Compte Rendu : Informatique Projet n°2

Noblanc Anabelle, Camaret Pierre-Louis – CapECL 1A

Abstract

Voici le compte rendu du projet d'informatique n°2. Vous trouverez dans ce document le cahier des charges, un description du projet, les difficultés rencontrées et les limitations de notre script actuel. Ce projet consiste en la programmation d'un jeu : le Memory. Deux joueurs s'affrontent pour révéler le plus de paires de cartes possibles. Une paire de carte est formée par deux cartes de la même couleur (script classique) ou par deux cartes de même couleur ET de même forme (script formes). L'objectif principal est de garantir une expérience utilisateur simple et efficace tout en maintenant un code clair et optimisé.

Table of Contents

Cahier des Charges

Le projet consiste à développer en Python un jeu de Memory. Les cartes sont disposées face cachée pour former une grille. À tour de rôle, chaque joueur en retourne deux : si elles forment une paire, elles sont retirées du jeu ; sinon, elles sont remises face cachée. La partie se termine lorsque toutes les paires sont découvertes, et le joueur ayant trouvé le plus de paires l'emporte.

Pour rendre le jeu à la fois ergonomique pour l'utilisateur et clair pour les développeurs, nous utilisons un fichier de configuration texte. Il contient les informations de chaque carte (forme, coordonnées de départ, largeur, longueur et couleur), que nous stockons dans un dictionnaire. Chaque clé correspond à une carte et associe une liste de données. Ensuite, nous calculons les coordonnées finales de la forme et écrivons un fichier (CSV dans notre cas) contenant l'ensemble des informations nécessaires, avant de remplacer la liste de données dans le dictionnaire par le nom de ce nouveau fichier.

Cela permet une gestion simple et un accès rapide aux données de chaque carte au moment de les tracer avec Matplotlib.

Avancement du Projet

Deux scripts sont prêts à l'usage :

1. **Version classique** :
    - Demande le nom des joueurs.
    - Comptabilise leurs points.
    - Indique clairement qui doit jouer.
    - Affiche un GIF festif quand un joueur gagne.
    - Chronomètre la durée de la partie.
    - Permet de mélanger aléatoirement la disposition des couleurs au démarrage, offrant ainsi une rejouabilité accrue.
2. **Version simplifiée mais avec cartes de différentes formes** :
    - Supprime l'affichage d'un GIF en fin de partie.
    - Plus de création de fichier CSV pour chaque carte.
    - Génération dynamique du fichier `config_shapes.txt`.
    - Introduit des formes géométriques (carré, rond, triangle) et réduit à six couleurs.
    - Augmente le total à 36 cartes (au lieu de 24), ce qui ajoute de la difficulté et de la variété.

Pour tester ces scripts, il suffit d'exécuter `main.py`, qui propose de choisir la version désirée. On saisit ensuite le nom des deux joueurs et la partie peut démarrer ! (Voir plus d'info dans la section Annexe.)

Difficultés et Solutions Trouvées

1. **Tracé des cartes avec Matplotlib** :
    - **Problème** : Nous n'avions pas le droit d'utiliser la fonction prédéfinie de Matplotlib pour dessiner un rectangle, ce qui nous a obligés à comprendre en profondeur la façon dont les formes sont tracées.
    - **Solution** : Nous avons tiré parti des coordonnées figurant dans le fichier CSV de chaque carte. Grâce à ces points de référence, nous avons pu générer manuellement les bords de chaque rectangle (ou autre forme) et gérer leur rendu dans Matplotlib. Ce processus a nécessité quelques allers-retours entre la lecture des données et le test d'affichage, mais il nous a finalement permis d'obtenir un rendu sur mesure.
2. **Gestion des clics et vérification des cartes** :
    - **Problème** : Il fallait identifier sur quelle carte un joueur clique, vérifier si elle est déjà retournée et gérer l'ordre des clics (premier, deuxième).
    - **Solution** : Nous avons utilisé les événements de clic fournis par Matplotlib (`on_click(event)`) pour repérer la position du pointeur. Ensuite, il a fallu vérifier si le joueur a cliqué sur une carte ou non, et si oui, sur quelle carte. Enfin, un contrôle de l'état de la carte (retournée, découverte ou encore cachée) nous a permis de déterminer l'action à réaliser.
3. **Mélange aléatoire des couleurs** :
    - **Problème** : Il fallait s'assurer que chaque partie bénéficie d'une répartition différente des couleurs pour que le jeu reste imprévisible. Il fallait faire attention à ce qu'il reste bien deux cartes de chaque couleur pour que le jeu reste fonctionnel.
    - **Solution** : La structure de notre gestion des cartes nous a permis assez facilement de faire un `random.suffle` pour mélanger l'indice des cartes de la liste.

Limites et Perspectives

1. **Nombre de joueurs**
    - **Limite actuelle** : Seuls deux joueurs peuvent participer simultanément.
    - **Perspectives** : Permettre un plus grand nombre de joueurs, par exemple en ajoutant un tableau de scores dynamiques. Toutefois, le jeu deviendrait plus long, et l'équilibrage de la difficulté serait à repenser.
2. **Mode Solo contre l'IA**
    - **Limite actuelle** : Le projet ne propose que du multijoueur local.
    - **Perspectives** : Créer un adversaire virtuel disposant d'une "mémoire" pour se souvenir des cartes déjà retournées. On pourrait ajouter plusieurs niveaux de difficulté (20 %, 50 % ou 80 % des cartes mémorisées), ou même intégrer des stratégies plus avancées (prioriser certaines cartes révélées, gérer la probabilité de réussir une paire, etc.).
3. **Extension à d'autres formes ou mécaniques de jeu**
    - **Limite actuelle** : Les formes sont actuellement limitées (carré, rond, triangle) et les couleurs relativement restreintes.
    - **Perspectives** : Ajouter de nouveaux types de cartes (formes plus complexes, images personnalisées, motifs spéciaux), voire introduire des "cartes bonus" qui pourraient renverser le cours de la partie (par exemple, un bonus qui permet de retourner trois cartes au lieu de deux).
4. **Performances et affichage**
    - **Limite actuelle** : Matplotlib fonctionne parfaitement pour nos besoins et respecte les consignes initiales du projet, mais il peut devenir plus lent si le nombre de cartes augmente considérablement.
    - **Perspectives** : Si l'on veut monter à 48 ou 60 cartes, il pourrait être pertinent d'explorer d'autres bibliothèques (pygame, turtle ou même tkinter) ou d'optimiser le tracé pour éviter tout ralentissement à l'écran.

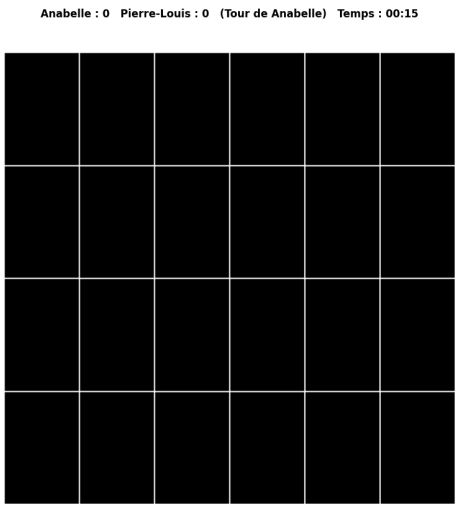
Annexe

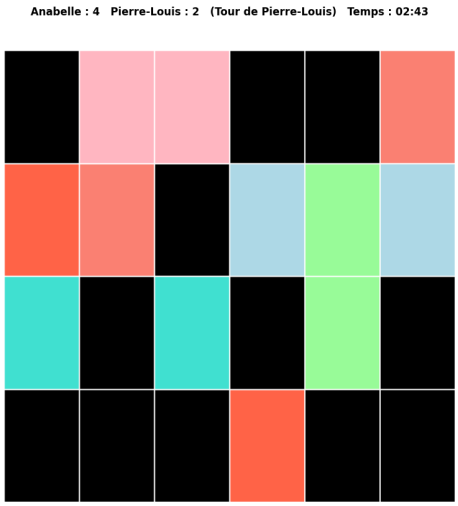
Informations supplémentaires

- **Installation** : Reportez-vous au README du dépôt GitHub pour les détails d'installation (versions de Python, bibliothèques requises, etc.).
- **Exécution** : Le script `main.py` permet de choisir la version du jeu (complète ou simplifiée). Vous n'aurez ensuite qu'à saisir le nom des joueurs dans la console pour commencer.
- **Voir sur GitHub** : projet-info-2

Screenshots

Version 1 (classique) :




Anabelle : 4 Pierre-Louis : 2 (Tour de Pierre-Louis) Temps : 02:43


Anabelle : 9 Pierre-Louis : 3 (Tour de Anabelle) Temps : 01:30
FÉLICITATIONS !
Toutes les paires sont trouvées !
Anabelle a gagné contre Pierre-Louis
I
WIN
!
Anabelle
Pierre-Louis

Version 2 (formes) :