Slide Puzzle Game

Presenté par : ABBAOUI Achraf
Encadré par : Pr. ELHAJJAMY OUSSAMA

Plan de la presentation

I. Présentation des designs pattern utilisé:
    1. Design Bridge:
    2. Design Observer:
    3. Design Stratégie:
II. Diagramme de classe de l'application.
III. Explication de besoin + utilisation:
    1. Design Bridge:
    2. Design Observer:
    3. Design Stratégie:
IV. Capture d'écran de l'application.

I.

Présentation des designs pattern utilisé

Design Bridge

- Bridge est un patron de conception structurel qui permet de séparer une grosse classe ou un ensemble de classes connexes en deux hiérarchies – abstraction et implémentation – qui peuvent évoluer indépendamment l'une de l'autre.

Design Observer

- L'Observateur est un patron de conception comportemental qui permet de mettre en place un mécanisme de souscription pour envoyer des notifications à plusieurs objets, au sujet d'événements concernant les objets qu'ils observent.

Design Stratégie

- Stratégie est un patron de conception comportemental qui permet de définir une famille d'algorithmes, de les mettre dans des classes séparées et de rendre leurs objets interchangeables.

Diagramme de class

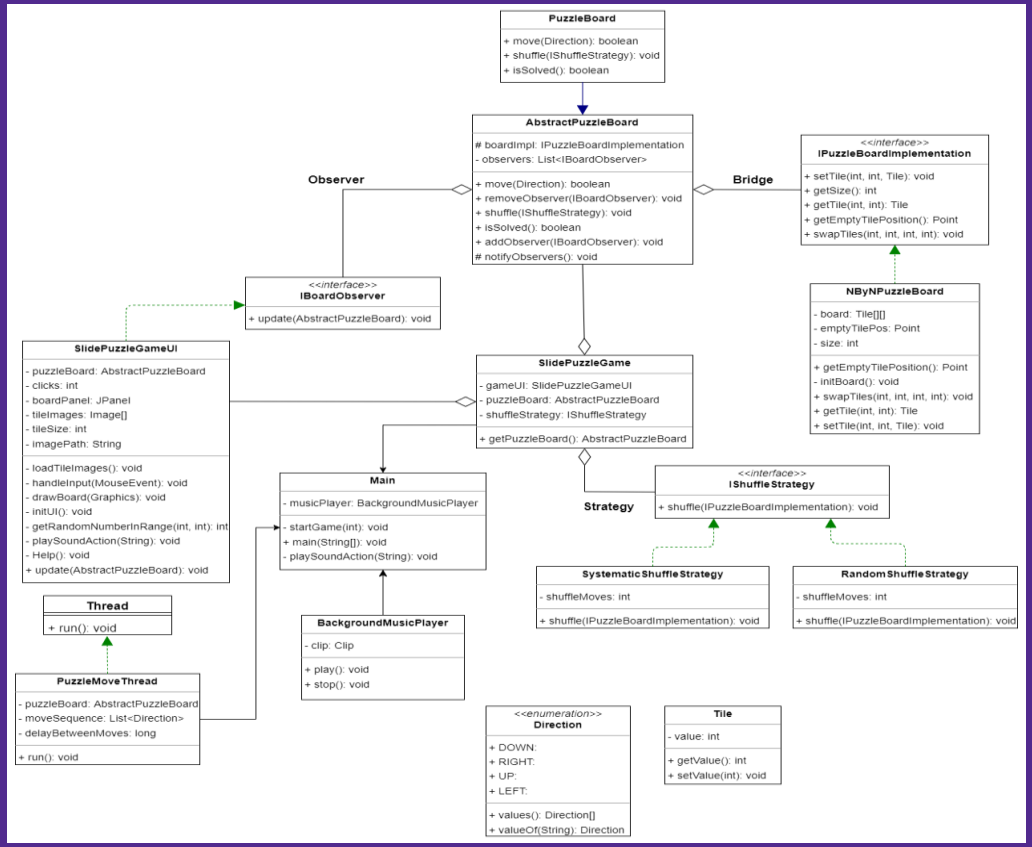
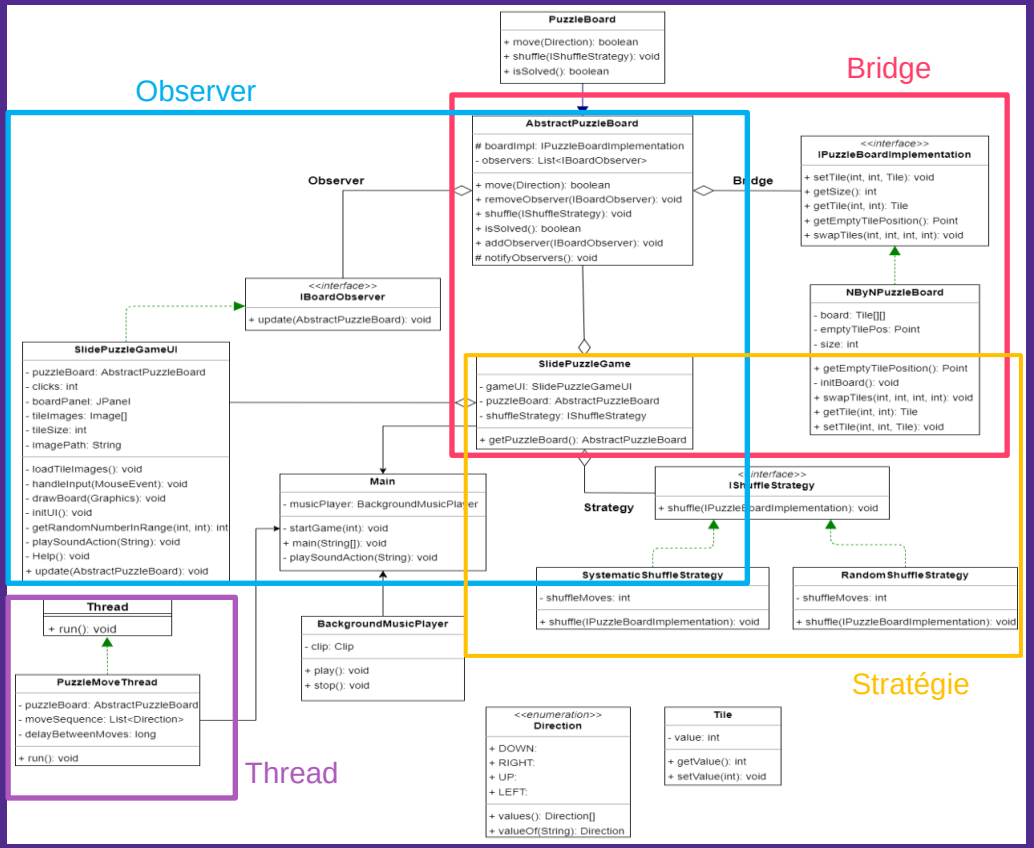


Diagramme de class



Pattern Bridge!

Le pattern Bridge a été utilisé pour séparer l'abstraction de l'implémentation dans notre application de puzzle. L'abstraction est représentée par la classe AbstractPuzzleBoard, qui définit les opérations haut niveau telles que le déplacement des tuiles, le mélange du puzzle et la vérification de la résolution du puzzle. D'autre part, l'implémentation est fournie par la classe IPuzzleBoardImplementation, qui fournit une implémentation concrète de ces opérations. Par exemple, si vous souhaitiez implémenter un autre type de plateau de puzzle vous pourriez créer une nouvelle implémentation de IPuzzleBoardImplementation et l'utiliser avec le code AbstractPuzzleBoard existant.

Pattern Oberser

Le pattern Observer a été utilisé pour mettre en place la relation observateur-observable entre la grille de puzzle (AbstractPuzzleBoard) et l'interface utilisateur (SlidePuzzleGameUI). La classe IBoardObserver définit la méthode update() qui est appelée par l'observable (AbstractPuzzleBoard) lorsque son état change. SlidePuzzleGameUI implémente IBoardObserver et fournit une implémentation concrète pour mettre à jour l'interface utilisateur lorsque la grille change.

Pattern Stratégie

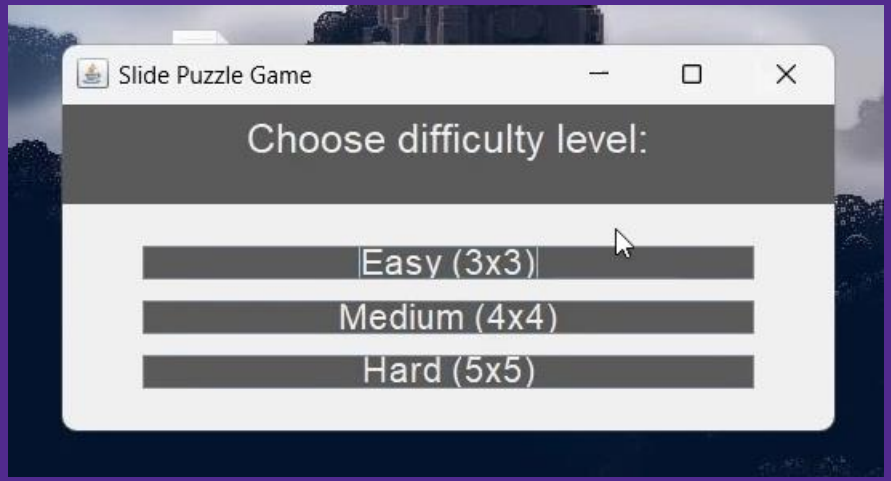
Le pattern Strategy a été utilisé pour encapsuler différentes stratégies de mélange (RandomShuffleStrategy et SystematicShuffleStrategy) et les rendre interchangeables. L'interface IShuffleStrategy définit la méthode shuffle(), et les stratégies de mélange concrètes (RandomShuffleStrategy et SystematicShuffleStrategy) implémentent cette interface.

Threads

Threads ont été utilisés pour permettre un déplacement fluide des tuiles de puzzle dans l'interface utilisateur, et précisément pour intégrer l'option de 'Help'. La classe PuzzleMoveThread étend Thread et définit la logique pour déplacer une tuile de puzzle dans une serie de directions spécifique.

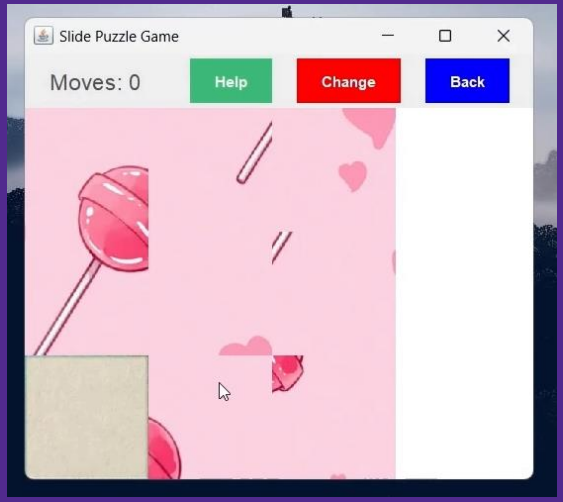
Capture d'écrant

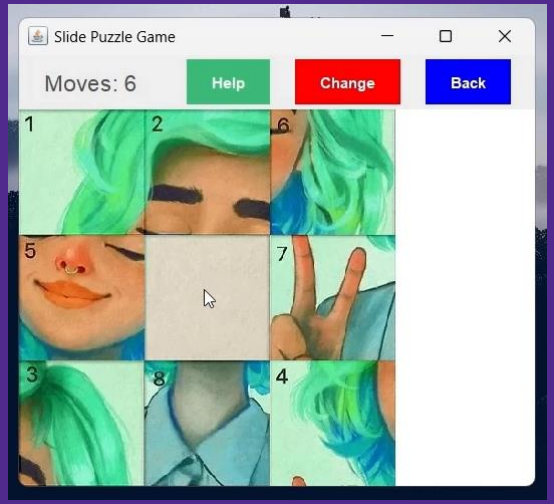
Menu principale

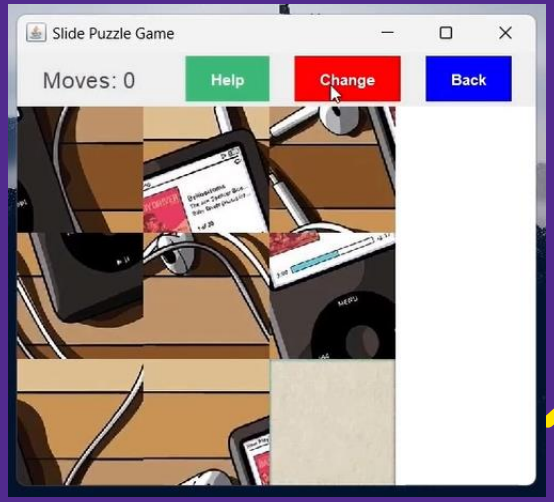


Capture d'écrant

Easy 3x3

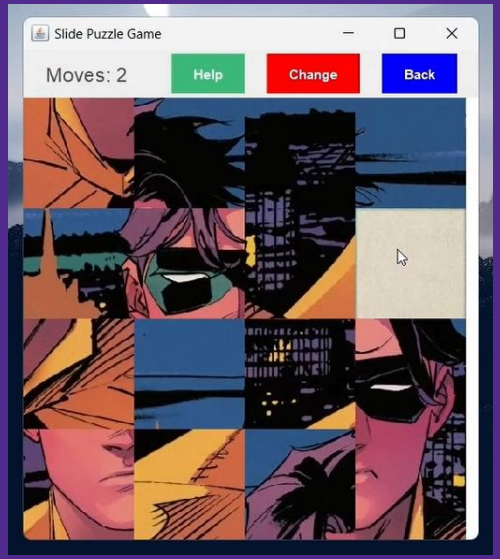


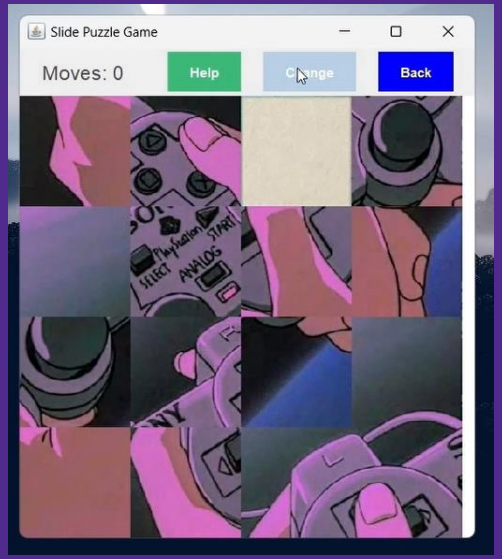


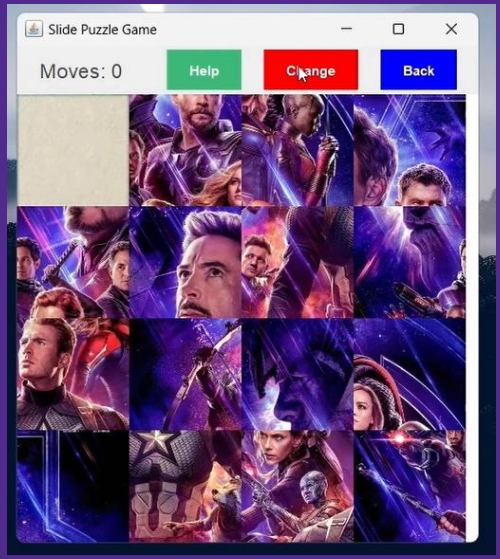


Capture d'écrant

Medium 4x4

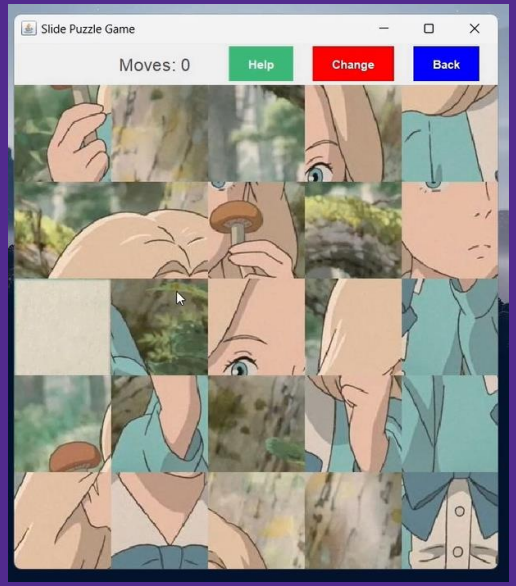


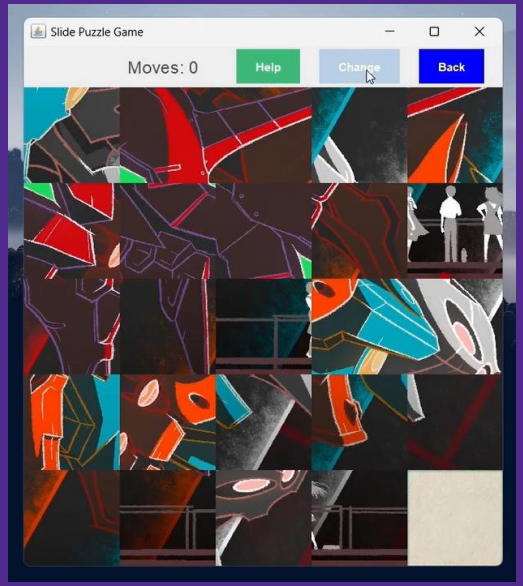


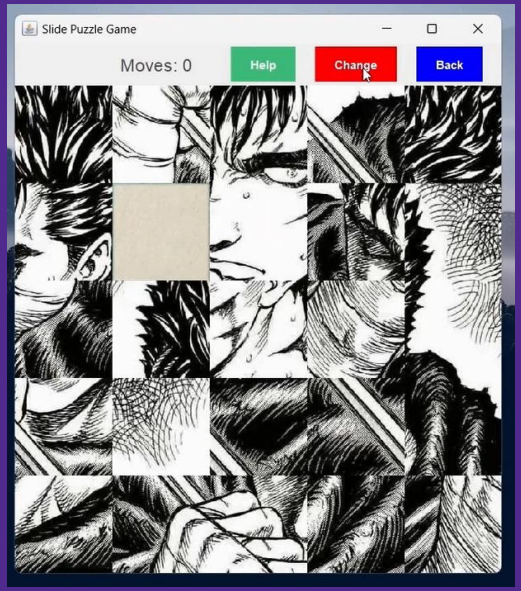


Capture d'écrant

Hard 5x5

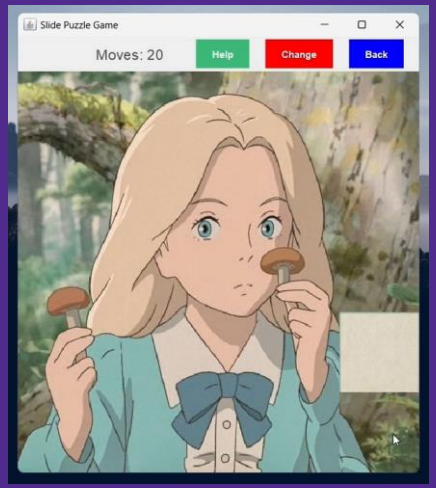






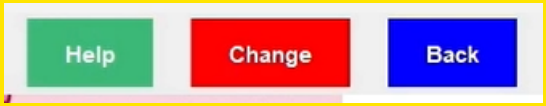
Capture d'écrant

Resoudre le jeu



Récompense

Explication de certain fonctionnalité



Help : Pour faire tourné le voisinage 2x2 de la *tuile* et aide le joueur.
Change : Changer l'image de la puzzle
Back : Revient au menu principale

- Un vidéo explicative est inclue avec les documents

- Source de la music / 'Sound effect' implémenté dans le jeu : https://pixabay.com/sound-effects/search/

Merci pour votre attention!