

AIX-MARSEILLE UNIVERSITE
FACULTE DES SCIENCES - CENTRE DE TELE-ENSEIGNEMENT SCIENCE
LICENCE 3 MATHEMATIQUES INFORMATIQUE

**Simulation d'un feu de forêt à l'aide des automates
cellulaires**
Projet Mathematiques-Informatique

Gladys DJOUFACK TADONKA
Maeva DHAYNAUT
Amine HADDOU
Nassim amar ROUAG
Franck SOULON

Encadrant: **Omar BOUCELMA**
Année académique: **2022/2023**

Mai 2023

Table des matières

| | |
|--|-----------|
| Table des figures | I |
| Introduction | 1 |
| 1 Revue de la littérature | 2 |
| 2 Quelques concepts théoriques | 5 |
| 2.1 Automate cellulaire | 5 |
| 2.2 Diagramme de <i>Gantt</i> | 5 |
| 2.3 UML | 6 |
| 2.4 Git | 6 |
| 2.5 Java | 6 |
| 3 Cas d'étude : Simulation graphique d'un feu de forêt avec le langage Java | 8 |
| 3.1 Modélisation mathématique du problème | 8 |
| 3.1.1 Impact du voisinage | 9 |
| 3.1.2 Impact de la direction du vent | 10 |
| 3.1.3 Impact du climat | 10 |
| 3.1.4 Impact de l'humidité du sol | 11 |
| 3.1.5 Type de végétation | 11 |
| 3.2 Gestion de projet | 11 |
| 3.2.1 Diagramme de Gant | 11 |
| 3.2.2 Trello | 12 |
| 3.2.3 Git | 12 |
| 3.2.4 Méthodologie Agile | 13 |
| 3.3 Modélisation informatique du problème | 13 |
| 3.3.1 Construction et amélioration du diagramme de classes | 13 |
| 3.3.2 Description du diagramme de classes | 14 |
| 3.4 Design de l'interface graphique | 15 |
| 3.5 Mise en œuvre de la solution à l'aide du langage Java | 16 |
| 3.5.1 Algorithme de la boucle de simulation | 16 |
| 3.5.2 Algorithme de calcul de l'état futur d'une cellule | 16 |
| 3.5.3 Algorithme de calcul de la probabilité de propagation | 17 |
| Conclusion | 18 |
| Bibliographie | 19 |
| Annexes | A1 |
| A Design de l'interface graphique | A1 |
| B Interface graphique implementé | A2 |

| | |
|---|----|
| C Algorithme de la boucle de simulation | A3 |
| D Algorithme de calcul de l'état futur d'une cellule | A4 |
| E Algorithme de calcul de la probabilité de propagation | A5 |

Table des figures

| | |
|--|----|
| Figure 3.1 Voisinage de Von Neumann | 9 |
| Figure 3.2 Diagramme de Gantt du projet | 12 |
| Figure 3.3 Stratégie des branches et historique des commit | 13 |
| Figure 3.4 Version finale du diagramme de classes | 14 |
| Figure A.1 Design Figma de l'interface graphique | A1 |
| Figure B.1 Interface graphique réellement implémenté | A2 |
| Figure C.1 Code de la méthode <code>Grille.simulation()</code> | A3 |
| Figure D.1 Code de la méthode <code>Arbre.calculeEtatFutur()</code> | A4 |
| Figure E.1 Code de la méthode <code>Arbre.calculeP()</code> | A5 |

Liste des tableaux

| | |
|--|---|
| Tableau 3.1 Types de cellules | 8 |
|--|---|

Introduction

L'augmentation régulière de la surface des forêts dévastées par les feux ces dernières années met en lumière l'urgence d'apporter des solutions pour mieux prévenir et gérer ces catastrophes naturelles. En France, l'année 2022 par exemple a été marquée par un record de surface végétative brûlée, avec plus de 62 000 hectares de forêt détruits lors de vagues de chaleur estivales. Face à l'urgence de ce phénomène qui ne cesse de s'aggraver chaque année, il est primordial de disposer d'outils permettant de mieux comprendre et maîtriser la propagation des feux de forêt.

Dans ce contexte, la simulation de la propagation des incendies de forêt peut offrir une solution prometteuse pour mieux comprendre et anticiper l'expansion de ces feux. Ce projet a pour objectif de proposer une interface graphique en langage Java pour simuler la propagation d'un feu de forêt en fonction des règles définies dans un automate cellulaire. L'interface permettra à l'utilisateur de configurer la parcelle de forêt et les positions initiales des arbres en feu, puis de visualiser l'évolution de l'incendie selon les règles de transition de l'automate cellulaire.

Ce rapport détaillera l'ensemble des étapes nécessaires à la réalisation de cette simulation, en passant en revue les concepts théoriques des automates cellulaires, le voisinage de Von Neumann, la modélisation mathématique et informatique du problème, ainsi que la gestion de projet et la conception de l'interface graphique. Nous explorerons également l'impact de différents facteurs sur la propagation du feu, tels que la direction du vent, l'humidité du sol, le climat et le type de végétation. Enfin, nous discuterons des différentes améliorations possibles de notre simulation, en explorant d'autres facteurs qui influencent le feu de forêt et en proposant des pistes pour intégrer des technologies plus avancées, telles que l'intelligence artificielle.

1 Revue de la littérature

La revue de la littérature effectuée dans le cadre de ce rapport a permis de recenser plusieurs études portant sur la modélisation de la propagation des incendies de forêt à l'aide des automates cellulaires. Parmi ces études, nous avons identifié neuf articles scientifiques qui ont retenu notre attention. Ces études présentent différentes approches et méthodologies pour la modélisation de la propagation des incendies de forêt, en utilisant les automates cellulaires comme outil de simulation. Les articles identifiés incluent : Hernandez Encinas et al. (2006) [1], Karafyllidis et Thanailakis (1996) [2], Alexandridis et al. (2008) [3], Berjak et Hearne (2001) [4], Drossel et Schwabl (1992) [5], Ghisu et al. (2015) [6], Mutthulakshmia et al. (2020) [7], Freire et DaCamara (2019) [8] et Hernandez Encinas et al. (2006) [9].

L'étude de Hernandez Encinas et al. (2006) [1] présente un modèle de propagation des incendies forestiers utilisant des automates cellulaires hexagonaux. Les auteurs ont intégré des paramètres tels que la densité des arbres, la vitesse et la direction du vent, ainsi que la présence de routes et de cours d'eau pour simuler la propagation des incendies. Les résultats montrent une corrélation significative entre les conditions environnementales et la vitesse de propagation de l'incendie.

L'étude de Karafyllidis et Thanailakis (1996) [2] a proposé un modèle de prédiction de la propagation des incendies forestiers en utilisant des automates cellulaires à deux dimensions. Le modèle a été développé en tenant compte des caractéristiques de la végétation, des conditions météorologiques et topographiques, ainsi que des comportements humains. Les résultats ont montré une corrélation significative entre la vitesse de propagation des incendies et la densité de la végétation. Cette étude a utilisé un voisinage de Moore pour la propagation de l'incendie.

L'étude de Alexandridis et al. (2008) [3] a présenté un modèle de prédiction de la propagation des incendies forestiers à l'aide d'automates cellulaires à deux dimensions. Le modèle a été développé pour simuler l'incendie qui a balayé l'île de Spetses en 1990. Les auteurs ont inclus des paramètres tels que la densité de la végétation, la direction et la vitesse du vent, ainsi que l'influence des facteurs topographiques. Les résultats ont montré une corrélation significative entre la vitesse de propagation de l'incendie et la densité de la végétation ainsi que la direction et la vitesse du vent.

L'étude de Berjak et Hearne (2001) [4] a présenté un modèle d'automate cellulaire amélioré pour simuler les incendies dans un système de savane spatialement hétérogène. Les auteurs ont pris en compte les effets des paramètres tels que la densité de la végétation, la direction et la vitesse du vent, la densité du combustible et la topographie. Les résultats ont montré une corrélation significative entre la vitesse de propagation de l'incendie et la densité de la végétation ainsi que les conditions météorologiques. Cette étude a utilisé un voisinage de von Neumann pour la propagation de l'incendie.

L'article de Drossel et Schwabl (1992) [5] introduit un modèle de propagation d'in-

cendie basé sur l'idée de la critique auto-organisée (self-organized criticality). Ce modèle suppose que les conditions de départ, comme la densité des arbres ou la présence d'herbes sèches, sont distribuées de manière aléatoire, ce qui conduit à une propagation d'incendie auto-organisée, c'est-à-dire que la propagation suit une loi de puissance plutôt que d'être régulière. Les auteurs ont également examiné la distribution spatiale de la taille des incendies, qui suit également une loi de puissance. Bien que ce modèle soit assez différent de celui que nous avons utilisé dans notre étude, il met en évidence l'importance des facteurs aléatoires et de la complexité des interactions dans la propagation des incendies.

Ghisu et al. (2015) [6] ont proposé quant à eux un algorithme optimisé pour la simulation de la propagation des incendies de forêt en utilisant un automate cellulaire testé sur un ensemble de données expérimentales recueillies sur le terrain. Les résultats ont montré que la simulation basée sur l'algorithme proposé est capable de reproduire la propagation réelle du feu avec une bonne précision. Les auteurs ont également comparé leur méthode avec d'autres approches d'automates cellulaires existantes et ont montré que leur algorithme est plus efficace et plus précis. Ce travail met en avant l'importance de l'optimisation des algorithmes pour la simulation de la propagation des incendies de forêt, ce qui peut être utile pour la gestion et la prévention des incendies de forêt. De plus, la méthodologie utilisée dans cet article peut être applicable à d'autres domaines qui utilisent des automates cellulaires pour simuler des phénomènes naturels.

L'étude de Mutthulakshmia et al. (2020) [7] propose une simulation de la propagation des feux de forêt à l'aide d'automates cellulaires basé sur des règles qui décrivent l'ignition, la propagation et l'extinction des feux. Ils ont également incorporé des règles pour la lutte contre les incendies, telles que l'utilisation de coupe-feu, d'eau et de produits chimiques. La simulation a été réalisée sur une zone de 1000x1000 cellules et les résultats ont été validés à l'aide de données satellitaires réelles. Les résultats ont montré que le modèle proposé est capable de reproduire avec précision les schémas de propagation du feu observés dans la réalité. Cette étude est intéressante car elle montre l'importance de prendre en compte des facteurs externes dans la modélisation des feux de forêt, tout en mettant en avant les avantages des automates cellulaires pour ce type de simulation.

Enfin, Les études menées respectivement par Freire et DaCamara (2019) [8], et par Hernandez Encinas et al. (2006) [9] utilisent elles aussi les automates cellulaires pour simuler la propagation des incendies de forêt. La première utilise une méthode de modélisation basée sur la propagation de la chaleur et sur la détermination des zones critiques. Les auteurs ont testé leur modèle en le comparant à des données réelles de feux de forêt et ont constaté que le modèle avait une précision satisfaisante. La deuxième étude quant à elle, utilise une approche similaire à celle de Hernandez Encinas et al. (2006) [1] en utilisant un voisinage de Von Neumann pour déterminer la probabilité de propagation de l'incendie de forêt. Les auteurs ont également proposé un algorithme optimisé pour améliorer la précision de la simulation.

Ces études ont toutes pour point commun l'utilisation des automates cellulaires pour modéliser la propagation des incendies de forêt. Elles diffèrent toutefois dans les

choix qu'elles font quant aux règles de propagation du feu. Certaines d'entre elles, telles que l'étude de Drossel et Schwabl (1992) [5], utilisent des règles relativement simples pour modéliser la propagation du feu, tandis que d'autres, comme l'étude de Ghisu et al. (2015) [6], ont recours à des règles plus complexes pour prendre en compte différents facteurs environnementaux.

Notre choix de nous concentrer sur le voisinage de Von Neumann et la prise en compte de facteurs externes dans le calcul de la probabilité de propagation est directement lié aux résultats et approches présentés dans ces études. En effet, plusieurs de ces études ont utilisé des approches similaires pour modéliser la propagation des incendies de forêt, en mettant l'accent sur l'influence des facteurs environnementaux et de la topographie dans la propagation du feu. Nous avons donc cherché à nous appuyer sur ces résultats pour réaliser une implémentation graphique d'un modèle simplifié de propagation des incendies de forêt à l'aide des automates cellulaires.

2 Quelques concepts théoriques

Dans cette partie, nous aborderons quelques concepts théoriques nécessaires à la compréhension de notre cas d'étude de simulation graphique d'un feu de forêt avec le langage Java. Nous commencerons par explorer le concept d'automate cellulaire, qui est une méthode de modélisation mathématique utilisée pour simuler des systèmes complexes tels que les feux de forêt. Ensuite, nous parlerons du diagramme de *Gantt*, un outil de gestion de projet qui nous permettra de planifier les différentes tâches à réaliser pour mener à bien notre projet. Nous aborderons également le langage de modélisation unifiée (UML) qui sera utilisé pour établir une modélisation informatique de notre projet. Nous verrons également l'importance de *Git*, un système de contrôle de version qui facilite le travail collaboratif sur un projet. Enfin, nous étudierons le langage de programmation *Java*, qui sera utilisé pour mettre en œuvre la simulation graphique du feu de forêt.

2.1 Automate cellulaire

Les automates cellulaires sont des modèles mathématiques permettant de simuler des phénomènes qui se déroulent dans des espaces discrets.[10] Ils sont largement utilisés dans la modélisation de systèmes complexes tels que la météorologie, l'écologie, la biologie et la physique. Dans notre cas d'étude, nous utiliserons les automates cellulaires pour simuler l'évolution d'un feu de forêt en fonction de différents paramètres tels que le vent, l'humidité, le type de végétation et le climat.

Le voisinage de *Von Neumann* est l'une des méthodes les plus courantes pour la modélisation de l'environnement à travers les automates cellulaires.[11] Cette méthode suppose que chaque cellule est en contact avec ses quatre voisins immédiats : en haut, en bas, à gauche et à droite. La règle de transition est alors appliquée à chaque cellule en fonction de l'état de ses voisins. Pour notre simulation de feu de forêt, le voisinage de *Von Neumann* nous permettra de prendre en compte l'influence des cellules environnantes sur le comportement du feu.

L'utilisation des automates cellulaires et du voisinage de *Von Neumann* nous permettra de simuler de manière réaliste l'évolution d'un feu de forêt et d'analyser l'impact de différents facteurs tels que la direction du vent, l'humidité du sol, le type de végétation et le climat sur la propagation du feu.

2.2 Diagramme de *Gantt*

Le diagramme de Gantt est un outil de gestion de projet qui permet de planifier les différentes étapes d'un projet et de suivre l'avancement des tâches. Il a été inventé au début du 20ème siècle par Henry Gantt pour planifier les activités de production et a depuis été adapté pour être utilisé dans une variété de projets et d'industries.

Dans notre projet de simulation de feu de forêt, nous avons utilisé le diagramme de Gantt pour identifier les tâches à effectuer, estimer leur durée et les organiser dans une chronologie cohérente. Cela nous a permis de suivre l'avancement du projet, de respecter

les délais et de détecter les retards potentiels.

Dans l'ensemble, le diagramme de Gantt a été un outil très utile pour la gestion de notre projet de simulation de feu de forêt. Nous recommandons son utilisation pour toute personne impliquée dans la gestion de projets, qu'ils soient personnels ou professionnels.

2.3 UML

UML (Unified Modeling Language) est un langage visuel pour modéliser les systèmes logiciels. Il est utilisé dans l'industrie pour représenter graphiquement les modèles de systèmes logiciels.

Différents types de diagrammes UML peuvent être créés pour représenter différents aspects d'un système. Le diagramme de classes est un type de diagramme qui représente les classes et leurs relations dans un système. Les classes sont représentées par des rectangles contenant leur nom, attributs et méthodes, et les relations sont représentées par des flèches. Les flèches peuvent être unidirectionnelles ou bidirectionnelles.

Le diagramme de classes est un outil essentiel pour décrire la structure d'un système orienté objet. Ce qui permet aux membres de l'équipe de se représenter mentalement le fonctionnement général du programme et d'avoir une meilleure perception des résultats attendus.

2.4 Git

Git est un système de contrôle de version très populaire pour le développement de logiciels. Il permet à plusieurs développeurs de travailler ensemble sur un même projet tout en conservant un historique de toutes les modifications apportées au code.

Git utilise une approche décentralisée, dans laquelle chaque développeur possède une copie locale du code source et peut partager les modifications apportées via un référentiel Git centralisé. Git permet de fusionner facilement les modifications apportées par différents développeurs, ce qui facilite la collaboration.

Nous avons utilisé Git pour notre projet de simulation de feu de forêt afin de gérer le code source et faciliter la collaboration entre les membres de l'équipe. Nous avons créé un référentiel Git centralisé et utilisé les commandes Git de base pour gérer les modifications apportées au code source.

En utilisant Git, nous avons pu conserver un historique de toutes les modifications apportées au code source et travailler simultanément sur différentes fonctionnalités sans risquer de perdre des modifications importantes.

2.5 Java

Java est un langage de programmation orienté objet très populaire dans le développement d'applications pour diverses plates-formes, grâce à sa portabilité. Il est souvent utilisé pour

développer des applications de bureau, des applications Web et des jeux vidéo.

Java dispose également d'une grande communauté de développeurs et de nombreuses bibliothèques open source qui facilitent le développement d'applications.

Pour la simulation de feu de forêt, Java est un choix judicieux car il peut être utilisé pour créer une interface graphique conviviale pour afficher la simulation en temps réel. Le langage est également suffisamment puissant pour permettre une manipulation facile des automates cellulaires et des calculs mathématiques nécessaires pour la simulation.

3 Cas d'étude : Simulation graphique d'un feu de forêt avec le langage Java

Dans cette partie, nous allons étudier un cas pratique de simulation graphique d'un feu de forêt en utilisant le langage de programmation Java. La simulation est basée sur des automates cellulaires et prend en compte plusieurs paramètres tels que le voisinage de *Von Neumann*, la direction du vent, l'humidité du sol, le type de végétation, et les conditions climatiques. Nous allons explorer la modélisation mathématique du problème, la gestion de projet, la modélisation informatique, la conception de l'interface utilisateur et la mise en œuvre de la solution. Cette étude permettra de mieux comprendre l'application des automates cellulaires dans la simulation de phénomènes naturels et de développer des compétences en programmation Java.

3.1 Modélisation mathématique du problème

La modélisation mathématique est une étape importante pour comprendre les mécanismes de la propagation du feu dans une forêt et prédire son évolution. Nous avons utilisé une grille de cellules pour représenter la forêt, où chaque cellule peut avoir trois états : sol nu, arbre vivant et arbre en feu. Nous avons défini des règles de propagation qui prennent en compte divers facteurs tels que le *voisinage*, la *direction du vent*, la *saison*, l'*humidité du sol* et le *type de végétation*. Ces règles ont été implémentées dans un modèle mathématique qui nous permet de simuler la propagation du feu dans la forêt et d'étudier les effets de chaque facteur sur la propagation de l'incendie.

La modélisation de la forêt a été effectuée à l'aide d'une grille de taille $n \times n$ contenant n^2 cellules. Chaque cellule peut avoir quatre états possibles tels que décrit dans le tableau 3.1. Des règles de propagation ont été établies afin de déterminer l'état futur de chaque cellule en fonction de son état actuel et de l'état de ses voisins. Si la cellule est un arbre en feu, son état futur est "arbre en cendre". Si elle est un arbre vivant et qu'aucun de ses voisins n'est en feu, alors son état futur est le même que son état actuel. Si elle est un arbre vivant et au moins un de ses voisins est en feu, alors sa probabilité de prendre feu est calculée. La probabilité initiale est fixée à 0.6, ce qui correspond à une probabilité raisonnable pour un incendie de forêt.

| Type de cellule | Couleur de la cellule | Code |
|-----------------|-----------------------|------|
| Sol nu | Ivoire | 0 |
| Arbre vivant | Vert | 1 |
| Arbre en feu | Rouge | 2 |
| Arbre en cendre | Gris | 3 |

Table 3.1 – Types de cellules

Cependant, cette probabilité fixe peut ne pas être réaliste dans toutes les situations. C'est pourquoi des facteurs externes ont été intégrés dans le modèle afin d'améliorer sa

précision. Ces facteurs incluent le voisinage, la direction du vent, la saison, l'humidité du sol et le type de végétation.

3.1.1 Impact du voisinage

Dans notre modèle de simulation de la propagation d'un incendie forestier, le voisinage d'une cellule est un élément important qui peut avoir un impact significatif sur l'état de la cellule elle-même. Le voisinage de *Von Neumann* a été choisi pour notre simulation, car il permet de prendre en compte les cellules voisines situées dans les quatre directions cardinales (nord, sud, est, ouest). Le voisinage de *Von Neumann* est défini comme l'ensemble des cellules situées à une distance de *Manhattan* égale à 1 de la cellule en question. Autrement dit, si nous considérons une cellule donnée dans notre grille, son voisinage de *Von Neumann* est composé de ses quatre voisins directs situés au nord, au sud, à l'est et à l'ouest. La figure 3.1 présente de façon graphique ce voisinage.

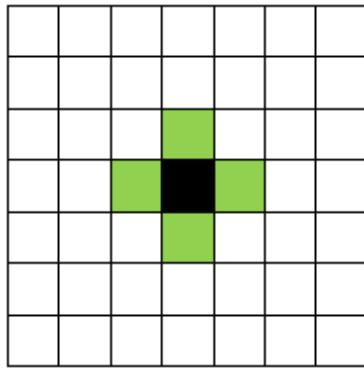


Figure 3.1 – Voisinage de Von Neumann

Lorsqu'une cellule vivante a au moins un voisin en feu, nous calculons la probabilité pour qu'elle aussi s'enflamme. Partant de la probabilité de base p initialement fixée à 0.6, la formule 1 est appliquée pour mettre à jour celle-ci de sorte qu'elle prenne en compte le nombre n de voisins en feu de la cellule.

$$p = p + (n \times 0.05) \quad (1)$$

Le voisinage de Von Neumann peut avoir un impact significatif sur la simulation de la propagation de l'incendie forestier, car si plusieurs cellules sont en feu dans le voisinage direct d'une cellule donnée, la probabilité pour qu'elle s'enflamme à son tour augmente considérablement. Cependant, si la cellule a des voisins qui ne sont pas en feu, cela peut diminuer la probabilité qu'elle aussi s'enflamme. En prenant en compte le voisinage de Von Neumann dans notre simulation, nous pouvons avoir une représentation plus réaliste de la propagation de l'incendie forestier. Cela nous permet également d'explorer les différents facteurs qui peuvent influencer cette propagation et d'ajuster notre modèle en conséquence.

3.1.2 Impact de la direction du vent

La direction du vent est un facteur important dans la propagation du feu en forêt. Dans notre modélisation, nous avons pris en compte la direction du vent pour déterminer la probabilité qu'une cellule vive prenne feu en fonction de ses voisins. Le vent peut être défini par rapport aux quatre points cardinaux et peut influencer la propagation du feu de différentes manières. Si le vent souffle dans la direction opposée à celle d'une cellule en feu, la probabilité de propagation diminue car le vent peut empêcher la propagation en poussant la fumée et les flammes dans la direction opposée. En revanche, si le vent souffle dans la même direction que celle d'une cellule en feu, la probabilité de propagation augmente car le vent peut alimenter le feu en oxygène.

Dans notre modélisation, nous avons utilisé une valeur binaire pour indiquer si le vent pousse le feu vers la cellule en question. Si la valeur est 1, la probabilité que la cellule prenne feu augmente de 0,5, tandis que si la valeur est 0, la probabilité diminue de 0,1. La formule 2 résume cet impact.

$$p = \begin{cases} p + 0.5 & \text{si } b = 1 \\ p - 0.1 & \text{si } b = 0 \end{cases} \quad (2)$$

L'utilisateur peut choisir *INDIFFERENT* comme direction du vent pour ignorer son impact. Dans ce cas, la probabilité de propagation dépendra uniquement de l'état des voisins de la cellule en question, sans considérer la direction du vent.

En somme, la direction du vent est un facteur important à prendre en compte pour modéliser la propagation du feu dans la forêt, et nous avons tenu compte de cet impact dans notre modèle en ajustant la probabilité de propagation en fonction de la direction du vent et de la position des cellules en feu, créant ainsi une simulation plus réaliste.

3.1.3 Impact du climat

Les saisons ont un impact important sur la propagation des feux de forêt, car les variations de température et d'humidité peuvent considérablement changer la probabilité de propagation d'un feu dans une forêt.

Dans notre modèle, nous avons donc pris en compte cet impact en modifiant la probabilité initiale pour chaque cellule. En hiver, la probabilité initiale de prendre feu a été réduite de 10% pour les arbres, car les températures sont basses et l'humidité est souvent plus élevée. En été, la probabilité initiale de prendre feu a été augmentée de 20% pour les arbres, car les températures sont élevées et l'humidité est généralement plus faible. Pour le printemps et l'automne, nous avons considéré que les conditions étaient relativement neutres et n'avaient donc pas d'impact significatif sur la probabilité de propagation d'un feu.

En incluant l'impact des saisons sur la probabilité initiale de prendre feu, notre modèle est plus réaliste et peut être utilisé pour explorer différents scénarios pour les

feux de forêt dans différentes saisons, ce qui est utile pour la planification des mesures de prévention et de lutte contre les feux de forêt.

3.1.4 Impact de l'humidité du sol

L'humidité du sol peut ralentir ou éteindre les incendies de forêt, car l'eau absorbe la chaleur et réduit la température autour des arbres en feu.

Dans notre modèle, nous avons réduit la probabilité qu'un arbre prenne feu lorsque le sol est humide. Cela peut avoir un impact significatif sur la propagation des incendies, en fonction des conditions météorologiques et environnementales locales. Si le sol est humide, cela peut aider à ralentir la propagation des flammes, tandis que si le sol est sec, cela peut aggraver la situation.

En conclusion, l'humidité du sol doit être prise en compte dans la modélisation des incendies de forêt pour mieux simuler leur propagation.

3.1.5 Type de végétation

Le type de végétation est un facteur important qui peut affecter la propagation du feu dans une forêt. Certaines espèces d'arbres sont plus résistantes au feu que d'autres, par exemple les conifères sont plus inflammables que les feuillus.

Dans notre modèle, nous avons ajouté une variable booléenne *arbre peu inflammable* pour représenter la résistance au feu des arbres. Si cette variable est vraie, alors la probabilité de propagation du feu est réduite de 20%. Nous avons également pris en compte l'impact de l'humidité du sol sur la résistance au feu des arbres, car les arbres dans des zones humides ou avec un accès à de l'eau souterraine sont souvent plus résistants aux incendies que ceux dans des régions arides.

En modifiant la résistance au feu des arbres, notre modèle peut simuler différents types de forêts et étudier leur comportement face aux incendies, ce qui aidera à élaborer des stratégies de prévention et de lutte contre les incendies.

3.2 Gestion de projet

La gestion de projet est un aspect essentiel de tout travail collaboratif. Dans notre projet de modélisation d'incendies de forêt, nous avons utilisé plusieurs outils de gestion de projet pour nous aider à rester organisés, à suivre les progrès et à gérer les tâches de manière efficace. Les quatre technologies que nous avons utilisées sont le diagramme de *Gantt*, *Trello*, *Git* et la méthodologie *agile*.

3.2.1 Diagramme de Gantt

Le diagramme de Gantt est un outil de gestion de projet utilisé pour planifier et suivre les tâches à accomplir dans un projet. Il permet de visualiser les tâches, leur durée et leur dépendance les unes par rapport aux autres. Cet outil a été très utile pour notre projet car il nous a permis de suivre le progrès et de voir les tâches restantes à accomplir.

Nous avons pu identifier les tâches qui prenaient plus de temps que prévu et ajuster notre planification en conséquence. Le diagramme de Gantt a également aidé à définir les rôles et les responsabilités de chaque membre de l'équipe. La figure 3.2 présente une capture ponctuelle du diagramme de Gantt que nous avons utilisé.

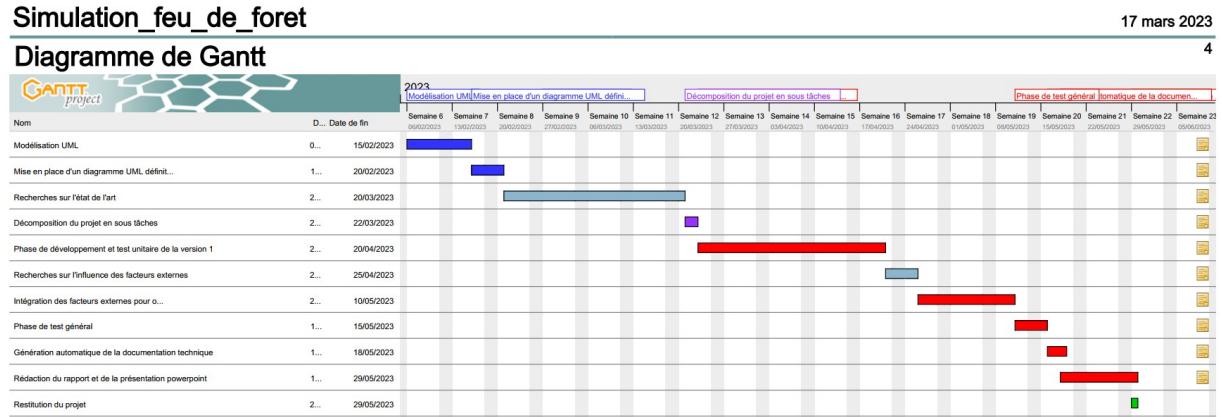


Figure 3.2 – Diagramme de Gantt du projet

3.2.2 Trello

Trello est un outil de gestion de projet avec des tableaux kanban. Il permet de créer des listes de tâches, des cartes pour chaque tâche, et des étiquettes pour classer les tâches. Nous avons utilisé Trello pour suivre les tâches quotidiennes et garder une trace des progrès. Nous avons pu ajouter des commentaires et des notes pour que chaque membre de l'équipe reste informé de l'avancement. Voici une capture d'écran de notre tableau Trello montrant les différentes tâches et leurs statuts.

3.2.3 Git

Git est un outil de gestion de version pour le code source. Il permet de suivre les modifications du code, de restaurer les versions précédentes, et de collaborer sur le code avec plusieurs membres de l'équipe. Nous avons utilisé Git pour notre code de modélisation d'incendies de forêt [12], ce qui a facilité la collaboration entre les membres de l'équipe. Nous avons pu travailler sur différentes parties du code en même temps sans craindre de perdre des modifications importantes. Nous avons également utilisé les branches de Git pour développer de nouvelles fonctionnalités sans interrompre le développement actuel. Plus concrètement, les branches suivantes ont été créées :

1. Une branche *master* pour la version finale du code
2. Une branche *test* pour les fonctionnalités prêtes à être validées par les membres de l'équipe
3. Une branche *dev* pour les fonctionnalités consolidées mais qui doivent être intégrées au projet dans son ensemble
4. Une branche pour chaque membre d'équipe pour le développement proprement dit des tâches.

La figure 3.3 présente une représentation graphique de la stratégie de branches que nous avons adopté ainsi qu'un aperçu ponctuel de l'historique des commit.

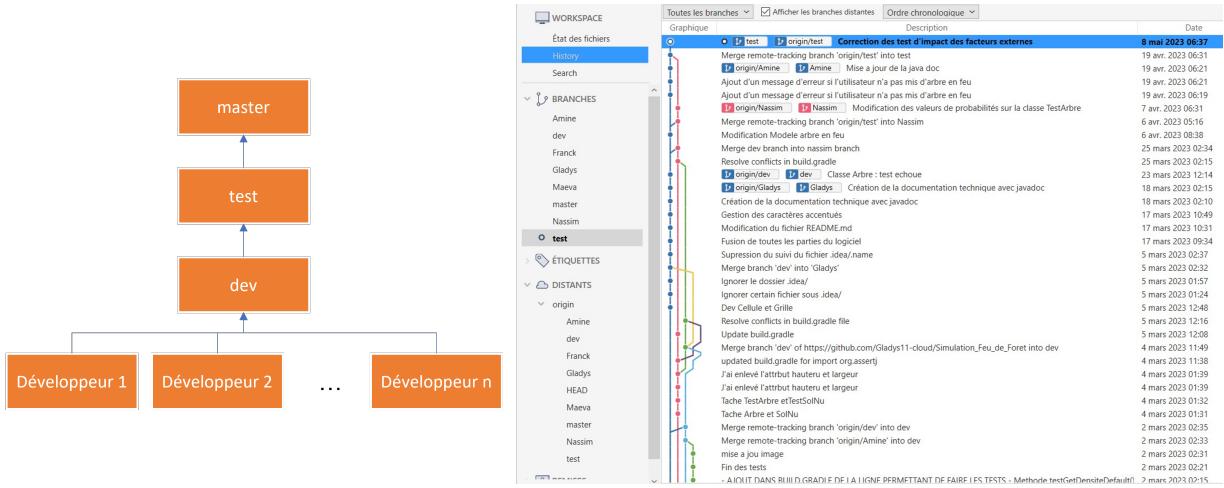


Figure 3.3 – Stratégie des branches et historique des commit

3.2.4 Méthodologie Agile

Nous avons utilisé la méthodologie agile pour gérer notre projet de modélisation d'incendie de forêt.

Cette méthode implique des itérations courtes appelées sprints, qui durent entre une et quatre semaines. Chaque sprint a un objectif spécifique et des tâches sont assignées aux membres de l'équipe pour atteindre cet objectif.

Cette méthode permet une collaboration étroite avec les parties prenantes, une meilleure visibilité sur la progression du projet et encourage l'amélioration continue. En fin de compte, la méthodologie agile a grandement amélioré notre efficacité en tant qu'équipe.

3.3 Modélisation informatique du problème

La modélisation informatique est une étape cruciale dans la conception d'un logiciel. Elle permet de représenter les différentes composantes du système ainsi que les interactions entre elles. Dans ce projet, nous avons utilisé le diagramme de classe de l'UML pour modéliser notre système de simulation de feu de forêt. Ce diagramme nous a permis de visualiser les différentes classes du système, les relations entre elles ainsi que les attributs et les méthodes de chaque classe. De plus, cette modélisation nous a permis de mieux comprendre le fonctionnement de notre système et ainsi de faciliter la phase d'implémentation.

3.3.1 Construction et amélioration du diagramme de classes

La construction du diagramme de classe UML a été une étape importante pour modéliser notre projet de simulation d'incendie de forêt. Chacun des membres de l'équipe a travaillé sur un diagramme de classes individuel pour représenter sa vision du projet.

Après discussion et prise en compte des meilleures idées, nous avons sélectionné la version initiale du diagramme de classes qui était la plus complète et intuitive. Cette version initiale a servi de base pour l'implémentation de notre programme.

Cependant, nous avons réalisé qu'il était possible de simplifier le diagramme et de le rendre encore plus efficace en termes de qualité de code. Nous avons donc amélioré le diagramme en prenant en compte les principes SOLID de programmation pour garantir une structure simple et claire. Nous avons regroupé les différentes classes d'arbres en une seule classe avec un attribut état pour simplifier le diagramme.

La version finale du diagramme de classes a été adoptée par l'équipe pour obtenir une structure plus simple et claire pour le projet. Cela nous a permis d'obtenir un code plus facile à comprendre et à implémenter.

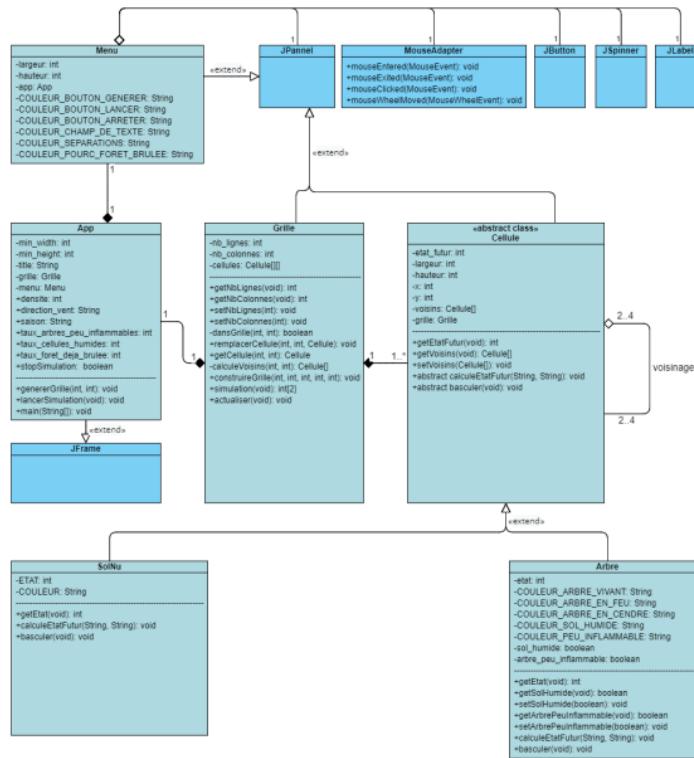


Figure 3.4 – Version finale du diagramme de classes

3.3.2 Description du diagramme de classes

Le diagramme de classes final comprend quatre classes principales et deux autres qui en spécialisent une. Ces classes sont les suivantes :

1. Classe **App** : Cette classe est la fenêtre principale du projet, point de départ de l'application. Elle contient la grille de simulation et le menu, et gère la simulation. Elle possède plusieurs attributs tels que la grille, le menu, la densité des arbres, la direction du

vent, la saison, le taux d'arbres peu inflammables, le taux de cellules humides, le taux de forêt déjà brûlée et un booléen qui permet de savoir si l'utilisateur a cliqué sur le bouton *Arrêter*. Elle a également des méthodes qui permettent de générer la grille et de lancer la simulation de façon itérative jusqu'à ce qu'il n'y ait plus de changement dans la grille ou que l'utilisateur clique sur le bouton *Arrêter*.

2. Classe **Menu** : Cette classe gère les interactions de l'utilisateur avec l'interface graphique. Elle contient les boutons, les champs de texte et les lignes de séparation qui permettent de configurer la grille et de lancer la simulation. Elle a des attributs tels que la largeur et la hauteur minimale du panneau, ainsi que des couleurs en hexadécimal pour les boutons et les champs de texte.

3. Classe **Grille** : Cette classe implémente la grille de simulation et permet de la mettre à jour. Elle a des attributs tels que le nombre de lignes et de colonnes de la grille, ainsi qu'un tableau à deux dimensions contenant toutes les cellules de la grille. Elle a également des méthodes qui permettent de construire la grille et de mettre à jour la grille lors de la simulation. Elle possède des méthodes utiles permettant d'accéder à une cellule spécifique de la grille, d'en déterminer les voisins et même de la modifier.

4. Classe **Cellule** : Cette classe abstraite représente les différentes composantes de la forêt, telles que le sol nu, l'arbre vivant, l'arbre en feu et l'arbre en cendres. Elle a une méthode abstraite, `calculeEtatFutur()`, qui permet de calculer l'état futur d'une cellule en se basant sur plusieurs critères tels que présentés dans la section 3.1. Les classes **SolNu** et **Arbre** héritent de cette classe.

5. Classe **SolNu** : Cette classe hérite de la classe **Cellule** et permet de modéliser un sol nu, qui correspond à une parcelle de terrain sans arbre. Sa méthode principale est `calculeEtatFutur()`, qui retourne simplement l'état courant de la cellule car un sol nu ne change pas d'état après le passage du feu.

6. Classe **Arbre** : Cette classe hérite également de la classe **Cellule** et permet de modéliser un arbre. Ses méthodes permettent de construire des arbres ayant des caractéristiques différentes afin de pouvoir observer le comportement du feu faces à divers types de forêts. Cette classe implémente également la méthode `calculeEtatFutur()`.

3.4 Design de l'interface graphique

Le design de l'interface graphique est une étape cruciale du développement d'un logiciel car il permet de concevoir visuellement la façon dont les utilisateurs vont interagir avec l'application.

Pour notre simulateur d'incendie de forêt, nous avons utilisé l'outil *Figma* pour concevoir l'interface graphique. La grille représente la forêt et est subdivisée en cellules, chacune représentant un arbre ou un sol nu. Le menu de configuration est subdivisé en quatre parties principales, permettant à l'utilisateur de configurer la forêt, les facteurs externes, et la simulation. L'interface graphique est conçue de manière claire et logique,

ce qui permet à l'utilisateur de comprendre rapidement les différentes fonctionnalités et de les utiliser efficacement.

La figure A.1 présente le design préalable tandis que la figure B.1 présente l'interface effectivement implémentée.

3.5 Mise en œuvre de la solution à l'aide du langage Java

Cette partie présente trois algorithmes essentiels pour le bon fonctionnement de la simulation : la boucle de simulation, le calcul de l'état futur d'une cellule et le calcul de la probabilité de propagation du feu sur une cellule. Nous allons décrire ces algorithmes en détail pour mieux comprendre leur fonctionnement et leur rôle dans la simulation.

3.5.1 Algorithme de la boucle de simulation

L'algorithme de la boucle de simulation de la figure C.1 est l'un des éléments clés de la solution. Cet algorithme est responsable de parcourir toutes les cellules de la grille et de calculer leur état futur en fonction de la direction du vent et de la saison. La méthode renvoie un tableau de deux entiers, le premier étant 0 si la simulation doit s'arrêter (s'il n'y a plus d'arbres en feu) et 1 sinon. Le deuxième entier représente le pourcentage de la forêt déjà brûlée.

La boucle de simulation commence par initialiser trois variables : continuerSimulation, nombreTotalArbre et nombreArbreEnCendre. La variable continuerSimulation est initialement à 0, mais elle sera mise à 1 si une cellule en feu est trouvée. Les variables nombreTotalArbre et nombreArbreEnCendre sont utilisées pour calculer le pourcentage de la forêt déjà brûlée. La boucle itère ensuite sur toutes les cellules de la grille. Pour chaque cellule, la méthode calculeEtatFutur est appelée, ce qui calcule l'état futur de la cellule en fonction de la direction du vent et de la saison. Si l'état futur de la cellule est 2 (en feu), la variable continuerSimulation est mise à 1. Enfin, les variables nombreTotalArbre et nombreArbreEnCendre sont mises à jour en fonction de l'état actuel et futur de la cellule. Le pourcentage de la forêt déjà brûlée est calculé en divisant le nombre d'arbres en cendres par le nombre total d'arbres et en multipliant par 100. Le tableau de deux entiers est ensuite retourné.

L'algorithme de la boucle de simulation est un élément essentiel de la solution car il permet de simuler la propagation du feu dans la forêt en calculant l'état futur de chaque cellule. Cet algorithme, ainsi que les autres algorithmes utilisés dans la solution, ont été implémentés en Java.

3.5.2 Algorithme de calcul de l'état futur d'une cellule

Cette méthode visible sur la figure D.1 utilise plusieurs critères pour déterminer l'état futur d'une cellule : le voisinage, la direction du vent, la saison, l'humidité du sol et le type de végétation.

Tout d'abord, si la cellule est un arbre déjà en cendre, son état futur reste le même.

Si la cellule est un arbre en feu, son état futur est défini comme étant en cendre. Si la cellule est un arbre vivant et qu'aucun de ses voisins n'est en feu, elle reste dans son état actuel. Cependant, si la cellule est un arbre vivant et qu'au moins un de ses voisins est en feu, un calcul de probabilité est effectué pour déterminer si le feu va se propager à cette cellule. La probabilité de propagation du feu vers la cellule est calculée à l'aide de la méthode `calculeP()` qui prend en compte la direction du vent, la saison, l'humidité du sol et le type de végétation. Cette probabilité est ensuite utilisée pour faire passer le cas échéant la cellule à l'état en feu.

3.5.3 Algorithme de calcul de la probabilité de propagation

L'algorithme de calcul de la probabilité de propagation du feu présenté sur la figure E.1 est relativement simple. La méthode prend en entrée la direction du vent et la saison courante et calcule la probabilité que l'arbre prenne feu en se basant sur plusieurs critères.

Tout d'abord, la méthode prend en compte l'impact du voisinage sur la probabilité p . Si plusieurs cellules voisines sont en feu, alors la probabilité que l'arbre prenne feu sera plus grande. Ainsi, pour chaque voisin en feu, la probabilité p est augmentée de 0.05. Ensuite, la méthode prend en compte l'impact de la direction du vent sur la probabilité p . Si le vent souffle dans la direction de la cellule, la probabilité que l'arbre prenne feu sera plus grande. Ainsi, si le vent souffle dans la direction de la cellule, la probabilité p est augmentée de 0.5. Si le vent souffle dans une autre direction, la probabilité p est diminuée de 0.1.

La méthode prend également en compte l'impact de la saison sur la probabilité p . Si la saison est l'été, alors la probabilité que l'arbre prenne feu sera plus grande. Ainsi, si la saison est l'été, la probabilité p est augmentée de 0.2. Si la saison est l'hiver, la probabilité p est diminuée de 0.1. Ensuite, la méthode prend en compte l'impact de l'humidité du sol sur la probabilité p . Si le sol est humide, alors la probabilité que l'arbre prenne feu sera plus faible. Ainsi, si le sol est humide, la probabilité p est diminuée de 0.1. Enfin, la méthode prend en compte l'impact du type de végétation sur la probabilité p . Si l'arbre est peu inflammable, alors la probabilité que l'arbre prenne feu sera plus faible. Ainsi, si l'arbre est peu inflammable, la probabilité p est diminuée de 0.2.

En sortie, la méthode retourne la probabilité p , qui est comprise entre 0 et 1.

Conclusion

En conclusion, ce projet de simulation de feu de forêt a permis de mettre en place un modèle simple mais réaliste de propagation du feu, en se basant sur plusieurs critères tels que la direction du vent, la saison, l'humidité du sol et le type de végétation. Le modèle a été implémenté en utilisant le langage de programmation Java et en se basant sur le principe de la programmation orientée objet.

Nous avons vu comment l'algorithme de calcul de l'état futur d'une cellule ainsi que celui de la probabilité de propagation du feu ont été développés. Ces deux algorithmes sont les pierres angulaires de la simulation de feu de forêt et sont essentiels pour comprendre comment le feu se propage et comment il peut être maîtrisé.

Il est important de noter que d'autres facteurs peuvent également influencer la propagation du feu de forêt, tels que la topographie, la densité de la forêt, la distance entre les arbres, la présence d'obstacles tels que des rivières ou des routes, et bien d'autres encore. Ces facteurs peuvent être intégrés dans la simulation pour en augmenter la précision.

En outre, il serait possible d'intégrer des techniques d'intelligence artificielle pour prédire la direction du vent en se basant sur les données des feux de forêt passés. Cela pourrait permettre de mieux prévoir la propagation du feu et d'adopter des mesures de prévention plus efficaces.

Enfin, ce projet de simulation de feu de forêt est un exemple de la façon dont la programmation peut être utilisée pour simuler des phénomènes naturels complexes et pour aider à mieux comprendre les mécanismes qui les sous-tendent. Il est donc possible d'explorer d'autres domaines en utilisant cette approche et de créer des modèles de simulation pour comprendre des phénomènes encore plus complexes.

Bibliographie

- [1] L. HERNÁNDEZ ENCINAS et al. « Modelling forest fire spread using hexagonal cellular automata ». In : *Applied Mathematical Modelling* 31.6 (2007), p. 1213-1227. DOI : 10.1016/j.apm.2006.04.001. URL : <https://doi.org/10.1016/j.apm.2006.04.001>.
- [2] I. KARAFYLLIDIS et A. THANAILAKIS. « A model for predicting forest fire spreading using cellular automata ». In : *Ecological modeling* 99.1 (1997), p. 87-97. DOI : 10.1016/s0304-3800(96)01942-4. URL : [https://doi.org/10.1016/s0304-3800\(96\)01942-4](https://doi.org/10.1016/s0304-3800(96)01942-4).
- [3] A. ALEXANDRIDIS et al. « A cellular automata model for forest fire spread prediction : The case of the wildfire that swept through Spetses Island in 1990 ». In : *Applied Mathematics and Computationg* 204.1 (2008), p. 191-201. DOI : 10.1016/j.amc.2008.06.046. URL : <https://doi.org/10.1016/j.amc.2008.06.046>.
- [4] S. G. BERJAK et J. W. HEARNE. « An improved cellular automaton model for simulating fire in a spatially heterogeneous Savanna system ». In : *Ecological Modelling* 148.2 (2002), p. 133-151. DOI : 10.1016/s0304-3800(01)00423-9. URL : [https://doi.org/10.1016/s0304-3800\(01\)00423-9](https://doi.org/10.1016/s0304-3800(01)00423-9).
- [5] B. DROSSEL et F. SCHWABL. « Self-Organized Critical Forest-Fire Model ». In : *Physical Review Letters* 69.11 (1992), p. 1629-1632. DOI : 10.1103/physrevlett.69.1629. URL : <https://doi.org/10.1103/physrevlett.69.1629>.
- [6] T. GHISU et al. « An optimal Cellular Automata algorithm for simulating wildfire spread ». In : *Environmental Modelling Software* 71 (2015), p. 1-14. DOI : 10.1016/j.envsoft.2015.05.001. URL : <https://doi.org/10.1016/j.envsoft.2015.05.001>.
- [7] K. MUTTHULAKSHMI et al. « Simulating forest fire spread and fire-fighting using cellular automata ». In : *Chinese Journal of Physicse* 65 (2020), p. 642-650. DOI : 10.1016/j.cjph.2020.04.001. URL : <https://doi.org/10.1016/j.cjph.2020.04.001>.
- [8] J. G. FREIRE et C. C. DA CAMARA. « Using cellular automata to simulate wildfire propagation and to assist in fire management ». In : *Natural Hazards and Earth System Sciences* 19.1 (2019), p. 169-179. DOI : 10.5194/nhess-19-169-2019. URL : <https://doi.org/10.5194/nhess-19-169-2019>.
- [9] A. HERNÁNDEZ ENCINAS et al. « Simulation of forest fire fronts using cellular automata ». In : *Advances in Engineering Software* 38.6 (2007), p. 372-378. DOI : 10.1016/j.advengsoft.2006.09.002. URL : <https://doi.org/10.1016/j.advengsoft.2006.09.002>.
- [10] Andrew WUENSCHÉ et Michael LESSER. *Cellular Automata : A Discrete View of the World*. John Wiley & Sons, 2000.
- [11] G. FRANCESCHINI et et AL. « Cellular Automata Models of Fire Spread ». In : *International Journal of Wildland Fire* (2020).

- [12] *Project git repository.* 2023. URL : https://github.com/Gladys11-cloud/Simulation_Feu_de_Foret (visité le 29/05/2023).

A Design de l'interface graphique



Figure A.1 – Design Figma de l'interface graphique

B Interface graphique implementé

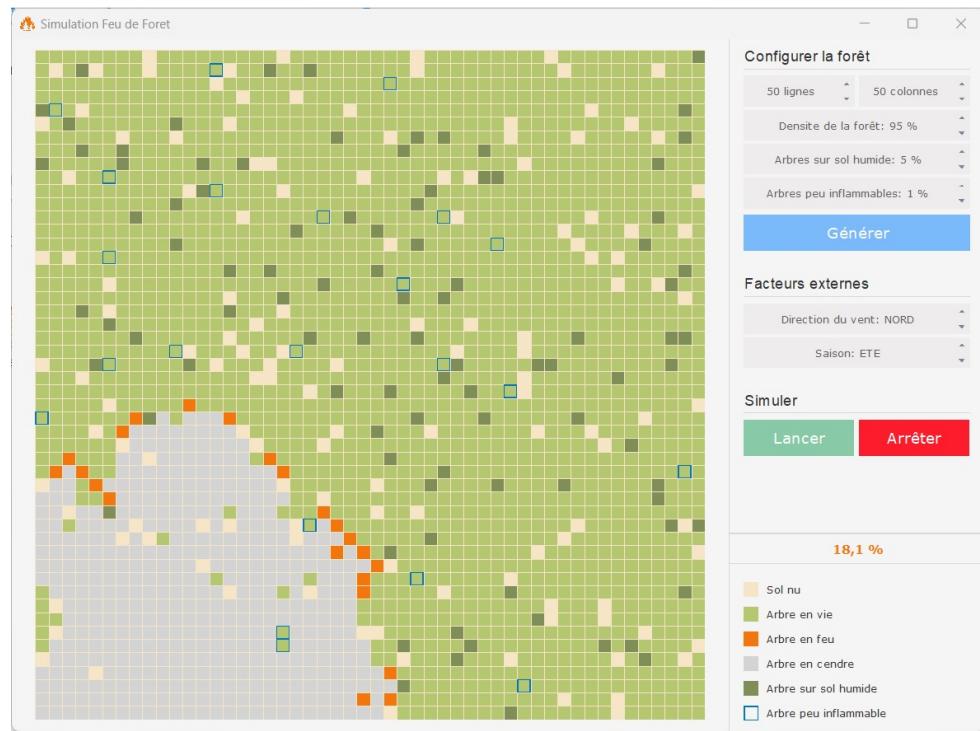


Figure B.1 – Interface graphique réellement implémenté

C Algorithme de la boucle de simulation

```
public double[] simulation(String direction_vent, String saison){  
    double continuerSimulation = 0;  
    double nombreTotalArbre = 0;  
    double nombreArbreEnCendre = 0;  
  
    for(int i = 0; i < this.nb_lignes; i++) {  
        for( int j = 0; j < this.nb_colonnes; j++) {  
            Cellule cellule = this.getCellule(i, j);  
            cellule.calculeEtatFutur(direction_vent, saison);  
            if(cellule.getEtatFutur() == 2)  
                continuerSimulation = 1;  
            if(cellule.getEtat() == 1 || cellule.getEtat() ==  
                2 || cellule.getEtat() == 3) nombreTotalArbre  
                += 1;  
            if(cellule.getEtat() == 3 || cellule.getEtatFutur()  
                () == 3) nombreArbreEnCendre += 1;  
        }  
    }  
  
    return new double[]{continuerSimulation, ((  
        nombreArbreEnCendre/nombreTotalArbre)*100)};  
}
```

Figure C.1 – Code de la méthode Grille.simulation()

D Algorithme de calcul de l'état futur d'une cellule

```
@Override
public void calculeEtatFutur(String direction_vent, String
saison){
    // p : probabilité de propagation du feu vers la cellule
    double p;

    if(this.etat == 3) this.etat_futur = etat;
    else if(this.etat == 2) this.etat_futur = 3;
    else if(this.etat == 1){
        if(this.nombreDeVoisinsEnFeu() == 0) this.etat_futur =
            this.etat;
        else{
            p = this.calculeP(direction_vent, saison);
            if(this.feuVaSePropager(p)) this.etat_futur = 2;
            else this.etat_futur = etat;
        }
    }
}
```

Figure D.1 – Code de la méthode Arbre.calculeEtatFutur()

E Algorithme de calcul de la probabilité de propagation

```
public double calculeP(String direction_vent, String saison){  
    // Impact du voisinage sur p.  
    double p = 0.6 + this.nombreDeVoisinsEnFeu() * 0.05;  
    // Impact de la direction du vent sur p.  
    if(!this.feuVersCellule(direction_vent) && direction_vent  
        != null) p -= 0.1;  
    if(this.feuVersCellule(direction_vent)) p += 0.5;  
    // Impact de la saison sur p  
    if(saison == null){}  
    else if(saison.equals("HIVER")) p -= 0.1;  
    else if(saison.equals("ETE")) p += 0.2;  
    // Impact de l'humidité du sol sur p.  
    if(this.soleEstHumide()) p -= 0.1;  
    // Impact du type de végétation sur p.  
    if(this.arbreEstPeuInflammable()) p -= 0.2;  
    // La probabilité doit être comprise entre 0 et 1.  
    if(p < 0) p = 0;  
    if(p > 1) p = 1;  
  
    return p;  
}
```

Figure E.1 – Code de la méthode Arbre.calculeP()