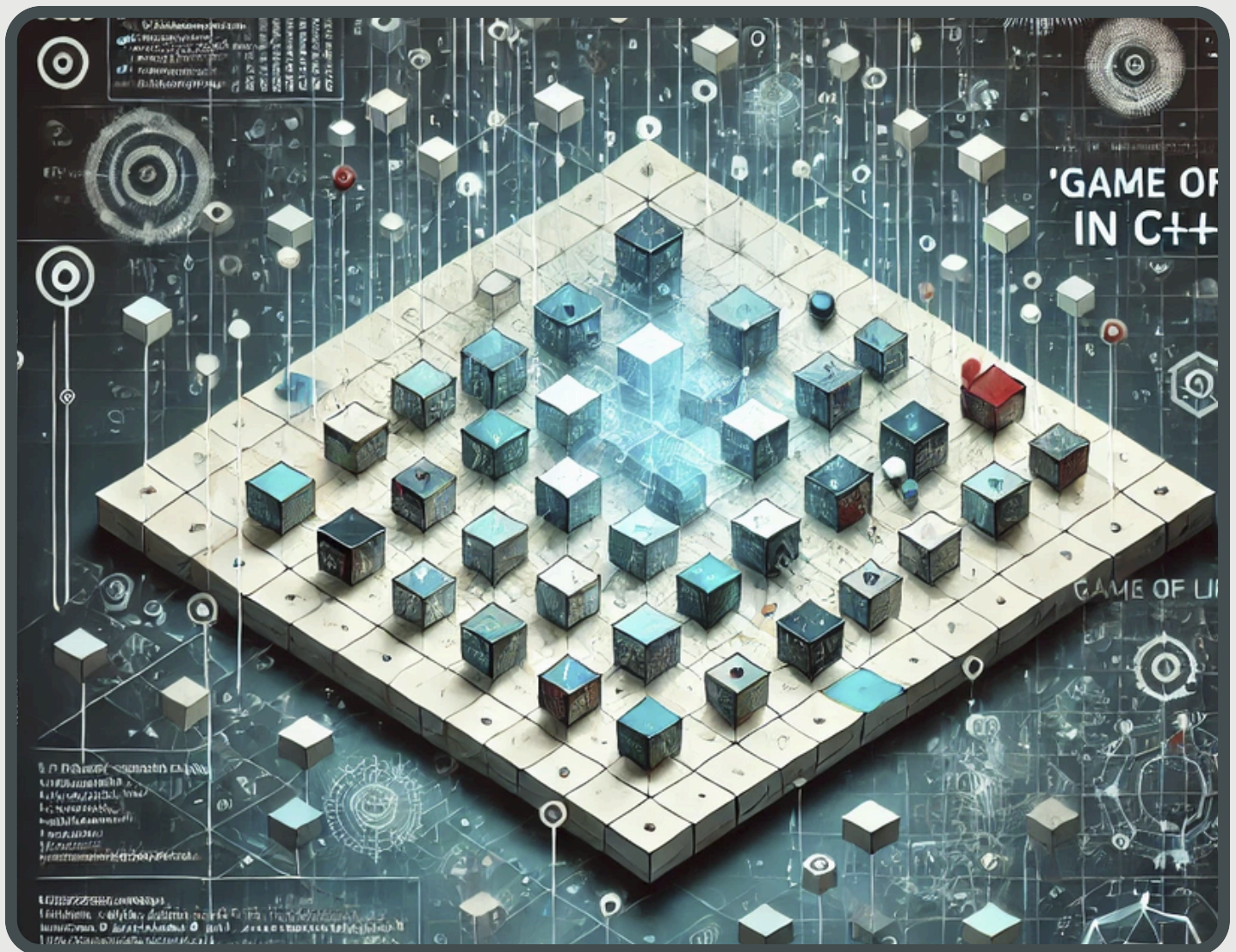


Livrable 2 : Diagramme POO

Bastien / Valentin



Sommaire

01

Introduction

02

Diagramme de cas d'utilisation

03

Diagramme de classe

04

Diagramme de séquence

05

Diagramme d'activité

06

Conclusion

Introduction

Dans le cadre de ce projet, nous avons conçu et développé une application mettant en œuvre le "Jeu de la vie", un automate imaginé par le mathématicien John Conway. Ce projet, réalisé en C++ avec l'appui de la bibliothèque graphique SFML, repose sur les principes de la programmation orientée objet. L'objectif est de simuler l'évolution d'une population de cellules dans une grille bidimensionnelle, en appliquant des règles précises pour chaque itération.

Notre approche s'articule autour de plusieurs axes : une phase de conception incluant des diagrammes UML (cas d'utilisation, classes, activités, séquences), un développement orienté objet, et une double modalité pour l'exécution. En mode console, l'application génère les états successifs des cellules sous forme de fichiers exploitables pour des analyses ultérieures. En mode graphique, l'utilisateur bénéficie d'une interface visuelle dynamique permettant de suivre en temps réel l'évolution de la grille.

Nous avons porté une attention particulière à la qualité du code, à la robustesse de l'application et à la conformité avec les spécifications fonctionnelles et techniques.

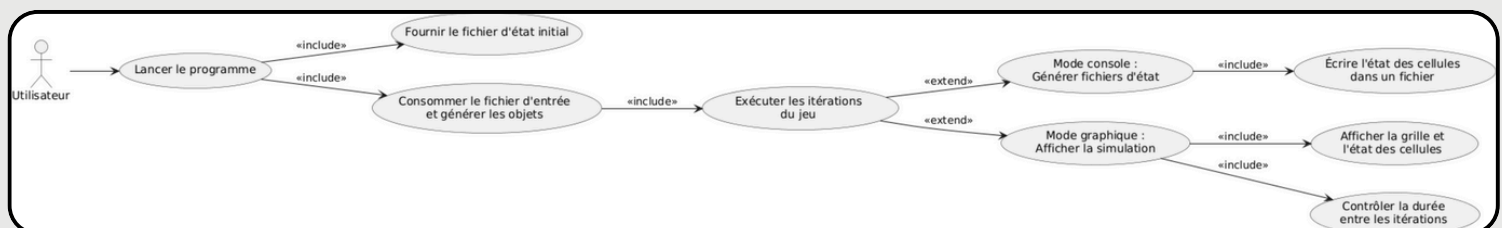
02

Diagramme de cas d'utilisation

Ce diagramme montre comment l'utilisateur interagit avec notre programme. Tout commence par lancer le programme, où l'utilisateur fournit un fichier d'état initial. Ensuite, le programme génère les objets à partir de ce fichier pour créer la grille.

On a prévu deux modes :

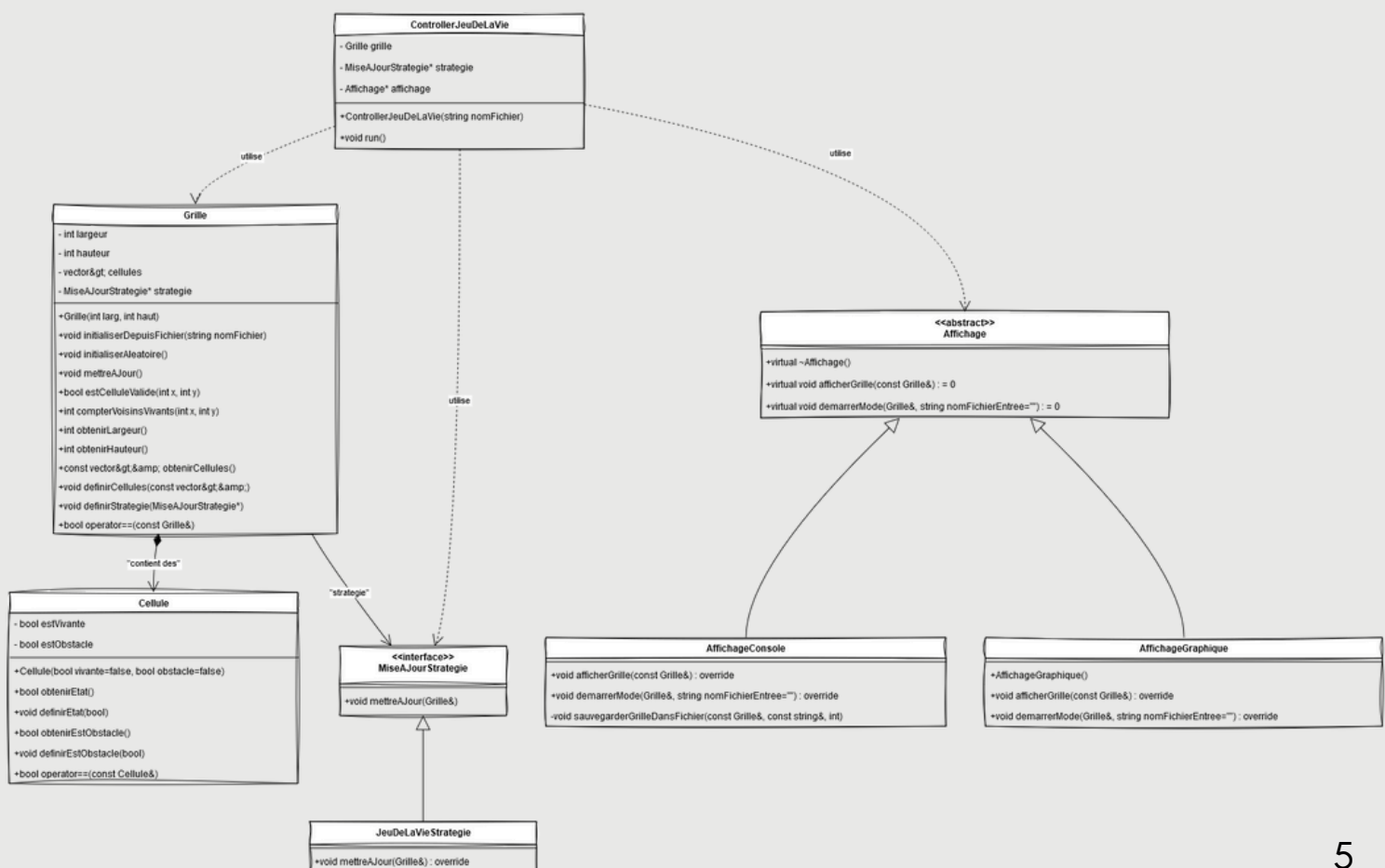
- **Mode console** : Génère un fichier avec l'état des cellules après chaque itération.
- **Mode graphique** : Affiche une simulation visuelle en temps réel.



03

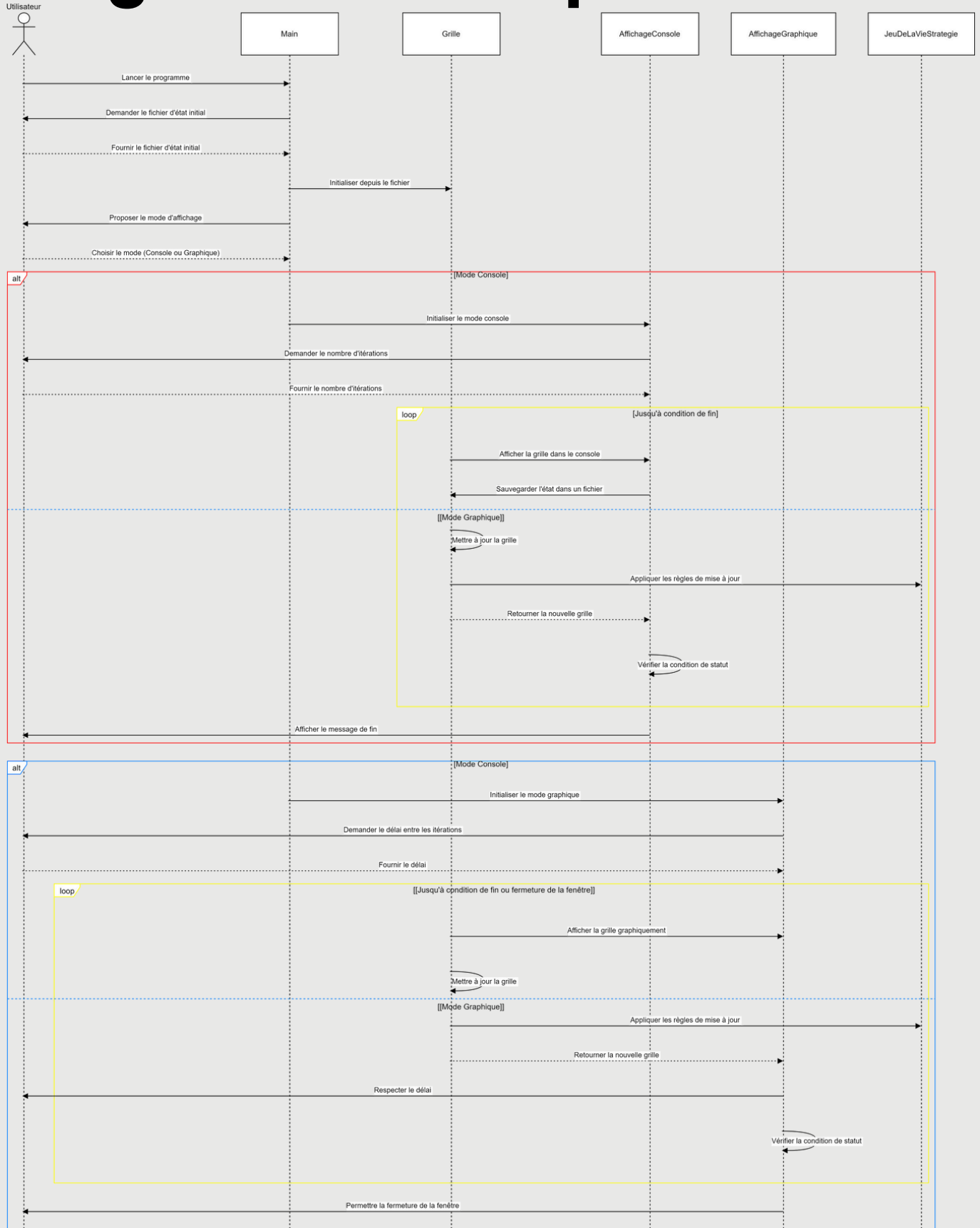
Diagramme de classe

1. Le diagramme suit une architecture MVC, où le Modèle gère les données et la logique (Grille, Cellule), la Vue affiche les données (console ou graphique), et le Contrôleur coordonne les interactions.
2. La stratégie de mise à jour (MiseAJourStrategie) est définie comme une interface, avec une implémentation concrète (JeuDeLaVieStrategie), permettant d'ajouter facilement de nouvelles règles au jeu.
3. La classe abstraite Affichage assure une flexibilité dans la vue, permettant de choisir entre un affichage en console ou graphique.



04

Diagramme de séquence




Ce diagramme décrit les étapes d'exécution de notre programme selon les deux modes disponibles :

1. **Mode Console :**

- L'utilisateur initialise le mode console.
- La grille est affichée dans le terminal et enregistrée dans un fichier à chaque itération.
- Les règles d'évolution sont appliquées pour mettre à jour la grille jusqu'à ce qu'une condition d'arrêt soit atteinte.
- Un message final est affiché.

2. **Mode Graphique :**

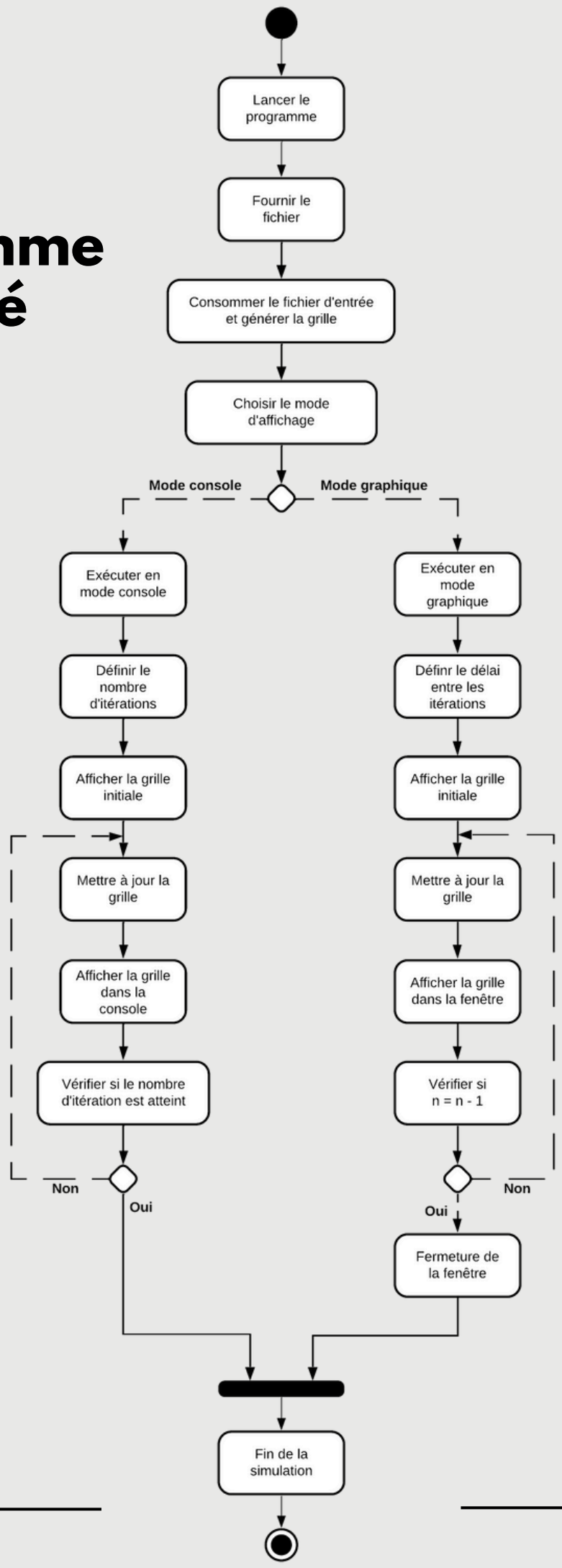
- L'utilisateur initialise le mode graphique.
- La durée entre les itérations est demandée, et une fenêtre graphique s'ouvre.
- La grille est affichée en temps réel, mise à jour à chaque itération selon les règles d'évolution.
- La simulation continue jusqu'à la fermeture de la fenêtre.



Ce diagramme d'activité montre comment nous avons structuré le fonctionnement de notre programme du Jeu de la Vie. On commence par fournir un fichier d'entrée décrivant l'état initial de la grille, qui est ensuite consommé pour générer les cellules et la grille. Nous proposons deux modes : en mode console, on affiche et sauvegarde les états successifs dans des fichiers, tandis qu'en mode graphique, on visualise l'évolution de la grille de manière interactive avec un contrôle du délai entre les itérations. La simulation s'arrête lorsque la grille devient stable, atteint un nombre d'itérations fixé ou, en mode graphique, si on ferme la fenêtre. Ce diagramme illustre clairement les étapes que nous avons définies pour répondre au sujet tout en mettant en avant une organisation modulaire et orientée objet.

05

Diagramme d'activité



06

Conclusion

En conclusion, ce livrable met en avant notre démarche pour concevoir et implémenter une simulation fidèle du Jeu de la Vie en C++. Nous avons structuré notre projet autour des principes de la programmation orientée objet, avec une architecture modulaire et claire, soutenue par des diagrammes UML (cas d'utilisation, classes, séquences et activité). L'application répond aux spécifications grâce à deux modes d'exécution : console pour une génération des états sous forme de fichiers exploitables, et graphique pour une simulation visuelle interactive. Nous avons veillé à la qualité du code, à la robustesse du programme et à la conformité avec les attentes, garantissant ainsi un résultat fonctionnel, cohérent et bien documenté.