Partie 1

Interface Homme-Machine en Python avec Pygame - documents

Objectif : Comprendre les bases des interfaces graphiques et du paradigme événementiel pour préparer la création d'un jeu type escape game/RPG

Pygame est une bibliothèque Python spécialement conçue pour créer des jeux 2D et des applications graphiques interactives. Pygame propose une gestion simplifiée des graphismes, sons et événements

---

1.1 Pygame : Les événements

Un **événement** est une action effectuée par l'utilisateur (ou par le système) que le programme peut détecter et traiter.

En langage python, le type d'évènement est accessible à l'aide de l'instruction : `event.type`. On compare alors `event.type` à l'évènement observé :

```
# retourne True si le bouton souris est appuyé
event.type == pygame.MOUSEBUTTONDOWN
```

```
# True si le bouton est relaché
event.type == pygame.MOUSEBUTTONUP
```

Types d'événements courants : - `MOUSEBUTTONDOWN` : L'utilisateur clique avec la souris - `MOUSEBUTTONUP` : L'utilisateur relâche le bouton de la souris - `MOUSEMOTION` : L'utilisateur déplace la souris - `KEYDOWN` : L'utilisateur appuie sur une touche du clavier - `KEYUP` : L'utilisateur relâche une touche - `QUIT` : L'utilisateur ferme la fenêtre

1.2 Structure minimale d'un programme Pygame

Voici le squelette de base que vous utiliserez pour tous vos projets Pygame :

```
import pygame

# ============================================
# INITIALISATION
# ============================================
pygame.init()

# Création de la fenêtre: ZONE 1
LARGEUR = 600
HAUTEUR = 600
screen = pygame.display.set_mode((LARGEUR, HAUTEUR))
pygame.display.set_caption("Mon premier programme Pygame")

# Horloge pour contrôler les FPS (images par seconde)
```

```
clock = pygame.time.Clock()

# Variable de contrôle de la boucle: ZONE 2
running = True

# ============================================
# BOUCLE PRINCIPALE
# ============================================
while running:
    # GESTION DES ÉVÉNEMENTS: ZONE 3
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # LOGIQUE DU JEU: ZONE 4
    # (mise à jour des positions, calculs, etc.)

    # AFFICHAGE: ZONE 5
    screen.fill((0, 0, 0))  # Efface l'écran avec du noir

    # Dessiner vos éléments ici

    # Met à jour l'affichage
    pygame.display.flip()

    # CONTRÔLE DU FRAMERATE
    clock.tick(60)  # 60 images par seconde

# ============================================
# FERMETURE PROPRE
# ============================================
pygame.quit()
```

Partie 2

Exercice 1 : Position dans la fenêtre graphique

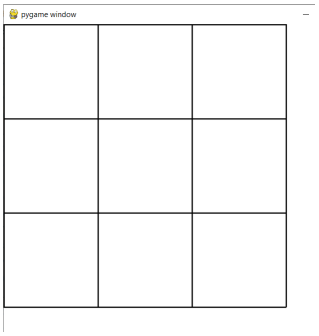

FIGURE 1 – jeu du morpion avec pygame

1. Coordonnées absolues On ouvre une fenêtre pygame de dimension 600×600. Déterminer les positions des centres de chacune des cases. Nommer les coordonnées de ces centres de la manière suivante : x1, y1, x2, y2, ... x9, y9.

2. Coordonnées calculées. La fenêtre est ouverte à l'aide de l'instruction `screen = pygame.display.set_mode((LARGEUR, HAUTEUR))`. Determiner cette fois les centres de chaque case à l'aide d'une expression mathématique, fonction de LARGEUR, HAUTEUR.

3. Dessiner un rectangle

La fonction `pygame.draw.rect(screen, color, (x, y, width, height))` dessine un rectangle. `(x, y, width, height)` est un tuple Python.

`x` et `y` sont les coordonnées du coin supérieur gauche. `width` et `height` sont la largeur et la hauteur du rectangle.

Par exemple, l'instruction `pygame.draw.rect(screen, (255,255,255), (0, 0, 50, 150))` va placer un rectangle blanc dans le coin supérieur gauche de la fenêtre.

Quelle instruction va placer un rectangle blanc au centre de la fenêtre 600×600 ? Ce rectangle doit être *centré*.

4. Ajouter une image Pour placer une image dans la fenêtre, on écrit une série d'instructions qui vont charger l'image (`load`), puis l'afficher à la position voulue (`blit`)

```
# charger l'image
door = pygame.image.load("datas/images/door1.png")
# Dessiner l'image dans le coin supérieur gauche
screen.blit(door, (0,0))
```

L'image *door1.png* a pour dimension $50times150$ pixels. Quelle instruction va placer l'image au centre de la fenêtre 600×600 ? L'image doit être *centrée*.

Partie 3

Exercice 2 : gestion des évènements

3.1 Couleurs

On peut effacer l'écran en le remplissant d'une couleur. Les couleurs en Pygame sont des tuples RGB : `(Rouge, Vert, Bleu)` avec des valeurs de 0 à 255.

Dans la zone 3, on place les lignes suivantes :

```
        if event.type == pygame.MOUSEBUTTONDOWN:
            couleur_fond = (random.randint(0, 255),
                            random.randint(0, 255),
                            random.randint(0, 255))
```

Dans la zone 5 on place :

```
    screen.fill(couleur_fond)
```

1. Que se passe t-il lorsque l'on clique dans la fenêtre ?
2. Quelles sont les couleurs associées aux différents tuples ci-dessous ?

(0, 0, 0)	(255, 255, 255)	(255, 0, 0)	(0, 255, 0)	(0, 0, 255)	(0, 127, 127)

3. Proposez une combinaison qui donnera un jaune d'intensité moyenne
4. Proposez une combinaison qui donnera un gris moyen.

3.2 Interactions avec des zones de la fenêtre

Pour détecter un clic sur une zone précise, on utilise l'attribut `event.pos` (une variable associée à l'objet `event`). Il s'agit d'un tuple, qui se manipule de la manière suivante :

```
# Dans la boucle d'événements (ZONE 3)
if event.type == pygame.MOUSEBUTTONDOWN:
    x, y = event.pos  # Position du clic

    # Vérifier si le clic est dans une zone rectangulaire
    # Zone de l'objet : x entre 275 et 325, y entre 225 et 375
    if 275 <= x <= 325 and 225 <= y <= 375:
        print("Vous avez cliqué sur l'objet !")
```

1. Quelles instructions permettent de changer la couleur de la porte (une forme rectangle 50×150 centrée dans la fenêtre 600×600) lorsque l'on clique sur celle-ci (et que l'on maintient appuyé), mais qui remet la première couleur lorsque l'on relâche le bouton. Préciser la zone dans laquelle vous mettez ces instructions.
2. Quelles instructions permettent de changer la couleur de la porte lorsque l'on clique sur celle-ci. Il faudra à nouveau cliquer sur la porte pour retrouver la couleur d'origine.

3.3 Interaction avec les objets

Dans ce nouveau programme, on utilise une image .png pour représenter la porte.

- On place celle-ci dans la fenêtre à l'aide des instructions suivantes (zone 1) :

```
# charger l'image
door = pygame.image.load("datas/images/door1.png")
# creation d'une surface rectangle (un calque)
# que l'on nomme rect
```

```
# de meme dimension que l'image door
rect = door.get_rect()
# Position de rect dans la fenetre graphique (centré)
rect.center = LARGEUR//2, HAUTEUR//2
# Dessiner l'image à la même position que le calque
screen.blit(door, rect)
```

- Dans la boucle d'événements (zone 3), on utilise cette fois la méthode `collidepoint` associée au calque `rect` :

```
if event.type == pygame.MOUSEBUTTONDOWN:
    if rect.collidepoint(event.pos):
        print("Objet cliqué !")
```

1. On dispose de 2 images d'une même porte, aux contours différents. Quelles sont les instructions qui permettent de changer l'image de la porte lorsque l'on clique sur celle-ci (et que l'on maintient appuyé), mais qui remet la première image lorsque l'on relâche le bouton :
2. Quelles sont les instructions qui permettent de changer l'image de la porte lorsque l'on clique sur celle-ci. Il faudra à nouveau cliquer sur la porte pour retrouver l'image d'origine.

Partie 4

Exercice 3 : Afficher du texte

- Première étape : On créé un objet `pygame.font`, pour lequel on choisit une police de caractères (par defaut, laisser `None`) et une taille pour cette police :

```
my_font = pygame.font.SysFont('Comic Sans MS', 30)
```

- il faut créer une surface sur laquelle on écrit, à l'aide de la fonction `render` :

```
render(text, antialias, color, background=None) -> Surface
```

Exemple :

```
couleur = (255, 255, 255)
texte = "Hello World"
texte_surface = my_font.render(texte, True, couleur)
```

- Troisième étape : Afficher et positionner la surface avec la fonction `blit` :

```
screen.blit(texte_surface, (0,0))  # Position du texte
                                   # en haut à gauche
```

Exemple complet :

```
screen.fill(255,255,255)
font = pygame.font.Font(None, 36)  # None = police par défaut, 36 = taille
texte = "Bienvenue dans la salle mystérieuse"
texte_surface = font.render(texte, True, (0,0,0))  # True = antialiasing
screen.blit(texte_surface, (50, 500))  # Position du texte
```

Remarques :

- Chaque fois que le texte change, il faudra créer une nouvelle surface à placer sur l'ancienne.
- Pour changer de ligne, on ne pourra pas utiliser le symbole "\n". Il faudra créer une nouvelle surface pour chaque ligne, et la positionner de manière correcte.

1. Quelles instructions vont permettre d'afficher les 2 lignes de texte comme ci-dessous ?

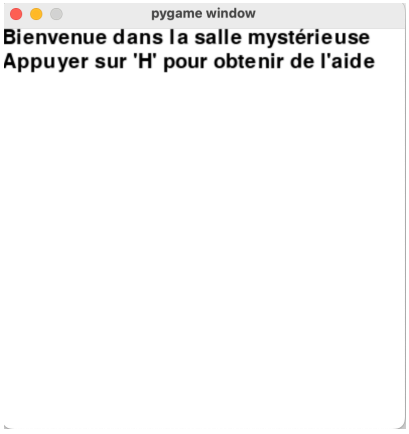

FIGURE 2 – Bienvenue dans la salle mysterieuse. Appuyer sur H pour obtenir de l aide.

2. Detecter l'appui sur une touche du clavier

Pour detecter l'appui sur la touche `"h"` du clavier, on utilisera la séquence d'instruction suivantes :

```
        if event.type == pygame.KEYDOWN:
            if pygame.key.name(event.key) == "h":
                print('HELP')
```

Modifier ces instructions pour afficher l'aide à la position (0,300), lorsque le joueur le demande. On suppose que l'aide est un texte d'une seule ligne, placé dans la variable `aide`.