Rapport du projet C++
Résolution d'un jeu de Sudoku

Achref Ben Ammar

29 décembre 2024

1 Introduction

Ce rapport présente le travail réalisé dans le cadre du projet de deuxième année à l'ENSAE en C++. L'objectif de ce projet était de créer un programme permettant à l'utilisateur de résoudre un problème de Sudoku et d'obtenir des indices de jeu pendant la partie, tout en intégrant une interface graphique réalisée à l'aide de la bibliothèque SFML (Simple and Fast Multimedia Library).

Dans le corps de ce rapport, nous décrirons le fonctionnement du programme et son architecture, ensuite nous nous intéresserons à la critique des problèmes rencontrés ainsi qu'aux solutions adaptées. La suite du rapport détaillera la description de l'architecture générale du programme.

2 Description du programme

Ce programme utilise la librairie SFML. Bien que l'installation de SFML ne soit pas nécessaire, le programme nécessite d'installer MSYS2 ucrt64 pour exécuter le code. Un fichier readme.md se trouve dans le dossier et contient les instructions pour lancer le programme.

Le projet Sudoku Solver en C++ a pour objectif de permettre à l'utilisateur de résoudre un puzzle de Sudoku via une interface graphique. Utilisant la bibliothèque SFML, l'application présente une grille interactive où l'utilisateur peut entrer des chiffres et d'obtenir des indices. Le programme repose sur un algorithme récursif de backtracking pour résoudre le Sudoku. Cet algorithme fonctionne en essayant récursivement des chiffres possibles dans les cellules vides, en validant à chaque étape que le chiffre respecte les règles du Sudoku. Si une valeur valide est trouvée, l'algorithme passe à la cellule suivante. Si aucune valeur valide n'est disponible, il revient en arrière pour tester d'autres solutions dans les cellules précédentes.

L'interface graphique permet à l'utilisateur de sélectionner des cellules, d'ajouter des chiffres dans ces cellules et de visualiser les erreurs en temps réel. En cas d'erreur, la cellule est mise en surbrillance en rouge. De plus, le programme offre une fonctionnalité d'indice où l'utilisateur peut cliquer sur une cellule pour afficher la valeur correcte. La résolution continue dans la console après la fermeture de la fenêtre graphique. Lorsque l'utilisateur ferme l'interface graphique, la résolution du puzzle ne s'arrête pas. Au lieu de cela, l'application passe en mode console, où l'utilisateur peut continuer à interagir avec la grille via des commandes textuelles

Enfin, l'application permet également de charger des puzzles à partir d'un fichier texte généré à partir du site : https ://qqwing.com/generate.html Stephen Ostermiller

3 Problèmes rencontrés et solutions envisagées

Une grande partie des difficultés rencontrées au cours du travail s'était présentée lors de l'implémentation de l'interface graphique. Principalement, j'ai rencontré des problèmes à la préparation de l'environnement de développement. Bien que j'aie pu faire fonctionner le programme sur d'autres machines, je n'ai pas eu l'occasion de l'essayer sur des machines non Windows 64 bits, et je soupçonne

qu'il pourrait avoir des problèmes à fonctionner sous MacOS ou Linux.

Afin de pouvoir faciliter l'exécution du code et de pouvoir le déboguer, j'ai utilisé CMAKE.

4 Architecture générale

Le programme présente plusieurs classes permettant de mieux organiser le fonctionnement du programme.

On trouve les deux classes **Subgrid** et **Grid** qui représentent la grille actuelle et intègrent les fonctionnalités nécessaires pour manipuler les valeurs de la grille.

La classe **SudokuSolver** s'intéresse principalement à la résolution automatique de la grille à l'aide d'un algorithme de backtracking alors que la gestion des interactions utilisateur est prise en charge par la classe **User**. Cette classe agit comme un intermédiaire entre les entrées de l'utilisateur et la mise à jour des données de la grille. Elle permet d'ajouter des chiffres à des cases spécifiques ou de demander des indices. En cas d'erreur, les cellules sont immédiatement mises en évidence, aidant l'utilisateur à corriger ses entrées.

A l'instant d'exécution du programme, une grille de sudoku est choisie au hasard. Les fonctions comme *readfilepuzzle* et *randomgrid* permettent de lire un fichier texte contenant plusieurs puzzles et de sélectionner aléatoirement une grille. La conversion de données textuelles en format exploitable se fait grâce à la fonction *puzzlefromtxt*, qui transforme une chaîne de caractères en une grille Sudoku sous forme de tableau à deux dimensions.

L'affichage et les interactions avec l'utilisateur sont gérés par la bibliothèque **SFML**. La fonction *drawGrid* est responsable de dessiner la grille Sudoku sur l'interface graphique. La fonction *drawButtons* complète cette interface en ajoutant un panneau de boutons pour les chiffres et une fonction de demande d'indices. Ces deux fonctions reflétant en temps réel les modifications apportées à la grille.

Finalement, une fois la fenêtre graphique fermée, le programme bascule sur une interface console pour permettre la poursuite de la résolution du puzzle.

5 Conclusion

Ce projet m'a permis de mieux comprendre et d'appliquer les principes de la programmation orientée objet, notamment l'encapsulation et l'utilisation des classes. En organisant les différentes fonctionnalités du programme en classes distinctes, j'ai pu mieux structurer mon code et apprendre à gérer les interactions entre les composants. L'organisation des fichiers a aussi été un élément clé pour rendre le projet plus lisible et modulaire. Ce projet m'a offert une bonne occasion d'apprendre à travailler avec ces concepts dans un contexte concret, tout en améliorant ma compréhension de la POO en C++.