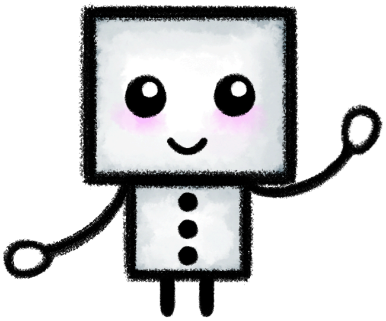
BORDEAUX INP Ensc

Livrable AI-Doku

Projet d'Informatique Individuel



Kloé Bonnet

Tuteur du projet : Angélique Tetelin

Décembre 2023 - Avril 2024

BORDEAUX INP Ensc

Sommaire

Introduction

Le jeu de Sudoku est un jeu de logique dont le but est de remplir une grille de 9x9 cases avec des chiffres. Il existe différents niveaux de ce jeu, les grilles devenant de plus en plus complexes au fur et à mesure que le nombre de chiffres de la grille diminue. Malgré plusieurs stratégies établies par le joueur, celui-ci peut se retrouver bloqué sur certaines grilles plus complexes. Il pourrait être tenté de demander un indice, mais les livrets de jeu ou applications ne proposent en général que la solution complète de la grille, sans donner d'astuces ou de méthodes au joueur pour que celui-ci s'améliore.

C'est alors qu'intervient ce projet. Le but de celui-ci est de créer une IA permettant d'assister un joueur de Sudoku. Celle-ci sera en mesure de proposer des grilles de sudoku de divers niveaux générées par elle-même. Elle pourra également aider le joueur en difficulté en lui fournissant des indices et des méthodes pour résoudre plus facilement son Sudoku. Enfin, si le joueur joue sur papier et qu'il souhaite demander de l'aide à l'IA, il pourra prendre en photo sa page de Sudoku et l'importer dans l'application.

Ce document présente de manière détaillée les différentes étapes et démarches du projet avec le choix des outils, le maquettage ainsi que la structure du code. Enfin, le document présente la gestion du projet ainsi que les perspectives.

Lancer le programme

Voici quelques étapes à réaliser pour lancer le programme. Pour toute interrogation sur l'utilisation du jeu, veuillez trouver les conseils d'utilisation en annexes.

Lancer le programme pour tester

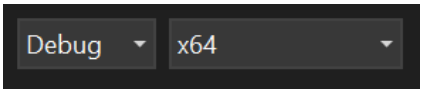
Si vous souhaitez tester le programme (sans vouloir accéder aux codes) :

- Rendez vous sur le dépôt GitHub suivant : https://github.com/kbonnet001/AI-Doku
- Télécharger en zip le projet et décompresser le OU cloner le projet (prévoir 213 Mo Mo)
- Rendez-vous dans le dossier AI-Doku-main>Project1>x64>Debug et lancer l'executable Project1.exe

Paramétrer le projet pour l'utiliser sur Visual Studio

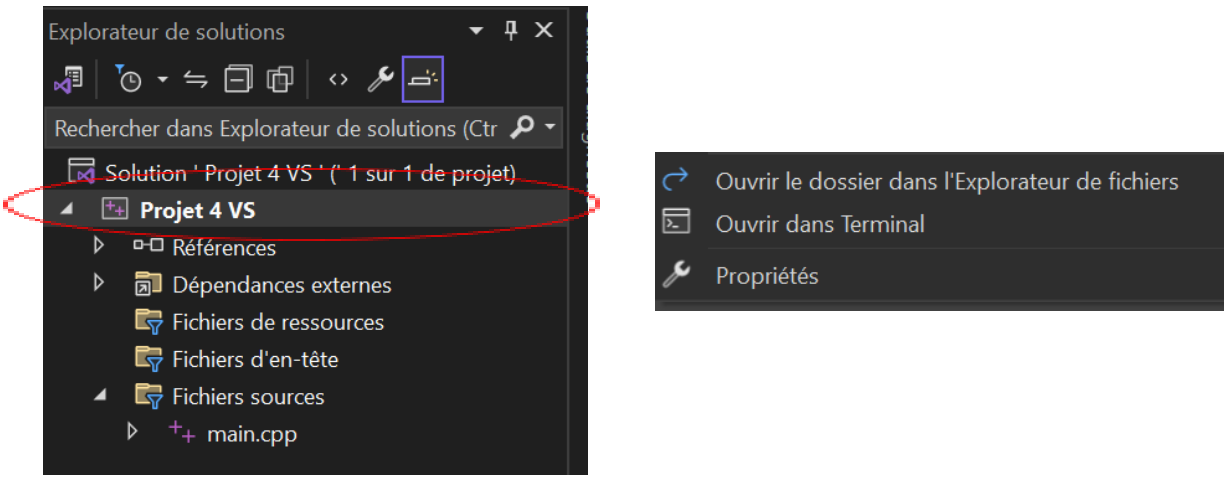
Si vous souhaitez ouvrir le projet avec Visual Studio et le lancer depuis cet éditeur de texte. Sachez que la tâche de paramétrage peut prendre du temps, notamment lorsque l'on est novice. Comptez entre 5 à 10 minutes pour cette étape :

- Rendez vous sur le dépôt GitHub suivant : https://github.com/kbonnet001/AI-Doku
- Téléchargez en zip le projet et décompressez le OU clonez le projet (prévoir 213 Mo)
- Ouvrez le projet avec VS
- Avant de commencer la configuration, veillez à avoir ceci en haut à gauche de votre écran :



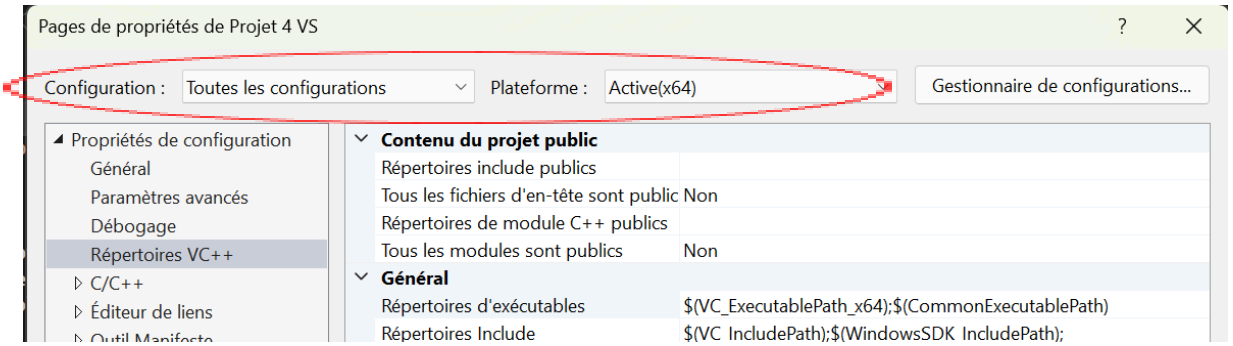
Capture d'écran 1 : Configuration VS

- Nous allons maintenant configurer : faites un clique droit sur le projet puis **Propriétés** tout en bas



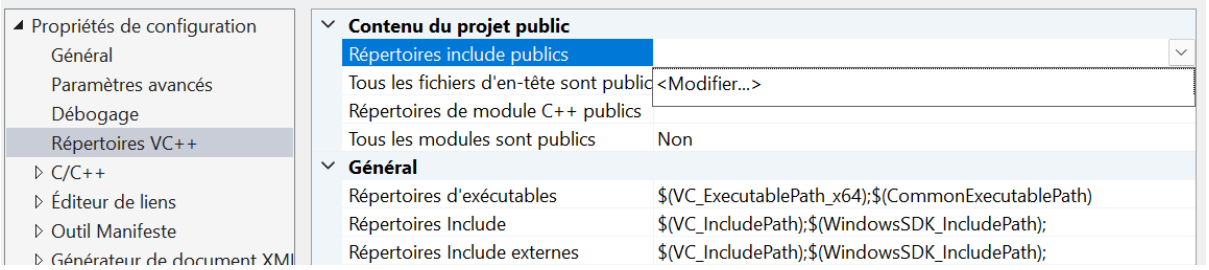
Capture d'écran 2 : Ouvrir les propriétés du projet

- **Ajouter les Include**
  - Changez la Configuration en haut à gauche par "Debug" et plateforme "Active(x64)" (vous pouvez aussi mettre "Toutes plateformes")



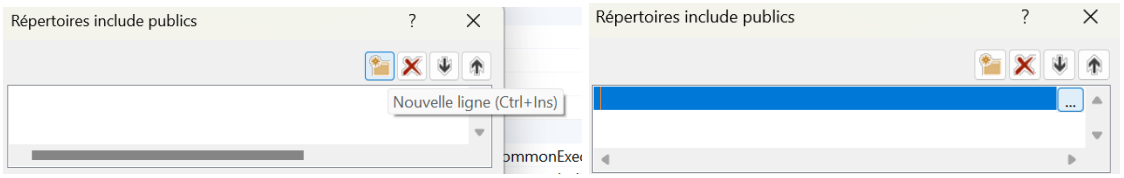
Capture d'écran 3 : Configurations de la fenêtre Propriétés

- Allez sur Répertoire VC++>Répertoires includes publics>*cliquer sur petite flèche à droite*>Modifier



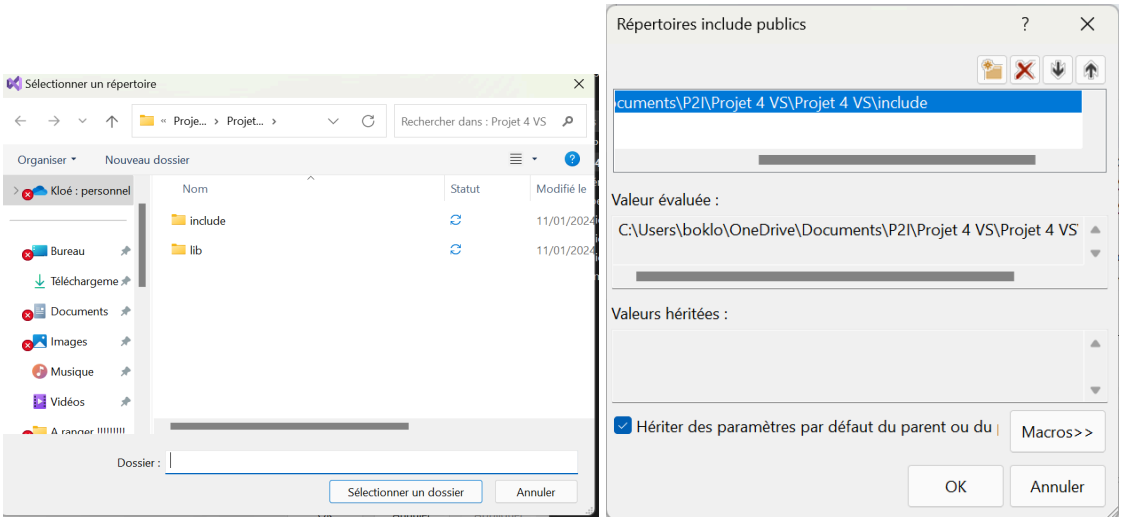
Capture d'écran 4 : Configuration - Répertoires include publics

- Une fenêtre s'ouvre, ajoutez une "nouvelle ligne" puis sur les trois petits points "..." pour pouvoir choisir un dossier



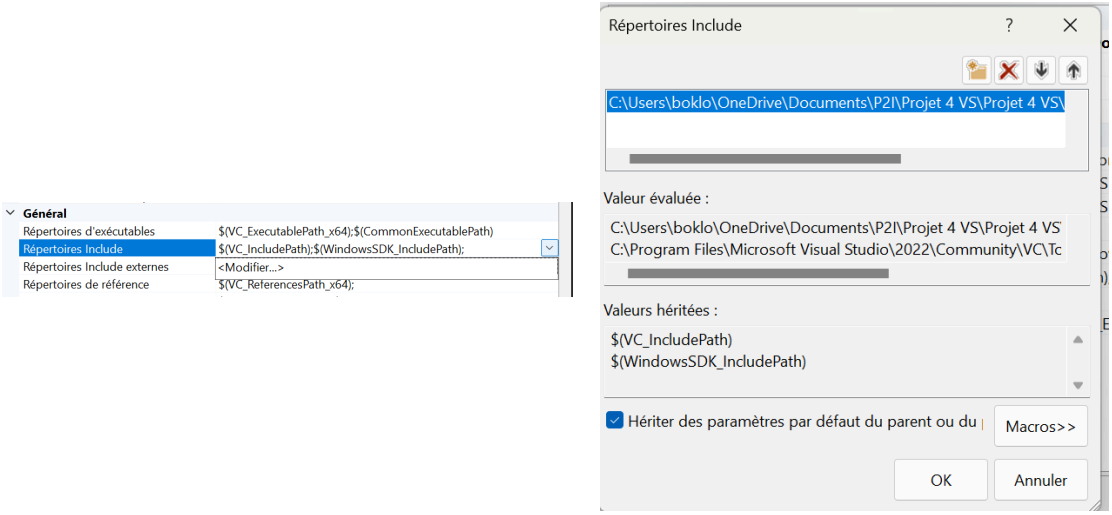
Capture d'écran 5 : Configuration - Répertoires include publics - Ajouter une ligne

- On sélectionne ensuite le dossier **Include** pour dire où il se trouve



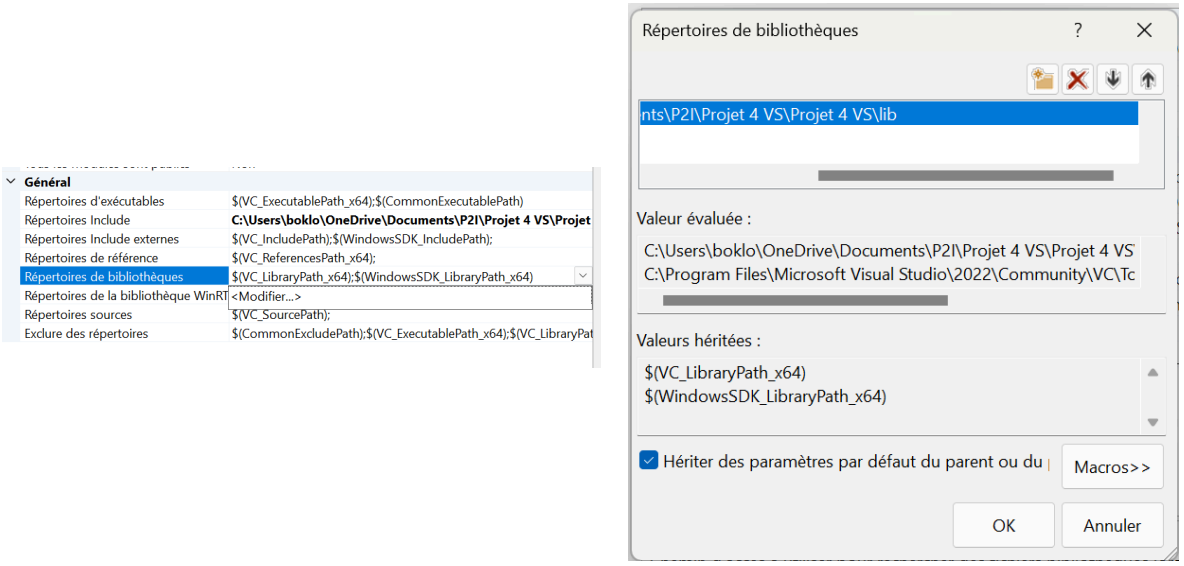
Capture d'écran 6 : Configuration - Répertoires include publics - Ajouter le dossier Include

- Pour éviter des confusion, on ajoute aussi l'**Include** dans "Répertoires Include"



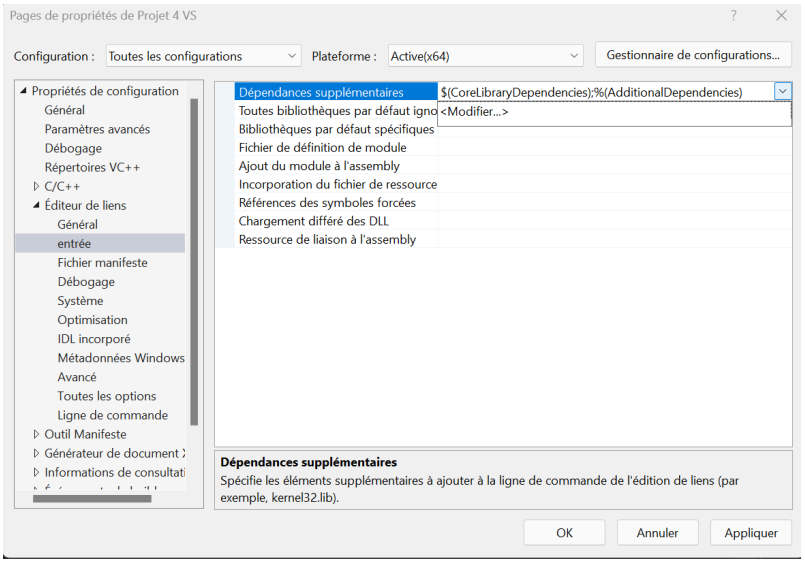
Capture d'écran 7 : Configuration - Répertoires include - Ajouter le dossier Include

- **Bibliothèque**
    - Rendez vous dans Répertoire de bibliothèques et ajoutez le dossier <**lib**> de la même façon que pour le dossier <include>



<u>Capture d'écran 8 :</u> Configuration - Répertoires de bibliothèques - Ajouter le dossier lib

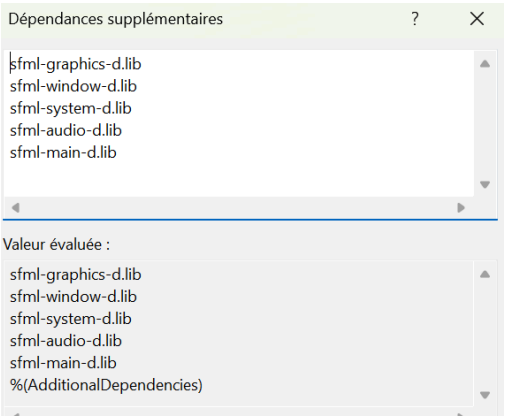
- **<bin>**
    - Rendez vous dans Editeur de liens>entrée>Dépendance supplémentaires>Modifier



<u>Capture d'écran 9 :</u> Configuration - Dépendance supplémentaires

- Une fenêtre s'ouvre, écrivez ces lignes de code puis faites OK

```
sfml-graphics-d.lib
sfml-window-d.lib
sfml-system-d.lib
sfml-audio-d.lib
sfml-main-d.lib
```



Capture d'écran 9 : Configuration - Dépendance supplémentaires - *Ajouter les dépendances*

- Grâce à ce paramétrage, vous devriez pouvoir run le projet.

Choix des outils

Ce projet devait initialement être codé avec Unity en C# mais, dans le but de s'exercer préalablement à un nouveau langage pour mon stage de 2e année, le C++ a été choisi.
De plus, afin de pourvoir réaliser une interface, la bibliothèque SFML a été choisie. En effet, cette dernière est la plus utilisée avec le langage C++. Il y a donc plusieurs ressources disponibles sur internet ce qui est favorable pour un apprentissage individuel demandé par ce projet.
Pour ce qui est de l'éditeur de code, le projet devait d'abord être réalisé sur Visual Studio Code. Après une installation fastidieuse du compilateur MinGW, il fallait ensuite installer la SFML. Cette étape étant particulièrement pénible et complexe sur VSC, de rapides tests ont donc été effectués sur Visual Studio, un autre éditeur de code. Le code test de la bibliothèque SFML fonctionne parfaitement avec VS, un changement de choix d'outil a ainsi eu lieu.
Finalement, pour résumer, ce projet est codé sur l'éditeur de texte Visual Studio en C++ et utilise la bibliothèque SFML.

Pour ce qui est du design de l'interface, des maquettes ont été réalisées sur Figma et les illustrations de l'IA sur Photoshop. Les animations ont ensuite été conçues à l'aide de l'outil de montage Filmora.

Le code du projet est disponible sur GitHub.

Maquettage de l’application

Cette section détaille les démarches et étapes de création de l’interface du jeu ainsi que de l’IA. Il est important de noter que, par manque de temps mais aussi parce que ce n’est pas l’objectif premier du projet, aucune méthode CCU n’a été appliquée. Ainsi les choix pour le design de la maquette ainsi que de l’IA sont totalement subjectifs et ne prennent pas en compte les besoins et préférences réels des utilisateurs.

Recherche d’un design pour l’IA

Plusieurs croquis ont été réalisés avant d’arriver à la version définitive du design de l’IA.

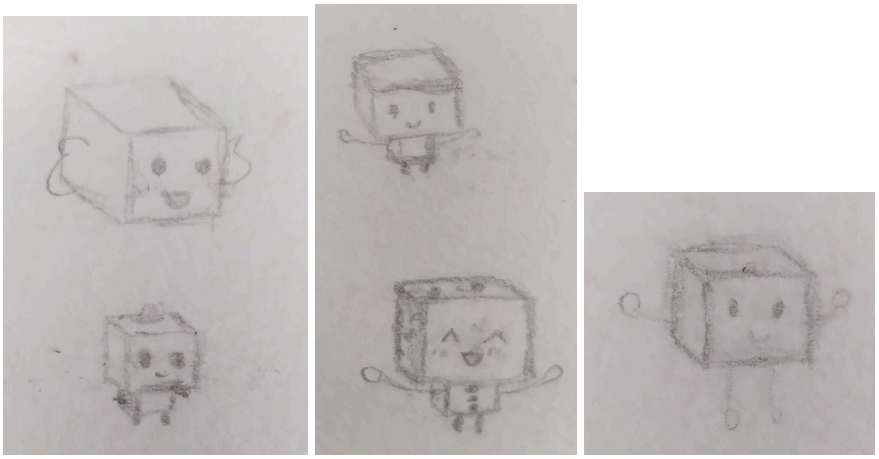


Figure 1 : Essais de croquis pour l’IA

Le choix final pour l’apparence de l’IA s'inspire de dés superposés : un premier dé à la base avec la face “3” représentant les boutons du personnage par ses trois ronds. Le second dé, placé au-dessus avec la face “2”, est doté de deux points symbolisant les yeux. Le trait est épais pour assurer une bonne visibilité sur une petite interface.

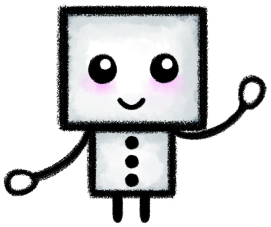


Figure 2 : Frame de l’animation “Coucou” de l’IA

D’autres dessins ont ensuite été réalisés afin d’être utilisés dans la maquette pour rendre le jeu plus attractif.

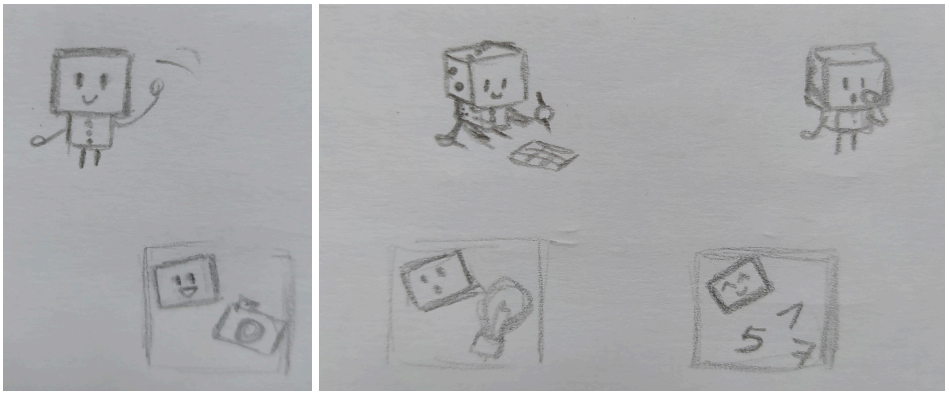


Figure 3 : Croquis de l’IA dans différentes attitudes

Interface de l’application

Un premier maquettage de l’application a été fait sur papier avant d’être numérisé sur Figma. La page d’accueil offrirait diverses fonctionnalités. L’utilisateur pourrait prendre en photo une grille de sudoku ou importer une photo déjà prise avec l’assistance de l’IA. Le bouton “Astuce” donnerait accès à différentes explications de stratégies de sudoku pour tous les niveaux. Enfin, le bouton “Joueur” permettrait de jouer au sudoku en choisissant préalablement le niveau.

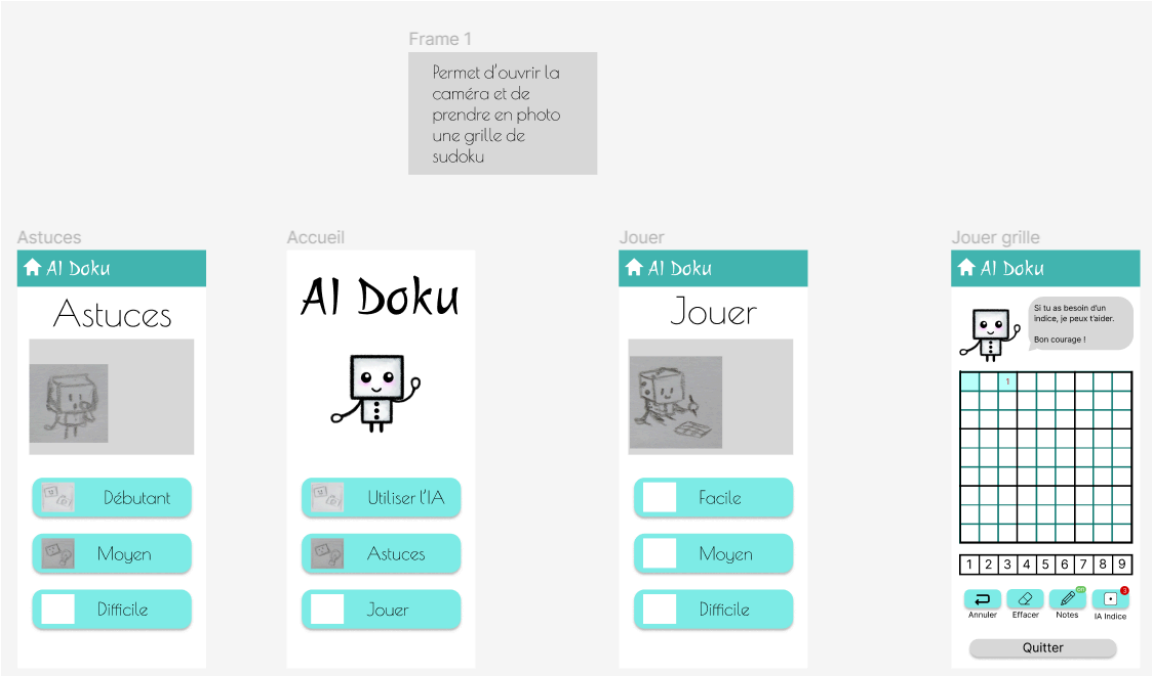


Figure 4 : Maquettage du jeu

En raison de l'ampleur du travail demandé pour la reproduction de cette maquette, la décision a été prise de se concentrer exclusivement sur la page présentant la grille de sudoku, avec l'IA positionnée au-dessus pour assister le joueur.

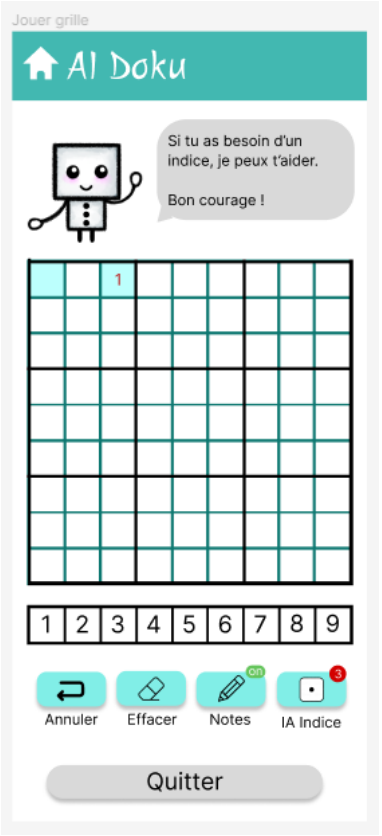


Figure 5 : Maquettage de la page "Jouer grille"

Cependant, lors de la phase de codage, quelques problèmes sont survenus. En effet, certains éléments ne peuvent pas être représentés, ou difficilement, avec la SFML. De plus, étant donné que ce projet priorise la phase de codage plutôt que l'aspect de l'interface et de l'esthétique, des simplifications ont été faites.

Par exemple, l'arrondi des boutons et l'ombre ont été supprimés afin de dégager plus de temps et d'attention sur leur fonctionnalité.

Figure 6 : Comparaison du bouton "Quitter" sur la maquette (à gauche) et sur l'interface réelle du jeu (à droite)

Structure du code

Diagramme

Vue globale

Avant de passer à une explication plus détaillée du code, voyons d'abord la structure de celui-ci. Nous sommes en programmation orientée objet (POO) avec plusieurs classes.
Tout d'abord, des classes pour faire l'affichage du sudoku avec de nombreuses classes boutons (**boutonClose**, **boutonEffacer**, etc) héritées de la classe **Bouton**.
Nous avons aussi des classes **Case** qui ressemblent à des boutons mais n'en sont pas exactement et sont les cases de la grille du sudoku au nombre de 81.
Il y a aussi **IaApparence** qui est une classe permettant d'afficher le personnage, la boîte de dialogue et de faire la gestion du dialogue. À noter qu'il y a eu des changements au cours du projet. En effet, au début, la classe **IaApparence** permettait seulement d'afficher le personnage de l'IA ainsi que la boîte de dialogue et utilisait une autre classe **GestionDialogue** dédiée uniquement à la gestion du dialogue de l'IA. Cependant, j'ai rencontré des problèmes de référence, il y avait trop de classe pour réaliser seulement les changements de paragraphe et de ligne. J'ai donc choisi de fusionner ces deux classes en une seule : **IaApparence**. Ainsi, l'IA peut ajouter des paragraphes dans **IaApparence** en fonction des stratégies qu'elle utilise. Puis, **Grille** fait le lien entre **IaApparence** et **BoutonNavigation** pour passer à la ligne suivante ou précédente.
Toutes ces classes interfaces utilisent la classe **Texte** qui permet de gérer plus facilement le texte des boutons notamment la police, de charger la fonte et de centrer le texte au milieu du cadre par exemple.
Il reste ensuite 3 classes principales qui permettent la résolution du sudoku par l'IA à savoir **Grille**, **IA** et **Sudoku**.

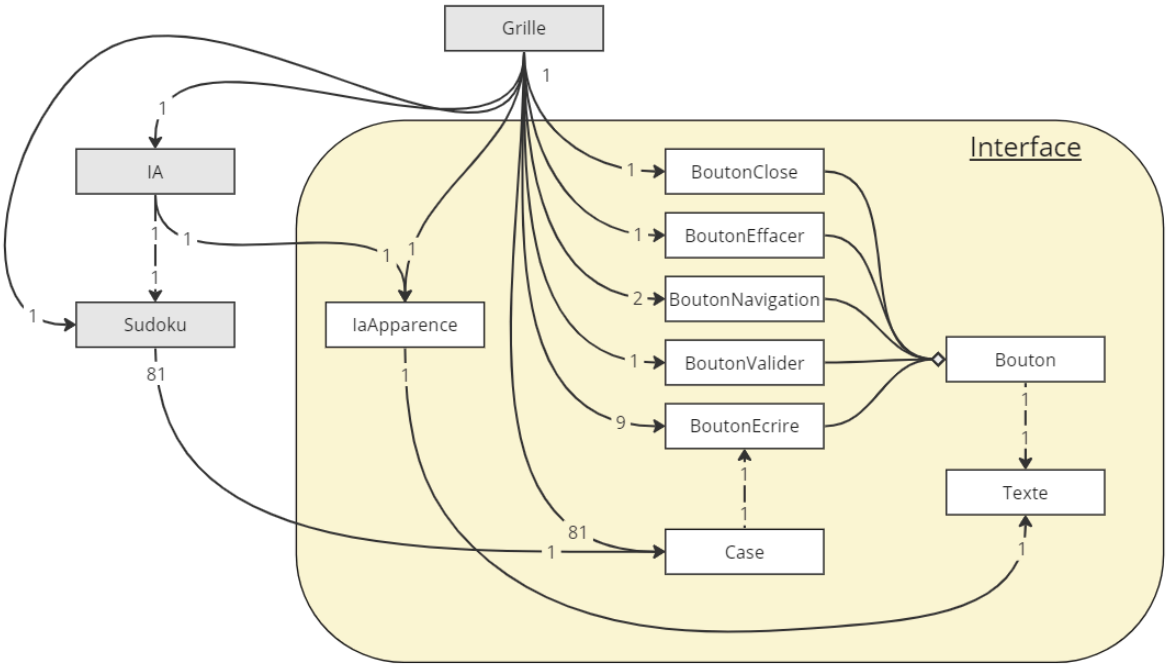


Figure 7 : Diagramme du code - vue globale

Classes principales

Regardons plus en détail ces trois dernières classes.

Nous avons d'abord une classe **Sudoku** qui permet de **générer des niveaux**. Comme l'exigence fonctionnelle EF_04 n'a pas été réalisée, il n'y a pas encore de génération automatique des niveaux. Les niveaux sont actuellement des grilles prédéfinies que j'ai inscrites moi-même à partir du site Sudoku.com.

Avec la classe **Sudoku**, on peut aussi **écrire,** c'est-à-dire écrire un chiffre dans une case, mais aussi **écrire des notes**. En effet, il est possible d'écrire en petit tous les chiffres possibles d'une case. Enfin on peut **mettre à jour l'état** notamment lorsque la grille du sudoku est entièrement complétée.

La classe **Grille** est la plus grosse classe du programme. Elle permet tout d'abord de dessiner l'interface grâce aux classes d'interface que nous avons vu précédemment. Mais aussi de faire **action grille**.

Ainsi, il est alors possible de faire

- soit une **résolution rapide avec l'IA** : on choisit un niveau et l'IA résout entièrement la grille en 1 à 2 secondes.

- On peut aussi faire la **résolution détaillée** : on choisit un niveau ou on entre sa grille incomplète, on appuie sur le bouton “Valider” et l’IA résout la grille, étape par étape, en expliquant à chaque fois la stratégie qu'elle utilise.

Voyons maintenant comment procède l’**IA** pour résoudre des sudokus. Tout d'abord elle prend des notes au tout début de la partie, c'est-à-dire qu'elle écrit en petit dans les cases tous les chiffres possibles. Elle met aussi à jour ses notes à chaque fois qu'elle écrit un chiffre dans une case. Enfin, elle utilise différentes stratégies. Actuellement elle n'en utilise que trois à savoir Singleton évident, Dernier chiffre possible et Paires Nues. Avec seulement ces trois stratégies, elle est capable de résoudre les niveaux faciles, moyens et difficiles de sudoku.com. Il reste encore de nombreuses stratégies à implémenter ce qui permettrait d'augmenter significativement le niveau de l'IA.

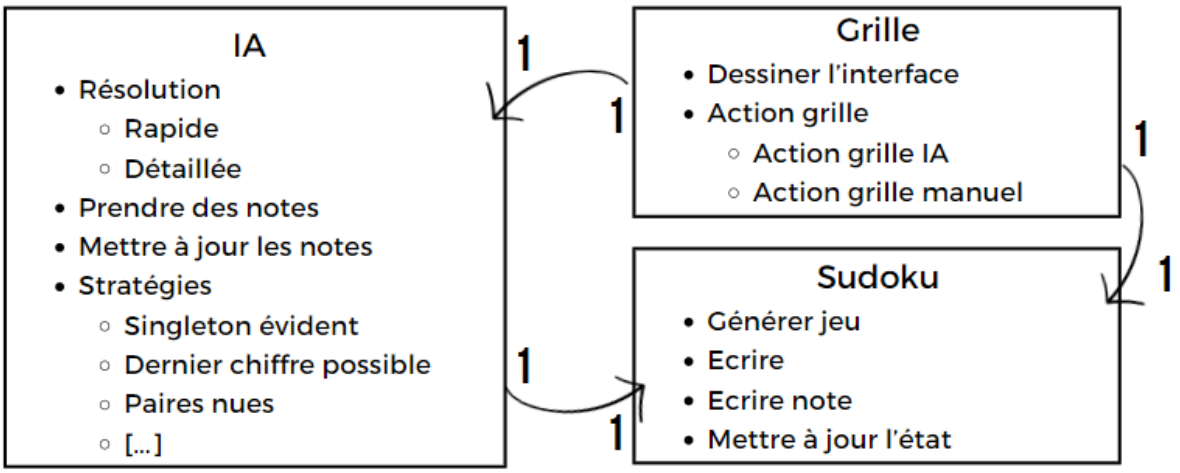


Figure 8 : Diagramme du code - classes principales

BORDEAUX INP Ensc

Résolution de sudoku par l'IA

Lorsque l'on joue au sudoku, plusieurs stratégies peuvent être appliquées. Ainsi, cumuler les stratégies permet de résoudre des niveaux plus difficiles. La conception de l'IA de ce projet se base sur le raisonnement humain afin que cette dernière puisse répondre aux sollicitations de l'utilisateur et lui expliquer quelles stratégies peuvent être utilisées pour continuer.

Ce projet a ainsi l'avantage d'avancer par paliers : à chaque palier atteint, à chaque nouvelle stratégie implémentée, l'IA améliore ses performances pour résoudre des niveaux de Sudoku de plus en plus complexes.

Avancement par niveau

Au début du projet, il n'est pas encore possible de générer automatiquement des grilles de sudoku. Il est donc nécessaire de les écrire manuellement à partir de grilles déjà existantes sur internet. Pour faciliter la recherche de grilles mais aussi d'évaluer leur difficulté sans avoir besoin de les résoudre, le site sudoku.com a été utilisé. Ce site propose des grilles de sudoku en ligne avec 5 niveaux différents (facile, moyen, difficile, expert, maître). Il ne propose en revanche pas les solutions. Ainsi, afin de connaître le niveau de l'IA, on utilise ces niveaux comme référence.

Avant de commencer

Avant de commencer l'utilisation de stratégies, les joueurs aguerris commencent toujours leur sudoku en prenant des notes. Cette méthode permet une vue d'ensemble des possibilités pour chaque case sans avoir à les mémoriser.
L'IA commence donc de la même manière avec la méthode **prendreNote** :

- Pour chaque case vide ij, elle regarde les possibilités de la ligne i
- Elle regarde ensuite les possibilités de la colonne j et ne garde que les chiffres en communs
- Elle regarde les possibilités du carré où se trouve la case ij, les compare et ne garde que celle en commun

- On obtient alors un tableau à 2 dimensions de vecteur de int contenant les possibilités de chaque cases.

De plus, lorsqu'une case est remplie par l'IA, celle-ci doit mettre à jour ses notes. Bien que la même méthode puisse être utilisée, elle n'est pas optimale. Une méthode distincte, **mettreAJourNote**, est spécialement conçue pour cette tâche :

- Pour la case ij où l'on vient d'écrire un chiffre k
- On regarde la ligne i, la colonne j et le carré de la case ij et on supprime le chiffre k des notes de ces cases
- Par sécurité, on vide finalement le contenu du vecteur note de la case ij.

Résoudre des sudokus de niveau facile

La première stratégie apprise à l'IA est le singleton évident ou singleton null grâce à la méthode **singletonEvident** :

- Pour chaque case vide, l'IA regarde ses notes
- Si le nombre de notes d'une case est de un, alors il n'y a qu'une possibilité dans cette case
- Elle écrit dans la case le seul chiffre des notes
- Elle met ensuite à jour ses notes

Avec cette première stratégie, l'IA peut résoudre trois sudokus de niveau très facile et facile :

Figure 9 : Résolution de grilles par l'IA (chiffres en rouge) - niveau très facile (à gauche) et facile (milieu et à droite)

Résoudre des sudokus de niveau moyen

On ajoute ensuite une nouvelle stratégie, celle de la dernière case de libre avec la méthode **dernierChiffrePossible** (utilisant **dernierChiffrePossibleLigne**, **dernierChiffrePossibleColonne**, **dernierChiffrePossibleCarre**) :

- On regarde les possibilités d'une ligne/colonne/carré
- Si un chiffre k n'apparaît que dans une seule case de la ligne/colonne/carré, c'est que c'est le bon
- On écrit le chiffre k dans la case correspondante
- On met à jour les notes

L'IA peut alors résoudre deux nouvelles grilles :

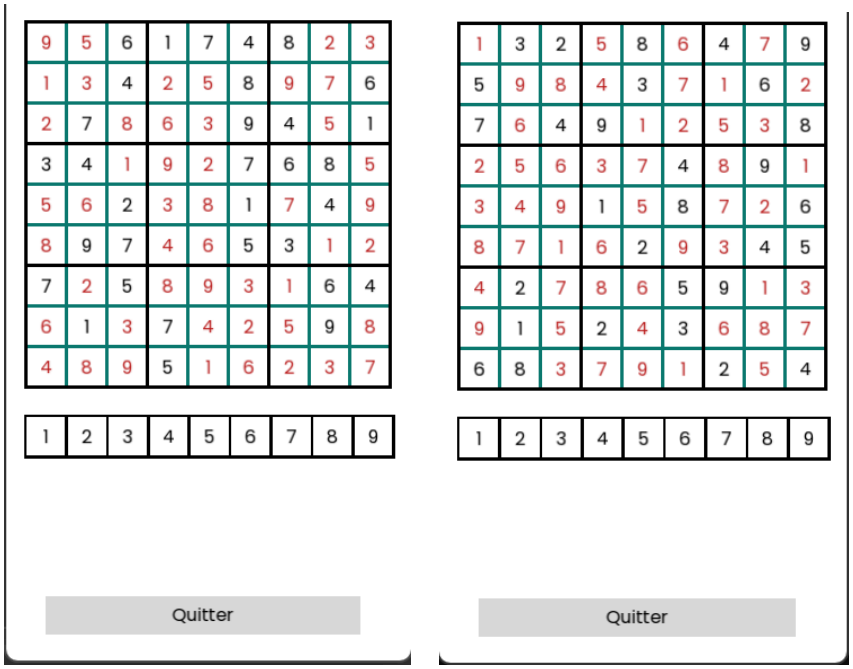


Figure 10 : Résolution de grilles par l'IA (chiffres en rouge) - niveau moyen

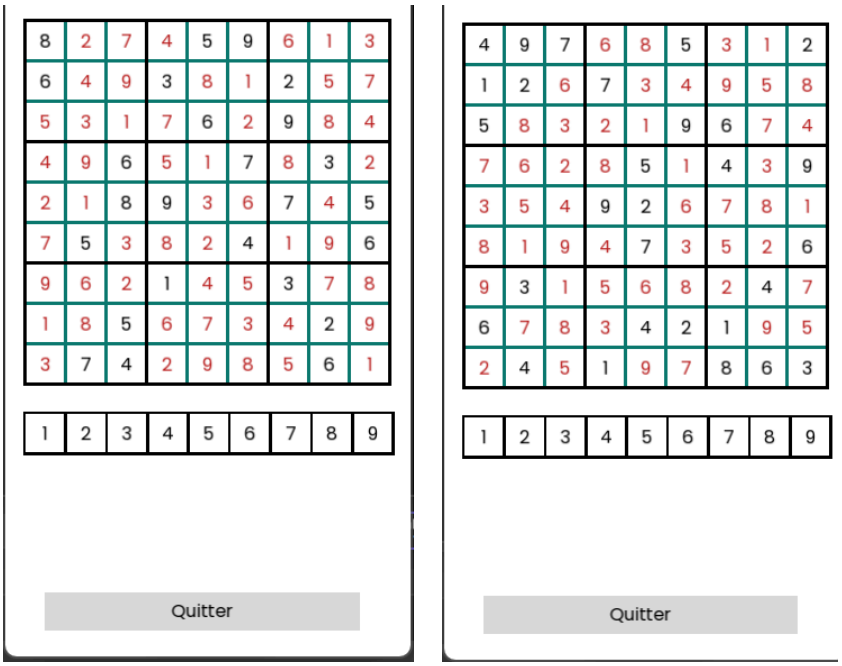
Résoudre des sudokus de niveau difficile

La troisième stratégie permet de retirer des notes inutiles avec la stratégie des paires nues en utilisant la méthode **pairesNues** (utilisant **pairesNuesLigne**, **pairesNuesColonne** et **pairesNuesCarre**) :

- Pour chaque ligne/colonne/carré on cherche un premier couple, une case vide avec seulement 2 chiffres en note.

- Lorsque l'on trouve un couple, on parcourt les autres cases de la ligne/colonne/carré et on cherche un second couple
- Lorsque l'on trouve un second couple, on le compare au premier
    - S'ils sont identiques, on a trouvé une paire ! On parcourt de nouveau toutes les cases de la ligne/colonne/carré et on supprime les chiffres du couple dans les cases (autre que les paires).
    - Sinon, on poursuit la recherche

L'IA peut ainsi terminer les sudokus de niveau difficile :



<u>Figure 11 :</u> Résolution de grilles par l'IA (chiffres en rouge) - niveau difficile

Dialogue de l’IA

Pour expliquer son raisonnement, l’IA possède un dialogue. En fonction de la stratégie qu’elle applique, elle ne dira pas les mêmes lignes de dialogues. On fait donc un texte contenant des paragraphes : on ajoute un nouveau paragraphe spécifique à chaque nouvelle stratégie utilisée. Chaque paragraphe contient des lignes. On peut naviguer entre les lignes grâce aux boutons de navigation.
Si on appuie sur le bouton de navigation pour passer à la ligne suivante alors qu’on est à la fin d’un paragraphe, il faut passer au paragraphe suivant. Or celui-ci n’existe pas encore. On demande alors à l’IA d’utiliser la stratégie la plus facile dans la situation actuelle et de rajouter un paragraphe spécifique à cette stratégie utilisée au texte de son dialogue.
Il existe plusieurs type de paragraphes possible :

- **Initial :** Paragraphe ajouté au lancement du programme (si modeIA=true) dans lequel l’IA se présente et explique le principe et son utilisation.
- **Note :** Paragraphe ajouté lorsque l'utilisateur appuie sur le bouton “Valider” pour enregistrer sa grille et la donner à l’IA pour qu’elle la résolve. La première stratégie à utiliser est toujours de prendre des notes.
- Il y a ensuite les paragraphes pour les stratégies si elles sont utilisées :
    - **SingletonEvident**
    - **DernierChiffrePossible**
    - **PairesNues**
- Enfin il y a deux types d’états finaux
    - **Final :** Si le sudoku a été entièrement complété.
    - **NiveauFaible :** Si l’IA n’a pas réussi à utiliser de stratégie, alors son niveau est trop faible. Elle ne sera pas en mesure de compléter le sudoku entièrement. On arrête alors la résolution et l’IA s’excuse.

Tests des stratégies

Les stratégies et les dialogues implémentés, il est maintenant possible de vérifier si l’IA repère correctement les stratégies à appliquer. Pour cela, j’ai réalisé quelques courts tests avec des grilles dédiées.

Test singletons nuls

On commence par la stratégie la plus simple, celle où il n'y a qu'une seule note dans une case. On fait ici un test avec un carré rempli de 1 à 8, il ne manque que le 9. L'IA prend des notes puis utilise la stratégie des singletons évidents et place le 9 manquant.

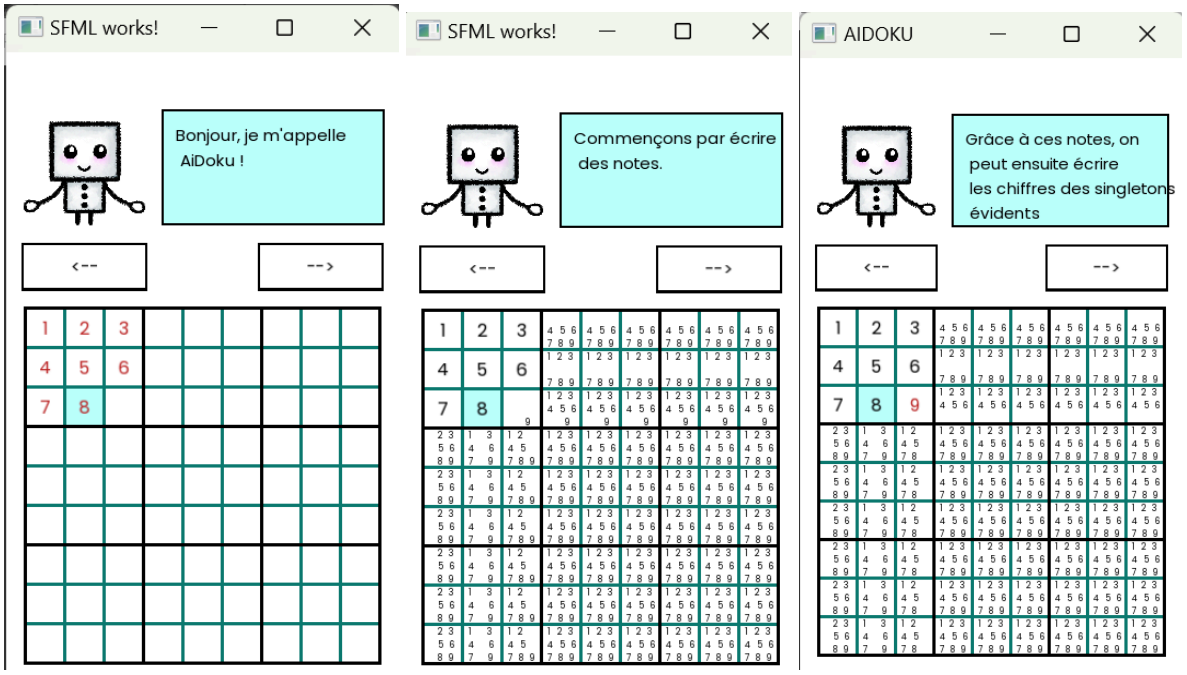


Figure 12 : Test de la stratégie Singleton Évident

Test dernier chiffre possible

On teste ensuite la stratégie du dernier chiffre possible. On place des chiffres dans une colonne en omettant le 5, le 7 et le 9. On place ensuite deux autres 7 de façon à ce que le 7 ne puisse être placé que dans une des cases de la colonne. On laisse ensuite l'IA résoudre. Celle-ci prend bien des notes et applique la stratégie correctement en plaçant le 7 dans la colonne.

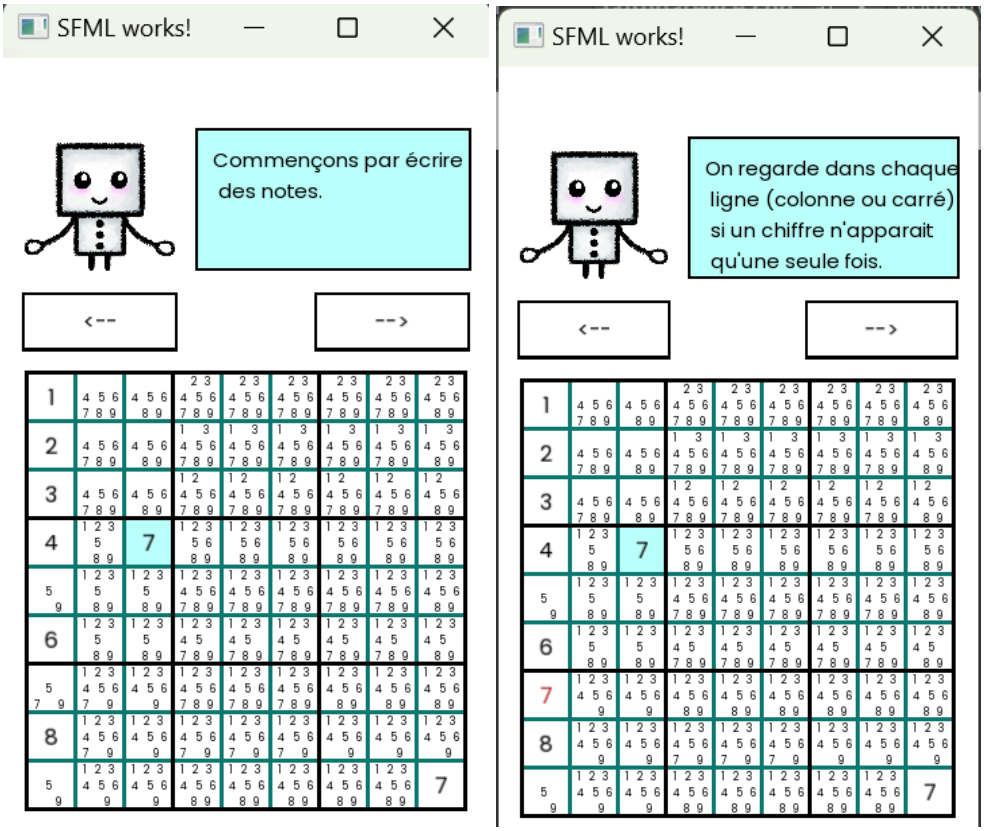


Figure 13 : Test de la stratégie Dernier Chiffre Possible

Test paires nues

On finit par tester la dernière stratégie de l'IA. Elle consiste à retirer des notes. On crée artificiellement une paire nue {2,3} dans le début d'une ligne. On lance ensuite l'IA, celle-ci prend des notes du début de la partie puis applique la stratégie et retire les {2,3} des trois cases de la fin de la ligne afin de ne laisser que les 2 et 3 de la paire nue.

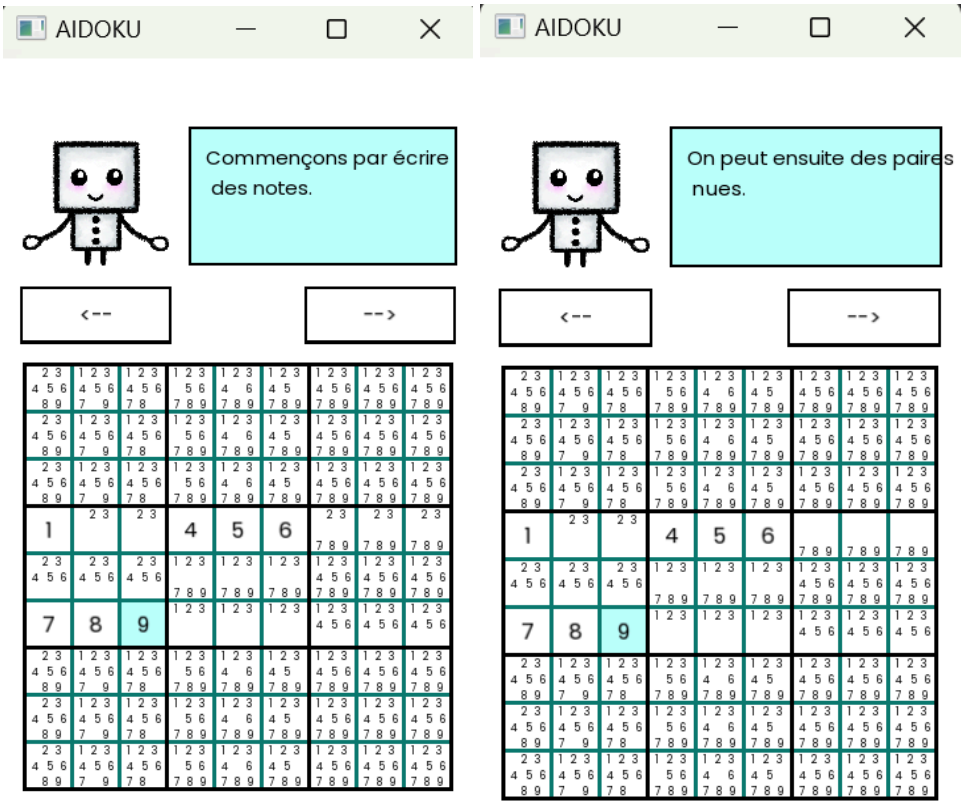


Figure 14 : Test de la stratégie Paires Nues

BORDEAUX INP Ensc

Gestion de projet

Exigences fonctionnelles et fonctionnalités optionnelles

Des objectifs ont été définis dès le début du projet. Ils sont répertoriés dans le tableau ci-dessous :

Exigences Fonctionnelles	
EF_01	Afficher une interface de jeu de Sudoku dans laquelle le joueur peut interagir (écrire ou effacer des chiffres)
EF_02	Résolution autonome de Sudoku par une IA utilisant un raisonnement proche de l'humain
EF_03	L'IA peut être sollicitée par le joueur lors de partie, elle lui donne alors des indices
EF_04	Génération automatique de grilles de sudoku
Fonctionnalités Optionnelles	
FO_01	L'IA possède un visuel, des animations et une voix
FO_02	Le joueur peut envoyer une photo d'une grille de Sudoku afin que l'IA puisse lui donner des conseils
FO_03	L'IA sait résoudre des grilles de Sudokus complexes

Tableau n°1 : Récapitulatif des exigences fonctionnelles et fonctionnalités optionnelles

A la fin de la première partie du projet, ce tableau a été revu. La fonctionnalité FO_02 a été supprimée pour allouer plus de temps à la fonctionnalité EF_03 qui est l'une des plus importantes du projet.

Finalement, à la fin du projet, l'exigence fonctionnelle EF_04 n'a pas été respectée (voir la section planning). La fonctionnalité optionnelle FO_01 a pu être entamée avec la réalisation d'un visuel pour l'IA et d'une animation. Cette dernière n'a cependant pas encore été implémentée dans le programme, le personnage de l'IA est actuellement statique.

Planning et difficultés rencontrées

Planning - Première partie

Afin d'avoir une vue globale du projet, un planning a été réalisé. La première partie du projet sera consacrée à l'apprentissage du langage de programmation C++ ainsi qu'au codage de la base du jeu de Sudoku. Le planning prévoit aussi les fonctionnalités optionnelles, détaillées précédemment dans la section précédente, en jaune.

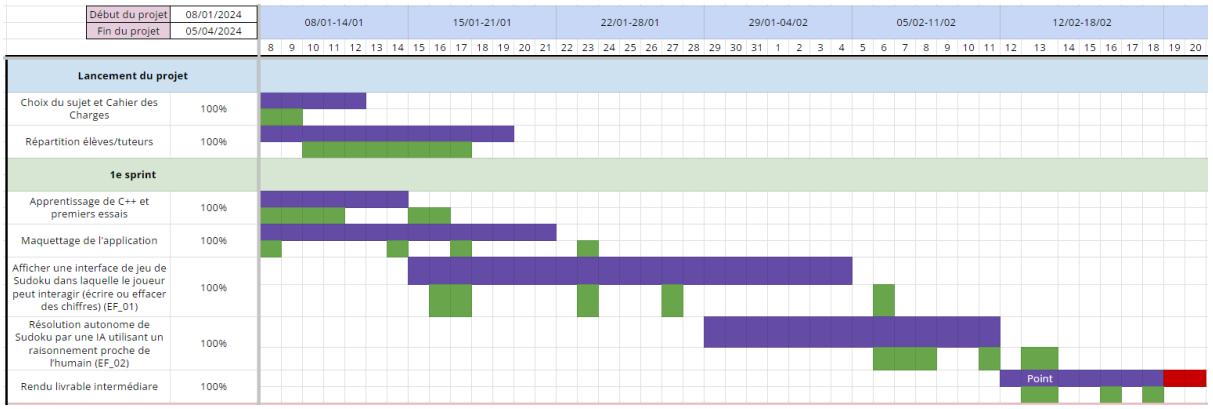


Figure 15-a : Planning du projet - partie 1

	Début du projet	13/09/2023	08/01-14/01	15/01-21/01	22/01-28/01	29/01-04/02	05/02-11/02	12/02-18/02
	Fin du projet	22/12/2023						
1e sprint								
Apprentissage de C++ et premiers essais		100%						
Maquettage de l'application		100%						
Afficher une interface de jeu de Sudoku dans laquelle le joueur peut interagir (écrire ou effacer des chiffres) (EF_01)		100%						
Résolution autonome de Sudoku par une IA utilisant un raisonnement proche de l'humain (EF_02)		100%						
Rendu livrable intermédiaire		100%						

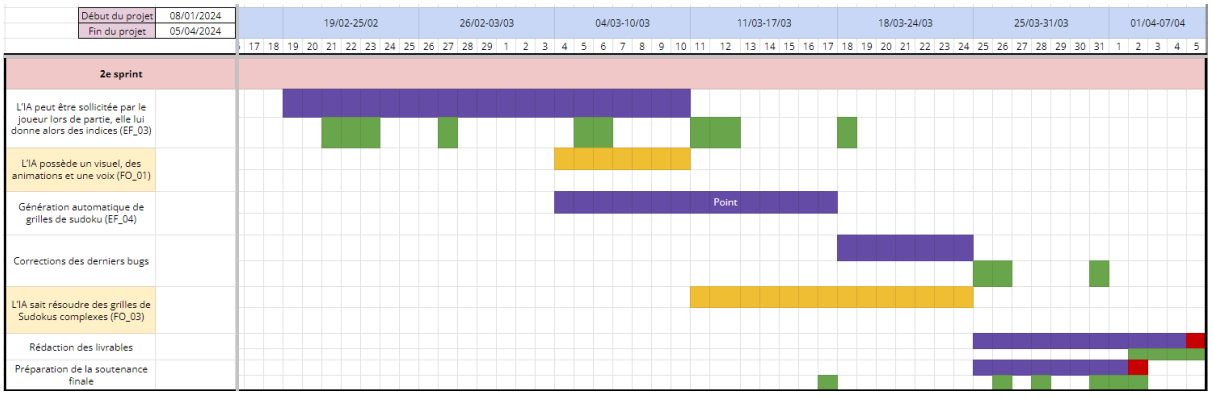
Figure 15-b : Planning du projet résumé - partie 1

Difficultés

Durant la première partie du projet, j'ai rencontré des difficultés sur l'affichage de l'interface avec l'utilisation de la SFML. En effet, cette dernière est difficile à prendre en main au début, parfois capricieuse… Ces difficultés ont donc entraîné un retard d'une semaine. Néanmoins, le retard a été rattrapé avec la résolution de sudoku par l'IA qui a pris moins de temps que prévu. Ainsi, les objectifs de la première partie du projet ont été respectés.

Planning - Seconde partie

Pour ce qui est de la seconde partie du projet, des modifications ont été apportées. Comme évoqué plus haut, la fonctionnalité FO_02 a été supprimée afin de laisser plus de temps pour la fonctionnalité EF_03 qui est l'une des plus importantes du projet.



<u>Figure 16-a :</u> Planning du projet - partie 2

		19/02-25/02	26/02-03/03	04/03-10/03	11/03-17/03	18/03-24/03	25/03-31/03	01/04-07/04
Début du projet	13/09/2023							
Fin du projet	22/12/2023							
2e sprint								
L'IA peut être sollicitée par le joueur lors de partie, elle lui donne alors des indices (EF_03)								
L'IA possède un visuel, des animations et une voix (FO_01)								
Génération automatique de grilles de sudoku (EF_04)								
Corrections des derniers bugs								
L'IA sait résoudre des grilles de Sudokus complexes (FO_03)								
Rédaction des livrables								
Préparation de la soutenance finale								

<u>Figure 16-b :</u> Planning du projet résumé - partie 2

Durant cette seconde et dernière partie, je n'ai pas pu faire la fonctionnalité EF_04. En effet, la fonctionnalité EF_03 m'a demandé plus de temps que prévue à cause de deux difficultés :

- Surcharge de mémoire : trop d'éléments étaient affichés. En effet, chacune des 81 cases possédait 10 textes (1 texte pour le chiffre de la case et 9 textes pour les notes de la case) soit déjà 810 textes pour seulement l'affichage de la grille sans compter les boutons et le dialogue. L'affichage ne se faisait alors par correctement, les textes "en surplus" étaient remplacés par des points :

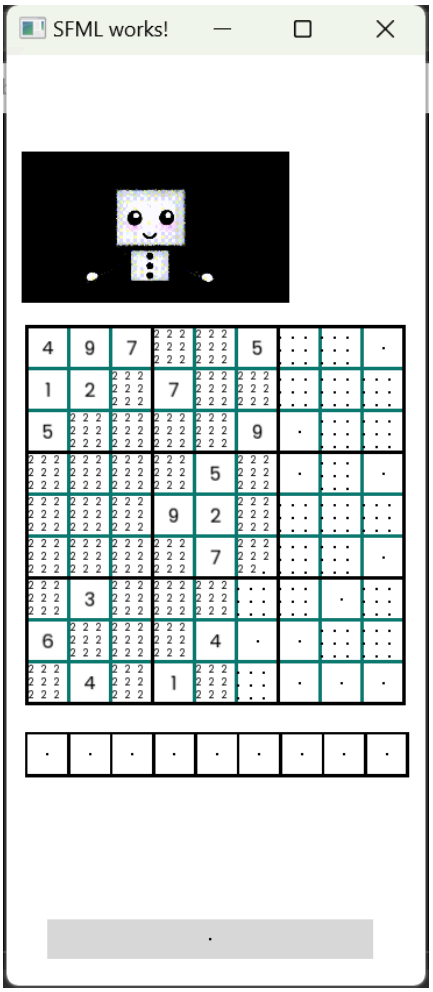


Figure 17 : Exemple de l'interface sans optimisation de l'affichage des textes

Il a donc fallu optimiser les textes. J'ai donc fusionné les 9 textes pour les notes d'une case en un seul texte avec des retours à la ligne (soit par exemple "1 2 3 \n4 5 6 \n7 8 9"). Ainsi, on passe de 810 textes pour le sudoku à 162 (1 texte par case pour le chiffre et 1 texte pour les notes avec des retours à la ligne).

- La seconde difficulté que j'ai rencontrée est un problème de référence. En effet, j'avais trop de classes pour faire la gestion du dialogue de l'IA. J'ai donc fusionné deux classes en une seule et le problème s'est résolu (voir la section diagramme).

Expérience personnelle

J'ai particulièrement apprécié ce projet qui m'a permis de travailler sur un sujet qui me passionne et d'apprendre en autonomie un nouveau langage. Cela m'a aussi permis de créer une intelligence artificielle étroite (ANI) dans le sens où elle est spécialisée uniquement sur la résolution de sudoku.

Je tire surtout des leçons de ce projet :

- Tout d'abord, qu'il est important de tester les bibliothèques avant de se lancer sur un gros projet. En effet, en travaillant sur ce projet avec la SFML je me suis rendu compte que cette bibliothèque n'était peut-être pas assez performante et adaptée pour ce que je souhaite faire à l'avenir. Aussi j'envisage de comparer les langages de programmation et les bibliothèques avant de continuer davantage dans la suite de ce projet.
- De plus, j'ai appris qu'il était important de ne pas être trop ambitieux sur le planning lorsque l'on a prévu d'utiliser de nouveaux outils. En effet, le temps prévu peut se révéler trop court pour corriger d'éventuelles erreurs.

BORDEAUX INP Ensc

Perspective du projet

Ce projet s'achève finalement après 13 semaines de travail en autonomie et un suivi hebdomadaire régulier avec ma tutrice. J'ai ainsi pu apprendre de nouvelles technologies telles que C++ et l'utilisation de la bibliothèque SFML. J'ai également développé une intelligence artificielle basée sur la stratégie humaine. Ceci utilise actuellement trois stratégies différentes, ce qui lui permet de résoudre les niveaux facile, moyen et difficile de Sudoku.com.

Le projet est bien avancé mais il n'est pas encore terminé et il reste de nombreuses choses intéressantes à faire.

- Tout d'abord, faire l'exigence fonctionnelle que je n'ai pas eu le temps de terminer à savoir générer automatiquement des niveaux.
- Il serait aussi intéressant de rendre l'IA un peu plus vivante. En effet, même si cette dernière possède un visuel, elle est cependant assez statique. Une animation a été réalisée mais elle n'a pas encore été implémentée, il serait donc judicieux de le faire. Enfin, cette dernière pourrait avoir une voix grâce à un générateur de voix comme par exemple "Narakeet", associé à l'utilisation d'Audacity.
- Ensuite, il serait pertinent d'accélérer le remplissage d'une grille. En effet, actuellement l'utilisateur est contraint de remplir manuellement sa grille sur l'interface, ce qui est très long et pénible. Il serait plus intéressant que l'utilisateur puisse prendre une photo de sa grille incomplète, l'envoyer au programme, posséder à un traitement de l'image et que la grille soit automatiquement remplie grâce à cette photo.
- Enfin, afin que l'IA soit plus performante, il serait nécessaire de coder des stratégies plus complexes, telles que le X-wing. Ainsi, elle pourrait alors résoudre n'importe quel niveau de sudoku, y compris le niveau master de sudoku.com.

Annexes

Sudoku et stratégies

Règles du Sudoku

Même si les règles du Sudoku sont connues par un grand nombre de personnes, nous allons les rappeler rapidement ici.

Un jeu de Sudoku, sur format papier ou digital sur des applications, se présente sous la forme d'une grille de 9x9, elle-même divisée en 9 carrés de 9 cases chacun.

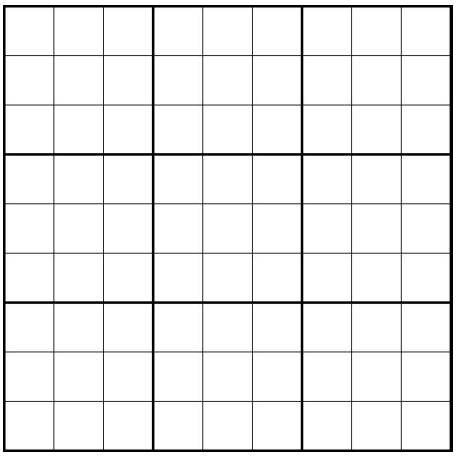


Figure 18 : Grille de sudoku vierge

Ces grilles sont ensuite complétées par des numéros allant de 1 à 9. Plus il y a de chiffres inscrits initialement dans la grille, plus le niveau du sudoku est relativement simple.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Figure 19 : Grille de sudoku remplie

Le joueur doit ensuite placer des chiffres allant de 1 à 9 dans les cases vides. Il doit cependant respecter plusieurs contraintes :

- Chaque ligne horizontale doit contenir les 9 chiffres de 1 à 9 sans aucune répétition
- Chaque colonne verticale doit contenir les 9 chiffres de 1 à 9 sans aucune répétition
- Chaque carré de 3x3 doit contenir les 9 chiffres de 1 à 9 sans aucune répétition

Nous sommes donc ici dans un problème de résolution de contrainte.

Stratégies simples (débutant et niveau moyen)

Pour des grilles de sudoku simples, le joueur va utiliser de nombreuses stratégies pour finir le jeu :

Notes

Le joueur peut écrire en petit dans chaque case tous les chiffres qui pourraient convenir. Pour cela, il peut regarder chaque ligne, colonne et carré dans l'ordre de son choix. C'est une stratégie par élimination. Cette méthode permet donc d'avoir une vision globale de toutes les possibilités de chaque case.

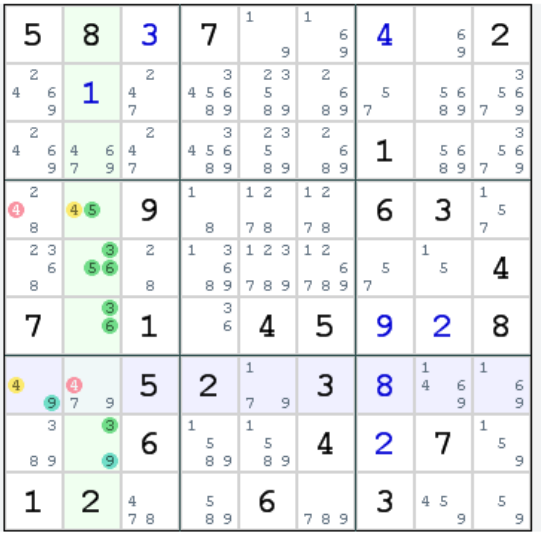


Figure 20 : Grille de sudoku avec notes

Singleton null

Si dans une ligne, une colonne ou un carré, une case ne contient qu'un seul chiffre possible, alors ce chiffre doit être placé dans cette case.

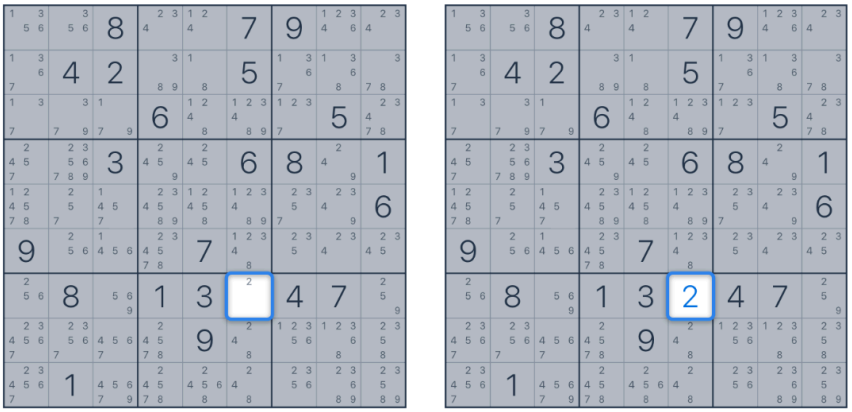


Figure 21 : Grille de sudoku - exemple de l'utilisation de la stratégie "singleton null"

Dernière case de libre

Si l'une des lignes, colonnes, carré ne contient qu'une seule case vide, il suffit de la compléter par le chiffre manquant.

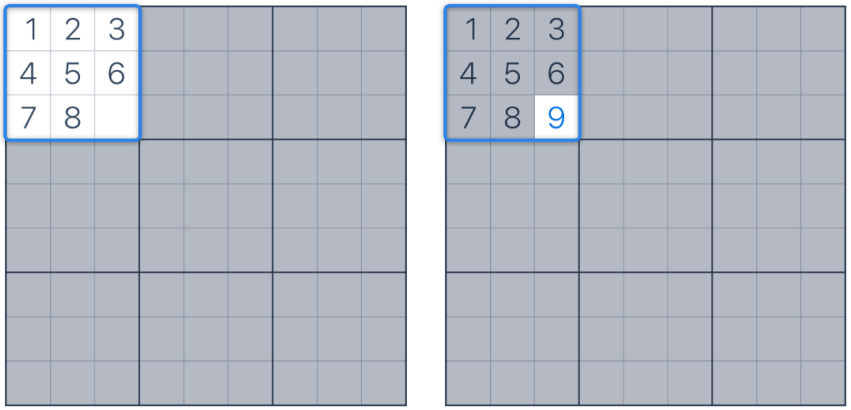
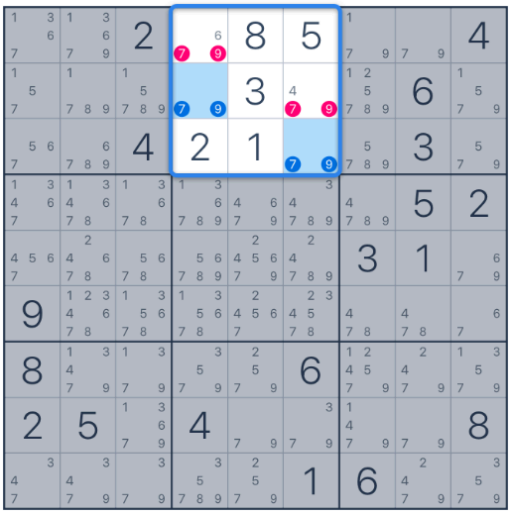


Figure 22 : Grille de sudoku - exemple de l'utilisation de la stratégie "dernière case de libre"

Paires nues

Si dans une ligne, une colonne ou un carré, on observe une paire de chiffres sans autre possibilité (des paires nues) alors tout autre chiffre de cette paire présent dans les autres cases de la ligne, colonne ou carré observé doit être retiré des notes.



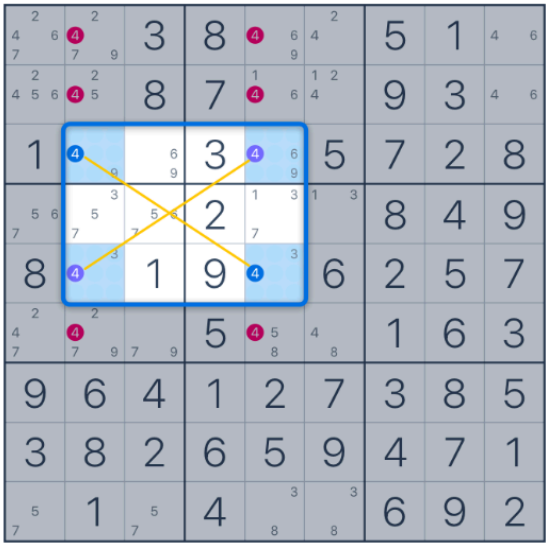
<u>Figure 23 :</u> Grille de sudoku - exemple de l'utilisation de la stratégie "paires nues"

Stratégies complexes

Si les stratégies simples que l'on vient de voir précédemment permettent de résoudre une grande majorité des sudokus, elles ne suffisent pas à résoudre des grilles plus complexes. Il existe de nombreuses méthodes de résolution plus complexes, nous allons en détailler quelques-unes ci-dessous.

X-wing

Il faut observer les lignes et les colonnes parallèles et s'intéresser à la redondance d'un chiffre spécifique. Si l'on trouve un chiffre se répétant dans chacun des angles d'un rectangle (voir image ci-dessous), alors tout autre itération de ce chiffre dans les colonnes et lignes peut être retirée (chiffre en rose dans l'image ci-dessous).



<u>Figure 24 :</u> Grille de sudoku - exemple de l'utilisation de la stratégie "X-wing"

BORDEAUX INP Ensc

Conseils d'utilisation

Généralités

Pour une utilisation optimal du programme :

- Ne mettez pas en plein écran la fenêtre de jeu du sudoku
- Soyez patient lorsque vous cliquez sur une case. Essayez avec le touchpad en cas de latence trop importante.

Choix des paramètres

Ce projet possède plusieurs mode à choisir au lancement du programme :

- Mode sans IA : Jouer au sudoku de façon classique
- Avec IA
    - Résolution rapide : L'IA résout la grille de sudoku instantanément.
    - Résolution détaillée : L'IA résout la grille donnée, étape par étape, en donnant des explications.

Dans tous les cas, vous devez choisir un niveau de sudoku :

- Niv 1 : Très facile
- Niv 2 et 3 : Facile
- Niv 4 et 5 : Moyen
- Niv 6 à 8 : Difficile
- Niv "0" : (sauf avec l'utilisation de la **Résolution rapide** par l'IA) : La grille de sudoku est vierge, le joueur doit entrer sa propre grille.

Détails des actions en jeu

La fenêtre de jeu comporte de nombreux boutons. Les voici ci-dessous présentées de façon exhaustives avec le mode IA - Résolution détaillée :

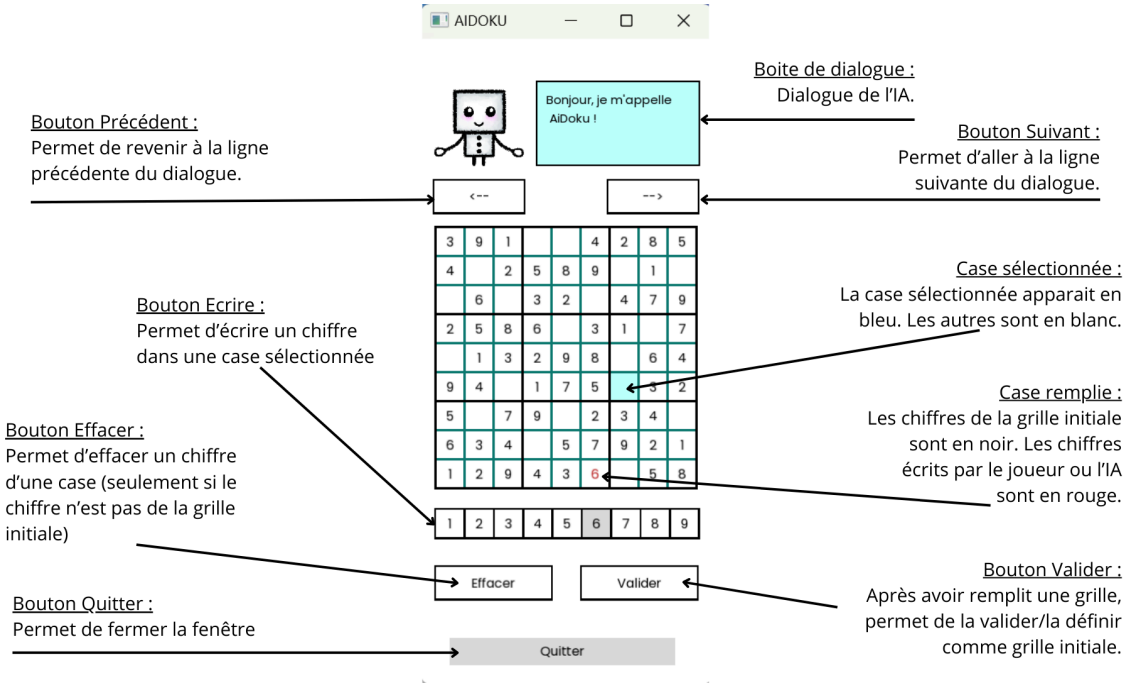


Figure 25 : Notice des actions possibles en jeu

Pour **écrire** un chiffre dans une case, il faut :

1) Cliquez sur la **case** souhaitée
2) Cliquez sur un des **bouton écrire**

Pour **effacer**, il faut :

1) Cliquez sur la **case** souhaitée
2) Cliquez sur le bouton "**Effacer**"