Coding Club

Duck Hunt

Réalisation d'un jeu vidéo en Python avec Pygame

Réalisé par: ***Paul\0 et Sworyz***

1. Introduction

L'objectif d'aujourd'hui sera de réaliser le célèbre jeu vidéo Duck Hunt (Ceux qui vivent dans une grotte et ne connaissent pas le jeu, allez regarder cette vidéo pour voir à quoi ressemble ce jeu).

Pour faire ce jeu nous allons utiliser Python ainsi que Pygame.

C'est quoi Python ?

Python est un langage de programmation. Tout comme les êtres humains utilisent plusieurs langages pour communiquer, en informatique il existe plusieurs langages pour communiquer avec sa machine, et le Python en fait partie.

L'avantage d'utiliser Python ? C'est que c'est un langage rapide à prendre en main, en effet, il suffit d'écrire ce que tu veux faire en anglais et en principe ça devrait fonctionner.

Il comporte néanmoins certaines particularités comparé aux autres langages, comme le fait qu'il est très à cheval sur l'**indentation** (le nombre d'espace – tabulation) et qu'il ne nécessite pas de **compilation**. C'est un langage **interprété**. On n'est pas obligé de convertir le fichier qui contient ton code en un fichier exécutable comme les « .exe ».

C'est quoi Pygame ?

Pygame est une bibliothèque. En informatique une bibliothèque est un ensemble de fonctions, qui, dans notre cas nous permettrons de créer une fenêtre, d'afficher une image à telle endroit, etc…

Si nous n'avions pas cette bibliothèque, il nous faudrait écrire directement en binaire (succession de 0 et de 1) que nous voudrions tel pixel à tel endroit dans notre programme, ce qui serait beaucoup plus fatiguant. Utiliser Pygame vas donc nous simplifier la tâche !

Duck Hunt est un jeu vidéo sorti sur la console NES en 1984 possédant une centaine de niveaux, des modes de jeu et de difficulté variable, etc…

C'est pourquoi dans un premier temps nous nous occuperons seulement de la partie jeu avec un canard qui bouge de gauche à droite sur l'écran.

1.1 Environnement de travail

Fait un dossier **My Game** sur le bureau de ton Ordinateur, c'est dans ce fichier que tu vas mettre tout ce dont tu auras besoin pour ton jeu.

Commence par créer un fichier Python qui te permettras d'écrire ton code (un fichier Python est un fichier dont l'extension est « .py », si tu vois toujours l'icône d'un fichier texte (.txt) ou que tu n'arrives pas à créer le fichier, **demande à un Cobra**, n'est pas honte, ils sont gentils), évidemment tu peux nommer ton fichier python comme tu le souhaites tant qu'il finit bien par « .py ».

Si à un moment dans la journée ton jeu a besoin d'utiliser des images ou des musiques, choisis-les sur Internet puis enregistre les, et n'oublie pas de les mettre dans ton dossier **My Game**.

Pour le moment, ton dossier « My Game » ne comporte que ton fichier Python ! Mais prend garde à comment tu comptes remplir ce dossier, pour éviter d'avoir toutes tes futurs images / musiques / etc… en vrac dans ton dossier, commence dès maintenant à créer un dossier « Image », « Music » où tu mettras toute tes images, musiques, …

Pour modifier votre fichier Python, on te recommande d'utiliser « IDLE », il te permettra d'écrire dans ton fichier tout en lançant ton programme Python.

Pour l'ouvrir, fait un **Clic droit** sur ton fichier Python -> **Edit with IDLE** puis tu peux éditer ton fichier ! Pour le lancer tu n'auras qu'a appuyer sur **Run** -> **Run Module** (en haut d'IDLE).

Il existe plusieurs autres « éditeur de texte », si tu es plus familier avec Notepad++, Sublime Text, Visual Code voire même le Bloc Note de Windows, tu peux les utiliser.

2. Créer son jeu

2.1 Ouvrir sa première fenêtre

Maintenant on arrive à la question que vous vous posez tous : Comment ouvre-t'on une fenêtre avec Pygame (et Python) ?

Nous avons besoin d'ouvrir une fenêtre pour notre jeu car on ne peut pas coller et faire bouger des images ou du texte comme ça sur notre machine, il nous faut au préalable ouvrir une fenêtre où l'on pourra y faire tout ce que l'on veut !

Pour ouvrir une fenêtre copie et colle ce code dans ton fichier Python.

```
# Import permet d'importer une bibliothèque pour utiliser ses fonctions, ici Pygame.
import pygame

# On initialise Pygame, chaque fonction de Pygame commence par « pygame. »
pygame.init()

# On fait une variable window de taille 800x600 pixels, ce seras notre fenêtre.
window = pygame.display.set_mode( (800, 600) )
# On assigne la phrase « Duck Hunt » à notre fenêtre.
pygame.display.set_caption("Duck Hunt")
# On fait une variable qui s'appelle « leave », et on lui assigne « False ».
leave = False

# While signifie « tant que », c'est une boucle, cela veut dire que tant que « leave »
# est égal à False, tout ce qui se trouve dans la boucle while va se ré-exécuté.
while not leave:
    # Ici Pygame cherche les évènements qui ont eu lieu lors de ce tour de boucle
    for event in pygame.event.get():
        # Si l'évènement est pygame.QUIT on met la variable « leave » a True
        if event.type == pygame.QUIT:
            leave = True

# Si on arrive ici c'est que la variable « leave » a été mis sur True
# On quitte Pygame et on termine le programme.
pygame.quit()
quit()
```

Attention ! Avec ce code vous devriez avoir seulement une fenêtre vide (noir ou blanche) qui ne fait rien d'autre que de se fermer quand vous cliquez sur la croix.

2.2 Le Temps

On a de la chance car Pygame gère le temps ! En programmation l'une des meilleurs pratique consiste à utiliser le temps pour « timer » ses actions, pour faire bouger ses images toute les X millisecondes par exemple.

Si on ne fait pas ça, la vitesse d'exécution du programme dépendra du processeur, donc entre cet ordinateur et celui de votre grand-mère, la vitesse ne serait pas la même, ce qui serait problématique quand vous voudrez lui montrer votre super jeu sur son Ordinateur !

2.3 Les Evènements

Pygame intègre une notion d'Event (Evènement en français). Pour faire simple c'est une information qui va déclencher une action.

Exemple : Si vous ne faites rien il n'y aura pas d'évènement, mais dès lors que tu appuieras sur une touche de ton clavier, bougeras ta souris, etc… un évènement lié à une action surviendras !

Essayes de copier-coller ce code dans ton fichier Python, et regardes les évènements qui apparaissent dans ta console.

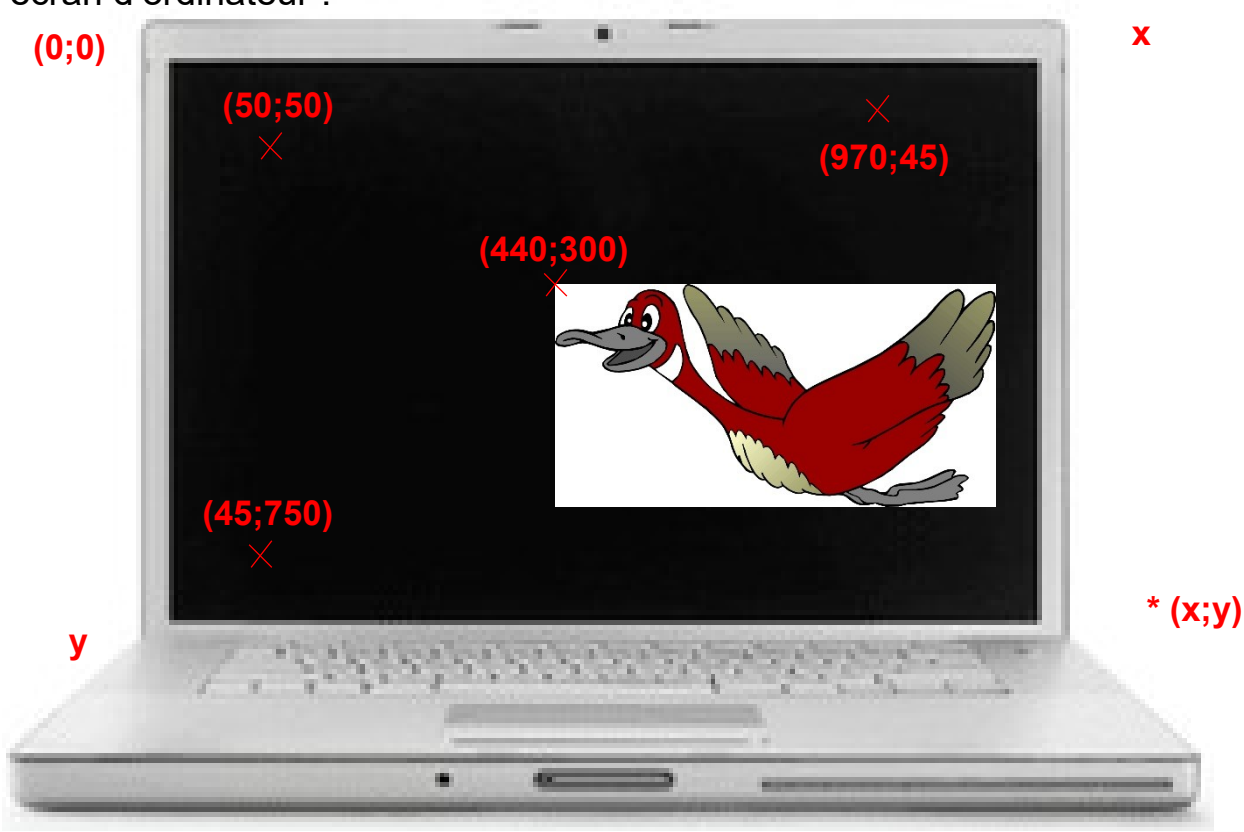
```
import pygame

pygame.init()

window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Mon super jeu vidéo")
# Fait une variable appelé « clock », ça va être notre compteur de temps.
clock = pygame.time.Clock()
leave = False

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
        # Affiche l'évènement trouvé dans la console.
        print(event)
    # Permet de limiter à 60 par seconde le nombre de fois qu'on passe
    # dans le while. On exécutera le code dans le while 1 fois toutes
    # les 1 / 60 = 0.17 ms (signifie une limitation à 60 FPS).
    clock.tick(60)

pygame.quit()
quit()
```

2.4 Les Images

Nous voilà enfin à la partie que vous attendiez tous ! Ainsi que la partie la plus intéressante de Pygame, j'ai nommé : La gestion des images !

Avant d'en dire plus, voici un petit récapitulatif du système de coordonnées en informatique:

Voyez-vous les graphs qu'on utilise pour représenter les fonctions en cours de maths ? (Si tu ne vois pas ou t'en souviens pas t'en fait pas, appelle un Cobra, **il sera ravi de t'éclairer**). Eh bien en informatique on utilise le même système de coordonnées, si ce n'est que l'axe des abscisses (y) est inversé ! Voici quelques coordonnées de point sur un écran d'ordinateur :



Attention ! En informatique les coordonnées d'une image ne sont représenté que par 1 seul point: celui le plus en haut à gauche de celle-ci (Soit (440 ; 300) dans notre exemple).

Ce qui signifie que pour savoir si un pixel se trouve dans la zone de notre image on doit vérifier si le point en x est entre la position en x de l'image et sa position en x + la taille de l'image en x (et pareille pour l'axe des abscisses) (Appelle un Cobra pour plus de détails si tu ne comprends pas)

Pour afficher une image sur ton écran, il te faudra passer par 2 étapes, la première sera de **blit** (Signifie « Apparaitre) ton image sur ta fenêtre, puis d'**actualiser** celle-ci.

Pour bien comprendre, imagine que ta fenêtre est une scène de théâtre. Quand tu blit quelque chose sur ta fenêtre, tu places quelque chose sur ta scène, mais pour que le public voit les changements, il te faudra ouvrir les rideaux (l'actualisation).

Si toi aussi t'as pas compris, appelle un cobra ou clic ici \'o'/

Entre chaque actualisation, l'image affichée resteras la même, donc chaque blit a l'écran ne sera affiché qu'une fois la fenêtre actualisée, mais attention ! L'ancienne image seras toujours présente, donc pense bien à effacer l'ancienne avant de réafficher ton image (dans le cas d'un déplacement). Ça te permettra d'éviter l'effet de « trainé ».

Si tu blit plusieurs fois avant d'update, pense bien que si tu mets une image sur la position d'une autre, c'est la dernière blit qui sera par-dessus !

```
import pygame

pygame.init()

window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Mon super jeu vidéo")
# Créer une variable contenant l'image Background.png (situé dans
# notre dossier « My Game/Image »).
background = pygame.image.load("Image/Background.png")
clock = pygame.time.Clock()
leave = False

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
    clock.tick(60)
    # Blit "background" en position (0;0).
    window.blit(background, (0, 0))
    # Rafraichis la fenêtre.
    pygame.display.update()

pygame.quit()
quit()
```

2.5 Le canard

Vous l'attendiez tous ! La star apparaaaaît !

Vieille référence a Tarzan ou Tok arrive en citant cette phrase…

Avec tout ce qui a été dit, vous devriez être en mesure d'afficher et de faire bouger votre canard, mais voici comment faire pour les moins confiant d'entre vous :

```
import pygame

pygame.init()

window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Mon super jeu vidéo")
background = pygame.image.load("Image/Background.png")
# Créer une variable contenant l'image Duck.png (situé dans
# Notre dossier « My Game/Image »)
duck = pygame.image.load("Image/Duck.png")
clock = pygame.time.Clock()
leave = False
position_x = 0
position_y = 0

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
    position_x = position_x + 1
    clock.tick(60)
    window.blit(background, (0, 0))
    # Blit "background" aux coordonnées (position_x ; position_y).
    window.blit(duck, (position_x, position_y))
    pygame.display.update()

pygame.quit()
quit()
```

As-tu remarqué ? Le canard file à toute allure puis disparais ! Pour être plus précis, il ne disparait pas, il continue sa route en dehors de l'écran mais nous ne le voyons plus !

Comment faire pour qu'il réapparaisse à gauche de l'écran quand il a atteint l'autre bord ?

La réponse est très simple ! Il suffit de vérifier les coordonnées du canard à chaque tour de boucle, et si ces dernières ont été dépassés au niveau du bord droit de la fenêtre, pouf ! On remet la **position_x** de notre canard à 0.

On peut même utiliser **random** pour nous donner un nombre aléatoire pour placé notre canard après sa « mort » sur **position_y** !

```
import pygame
# Importe seulement la fonction randint de la bibliothèque random.
from random import randint

pygame.init()

window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Mon super jeu vidéo")
background = pygame.image.load("Image/Background.png")
duck = pygame.image.load("Image/Duck.png")
clock = pygame.time.Clock()
leave = False
position_x = 0
position_y = 0

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
    position_x = position_x + 1
    # Si la position_x du canard à dépasser la taille en x de la fenêtre.
    if position_x >= 800:
        # Met la position en x du canard a -300, donc en dehors de l'écran
        # Pour pouvoir le voir arriver
        position_x = -300
        # randint nous donneras un nombre aléatoire entre 0 et 600 – 110
        # 600 représente la taille max de la fenêtre, et on enlève 110 pour
        # Eviter d'avoir un canard qui « mange le sable ».
        position_y = randint(0, 600 - 110)
    clock.tick(60)
    window.blit(background, (0, 0))
    window.blit(duck, (position_x, position_y))
    pygame.display.update()

pygame.quit()
quit()
```

3. Détection du clic de la souris

Maintenant il serait temps de « tuer » le canard ne trouvez-vous pas ? C'est possible grâce à 2 fonctions de Pygame :

- pygame.mouse.get_pos()
- pygame.mouse.get_pressed()

```
import pygame
from random import randint

pygame.init()

window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Duck Hunt")
background = pygame.image.load("Image/Background.png")
duck = pygame.image.load("Image/Duck.png")
clock = pygame.time.Clock()
leave = False
position_x = 0
position_y = 0

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
    # Assigne les variables mouse_x/y aux coordonnées en x/y de la souris.
    mouse_x, mouse_y = pygame.mouse.get_pos()
    # Si la souris est sur un pixel de l'image et que l'utilisateur clic, remet
    # les coordonnées du canard en dehors de l'écran.
    if (mouse_x >= position_x and mouse_x <= position_x + 110
    and mouse_y >= position_y and mouse_y <= position_y + 110
    and pygame.mouse.get_pressed()[0] is 1):
        position_x = 0
        position_y = randint(0, 600 - 110)
    position_x = position_x + 1
    if position_x >= 800:
        position_x = -300
        position_y = randint(0, 600 - 110)
    clock.tick(60)
    window.blit(background, (0, 0))
    window.blit(duck, (position_x, position_y))
    pygame.display.update()

pygame.quit()
quit()
```

Mais du coup, comment faire la différence entre quand il franchit l'écran et quand on lui clic dessus ?! Il nous faudrait un système de score ou de vie !

Il est enfin temps que notre jeu ressemble à un vrai jeu (avec un système de score ET de vie, histoire d'avoir un but, sinon ce n'est pas drôle). Pour cela, il nous suffit de rajouter 2 variables, l'une désignant notre vie : on perdrait 1 point de vie par canard qui franchis l'écran. Le second de score qui compterais le nombre de canard qu'on a plumé !

```
import pygame
from random import randint

pygame.init()
window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Mon super jeu vidéo")
background = pygame.image.load("Image/Background.png")
duck = pygame.image.load("Image/Duck.png")
# Définition la police du texte (arial) ainsi que sa taille (32 pixel).
font = pygame.font.SysFont("arial", 32)
clock = pygame.time.Clock()
leave = False
position_x = 0
position_y = 0
score = 0
life = 3

while not leave:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            leave = True
    mouse_x, mouse_y = pygame.mouse.get_pos()
    if (mouse_x >= position_x and mouse_x <= position_x + 110
    and mouse_y >= position_y and mouse_y <= position_y + 110
    and pygame.mouse.get_pressed()[0] is 1):
        position_x = 0
        position_y = randint(0, 600 - 110)
        # Incrémente le score vu qu'on a touché un canard.
        score = score + 1
    # Si on a plus de vie, on quitte le jeu.
    if life <= 0 :
        leave = True
    position_x = position_x + 1
    if position_x >= 800:
        position_x = -300
        position_y = randint(0, 600 - 110)
        # Décrémente les points de vie vu qu'on en a laissé filé un.
        life = life – 1
    # Crée les textes, d'épaisseur 3 et de couleur (255, 255, 255), soit blanc.
    text_score = font.render(str(score), 3, (255, 255, 255))
    text_life = font.render(str(life), 3, (255, 255, 255))
    clock.tick(60)
    window.blit(background, (0, 0))
    window.blit(duck, (position_x, position_y))
    # Blit le score et la vie aux positions 10;10 pour le score et 500;10 pour la vie.
    window.blit(text_score, (10, 10))
    window.blit(text_life, (500, 10))
    pygame.display.update()

pygame.quit()
quit()
```

4. Système de texte

Si tu as scroll jusqu'ici pour voir à quoi ressemble la fin, voici une photo d'Sarko qui fait du cheval (histoire que tu ne sois pas venu jusqu'ici pour rien =D)

Ah ! Tu as fini… Et bien dans ce cas utilise le temps qu'il te reste pour faire ton propre jeu vidéo ! Genre une sorte de mini RPG ou tu déplacerais ton personnage avec les touches du clavier, et pourrais ainsi tuer les méchants champignons qui ont kidnappé la princesse ! (Mario… Oui oui)

Fin sérieux quoi, tu es venu au Coding Club pour faire « un jeu vidéo en Python », pas pour faire un simple Duck Hunt, donc lance toi !