SNT - ELLA

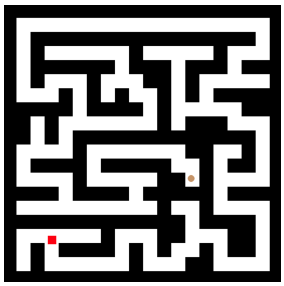
PROGRAMMATION PYTHON

SÉANCE 6
PROGRAMMER UN JEU EN PYTHON

Nom : .... **Classe :** .....

Le module **pygame** permet de programmer facilement des jeux en python.

PARTIE 1: STRUCTURE DU PROJET



• La structure de notre programme va être une boucle qui se répète indéfiniment :

```
...
run = True
while run:      # boucle infinie
    # On dessine l'écran
    # On dessine le décor
    # On dessine le personnage et le trésor
    # On dessine les murs...
```

PARTIE 2: CRÉATION DU LABYRINTHE

1 IMPORTATION DES BIBLIOTHÈQUES NÉCESSAIRES

• Pour pouvoir utiliser Pygame, il suffit d'importer les modules suivants en début de programme.

```
import pygame
from pygame.locals import *
```

2 CRÉATION DU LABYRINTHE

• Pour commencer, on définit le labyrinthe et la taille des éléments du décor.

```
LABYRINTHE = ["XXXXXXXX",          # Labyrinthe est carré
              "X   X TX",          # Le départ est en D
              "X X X XX",          # Le trésor est en T
              "X X X XX",          # Les murs sont des X
              "X X   XX",
              "X XXXXXX",
              "X     DX",
              "XXXXXXXX"]
LARGEUR = 700
HAUTEUR = LARGEUR
TAILLE_ECRAN = (LARGEUR, HAUTEUR)
TAILLEMUR = LARGEUR//len(LABYRINTHE)
TAILLEJOUEUR = TAILLEMUR * 70//100            # La taille du joueur est 70% de celle des murs
TAILLETRESOR = TAILLEMUR * 50//100            # La taille du tresor est 50% de celle des murs
```

2 CRÉATION DE LA CLASSE MUR PERMETTANT DE FABRIQUER ET D'AFFICHER UN MUR

```
class Mur:
    def __init__(self):
        self.x = 0
        self.y = 0
        self.couleur = ( 0, 0, 0 )
        self.taille = TAILLEMUR

    def dessine(self):
        pygame.draw.rect(screen, self.couleur, (self.x, self.y,  self.taille,  self.taille))
```

Quand on créera un **mur** avec la commande `mur = Mur()`,
par défaut, ce mur aura les propriétés définies par la méthode **`__init__(self)`**.

Ainsi :

- `mur.x` et `mur.y` donneront la position du mur sur l'écran
- `mur.dessine()` permettra de dessiner le mur

3 CRÉATION DU LABYRINTHE

On va créer une fonction qui va parcourir LABYRINTHE et fabriquer une liste contenant tous les murs du labyrinthe.

• Pour cela, à la suite du code précédent, il suffit de saisir :

```
def fabriqueLabyrinthe():
    y = 0
    for ligne in LABYRINTHE:
        x = 0
        for case in ligne:
            if case == "X":
                mur = Mur()             #On crée un nouveau mur
                mur.x = x
                mur.y = y
                listeMurs.append(mur)
            x = x + TAILLEMUR
        y = y + TAILLEMUR
```

4 AFFICHAGE DU LABYRINTHE

On va utiliser une boucle infinie qui va afficher le labyrinthe.

• Pour cela, à la suite du code précédent, il suffit de saisir :

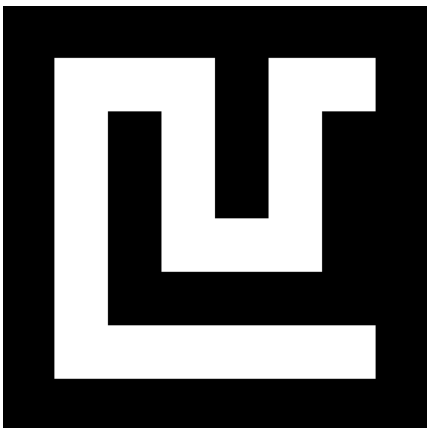
```
pygame.init()                                   # On initialise PYGAME
screen = pygame.display.set_mode(TAILLE_ECRAN)  # On crée un écran d'affichage
listeMurs = []                                  # Cette liste contiendra tous les murs du labyrinthe
fabriqueLabyrinthe()                            # On remplit la liste listeMurs
Horloge = pygame.time.Clock()                   # On crée  une horloge d'affichage
run = True
while run:                                      # Boucle infinie
    Horloge.tick(60)                            # On effectue la boucle 60x/seconde
    screen.fill((255, 255, 255))                # On colorie tout l'écran en blanc
    for mur in listeMurs:
        mur.dessine()

    for evenement in pygame.event.get():        # Fin du jeu en fermant la fenêtre
        if evenement.type == QUIT:
            run = False

    pygame.display.update()                     # On rafraichit l'affichage
```

Sauvegarder le programme sous le nom **jeu.py** avant de poursuivre.

DEPÔT 1 **déposer jeu.py sur https://entraide-ella.fr**



Sur le modèle de la classe Mur, nous allons créer la classe Joueur qui définit les attributs et méthodes du joueur.

1 LA CLASSE JOUEUR

• Pour commencer, à la suite de la classe **Mur**, il faut définir la classe **Joueur** à l'aide du code ci-dessous.

```
class Joueur:
    def __init__(self):
        self.x = 0
        self.y = 0
        self.vx = 0    #Vitesse horizontale
        self.vy = 0    #Vitesse verticale
        self.taille = TAILLEJOUEUR

    def dessine(self):
        couleur=(255, 0, 0)   #rouge
        pygame.draw.rect(screen, couleur, (self.x, self.y, self.taille, self.taille))

    def deplace(self):
        self.x = self.x + self.vx
        self.y = self.y + self.vy
```

• Ensuite, il faut placer le joueur à sa position définie par le LABYRINTHE.
Pour cela, dans la fonction **def fabriqueLabyrinthe()** il faut rajouter le test **if case == "D"** comme ci-dessous.

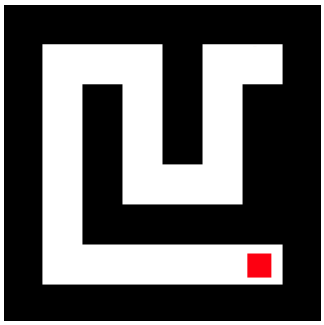
```
def fabriqueLabyrinthe():
    ...
        for case in ligne:
            if case == "X":
                ...
            if case == "D":           # <---- ligne à rajouter dans le code
                joueur.x = x + 5      # <---- ligne à rajouter dans le code
                joueur.y = y + 5      # <---- ligne à rajouter dans le code
                joueur.vx = 0         # <---- ligne à rajouter dans le code
                joueur.vy = 0         # <---- ligne  à rajouter dans le code
            x = x + TAILLEMUR
        y = y + TAILLEMUR
```

• Puis , il faut créer le joueur juste après avoir créé les murs :

```
...
listeMurs = []
joueur = Joueur()               # <----- Ligne à rajouter pour créer le joueur
fabriqueLabyrinthe()
...
```

• Enfin, il faut rajouter **joueur.deplace()** et **joueur.affiche()** dans la boucle infinie :

```
...
screen.fill((255, 255, 255))
joueur.deplace()         # <----- Ligne à rajouter
joueur.dessine()         # <----- Ligne à rajouter
for mur in listeMurs:
    mur.dessine()
...
```



2 DÉPLACEMENT DU JOUEUR AVEC LE CLAVIER

Il suffit de faire en sorte qu'un appui sur les flèches du clavier modifient les vitesses **joueur.vx** et **joueur.vy**

Pour cela, dans la boucle **for evenement in pygame.event.get()** il suffit de rajouter :

```
if evenement.type == pygame.KEYDOWN:
    if evenement.key == pygame.K_DOWN:
        joueur.vy = joueur.vy + 1
        joueur.vx = 0

    elif evenement.key == pygame.K_UP:
        joueur.vy = joueur.vy - 1
        joueur.vx = 0

    elif evenement.key == pygame.K_LEFT:
        joueur.vx = joueur.vx - 1
        joueur.vy = 0

    elif evenement.key == pygame.K_RIGHT:
        joueur.vx = joueur.vx + 1
        joueur.vy = 0
...
```

PARTIE 4: CRÉATION DU TRÉSOR

1 LA CLASSE TRESOR

- Copier/coller la classe **Mur** pour créer la classe **Tresor** en tout point semblable.
- Modifier la propriété **self.couleur** et **self.taille** comme ci-dessous.

```
class Tresor:
    def __init__(self):
        self.couleur = ( 200, 150, 100)
        self.taille = TAILLETRESOR

    def dessine(self):
        pygame.draw.ellipse(screen, self.couleur, (self.x, self.y,  self.taille,  self.taille))
```

2 POSITIONNEMENT DU TRÉSOR

- Rajouter une condition dans la fonction **def labriqueLabyrinthe()** afin de positionner le trésor

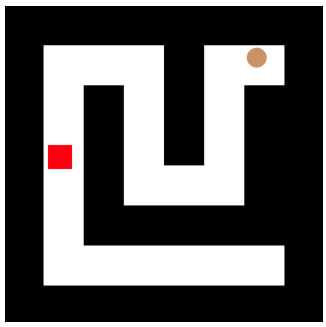
```
if case == "T":
    tresor.x = x + 6
    tresor.y = y + 6
```

- Créer la variable **tresor** avant la boucle infinie

```
joueur = Joueur()
tresor = Tresor()                # <------- ligne à rajouter
fabriqueLabyrinthe()
```

- Afficher le trésor dans la boucle infinie.

```
joueur.dessine()
tresor.dessine()                # <------- ligne à rajouter
for mur in listeMurs:
    mur.dessine()
```



- Lancer le programme et vérifier que le joueur se déplace avec les touches du clavier.
- Sauvegarder le programme sous le nom **jeu.py** avant de poursuivre.

DEPÔT 2 déposer jeu.py sur https://entraide-ella.fr

PARTIE 5: GESTION DES COLLISIONS ET GAME OVER

Le but de cette partie est de faire en sorte que si le joueur heurte un mur, il perde la partie et que s'il touche le trésor. il gagne la partie.

1 GESTION DES COLLISIONS

• Rajouter la méthode **def collision** aux classes **Mur** et **Tresor** :
Le but de cette méthode est de renvoyer **True** si le joueur touche le mur ou le trésor.

```
def collision(self, joueur):
    touche = -self.taille<self.x-joueur.x<joueur.taille and -self.taille<self.y-joueur.y<joueur.taille
    return touche
```

• Rajouter le test de collision avec un mur ou avec le joueur dans la boucle infinie.

```
for mur in listeMurs:
    mur.dessine()
    if mur.collision(joueur):          # <--- Ligne à rajouter
        print("Game Over")                                    # <--- Ligne à rajouter

if tresor.collision(joueur):             # <--- Ligne à rajouter
    print("Trésor trouvé")                                   # <--- Ligne à rajouter
```

• Lancer le programme et vérifier les affichages de la console

2 ARRÊT DU JEU TEMPORAIRE EN CAS DE COLLISION

• En cas de collision, on veut que le jeu fasse une pause puis redémarre. Pour cela, il suffit de compléter les tests de collision précédents.

```
...
print("Game Over")
pygame.time.delay(3000)    # Pause de 3 secondes
fabriqueLabyrinthe()

...
print("Trésor trouvé")
pygame.time.delay(3000)      # Pause de 3 secondes
fabriqueLabyrinthe()
```

PARTIE 6: CRÉATION DE SON PROPRE LABYRINTHE

• Modifier la liste LABYRINTHE en début de programme pour avoir un labyrinthe plus compliqué.
• Sauvegarder le programme sous le nom **jeu.py**

DEPÔT 3 déposer jeu.py sur https://entraide-ella.fr