PRENSENTEE PAR : M'BARK LAHNOUD
ENCADREE PAR : ASMAE EL OUAZZANI

مدن المهن و الكفاءات
ⵜⵉⵎⴰⵣⵉⵖⵉⵏ ⵏ ⵜⵎⵙⵙⵓⵏⵉ ⴰ ⵜⵎⵓⵔⵜ
Cités des Métiers et des Compéter

DEV102-MEMORYGAME

PROJET PYTHON AVEC INTERFACE GRAPHIQUE

WWW.DEV102MEMORYGAME.MA

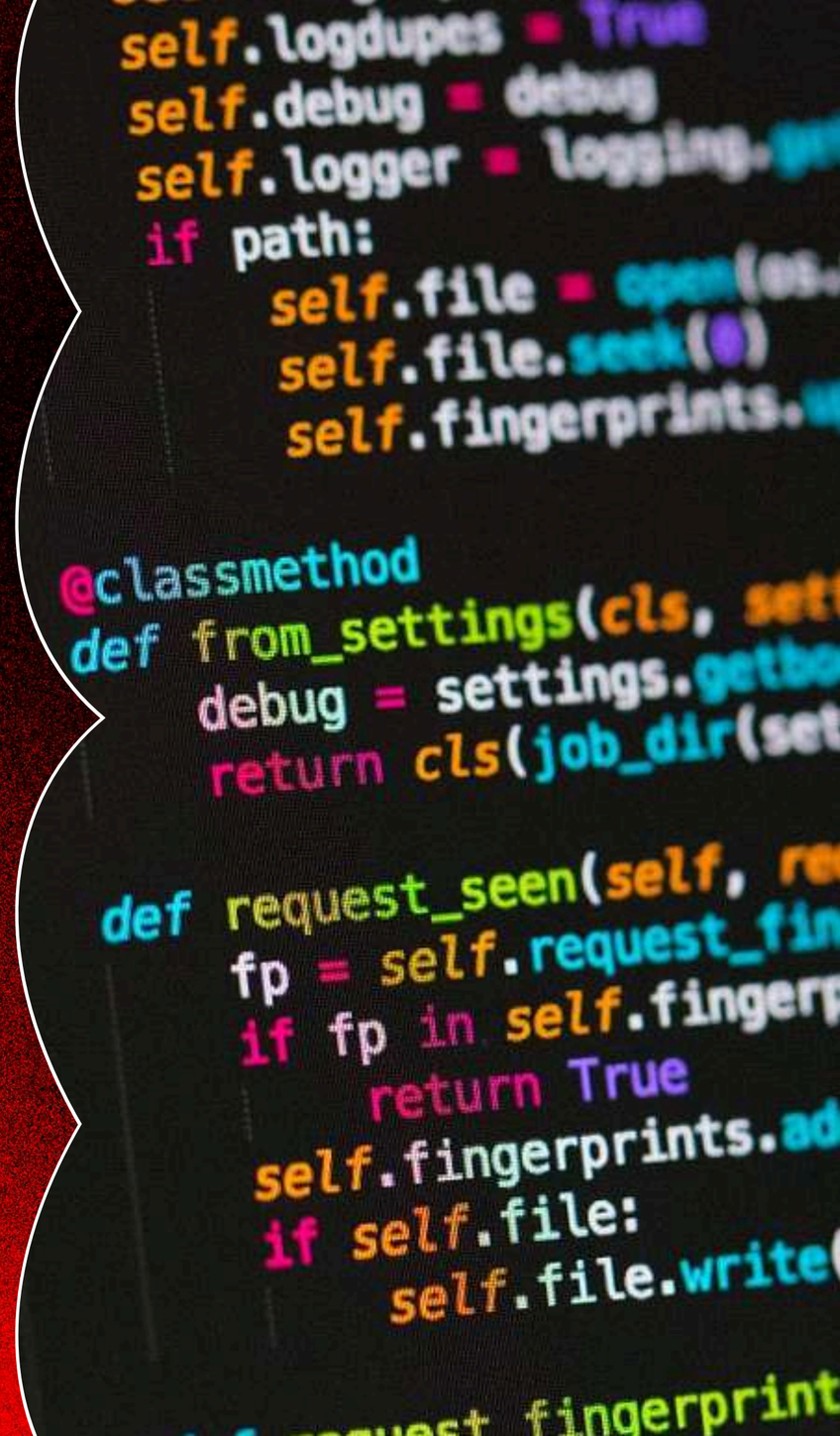
python

I – INTRODUCTION :

Ce projet est un jeu de mémoire conçu pour tester et améliorer la mémoire du joueur. L'objectif est de retrouver toutes les paires de cartes dans le moins de tentatives possibles. Le jeu sauvegarde le meilleur score pour suivre les performances du joueur. Dans cette présentation, nous aborderons les technologies utilisées, l'architecture du projet, le package et les modules, le plateau de jeu et les interactions, la gestion du score et des images, les fonctionnalités et animations, les challenges et solutions, et enfin les perspectives et conclusions.

II - TECHNOLOGIES UTILISEES :

LE PROJET UTILISE PYTHON POUR TOUTE LA LOGIQUE DU JEU. TKINTER EST UTILISÉ POUR CRÉER UNE INTERFACE GRAPHIQUE SIMPLE MAIS EFFICACE. LA BIBLIOTHÈQUE PILLOW PERMET DE GÉRER LES IMAGES DES CARTES ET DU PLATEAU, TANDIS QUE DES FICHIERS TEXTE SONT UTILISÉS POUR SAUVEGARDER LE MEILLEUR SCORE DE MANIÈRE PERSISTANTE ENTRE LES PARTIES.



III – ARCHITECTURE DU PROJET :

L'APPLICATION EST ORGANISÉE SOUS FORME DE PACKAGE PYTHON. LE FICHIER MAIN.PY SERT À LANCER LE JEU ET À CENTRER LA FENÊTRE. LE DOSSIER MEMORYGAME CONTIENT LES MODULES ESSENTIELS : __INIT__.PY POUR L'INITIALISATION DU PACKAGE, CARDS.PY POUR GÉRER CHAQUE CARTE, ASSETS.PY POUR LES IMAGES ET RESSOURCES, GAME.PY POUR LA LOGIQUE DU JEU, ET UTILS.PY POUR LES FONCTIONS UTILITAIRES COMME LA GESTION DU SCORE. CETTE STRUCTURE PERMET UN CODE CLAIR, MODULAIRE ET FACILEMENT MAINTENABLE.

IV – LE PLATEAU DE JEU :

Le plateau est constitué d’une grille 4x5 soit 20 cartes, formant 10 paires. Les cartes sont mélangées aléatoirement à chaque partie pour rendre le jeu imprévisible et stimulant. Chaque carte peut être retournée en cliquant dessus et reste visible si elle correspond à une autre carte identique.

V – CARD CLASS :

La classe Card représente chaque carte du plateau. Chaque carte connaît son image de face et de dos, son état (retournée ou non, appariée ou non) et son bouton Tkinter associé. Elle fournit des méthodes pour afficher l'avant (show_front) ou l'arrière (show_back) de la carte. Cela permet de gérer chaque carte de manière autonome et claire.

```python
import tkinter as tk

class Card:
    def __init__(self, frame, image_back, image_front, index, co
        self.frame = frame
        self.back_image = image_back
        self.front_image = image_front
        self.index = index
        self.button = tk.Button(frame, image=self.back_image, co
        self.button.pack(fill="both", expand=True)
        self.current_image = self.back_image
        self.matched = False

    def show_front(self):
        self.button.config(image=self.front_image)
        self.current_image = self.front_image

    def show_back(self):
        self.button.config(image=self.back_image)
        self.current_image = self.back_image
```

VI - ASSETS CLASS :

assets.py contient toutes les ressources du jeu, notamment les images des cartes et le logo. Les images sont redimensionnées à 100x100 pixels pour uniformité. Le fichier gère aussi le chemin vers le meilleur score et fournit des fonctions pour charger et sauvegarder les images et le score. Cela centralise toutes les ressources et simplifie leur utilisation dans le jeu.

```
from PIL import Image, ImageTk

CARD_SIZE = 100

PACKAGE_FOLDER = os.path.dirname(os.path.abspath(__file__))
IMAGE_FOLDER = os.path.join(PACKAGE_FOLDER, "..", "images")
BEST_SCORE_FILE = os.path.join(PACKAGE_FOLDER, "..", "best_scor

LOGO_FILE = os.path.join(IMAGE_FOLDER, "logo.png")

WINDOW_ICON_FILE = os.path.join(IMAGE_FOLDER, "logo_round.png")

IMAGE_FILES = [os.path.join(IMAGE_FOLDER, f'IMG{i}.png') for i

def load_image(path):

    img = Image.open(path).resize((CARD_SIZE, CARD_SIZE))
    return ImageTk.PhotoImage(img)
```

VII – UTILS CLASS :

Le module utils.py contient des fonctions pour gérer le meilleur score. Il permet de lire le score depuis un fichier texte et de le sauvegarder après chaque partie. Cela assure la persistance des données entre différentes sessions de jeu.

```
def load_best_score(file_path):
    try:
        with open(file_path, "r") as f:
            return int(f.read())
    except:
        return None

def save_best_score(file_path, score):
    with open(file_path, "w") as f:
        f.write(str(score))
```

VIII – MEMORYGAMECLASS :

La classe MemoryGame dans game.py gère l'ensemble du jeu. Elle initialise la fenêtre, charge les images, crée les cartes, affiche les labels de score, et gère les interactions avec le joueur. Les cartes sont mélangées aléatoirement et le clic est verrouillé pendant le délai de retournement des cartes pour assurer une interaction fluide. Le module gère aussi la fin de partie et met à jour le meilleur score si nécessaire.

```python
class MemoryGame:
    def __init__(self, root):
        self.root = root
        self.root.title("DEV102 Memory Game")
        self.root.resizable(False, False)


        if WINDOW_ICON_FILE:
            try:
                icon_img = Image.open(WINDOW_ICON_FILE)
                icon_img = icon_img.resize((64, 64))
                self.root.iconphoto(False, ImageTk.PhotoImage(icon_img))
            except Exception as e:
                print("Erreur icône fenêtre:", e)
```

```python
def check_match(self, first, second):
    card1, card2 = self.cards[first], self.cards[second]
    if card1.front_image == card2.front_image:
        card1.matched = True
        card2.matched = True
        card1.frame.config(highlightbackground="green", highlightthickness=3)
        card2.frame.config(highlightbackground="green", highlightthickness=3)
        self.score += 1
    else:
        card1.show_back()
        card2.show_back()
```

```python
def reset_game(self):
    self.first_choice = None
    self.lock = False
    self.score = 0
    self.attempts = 0
    self.attempts_label.config(text=f"Attempts: {self.attempts}")
    self.best_label.config(text=f"Best Score: {self.best_attempts if self.best_attempts else 'N/A'}")
    random.shuffle(self.images)
    for i, card in enumerate(self.cards):
        card.front_image = self.images[i]
        card.show_back()
        card.matched = False
        card.frame.config(highlightthickness=0)
```

IX – PROGRAMME PRINCIPALE :

Le fichier main.py est le point d'entrée du jeu. Il crée la fenêtre principale avec Tkinter et la configure à une taille de 600x550 pixels. Ensuite, il instancie la classe MemoryGame du module game pour lancer le plateau et les interactions du jeu. Enfin, mainloop() permet à la fenêtre de rester active et d'attendre les actions du joueur.

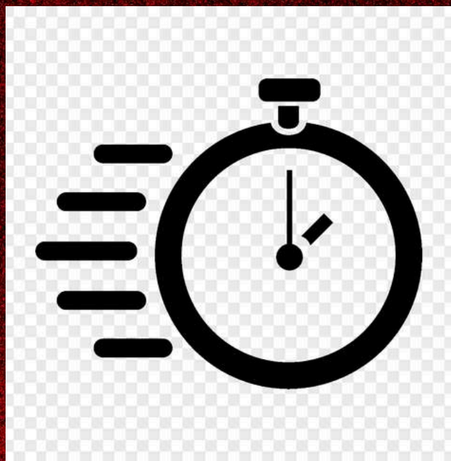
```python
import tkinter as tk
from .memory.game import MemoryGame

def main():
    root = tk.Tk()
    root.geometry("600x550+400+50")
    game = MemoryGame(root)
    root.mainloop()

main()
```

X – PERSPECTIVES D'AMELIORATION :

À l'avenir, il serait possible d'ajouter plusieurs niveaux de difficulté, un timer, des effets visuels supplémentaires pour les paires trouvées, différents thèmes d'images et un mode multi-joueur.



XI – POURQUOI POO :

La programmation orientée objet permet de gérer chaque carte et chaque partie du jeu comme des objets autonomes.

La modularité garantit que le code reste lisible, maintenable et facilement extensible pour ajouter de nouvelles fonctionnalités.

XII – DEMONTRATION :

Nous allons maintenant passer à la démonstration pour tester notre jeu et observer toutes ses fonctionnalités en action : navigation, plateau, interactions, suivi des scores et gestion des cartes.

Cette étape illustre la complétude et l'efficacité du projet avant la conclusion.

CONCLUSION GENERALE :

Nous arrivons maintenant à la fin : le projet DEV102 Memory Game est complet, modulable et facile à maintenir grâce à la POO et l'architecture en modules.

Il offre une expérience utilisateur fluide et agréable, avec un plateau dynamique, un suivi des scores et des interactions claires. La démonstration finale montre la complétude et l'efficacité du jeu.

DEV102-MemoryGame

مدن المهن و الكفاءات
ⵜⵉⵎⴰⵣⵉⵖⵉⵏ ⵏ ⵜⵎⵙⵙⵓⵔⵉ ⴰ ⵜⵓⵙⵙⵉ
Cités des Métiers et des Compétences

MERCI POUR VOTRE ATTENTION

WWW.DEV102MEMORYGAME.MA

python™