu deux fois
        win.blit(self.IMG, (self.x2, self.y)) # car une seule image représente la base
```

Rappel : Ajouter la class Base à la fonction main et appeler les fonctions move et draw au bon endroit. La valeur 730 sera la position de la base.

Toujours bloqué ? Il faut aussi créer une variable BASE en haut du fichier qui permet de récupérer l'image comme le BACKGROUND ou BIRD.

VII – La vie est vraiment injuste

Comme si ça n'allait déjà pas assez mal pour notre oiseau, voici que maintenant son chemin est semé d'embûches ou de tuyaux devrais-je dire ! Tu vas devoir ajouter la class Pipe à ton code.

Tu devrais avoir quelque chose qui ressemble à ça.

```
class Pipe:
    GAP = 200
    VEL = 5

    def __init__(self, x):
        self.x = x
        self.height = 0
        self.bottom = 0
        self.top = 0              # Initialisation des variables de notre class
        self.PIPE_TOP = pygame.transform.flip(PIPE, False, True)
        self.PIPE_BOTTOM = PIPE
        self.passed = False
        self.set_height()


    def set_height(self):    # Initialisation des positions des tuyaux
        self.height = random.randrange(50, 450) # ceux du haut et ceux du bas
        self.top = self.height - self.PIPE_TOP.get_height()
        self.bottom = self.height + self.GAP

    def move(self):
        self.x = self.x - self.VEL

    def draw(self, win):          # Fonction d'affichage
        win.blit(self.PIPE_TOP, (self.x, self.top))
        win.blit(self.PIPE_BOTTOM, (self.x, self.bottom))

    def collide(self, bird):
```

Il ne faut pas oublier de l'ajouter à la fonction main.

```
def main():
    run = True # Une variable qui nous permettra de savoir quand stoper le jeu
    win = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT)) # Initialisation de la fenêtre
    clock = pygame.time.Clock() # Permet de calculer le temps entre chaque boucle
    bird = Bird(230, 350)      # Création de notre oiseau
    base = Base(730)
    pipes = [Pipe(600)] # Création d'un tableau du tuyau

    while (run):
        clock.tick(30)
        for event in pygame.event.get():
            if event.type == pygame.QUIT: # Ferme la fenêtre si on clique sur la croix
                run = False               # en haut à droite
                pygame.quit()
                quit()

        pipe_ind = 0

        bird.move()
        add_pipe = False
        rem = []

        for pipe in pipes:                  # On vérifie dans tout le tableau si l'oiseau est
            if (pipe.collide(bird)):        # entré en collision avec un tuyau
                main()
            if (not (pipe.passed) and (pipe.x < bird.x)): # Si l'oiseau est passé on ne fait rien pour
                pipe.passed = True                        # le moment
                add_pipe = True

            if (pipe.x + pipe.PIPE_TOP.get_width() < 0):  # Si un tuyau dépasse la fenêtre, on peut le
                rem.append(pipe)                          # supprimer et continuer de déplacer les tuyaux
            pipe.move()

        if (add_pipe):                                    # Si un tuyau a été passé, on peut déjà en ajouter
            pipes.append(Pipe(600))                       # un autre à l'écran

        for r in rem:                   # Suppression d'un tuyau
            pipes.remove(r)
```

Tu as peut-être pu le voir, mais une fois que tu as fini de tout implémenter le code ne fonctionne pas. Pourquoi ? Parce tu dois compléter la fonction collide.

Tu devrais avoir un code qui ressemble à ça.

```python
class Pipe:
    GAP = 200
    VEL = 5

    def __init__(self, x):
        self.x = x
        self.height = 0
        self.bottom = 0
        self.top = 0            # Initialisation des variables de notre class
        self.PIPE_TOP = pygame.transform.flip(PIPE, False, True)
        self.PIPE_BOTTOM = PIPE
        self.passed = False
        self.set_height()


    def set_height(self):   # Initialisation des positions des tuyaux
        self.height = random.randrange(50, 450) # ceux du haut et ceux du bas
        self.top = self.height - self.PIPE_TOP.get_height()
        self.bottom = self.height + self.GAP

    def move(self):
        self.x = self.x - self.VEL

    def draw(self, win):        # Fonction d'affichage
        win.blit(self.PIPE_TOP, (self.x, self.top))
        win.blit(self.PIPE_BOTTOM, (self.x, self.bottom))

    def collide(self, bird):
        bird_mask = bird.get_mask()
        top_mask = pygame.mask.from_surface(self.PIPE_TOP)
        bottom_mask = pygame.mask.from_surface(self.PIPE_BOTTOM)

        top_offset = (self.x - bird.x, self.top - round(bird.y))
        bottom_offset = (self.x - bird.x, self.bottom - round(bird.y))

        b_point = bird_mask.overlap(bottom_mask, bottom_offset)
        t_point = bird_mask.overlap(top_mask, top_offset)

        if (t_point or b_point):
            return True
        return False
```

VIII – C’est un avion ? C’est superman ? Non c’est un oiseau !

Maintenant que tout est en place pour notre oiseau, il va devoir voler ! Pour cela, tu vas devoir récupérer les touches du clavier afin de voir ton oiseau s’envoler au loin… snif comme il grandit vite :’(

De plus, tu devras ajouter un score à ton écran pour chaque tuyau que ton oiseau franchira !

Réponse :

```
def draw_window(win, bird, base, pipes, score):         # Permet d'afficher les éléments souhaités.
    win.blit(BACKGROUND, (0, 0))     # L'ordre des draw est importante !
    for pipe in pipes:
        pipe.draw(win)
    text = STAT_FONT.render("Score: " + str(score), 1,(255, 255, 255))
    win.blit(text, (WIN_WIDTH - 10 - text.get_width(), 10)) # Afficher le score
    base.draw(win)
    bird.draw(win)                   # Afficher l'oiseau
    pygame.display.update()          # Permet d'actualiser la fenêtre

def main():
    run = True # Une variable qui nous permettra de savoir quand stoper le jeu
    win = pygame.display.set_mode((WIN_WIDTH, WIN_HEIGHT)) # Initialisation de la fenêtre
    clock = pygame.time.Clock() # Permet de calculer le temps entre chaque boucle
    bird = Bird(230, 350)      # Création de notre oiseau
    base = Base(730)
    pipes = [Pipe(600)] # Création d'un tableau du tuyau
    score = 0   # On débute la partie avec 0 de score

    while (run):
        clock.tick(30)
        for event in pygame.event.get():
            if event.type == pygame.QUIT: # Ferme la fenêtre si on clique sur la croix
                run = False               # en haut à droite
                pygame.quit()
                quit()
            if event.type == pygame.KEYDOWN: # On vérifie si l'utilisateur a appuyé sur une touche
                if event.key == pygame.K_SPACE: # On vérifie quelle touche, ici c'est espace
                    bird.jump()         # On appelle la fonction jump pour faire voler notre oiseau

        pipe_ind = 0

        bird.move()
        add_pipe = False
        rem = []

        for pipe in pipes:                  # On vérifie dans tout le tableau si l'oiseau est
            if (pipe.collide(bird)):        # entré en collision avec un tuyau
                main()
            if (not (pipe.passed) and (pipe.x < bird.x)): # Si l'oiseau est passé on ne fait rien pour
                pipe.passed = True                        # le moment
                add_pipe = True

            if (pipe.x + pipe.PIPE_TOP.get_width() < 0):  # Si un tuyau dépasse la fenêtre, on peut le
                rem.append(pipe)                          # supprimer et continuer de déplacer les tuyaux
            pipe.move()

        if (add_pipe):                                    # Si un tuyau a été passé, on peut déjà en ajouter
            pipes.append(Pipe(600))                       # un autre à l'écran
            score = score + 1                   # On ajoute +1 au score
```

Tu n'aurais pas encore oublié quelque chose ?

L'initialisation de ta font en haut du fichier, comme pour BASE, BACKGROUND.

Conclusion :

Félicitations ! Tu as réussi à faire voler ton oiseau et à lui faire surmonter les obstacles sur sa route. Cependant, ton oiseau aimerait bien vivre plus de sensations fortes c'est pourquoi, tu devrais ajouter des bonus !

IX – Bonus :

Ajouter une IA qui apprendra à voler toute seule à l'aide de NEAT.

Ajouter des bonus qui rendent invincible votre oiseau.

Ajouter un changement de décor après avoir dépassé un certain score.

Ajouter un effet de parallaxe.