



# Coding Club

---

## Duck Hunt



Réalisation d'un jeu vidéo en Python avec Pygame

Réalisé par: Paul\0



# 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 le jeu).

Pour cela nous allons utiliser Python ainsi que Pygame.

## C'est quoi Python ?:

Python est un langage. 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 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 comparées aux autres langages, comme le fait qu'il est très à cheval sur l'**indentation** (le nombre d'espace) et qu'il ne nécessite pas de **compilation** (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 c'est un ensemble de fonctions, qui dans notre cas nous permettront de créer une fenêtre, d'afficher une image à tel endroit, etc...

Si nous n'avions pas cette bibliothèque, il nous faudrait écrire directement en binaire (succession de 0 et de 1) ce que nous voudrions telle pixel à tel endroit de notre machine, ce qui est beaucoup plus fatigant. Utiliser Pygame va 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 de l'écran.**



## 1.1 Environnement de travail

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

Commence à créer un fichier Python qui te permettra 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, **appelle un Cobra =D**, n'est pas honte, ils sont gentil), 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 sur le PC, 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 toute tes futures images / musiques / etc... en vrac dans ton dossier, commence dès maintenant à créer un dossier « Image », « Music » ou tu mettras toute tes images, musiques, ...

Pour modifier votre fichier Python, nous vous recommandons d'utiliser « l'IDLE », il vous permettra d'écrire dans votre fichier tout en lançant votre 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'à appuyer sur **Run** -> **Run Module** (en haut de l'IDLE).



Il existe plein d'autres « éditeurs de texte », si vous êtes plus familiers avec Notepad++, Sublime Text, Visual Code voir même le Bloc Note de Windows, vous pouvez les utiliser.



## 2. Créer son jeu

### 2.1 Ouvrir sa première fenêtre

Maintenant on arrive à la question que vous vous posez tous :  
Comment ouvrir une fenêtre avec Pygame ?

Nous avons besoin d'ouvrir une fenêtre pour notre jeu car on ne peut pas cliquer 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 copier-coller ce code dans votre fichier Python.



```
# Import 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 crée une variable window de taille 800x600 pixels, ce sera notre fenêtre.
window = pygame.display.set_mode( (800, 600) )
# On assigne la phrase « Mon super jeu vidéo » à notre fenêtre.
pygame.display.set_caption("Mon super jeu vidéo")
# On crée une variable qui s'appelle « leave », et on dit qu'elle est égale à « False ».
leave = False

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

# (Si on arrive ici c'est que la variable « leave » a été mise à 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 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 meilleures pratiques consiste à utiliser le temps pour « timer » ses actions, pour faire bouger ses images toutes 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 sera problématique quand vous voudrez lui montrer votre super jeu sur son PC :'

## 2.3 Les Événements

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

**Exemple :** Si vous ne faites rien il n'y aura pas d'événement, mais dès lors que tu appuieras sur une touche de ton clavier, bougeras ta souris, etc... un événement relié à chaque action surviendra !

Essayer de copier-coller ce code dans votre fichier Python, et regarder les événements qui apparaissent dans votre console.



```
import pygame

pygame.init()

window = pygame.display.set_mode( (800,600) )
pygame.display.set_caption("Mon super jeu vidéo")
# Crée une variable appelée « 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'événement 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 = 0.017 ms (Ça revient à se limiter à 60 FPS).
    clock.tick(60)

pygame.quit()
quit()
```

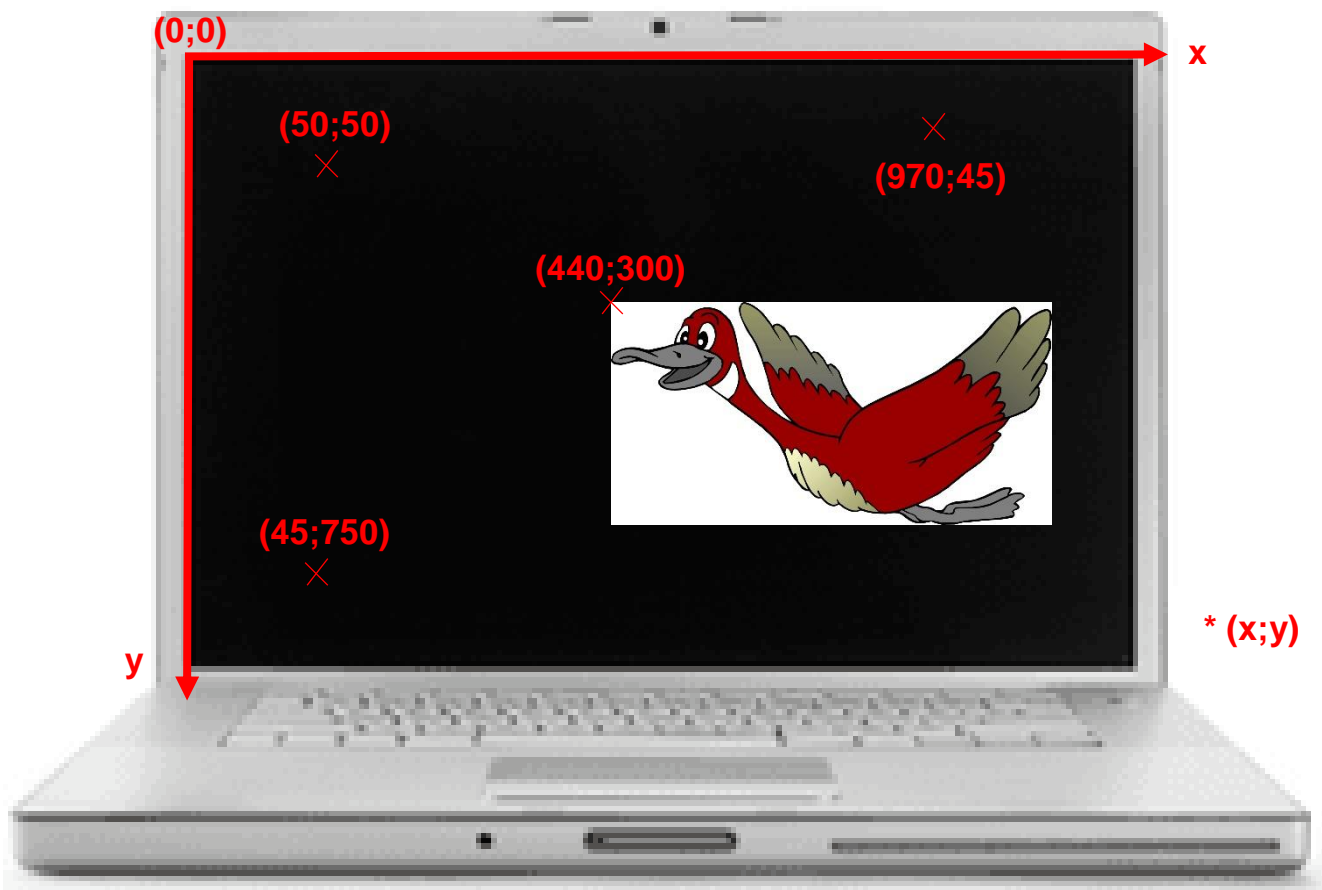


## 2.4 Les Images

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 =D**) Et bien en informatique on utilise le même système de coordonné, 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 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 =D)



Pour affiché une image sur votre écran, il vous faudra passer par 2 étapes, la première seras de **blit** (Signifie « Apparaître) votre image sur votre fenêtre, puis d'**actualisé** celle-ci.

Pour bien comprendre, imaginé que votre fenêtre est une scène de théâtre, et que quand vous blitez quelque chose sur votre fenêtre, vous placé quelque chose sur votre scène, mais pour que le public voit les changements, il vous faudra ouvrir les rideaux (l'actualisation).



Si toi aussi ta pas compris, appelle un cobra =D ou clic [ici](#) \o/



Entre chaque actualisation, l'image affiché resteras la même, donc chaque blit a l'écran ne seras affiché qu'une fois la fenêtre actualisé, mais attention ! L'ancienne image seras toujours présente, donc pense bien a effacé l'ancienne avant de réaffiché ton image (dans le cas d'un déplacement). Ça te permettras d'évité 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 apparaaaaait !



*Vieille référence a Tarzan ou Tok arrive en citant cette phrase...*

Avec tout ce qui a été dit, vous devriez être en mesure d'affiché et 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 disparaît ! Pour être plus précis, il ne disparaît 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 celle-ci ont dépassé celle 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 placer notre canard après sa « mort » !



```
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 a dépassé la taille en x de la fenêtre.
    if position_x >= 800:
        # Met la position en x du canard à -300, donc en dehors de l'écran
        # Pour pouvoir le voir arriver
        position_x = -300
        # randint nous donnera 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 « tué » 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("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
    # 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 !



## 4. Système de texte

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, et on perdrait 1 point de vie par canard qui franchis l'écran, et 1 de score qui compterait 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")
# Définit 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 filer 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()
```



**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 tout 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 !**

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