Paul Cottin

Jean-Laurent Brison

Projet de POO :

Nine Square Puzzle

**Présentation du problème :**

Le but de ce projet était d’implémenter en Java, un programme capable de résoudre un puzzle carré de 9 pièces carrées. A chacun des 4 côtés de chaque pièce est associé un nombre compris entre -4 et +4, le zéro étant exclu. Pour que le puzzle soit correctement formé, il faut disposer les pièces de telle sorte que la somme des 2 nombres associés aux cotés des 2 pièces adjacentes soit égale à zéro.

**Introduction**

Ayant compris assez rapidement la manière selon laquelle il nous fallait raisonner pour implémenter l’algorithme, nous nous sommes projetés assez rapidement un affichage graphique, qui allait nous permettre de rendre le projet plus intéressant et nous apporter une meilleure compréhension du problème.

**Méthode adoptée :**

Nous avons donc choisi de diviser le problème en différentes classes :

Il nous fallait donc :

Une classe ‘’Pièce’’ qui gère l’objet pièce, ainsi que toutes les méthodes qu’il comporte.

Une classe ‘’Pool’’ qui regroupe l’ensemble des pièces qui ne sont pas sur le plateau.

Une classe ‘’Board’’ qui gère l’ensemble des pièces disposées sur le plateau.

La classe principale : ‘’Main’’ qui sert à lancer l’algorithme.

La classe ‘’NineSquarePuzzle’’ qui effectue les différents tests et algorithmes qui placent les pièces correctement.

La classe ‘’Solution’’ gère les ensembles de pièces formant une solution.

La classe ‘’InstanceBoard’’ qui permet de sauvegarder l’ordre et l’orientation des pièces dans l’algorithme pour déterminer si une certaine configuration a déjà été testée.

La classe ‘’Resolution’’ qui crée un nouveau Thread pour avoir un Thread qui s’occupe de la fenêtre graphique et un autre de l’algorithme

La classe ‘’Instrumentation’’ qui a pour but de sauvegarder les statistiques de l’algorithme comme le nombre d’appels récursifs.

**Difficultés rencontrées :**

Ce projet nous a posés plusieurs problèmes auxquels nous avons trouvé des solutions :

|  |  |  |
| --- | --- | --- |
| 4 | 3 | 2 |
| 5 | 0 | 1 |
| 6 | 7 | 8 |

Après plusieurs tentatives sur l’ordre dans lequel on placerait les pièces sur le plateau, nous avons choisi de les placer une par une, en forme d’escargot, en partant de la pièce centrale, puis à droite, en haut, à gauche pour finir par la case en bas à droite.

Cette méthode effectuait le moins de tests, et surtout, était plus facile à implémenter, car si on considère les emplacements des pièces numérotées comme sur le schéma suivant.

On effectue un seul test pour les pièces paires (en commençant à 2), et deux tests pour les pièces impaires, (un avec la pièce n-1 et un avec la pièce centrale, 0)

Les pièces sont stockées dans une liste (Pool), à chaque fois qu’une pièce est posée sur le plateau, le reste de la liste situé après la pièce enlevée se déplace vers la gauche. Si on retire une pièce du plateau (Board), elle vient se replacer à la fin de la liste (Pool). Ce qui fait qu’après plusieurs tours de boucle, les pièces ne sont plus testées dans l’ordre initial (A, B, C, D …) et que l’ordre d’apparition des pièces sur le plateau change en permanence.

La plupart des difficultés rencontrées provenaient surtout de la coordination entre les parties algorithme et affichage graphique. Il est en effet difficile de gérer les deux, surtout l’affichage de la résolution en direct. Le codage de la partie graphique a consommé beaucoup de temps et d’énergie, mais l’affichage nous a permis de suivre en direct l’avancement de l’algorithme, et donc de gagner du temps sur la partie conception.

**Temps passé sur le projet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Conception | Codage | Tests | Rédaction du rapport |
| Jean-Laurent Brison | 7h | 20h | 10h | 3h |
| Paul Cottin | 15h | 50h | 3h | 1h |

**Conclusion :**

Finalement après quelques tâtonnements et beaucoup de mûres réflexions, nous avons réussi à concevoir un programme permettant de résoudre le puzzle et de pouvoir suivre son avancement en direct.

Néanmoins, même si nous avons offert à l'utilisateur une interface graphique avec de multiples fonctionnalités, lui permettant notamment de tester le programme de façon ludique, nous sommes assez déçus du temps de calcul (à notre goût assez médiocre), alors que nous avons passé pas mal de temps afin de l'améliorer. Nous pourrions avoir de meilleurs résultats, d'une part en baissant la complexité de notre algorithme qui reste assez élevée. Cela sera donc un point essentiel à revoir lors d'une éventuelle suite à donner à ce projet. En attendant, ce projet nous a permis de mettre en application certains points vus en cours et de découvrir comment implémenter des interfaces graphiques en langage Java.