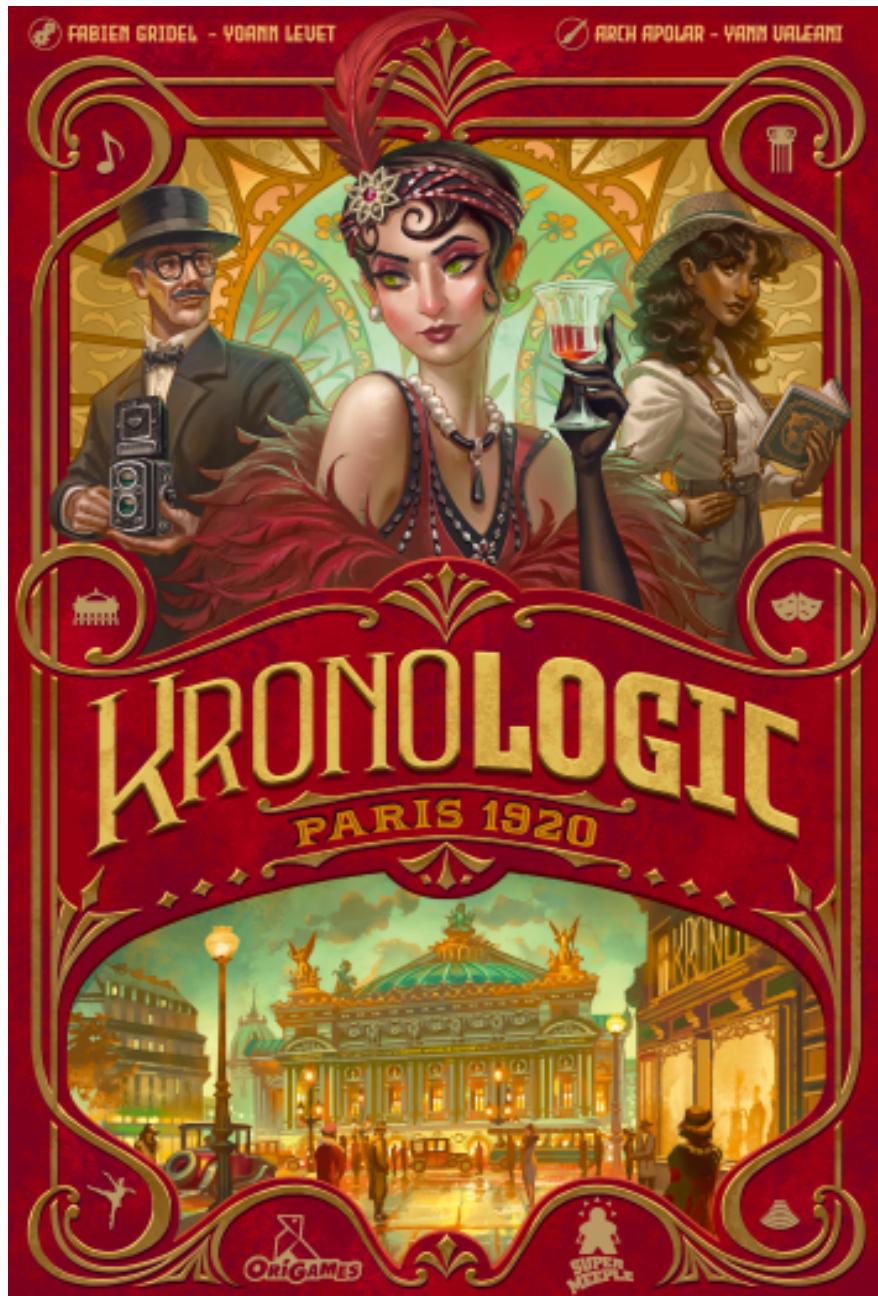


## Projet tutoré - Document de fin d'itération 4 Kronologic





# Table des matières

<b>Table des matières</b>	<b>3</b>
<b>1. Introduction</b>	<b>4</b>
1.1. Objet du document	4
1.2. Présentation du projet	4
<b>2. Analyse</b>	<b>5</b>
2.1. Découpage fonctionnel	5
2.2. Modèles UML utilisés	5
2.3. Évolution du projet par rapport à l'étude préalable	6
<b>3. Réalisation</b>	<b>7</b>
3.1. Architecture logicielle	7
3.2. Tests	7
3.2.1. Jeu	7
3.2.2. IA	7
3.3. Difficultés rencontrées	7
Problèmes identifiés :	8
<b>4. Planning du déroulement des 4 premières itérations</b>	<b>9</b>
4.1. Itération 1	9
4.2. Itération 2	9
4.3. Itération 3	10
4.4. Itération 4	11
<b>5. Répartition du travail</b>	<b>12</b>
5.1. Itération 1	12
5.2. Itération 2	12
5.3. Itération 3	13
5.4. Itération 4	14
<b>6. Nos réalisations</b>	<b>15</b>
6.1. Réalisation de Corentin	15
6.2. Réalisation de Mathieu	17
6.3. Réalisation d'Enzo	20
<b>7. Objectifs de fin de projet</b>	<b>24</b>
<b>8. Planning des trois dernières itérations</b>	<b>25</b>
8.1. Itération 5	25
8.2. Itération 6	25
8.3. Itération 7	25
<b>9. Conclusion</b>	<b>26</b>

# 1. Introduction

## 1.1. Objet du document

Ce document a pour objectif de faire le point sur l'avancement du projet en évaluant les progrès réalisés par rapport au planning établi lors de l'étude préalable. Il permettra ainsi de mesurer les écarts éventuels entre les prévisions et la réalité, afin d'identifier les ajustements nécessaires pour assurer la bonne conduite du projet.

Dans un premier temps, nous examinerons la répartition des tâches au sein de l'équipe ainsi que le travail accompli au cours des quatre premières itérations. Cette analyse nous aidera à mieux comprendre la dynamique de collaboration et l'efficacité de notre organisation.

Ensuite, chaque membre de l'équipe présentera une réalisation dont il est particulièrement fier. Ce partage d'expériences mettra en lumière des avancées significatives, des défis surmontés et des apprentissages majeurs ayant contribué à l'évolution du projet.

Dans la dernière partie, nous définirons les objectifs à atteindre d'ici la fin du projet et détaillerons le planning des trois dernières itérations. Enfin, nous conclurons en dressant un bilan global de notre progression et en évoquant les perspectives pour la suite.

## 1.2. Présentation du projet

Le projet tutoré "Kronologic" vise à concevoir une intelligence artificielle (IA) capable d'accompagner ou de remplacer un joueur dans un jeu de déduction complexe. Le jeu de plateau Kronologic, créé par Fabien Gridel et Yoann Levet et édité par SuperMeeple en 2024, plonge les joueurs dans une enquête criminelle où ils doivent reconstituer les événements autour d'un meurtre en collectant des indices et en déduisant l'identité du coupable. Pour ce faire, le joueur doit analyser les lieux, les moments clés et les déplacements des personnages, tout en respectant des règles strictes de déduction.

Les objectifs de ce projet sont :

- ❖ Assimiler les mécanismes et règles du jeu et les intégrer dans une interface intuitive et accessible ;
- ❖ Développer une IA capable d'assister le joueur en réalisant des déductions logiques à partir des indices, et garantir que ses raisonnements sont corrects ;
- ❖ Concevoir une IA joueuse qui puisse analyser les indices et identifier rapidement le coupable ;
- ❖ Élaborer un générateur de scénarios permettant de créer de nouvelles enquêtes avec des indices variés, en s'assurant que chaque problème

proposé offre une solution unique, accessible et exigeant une combinaison astucieuse d'informations pour être résolu.

## 2. Analyse

### 2.1. Découpage fonctionnel

Module	Description
Données	<i>Gestion des données du jeu et lecture des scénarios depuis un fichier JSON.</i>
IA	<i>Contient les différentes IA du projet : l'IA de déduction, l'IA d'assistance et l'IA joueuse.</i>
Jeu	<i>Regroupe tous les éléments du jeu, y compris les personnages, les lieux, les indices, le déroulement d'une partie, etc...</i>
MVC	<i>Structure dédiée à l'affichage et aux interactions avec le joueur, en assurant la séparation entre la logique du jeu et l'interface utilisateur.</i>

### 2.2. Modèles UML utilisés

**Diagramme de classes** : pour montrer la structuration des différents modules (Jeu, IA, MVC et Données) du projet en modélisant les relations entre les classes principales. Ce qui a permis d'organiser l'ensemble du projet.

**Diagramme de séquence** : pour illustrer les interactions entre les différentes classes et méthodes du projet, notamment entre le joueur et l'interface graphique pour la gestion des questions et des hypothèses, ainsi qu'entre la partie et la gestion des indices pour mettre à jour les connaissances au fur et à mesure du jeu. Ce qui nous a aidés à clarifier la logique des échanges et à optimiser les appels aux différentes méthodes des classes impliquées.

**Diagramme de cas d'utilisation** : pour illustrer les interactions principales entre le joueur, (poser des questions, formuler des hypothèses, interagir avec les différentes composantes du système), l'IA de Déduction (analyse les indices et propose des hypothèses logiques en fonction des informations disponibles), l'IA d'Assistance (aide le joueur en recommandant des questions optimales et en corrigeant ses déductions si nécessaire), l'IA Joueuse (simulate une partie en s'aidant de l'IA d'Assistance) et le Générateur de scénarios (permet de créer des scénarios d'enquête).

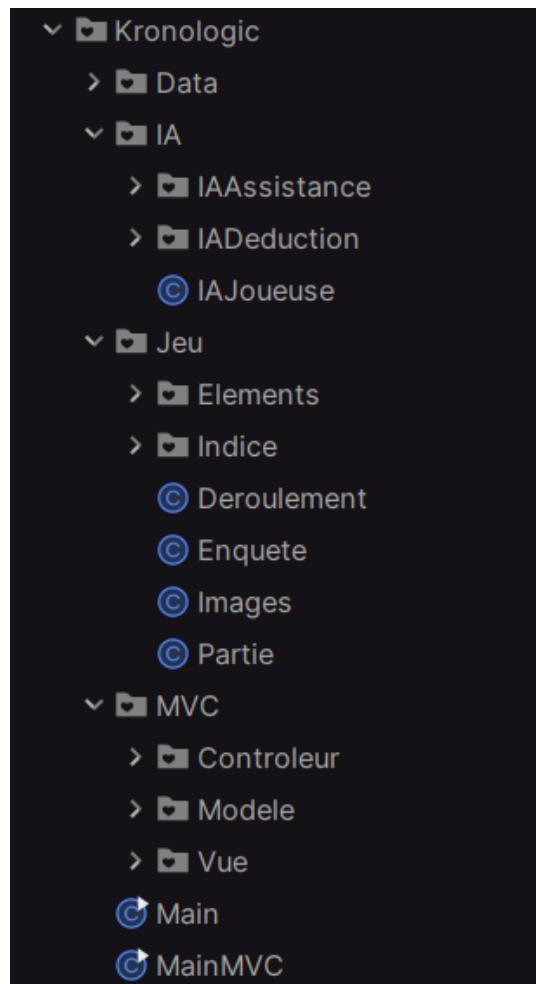
**Diagramme d'activité** : pour représenter le déroulement du jeu en se concentrant sur les actions du joueur et du système. Ce qui permet de détailler le déroulement général d'une partie, depuis le lancement de l'application jusqu'à la fin du jeu, les différentes interactions possibles, telles que poser une question, formuler une hypothèse ou consulter les indices, l'impact des choix du joueur sur l'état du jeu, notamment l'ajout et la mise à jour des indices en fonction des réponses obtenues, puis l'utilisation de l'IA Déduction et de l'IA Assistante, qui permettent d'affiner les hypothèses et d'aider le joueur dans son raisonnement.

## 2.3. Évolution du projet par rapport à l'étude préalable

Tâches	Itération prévue	Itération réalisée
Apprentissage de Choco-Solver	Itération 1	Itération 1
Prototypes de Choco-Solver	Itération 1	Itération 1
Encodage des données du jeu	Itération 1	Itération 1
Architecture MVC	Itération 1	Itération 1
Commencer le développement de l'interface graphique	Itération 1	Itération 1
Terminer l'interface graphique	Itération 2	Itération 4
Pose de question	Itération 2	Itération 1
Visualisation des indices	Itération 2	Itération 2
Réalisation d'hypothèses	Itération 2	Itération 4
Réalisation d'une déduction	Itération 2	Itération 2
Implémenter les notions de victoire et de défaite	Itération 2	Itération 2
Développer IA de Déduction Choco-Solver	Itération 3/4	Itération 3
Développer IA de Déduction Heuristique	Itération 3/4	Non achevée
Comparer les performances des deux IA de Déduction	Itération 3/4	Non commencée
Intégrer les suggestions de l'IA dans l'interface graphique	Itération 3/4	Non achevée
Développer IA d'Assistance Choco-Solver	Itération 5/6	Commencée
Développer IA d'Assistance Heuristique	Itération 5/6	Non commencée
Comparer les performances des deux IA d'Assistance	Itération 5/6	Non commencée
Permettre une interaction fluide entre l'IA d'Assistance et le Joueur	Itération 5/6	Itération 4
IA Joueuse	Optionnelle	Non commencée
Générateur de scénario	Optionnelle	Non commencée

## 3. Réalisation

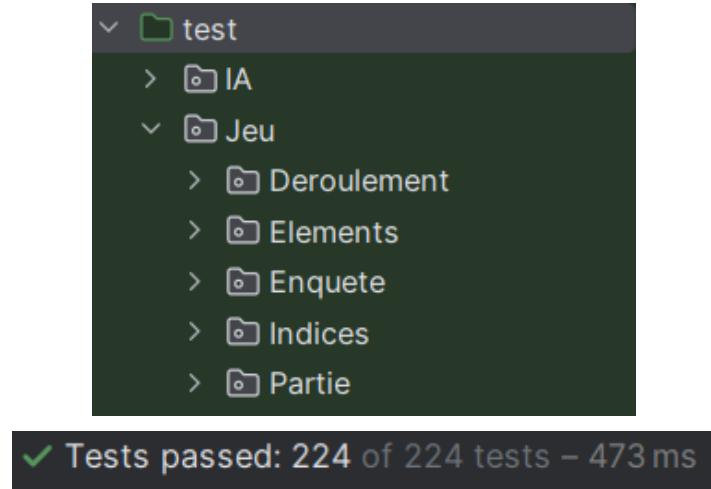
### 3.1. Architecture logicielle



### 3.2. Tests

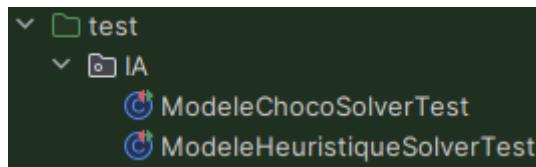
#### 3.2.1. Jeu

Pour les tests réalisés sur les bases du jeu et ces fonctionnalités, nous avons pris soin d'être exhaustif pour nous assurer du bon déroulement du développement. Voici ce que nous avons testé :



### 3.2.2. IA

Nous avons aussi testé l'IA de déduction Choco-Solver, afin de vérifier son bon fonctionnement sans devoir tout relancer à chaque fois. Ces tests ont permis de s'assurer de la cohérence des contraintes, de la pertinence des réductions de domaines, et de s'assurer de l'obtention des bons résultats. Pour ce qui est de l'IA de déduction heuristique, les tests n'ont pas encore été réalisés mais ils vont être faits lors de la cinquième itération.



## 3.3. Difficultés rencontrées

Le **placement automatique des pions** a constitué un défi majeur dans le développement du jeu, car il implique de respecter des contraintes spatiales et logiques tout en offrant une expérience utilisateur fluide. Cette problématique a évolué au fil des itérations du projet, chaque étape permettant d'affiner la solution.

### Problèmes identifiés :

- ❖ **Définition des positions initiales** : Il fallait déterminer comment et où placer automatiquement les pions au début du jeu en respectant les règles du scénario.
- ❖ **Interaction avec le joueur** : Assurer une transition fluide entre le placement automatique et le déplacement manuel des pions.
- ❖ **Synchronisation des affichages** : Garantir une cohérence entre la carte et le tableau des positions.
- ❖ **Placement des pions aux bons endroits** : Le placement automatique des pions a été un véritable défi. Notre problème principal a été de bien les placer dans les bonnes sous-zones. Cela s'est résolu en revoyant notre architecture et notre structure MVC.

## 4. Planning du déroulement des 4 premières itérations

### 4.1. Itération 1

L'objectif principal de cette première itération est de poser les fondations techniques du projet en développant les éléments fondamentaux du jeu.

- **Implémentation du jeu** : Définition des mécanismes de base du jeu et mise en place de l'environnement de développement.
- **Encodage des données du jeu** : Modélisation des données du jeu, telles que les personnages, les lieux et les indices, en structurant ces informations de manière efficace.
- **Pose de question** : Développement de la première fonctionnalité du jeu : la pose de question (avec *Lieu et Temps* et *Lieu et Personnage*).
- **Architecture MVC** : Adoption de l'architecture Modèle-Vue-Contrôleur (MVC) afin de garantir une séparation claire entre la logique métier, l'interface utilisateur et les contrôleurs.
- **Développement de l'interface graphique** : Création d'une première version de l'interface utilisateur, avec une mise en place des éléments visuels principaux.
- **Suivi du tutoriel Choco-Solver** : Approfondissement des connaissances sur Choco-Solver, un outil de résolution de contraintes, afin de mieux comprendre son intégration dans le projet.
- **Réalisation d'un prototype Choco-Solver** : Développement de prototypes utilisant Choco-Solver pour valider la faisabilité des mécanismes de déduction du jeu.

### 4.2. Itération 2

Cette phase est axée sur l'amélioration de l'affichage et sur l'introduction des fonctionnalités interactives du jeu.

- **Réalisation du changement d'affichage** : Amélioration de l'interface pour mieux visualiser les informations et offrir une expérience utilisateur plus fluide.
- **Formulation d'une déduction** : Développement du mécanisme permettant au joueur de réaliser une déduction avec les indices qu'il a récolté et ainsi essayer de gagner.
- **Mise en place du Drag and Drop des pions** : Ajout d'une fonctionnalité permettant aux joueurs de déplacer les pions sur les zones de jeu via un système de glisser-déposer.

- **Implémentation des pop-ups** : Création de fenêtres contextuelles fournissant des informations utiles, comme des résultats de déduction ou des indices débloqués.
- **Développement de l'intelligence artificielle (IA)** : Mise en place du Modèle Choco-Solver et de la première version de l'IA de déduction.
- **Définition des variables et contraintes initiales** : Mise en place des différentes variables et des contraintes initiales du temps 1.
- **Élaboration des contraintes de règles** : Mise en place des contraintes de déplacements (salles adjacentes et déplacement obligatoire).
- **Automatisation de l'ajout des contraintes en fonction des indices** : À chaque nouvel indice découvert, des nouvelles contraintes sont ajoutées à l'IA pour réduire les possibilités des positions des personnages.

### 4.3. Itération 3

Cette phase se concentre sur l'amélioration de l'expérience utilisateur, l'ajout de nouvelles mécaniques et l'optimisation de l'intelligence artificielle.

- **Affichage d'un pop-up expliquant l'énigme** : Mise en place d'un récapitulatif interactif expliquant le contexte et les objectifs du jeu.
- **Affichage des règles** : Intégration d'une section expliquant les règles du jeu de manière claire et accessible.
- **Affichage des films (terminal)** : Première version de l'historique des notes du joueur et du film réel de la partie.
- **Gestion des pions et des notes du joueur** : Introduction d'un système permettant aux joueurs de gérer leurs pions et de prendre des notes avec ceux-ci.
- **Création des zones de Drag and Drop (placement automatique des pions)** : Amélioration du système de glisser-déposer en définissant des zones précises pour un placement automatique.
- **Implémentation du placement automatique des pions au temps 1** : Ajout d'un système positionnant automatiquement les pions au début du jeu.
- **Réalisation de tests** : Validation exhaustive des fonctionnalités du jeu et de l'IA par des tests approfondis pour détecter et corriger d'éventuels bugs.
- **Synchronisation des deux affichages de prise de notes (Carte et Tableau)** : Amélioration de l'interface pour assurer une cohérence entre les deux différentes façons de visualiser les indices.
- **Contraintes des indices publics de l'IA de déduction Choco-Solver** : Gestion de l'ajout des contraintes des indices publics.
- **Ajout des variables du coupable** : Intégration des variables du personnage coupable, du lieu du crime et du temps du crime.
- **Ajout de la contrainte du coupable** : Mise en place de règles assurant que le coupable est le seul présent avec la victime au moment du crime.

- **Affichage du coupable** : Ajout d'un affichage permettant au joueur de voir qui est désigné comme coupable lorsque les indices sont suffisants.
- **Implémentation de l'IA de déduction heuristique** : Mise en place d'une seconde IA de Déduction utilisant des algorithmes heuristiques.
- **Définition des domaines en booléen** : Représentation en booléens pour simplifier le traitement des données.
- **Définition des positions initiales** : Mise en place des contraintes de placement initial de début de partie.

#### 4.4. Itération 4

Cette phase se concentre sur la synchronisation des éléments du jeu et l'ajout de nouvelles mécaniques permettant d'enrichir l'expérience du joueur.

- **Synchronisation des pions de nombre** : Assurer que les pions de nombre affiché soient toujours cohérent avec les informations en jeu. Assurer que la *Vue Carte* et la *Vue Tableau* représentent les mêmes données.
- **Gestion des pions d'absences et d'hypothèses (données et affichage)** : Ajout d'un système permettant de poser des pions d'absences et d'hypothèses (ou d'hypothèse d'absence) de manière dynamique.
- **Réalisation du film de la partie** : Création d'un récapitulatif visuel retraçant les étapes clés de la partie.
- **Réalisation du film du joueur** : Création d'un récapitulatif visuel retraçant les notes prises par le joueur à chaque question lui permettant de retracer ses prises de décision.
- **Amélioration de l'IA de déduction heuristique** : L'IA de déduction heuristique ne prenait en compte que les positions initiales. Elle est à présent capable de raisonner sur les nouveaux indices récupérés à chaque question.
- **Ajout d'un nouvel indice** : L'IA de déduction heuristique effectue de la réduction de domaines en fonction des indices récoltés.
- **IA Assistance Choco-Solver** : Intégration d'un système d'assistance utilisant Choco-Solver pour aider le joueur.
- **Correction des déductions du joueur** : Mise en place d'une correction des déductions du joueur en comparaison des déductions de l'IA de déduction.

## 5. Répartition du travail

### 5.1. Itération 1

Tâches	Mathieu	Corentin	Enzo
<i>Implémentation du jeu</i>			X
<i>Encodage des données du jeu</i>		X	X
<i>Pose de question</i>		X	
<i>Architecture MVC</i>		X	X
<i>Développement de l'interface graphique</i>		X	X
<i>Suivi du tutoriel Choco-Solver</i>	X		
<i>Réalisation d'un prototype Choco-Solver</i>	X		

### 5.2. Itération 2

Tâches	Mathieu	Corentin	Enzo
<i>Réalisation du changement d'affichage</i>			X
<i>Formulation d'une déduction</i>		X	X
<i>Mise en place du Drag and Drop des pions</i>		X	
<i>Implémentation des pop-ups</i>		X	
<i>Développement de l'intelligence artificielle</i>	X		
<i>Définition des variables et contraintes initiales</i>	X		
<i>Elaboration des contraintes de règles</i>	X		
<i>Automatisation de l'ajout des contraintes en fonction des indices</i>	X		

### 5.3. Itération 3

Tâches	Mathieu	Corentin	Enzo
Affichage d'un pop-up expliquant l'énigme			X
Affichage des règles		X	X
Affichage des films (dans le terminal)			X
Gestion des pions et des notes du joueur		X	X
Création des zones de Drag and Drop		X	
Implémentation du placement automatique des pions au temps 1		X	
Réalisation de tests	X		X
Synchronisation des deux affichages de prise de notes (Carte et Tableau)			X
Contraintes des indices publics de l'IA de déduction Choco-Solver	X		
Ajout des variables du coupable	X		
Ajout de la contrainte du coupable	X		
Affichage du coupable	X		
Implémentation de l'IA de déduction heuristique	X		
Définition des domaines en booléen	X		
Définition des positions initiales	X		

## 5.4. Itération 4

Tâches	Mathieu	Corentin	Enzo
<i>Synchronisation des pions de nombre</i>			X
<i>Gestion des pions d'absences et d'hypothèses (données et affichage)</i>		X	
<i>Réalisation du film de la partie</i>			X
<i>Réalisation du film du joueur</i>			X
<i>Continuation de l'IA de déduction heuristique</i>		X	
<i>Mise à jour lors d'un nouvel indice</i>		X	
<i>IA Assistance Choco-Solver</i>	X		
<i>Correction des déductions du joueur</i>	X		

# 6. Nos réalisations

## 6.1. Réalisation de Corentin

### L'IA de déduction heuristique :

#### A) Objectif de l'IA

L'IA de déduction heuristique de *Kronologic* repose sur une approche basée sur des règles logiques et des réductions de domaine pour déterminer où se trouvaient les personnages à différents moments, et ainsi identifier le coupable, le lieu du crime et le moment exact du meurtre. Contrairement à une approche basée sur un solveur de contraintes, cette IA utilise des heuristiques spécifiques pour exclure progressivement les options impossibles.

#### B) Représentation des données

L'IA stocke les informations sous la forme d'un tableau de booléens à trois dimensions :

domainesPersonnages[temps][personnage][lieu] = true ou false

- *temps* : Le moment dans la partie (6 étapes de temps).
- *personnage* : L'un des 6 personnages en jeu.
- *lieu* : Un des 6 lieux possibles.

Chaque cellule de ce tableau représente la possibilité qu'un personnage soit présent dans un lieu donné à un moment donné.

Au départ, toutes les combinaisons sont considérées comme possibles, puis l'IA réduit progressivement les domaines en appliquant des règles précises.

#### C) Les différentes règles à appliquer

L'IA applique différentes règles afin de réduire les domaines à chaque fois qu'une nouvelle question est posée.

##### 1. Positions initiales

Lors du démarrage, les positions initiales connues des personnages sont enregistrées :

- On connaît la position de tous les personnages au temps 1, ce qui permet de réduire tous les domaines correspondants au temps 1.
- Les règles de déplacement sont appliquées : un personnage doit obligatoirement changer de lieu entre deux temps successifs.

Exemple : Si le détective est dans la Salle A au Temps 1, il ne peut pas y être au Temps 2 (il doit obligatoirement se déplacer).

##### 2. Règles de déplacement

Le jeu impose des règles strictes sur les déplacements des personnages :

- Obligation de mouvement : Comme dit précédemment, un personnage ne peut pas rester au même endroit entre deux temps successifs.
- Salles adjacentes : Un personnage ne peut se déplacer que vers une salle adjacente à celle où il se trouve.

L'IA met donc à false toutes les positions impossibles dans le tableau domainesPersonnages en fonction des connexions entre les salles et des positions connues des personnages.

Exemple : Si un personnage est dans le Grand Foyer au Temps 1, il peut seulement se rendre au GrandEscalier ou à la Salle au Temps 2 (car ce sont les salles adjacentes). L'IA met alors à false les autres lieux pour ce personnage au Temps 2.

### **3. Règles sur les nombres de passages et le nombre de personnes dans une salle**

Lorsqu'un indice révèle le nombre de fois où un personnage est passé dans un lieu ou le nombre total de personnages présents dans un lieu à un moment donné, l'IA ajuste son modèle.

Elle met à jour les temps de passage obligatoires et supprime les lieux non compatibles pour les autres moments. Si un nombre exact de passages ou de présences est connu, l'IA s'assure que seuls ces événements restent possibles dans les domaines restants.

Lorsque toutes les présences dans un lieu sont connues à un moment donné, l'IA peut en déduire que les autres personnages n'y étaient pas et restreindre ainsi leurs domaines de déplacement.

#### **D) Pourquoi j'en suis fier ?**

J'ai rejoint la partie IA du projet alors qu'une architecture était déjà en place, avec une structure bien définie et un modèle utilisant Choco-Solver pour la résolution par contraintes. Mon défi a été d'implémenter une IA de déduction en utilisant une approche heuristique, qui fonctionne sans solveur externe.

Ce qui rend cette réalisation particulièrement satisfaisante :

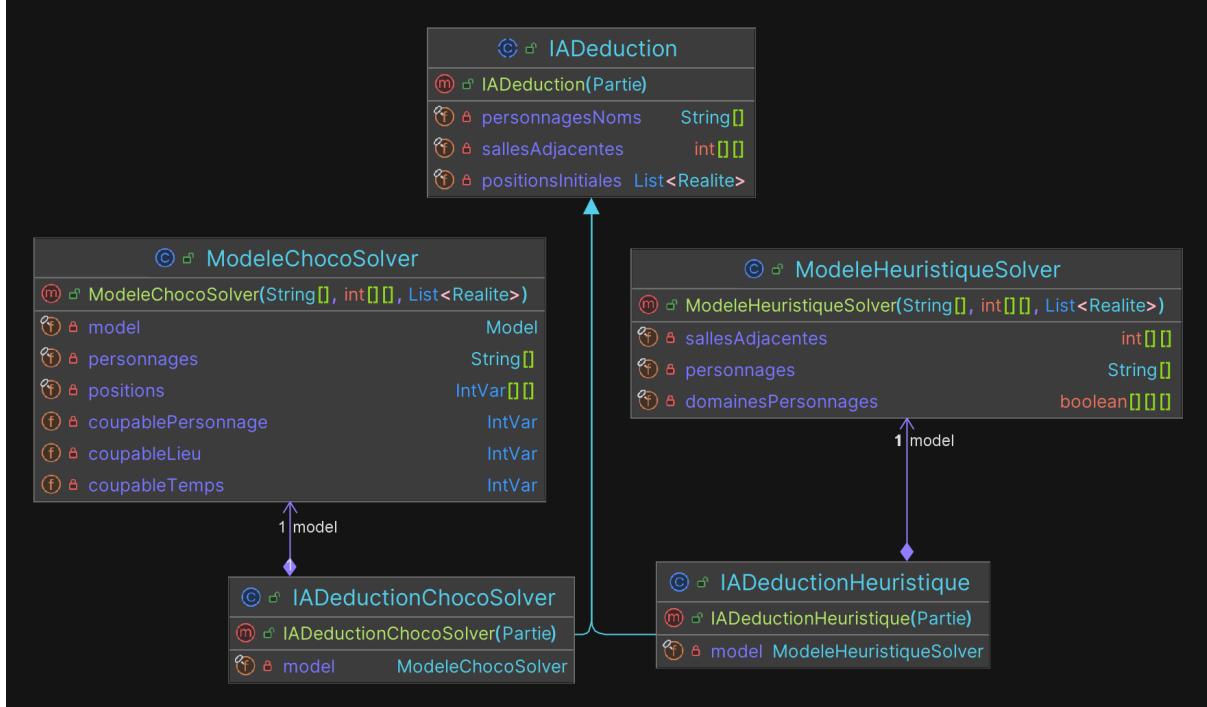
- J'ai dû comprendre et intégrer le fonctionnement des déplacements et des interactions dans le jeu.
- J'ai optimisé la gestion des contraintes pour exclure les solutions impossibles.

Il ne reste plus qu'à implémenter la recherche du coupable afin de la finaliser. Il sera aussi intéressant de comparer cette IA avec celle utilisant Choco-Solver afin de voir laquelle est plus optimisée.

## 6.2. Réalisation de Mathieu

### L'IA de Déduction Choco-Solver :

Cette IA a pour but de raisonner comme un joueur en analysant les indices fournis et en appliquant de la programmation par contraintes pour réduire progressivement l'ensemble des possibilités et identifier le coupable, à l'aide des règles du jeu.



### 1. Pourquoi Choco-Solver ?

Choco-Solver est une bibliothèque de programmation par contraintes, utilisée pour résoudre des problèmes complexes en modélisant les variables, leurs domaines de valeurs possibles, et les contraintes qui restreignent leurs relations.

Dans notre cas, nous l'avons choisi car il permet de :

- Gérer efficacement les contraintes du jeu, comme les déplacements autorisés, la présence des personnages et les indices obtenus.
- Réduire l'espace de recherche en éliminant rapidement les configurations impossibles.

### 2. Définition des contraintes

Pour permettre à l'IA de raisonner efficacement, nous avons défini plusieurs types de contraintes basées sur les règles du jeu et les indices obtenus. Ces contraintes permettent de réduire progressivement l'espace des possibilités.

### a. Variables utilisés

**Position des personnages** : Chaque personnage a une position définie à chaque instant du jeu, correspondant à l'une des salles.

**Le Personnage Coupable** : Le personnage qui a commis le crime, excluant le détective (D).

**Le lieu du meurtre** : La salle où le crime a eu lieu, parmi les six salles possibles.

**Le temps du meurtre** : Le moment où le crime a eu lieu, à partir du deuxième tour (le crime ne peut pas survenir au premier tour).

### b. Contraintes du jeu

**Déplacement obligatoire** : Chaque personnage doit se déplacer tous les tours.

**Déplacement dans des salles adjacentes** : Un personnage peut uniquement se déplacer dans une salle adjacente.

**Contraintes de début de partie** : Au début de la partie, la position initiale de chaque personnage est annoncée. Une contrainte fixe est appliquée pour chaque personnage.

**Contraintes au cours de la partie** : Les contraintes apparaissent au fur et à mesure de la partie et dépendent des indices fournis :

- **Indice Public (Temps choisi)** : Le joueur sait combien de personnages étaient présents dans une salle spécifique à un moment donné, sans connaître leur identité.
- **Indice Privé (Temps choisi)** : Le joueur apprend l'identité d'un personnage qui se trouvait dans une salle précise à ce moment-là. Il peut aussi obtenir la possibilité de poser une autre question.
- **Indice Public (Personnage choisi)** : Le joueur sait combien de fois un personnage est passé par un lieu donné au cours de la partie, sans savoir exactement quand.
- **Indice Privé (Personnage choisi)** : Le joueur apprend à quel moment précis un personnage était dans un lieu donné. Il peut aussi recevoir la possibilité de poser une autre question.

**Contraintes coupable** : Le coupable était seul avec le détective au moment du meurtre. C'est-à-dire que si deux personnages (dont le détective) étaient ensemble dans une pièce à un instant donné, le coupable est identifié comme étant l'autre personne.

### c. Réduction de domaine

Nous avons utilisé une propagation de domaine plutôt qu'une résolution directe des solutions afin de seulement avoir les domaines (plus utile pour le joueur).

### 3. Fonctionnement de l'IA avec Choco-Solver

L'IA évolue à chaque indice révélé et réduit progressivement l'ensemble des possibilités afin d'identifier le coupable.

#### 1) Initialisation :

- Chargement de tous les personnages, les salles, les temps et les contraintes de déplacement.
- Assignation des positions initiales connues.

#### 2) Ajout des indices révélés pendant la partie :

- Contraintes sur les lieux, les temps et les nombres de passages des personnages.
- Modification des domaines des variables en fonction des indices publics et privés.

#### 3) Propagation des contraintes :

- Mise à jour des possibilités en supprimant les solutions impossibles.

#### 4) Déduction finale :

- Vérification des variables coupables : si elles sont instanciées à une unique valeur, alors le crime est résolu.

The screenshot shows a window titled "Déductions de l'IA". The main area displays the deduction history:

```
===== 🔎 Impossible d'identifier un coupable pour le moment =====
⚠️ Continuez à poser des questions pour réduire les suspects !
===== 📊 Historique des Déductions =====
🕵️ B :
- Temps 1 : {1}
- Temps 2 : {2, 3}
- Temps 3 : {1, 2, 3, 4}
- Temps 4 : {1, 2, 3, 4, 5, 6}
- Temps 5 : {1, 2, 3, 4, 5, 6}
- Temps 6 : {1, 2, 3, 4, 5, 6}

🕵️ D :
- Temps 1 : {2}
- Temps 2 : {1, 3}
```

At the bottom, there are two buttons: "IA ChocoSolver" and "IA Heuristique".

## 6.3. Réalisation d'Enzo

### L'affichage des films du Joueur et de la Réalité :

#### 1. Film du Joueur

##### a. Objectifs

Au cours des premières itérations et à force de jouer au jeu, nous nous sommes aperçus que savoir quelles notes le joueur à placer à quel moment est très intéressant pour le joueur. Cela lui permet de s'apercevoir d'erreurs et de les corriger. En suivant les conseils de nos tuteurs, nous avons décidé de l'afficher à l'aide d'un nouveau bouton.

##### b. Planification

Pour réaliser cela, nous avons choisi une représentation qui nous semble la plus compréhensible possible. Ainsi, le joueur a, comme pour la Vue Carte, ses six cartes de jeu et un slider au-dessus qui lui permet de les actualiser. Le slider représente ce qu'on a appelé les tours du jeu : à chaque question, le joueur avance d'un tour dans les tours de jeu. Ainsi, le slider adapte sa taille au nombre de tour joué. Les notes ont aussi été ajoutées à une nouvelle variable qui nous sert d'historique. Cela nous permet d'observer l'ajout, la suppression et la modification des notes du joueur à chaque tour.

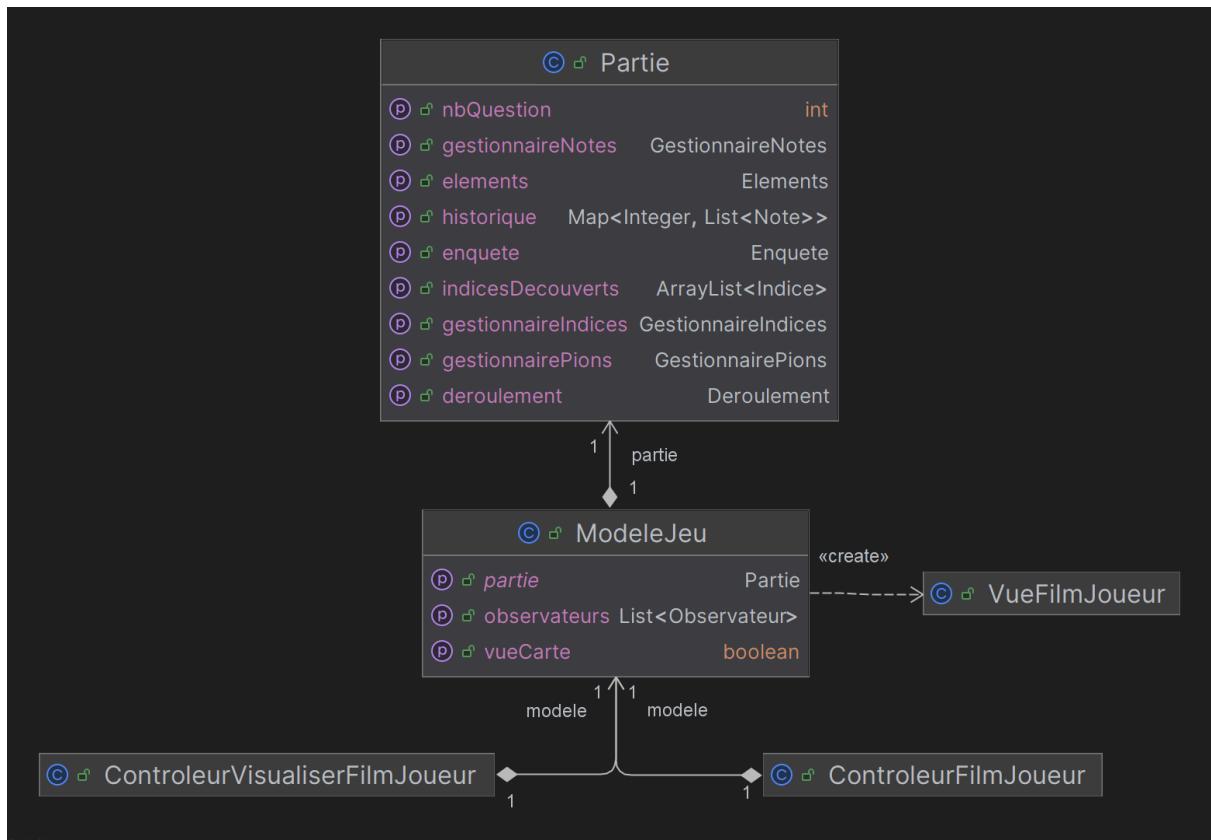


Diagramme de classe des classes et des fonctionnalités ajoutées

### c. Réalisation

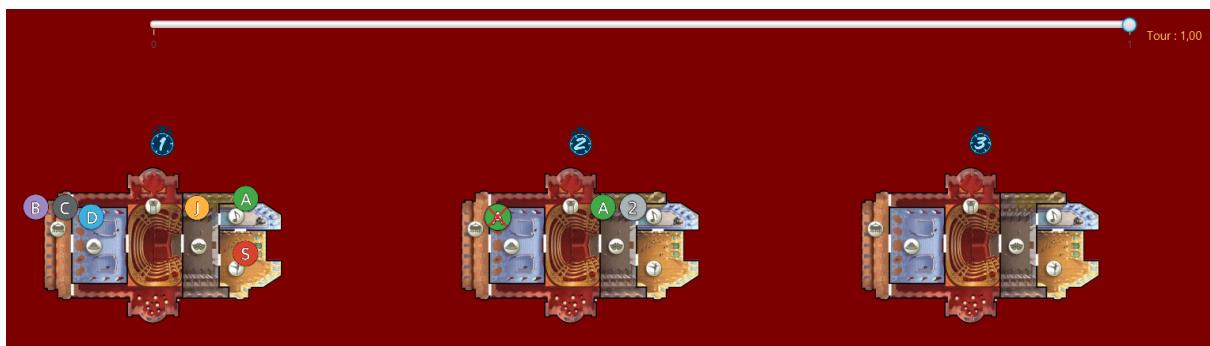
Pour illustrer cela, nous allons prendre un exemple (on sait que les notes du temps 1 sont toujours là puisque ce sont des informations données par le jeu) :

- ❖ Tour 0 : Le joueur décide de mettre 3 pions :
  - Un pion d'absence : L'aventurière au Temps 2 dans le Grand Escalier.
  - Deux pions d'hypothèse :
    - L'aventurière au Temps 2 dans la Scène et dans le Foyer de la danse.



Résultat observé au Tour 0

- ❖ Tour 1 : Le joueur a décidé de poser la question suivante : “*Combien de personnage(s) étais(en)t présent(s) au Temps 2 dans La Scène ?*”. Il a obtenu comme réponse que deux personnages sont présents et que l'un d'entre eux est bien l'Aventurière. Ainsi, il met à jour ses pions :
  - Il supprime le pion d'hypothèse de l'Aventurière au Temps 2 dans la Scène par un pion de présence.
  - Il supprime le pion d'hypothèse de l'Aventurière au Temps 2 dans le Foyer de la Danse.
  - Il ajoute aussi un pion de présence de valeur 2 au Temps 2 dans la Scène.



Résultat observé au Tour 1 (on observe la suppression des pions et leur modification)

## 2. Film de la Réalité

### a. Objectifs

En parallèle de la décision pour le film du joueur, nous avons aussi choisi de représenter de manière similaire le film de la partie, dit réalité. Le but de cette représentation est de permettre au joueur de comparer ses déductions avec le film réel de la partie.

### b. Planification

Pour réaliser cela, nous avons choisi une représentation qui nous semble la plus compréhensible possible. Ainsi, le joueur a accès à une seule des six cartes. C'est avec un slider au-dessus que la carte s'actualise. Le slider représente les temps de l'énigme, donc six temps au total. Le scénario de l'énigme est stocké dans une classe *Deroulement* et en utilisant cette dernière nous pouvons afficher le film exact de l'énigme.

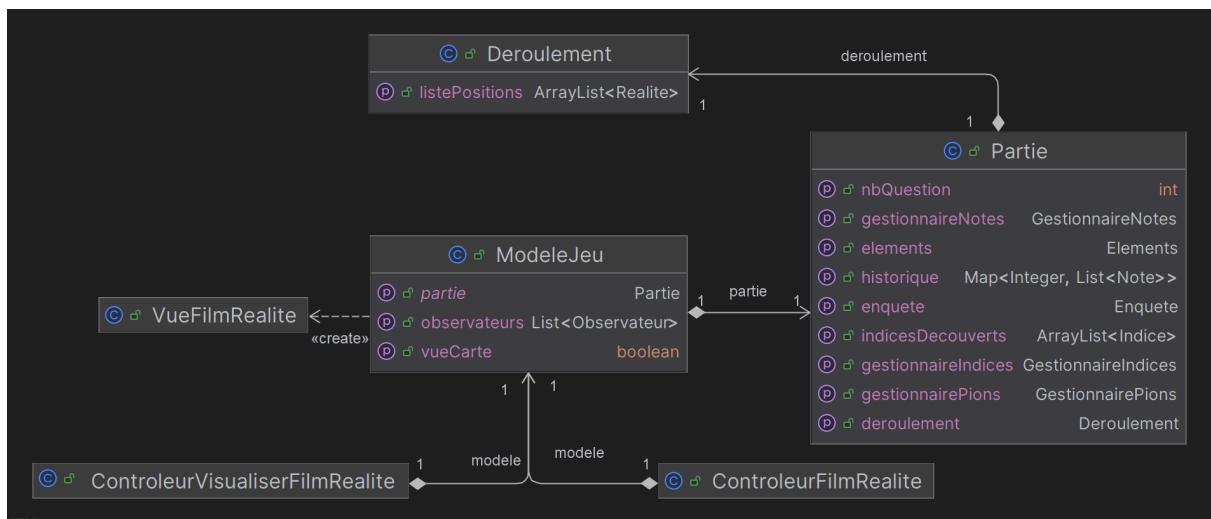
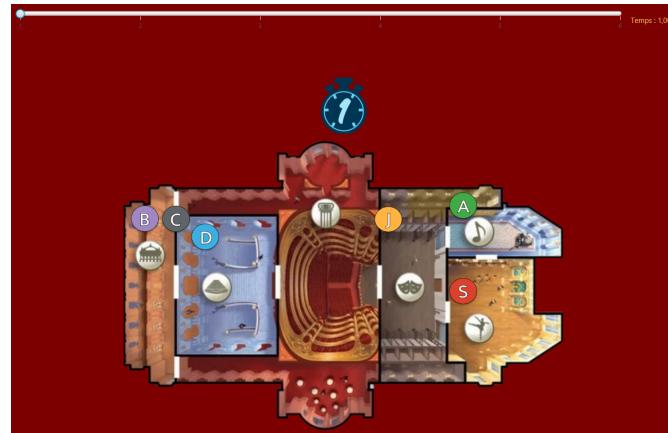


Diagramme de classe des classes et des fonctionnalités ajoutées

### c. Réalisation

Pour illustrer cela, je vais vous montrer différents états de la partie :

- ❖ Le premier exemple représente le premier temps que nous avons déjà vu auparavant.



*Résultat observé au Temps 1*

- ❖ Le deuxième exemple représente le temps suivant.



*Résultat observé au Temps 2*

- ❖ Le troisième exemple représente le dernier temps avec la réponse visuel à l'énigme.



*Résultat observé au Temps 6*

## 7. Objectifs de fin de projet

Nous sommes pour l'instant dans les temps car nous avons réussi à développer toutes les fonctionnalités voulues dans les quatre premières itérations. Durant les trois itérations restantes, nous allons donc nous concentrer sur les fonctionnalités importantes qu'il reste à développer ou à finaliser :

- ❖ **IA Déduction Heuristique** : Résoudre les derniers problèmes de déduction de cette IA.
- ❖ **IA Assistance Choco-Solver** : Permettre à l'IA de conseiller la meilleure question possible pour le joueur par rapport aux indices qu'ils possèdent.
- ❖ **IA Assistance Heuristique** : Développer une version de l'IA d'assistance basée sur une approche heuristique.
- ❖ **Comparaison Choco-Solver - Heuristique** : Comparer les performances entre l'IA utilisant Choco-Solver et l'IA heuristique pour identifier la solution la plus efficace.
- ❖ **IA Joueuse** : Implémenter une IA capable de jouer seule et de résoudre les enquêtes sans intervention humaine.
- ❖ **Générateur de scénario** : Développer un système permettant de générer automatiquement des scénarios de jeu variés.

## 8. Planning des trois dernières itérations

### 8.1. Itération 5

Pour cette itération, nous souhaitons :

- ❖ **Terminer** l'IA de Déduction Heuristique,
- ❖ **Comparer** l'efficacité des deux IA de Déduction,
- ❖ **Implémenter** l'IA d'Assistance Heuristique,
- ❖ **Finir** le développement de l'IA d'Assistance Choco-Solver,
- ❖ **Implémenter** l'IA Joueuse.

### 8.2. Itération 6

Pour cette itération, nous souhaitons :

- ❖ **Terminer** l'IA d'Assistance Heuristique,
- ❖ **Comparer** l'efficacité des deux IA d'Assistance,
- ❖ **Finir** l'IA Joueuse,
- ❖ **Implémenter** l'algorithme de création de scénario.

### 8.3. Itération 7

Pour cette itération, nous souhaitons :

- ❖ **Terminer** l'Algorithme de Création de Scénario,
- ❖ **Améliorer** l'interface graphique.

## 9. Conclusion

Les quatre premières itérations nous ont permis d'atteindre plusieurs objectifs majeurs, tout en réalisant l'ensemble des fonctionnalités prévues. En effet, la structure logicielle a été mise en place, intégrant une interface fonctionnelle et un modèle de jeu opérationnel. Nous avons également développé et testé plusieurs IA, notamment l'IA de Déduction Choco-Solver, qui repose sur la programmation par contraintes, ainsi que l'IA de Déduction heuristique, qui applique des règles logiques pour réduire progressivement l'espace des solutions.

Toutefois, plusieurs éléments restent encore à finaliser pour atteindre les objectifs de fin de projet :

- **Terminer l'IA de déduction heuristique**, afin qu'elle soit capable de trouver le coupable, le lieu du meurtre et le lieu du meurtre en fonction des indices récoltés.
- **Continuer de développer les IA d'assistance**, qui proposeront au joueur les meilleures questions à poser en fonction des indices collectés.
- **Comparer les performances des deux approches d'IA (Choco-Solver vs heuristique)** pour identifier la solution la plus efficace en termes de rapidité et de fiabilité.
- **Développer une IA joueuse autonome**, capable de résoudre une enquête en complète autonomie.
- **Implémenter un générateur de scénarios**, permettant de produire des scénarios d'enquête, tout en s'assurant qu'ils soient conformes.