Conception_CTF - Alex_SoapLand

Conception de l'épreuve de Capture The Flag (CTF) - SoapLand

Introduction

"Soap Land" est une épreuve de CTF qui invite les participants à télécharger et à lancer un jeu python. Le jeu fonctionne mal et pousse les participants à chercher le flag dans le code du jeu.

Description du Défi

Le challenge propose de télécharger un fichier .py qui, lorsqu'il est lancé, démarre le jeu Puzzle Bubble. Le jeu a des problèmes de fonctionnement qui sont susceptibles de générer de la frustration chez le joueur, le poussant ainsi à chercher le flag dans le code du jeu.

Voici le programme, inspiré du grand classique Puzzle Bubble

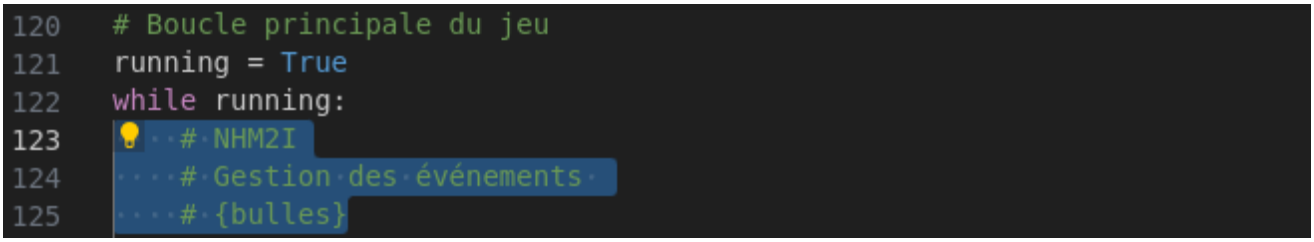
```
#Puzzle Bubble
#100 points : flag

import pygame
import random

# Initialisation de Pygame
pygame.init()

# Définition des constantes
screen_width = 640
screen_height = 480
bubble_radius = 20
num_bubbles = 20
bubble_speed = 5
bubble_colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0),
(0, 255, 255), (255, 0, 255)]
ball_radius = 10
ball_speed = 10
launcher_height = 20
black = (0, 0, 0)
white = (255, 255, 255)

# Création de la classe pour les bulles
class Bubble:
    def __init__(self, x, y, color):
```

```
        self.x = x
        self.y = y
        self.color = color

    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y),
bubble_radius)

    def move(self, dx, dy):
        self.x += dx
        self.y += dy

# Création de la classe pour la balle
class Ball:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def draw(self):
        pygame.draw.circle(screen, white, (self.x, self.y), ball_radius)

    def move(self, dx, dy):
        self.x += dx
        self.y += dy

# Création de la classe pour le lanceur de balle
class Launcher:
    def __init__(self):
        self.x = screen_width // 2
        self.y = screen_height - launcher_height

    def draw(self):
        pygame.draw.rect(screen, white, (self.x - 50, self.y, 100,
launcher_height))

    def move_left(self):
        self.x -= 10

    def move_right(self):
        self.x += 10

# Création de la fonction pour initialiser les bulles sur l'écran
def init_bubbles():
    global bubbles

    # Initialisation de la liste des bulles
    bubbles = []

    # Positionnement des bulles sur l'écran
```

```
    for i in range(num_bubbles):
        x = random.randint(bubble_radius, screen_width - bubble_radius)
        y = random.randint(bubble_radius, screen_height // 2)
        color = random.choice(bubble_colors)
        bubble = Bubble(x, y, color)
        bubbles.append(bubble)

# Fonction pour dessiner les bulles sur l'écran
def draw_bubbles():
    for bubble in bubbles:
        bubble.draw()

# Fonction pour déplacer les bulles sur l'écran
def move_bubbles():
    for bubble in bubbles:
        bubble.move(0, bubble_speed)

# Fonction pour vérifier si la balle a touché une bulle
def check_collision():
    global score

    for bubble in bubbles:
        distance = ((bubble.x - ball.x) ** 2 + (bubble.y - ball.y) ** 2)
** 0.5

        if distance <= bubble_radius + ball_radius:
            # Suppression de la bulle touchée de la liste des bulles
            bubbles.remove(bubble)
            # Incrémentation du score
            score += 10

# Fonction pour dessiner le score sur l'écran
def draw_score():
    font = pygame.font.SysFont(None, 25)
    score_text = font.render("Score: " + str(score), True, white)
    screen.blit(score_text, (10, 10))

# Initialisation de la fenêtre du jeu
screen = pygame.display.set_mode((screen_width, screen_height))

# Initialisation du score
score = 0

# Initialisation des objets du jeu
init_bubbles()
ball = Ball(screen_width // 2, screen_height - launcher_height -
ball_radius)
launcher = Launcher()
```

```python
# Boucle principale du jeu
running = True
while running:
    # NHM2I
    # Gestion des événements
    # {bulles}
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                launcher.move_left()
            elif event.key == pygame.K_RIGHT:
                launcher.move_right()
            elif event.key == pygame.K_SPACE:
                ball.move(0, -ball_speed)

    # Effacement de l'écran
    screen.fill(black)

    # Dessin des objets du jeu
    draw_bubbles()
    ball.draw()
    launcher.draw()
    draw_score()

    # Déplacement des objets du jeu
    move_bubbles()

    # Vérification des collisions
    check_collision()

    # Mise à jour de l'écran
    pygame.display.flip()

# Fermeture de Pygame
pygame.quit()
```

Le flag se situe ici :



Déroulement du Défi

Étape 1

La première étape consiste à télécharger le jeu puzzle.py et à l'installer (il faut installer la librairie pygame). Ensuite, le joueur doit lancer le jeu.

Étape 2

Comme le jeu fonctionne mal, le joueur est incité à chercher ailleurs pour trouver le flag.

Étape 3

Le flag se trouve dans le code du jeu.

Flag

Le flag est [NHM2I{bulles}].

Conclusion

"Soap Land" est un défi qui teste la patience et la curiosité des joueurs. Après avoir connu des échecs en essayant de jouer à Puzzle Bubble, le joueur doit avoir la curiosité de chercher le flag dans le code du jeu. Afin de ne pas donner d'indices supplémentaires sur où chercher, le challenge n'a pas été placé dans la catégorie "forensic - intro".