RAPPORT PROJET

J-Sim Forest

EXIA ANNEE 2

J-Sim Forest

Office National des Forêts

CESI exia
ÉCOLE SUPÉRIEURE D'INFORMATIQUE

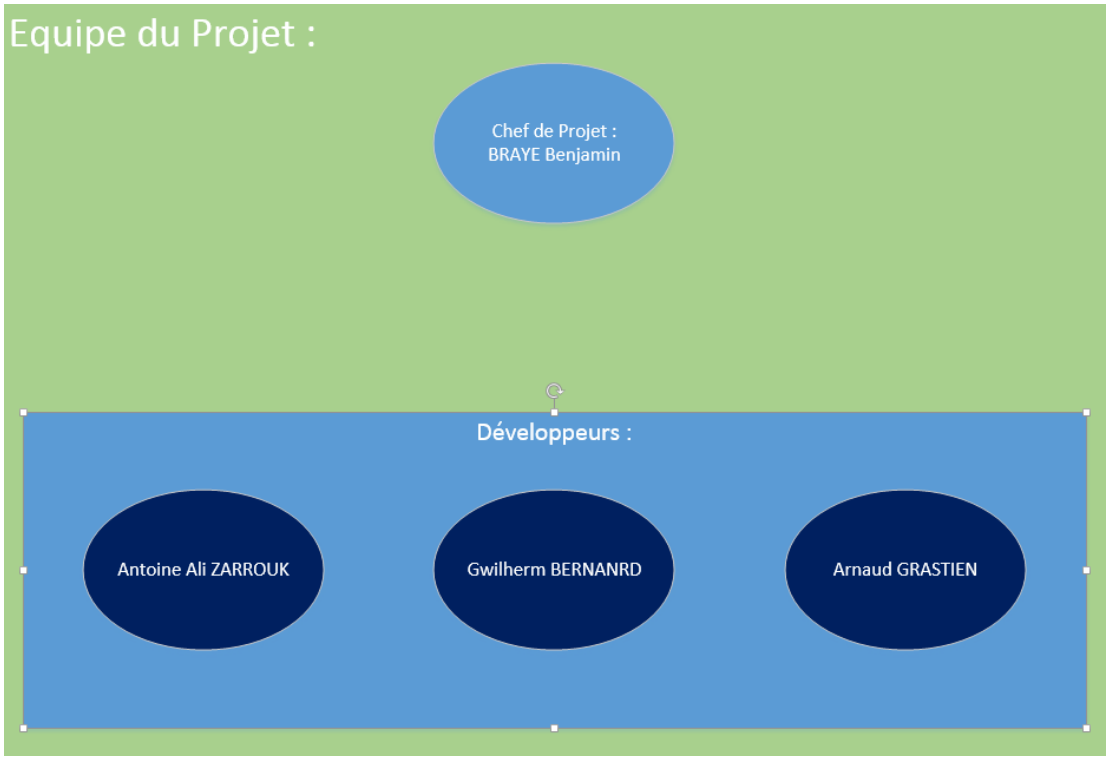
Table des matières

I. Partie 1 : Présentation

A. Introduction

Dans le cadre de notre cursus d'analyste programmeur, nous avons intégré depuis peu une nouvelle technologie, le Java SE. Ce projet a donc pour but de regrouper les différentes connaissances que nous avons acquises. En parallèle, vous trouverez dans ce rapport les différentes informations concernant la gestion de ce projet, ainsi que tous ce qui tourne autour de l'analyse fonctionnelle et UML de notre application.

B. Rôles des membres du groupe



C. Rappel de la demande

L'ONF (l'Office National des Forêts) a fait appel à vous pour le développement d'un outil qui permettrait de simuler le développement des arbres et végétaux dans une forêt. Ce type de simulation est très utile pour permettre aux décideurs de cet organisme de prendre des décisions en fonction des projections dans le temps que le logiciel fournira. Il peut permettre par exemple d'analyser l'occupation des sols, la diffusion d'un incendie, ou d'une invasion d'insectes. C'est à partir de ces besoins que le projet J-Sim Forest a vu le jour. Le langage Java a été choisi par l'ONF pour sa portabilité sur les différents OS.

II. Partie 2 : Gestion de projet

A. Planning prévisionnel

Planning Prévisionnel					
Jour/Date	Horaire	Antoine Ali Zarrouk	Gwilherm Bernard	Arnaud Grastien	Benjamin
	Après-Midi	Paramétrage du serveur de Versioning	Réalisation du Rapport n°1	Outils de Développements	Outils de Développements
Vendredi	Matin	BDD/UML	UML	Mockups	UML / Préparation des Livrables
	Après-Midi	Formation SQL Lite / Java 2D	Formation SQL Lite / Java 2D	Formation SQL Lite / Java 2D	Formation SQL Lite / Java 2D
Lundi	Matin	Démarrage Programmation avec Eclipse	Démarrage Programmation avec Eclipse	Démarrage Programmation avec Eclipse	Démarrage Programmation avec Eclipse
	Après-Midi	Développement de IHM	Création du package settings	Création du package settings	Développement IHM
Mardi	Matin	Classe Grille /Cellule	Package DataBase	Classe Grille /Cellule	Package DataBase
	Après-Midi	Classe Grille /Cellule	Package DataBase	Classe Grille /Cellule	Package DataBase
Mercredi	Matin	Association des Différents Packages	Association des Différents Packages	Association des Différents Packages	Association des Différents Packages
	Après-Midi	Intégration Java 2D	Intégration Java 2D	Intégration Java 2D	Intégration Java 2D
Jeudi 31	Matin	Intégration Java 2D	Intégration Java 2D	Intégration Java 2D	Documents du Projet
	Après-Midi	Correction Bugs	Correction Bugs	Correction Bugs	Documents du Projet
Vendredi	Matin	Finalisation de l'Automate	Finalisation de l'Automate	Finalisation de l'Automate	Documents du Projet
	Après-Midi	Finalisation de l'Automate	Finalisation de l'Automate	Finalisation de l'Automate	Documents du Projet
Lundi	Matin	SOUTENANCES			
	Après-Midi				

B. Planning (réalisé)

Planning Réalisé					
Jour/Date	Horaire	Antoine Ali Zarrouk	Gwilherm Bernard	Arnaud Grastien	Benjamin
Jeudi 31	Après-Midi	Paramétrage du serveur de Versioning	Outils de Développements	Outils de Développements	Outils de Développements
Vendredi	Matin	BDD/UML	UML/Analyse Fonctionnelle	Mockups	UML/Analyse Fonctionnelle
	Après-Midi	Formation SQL Lite / Java 2D	Formation SQL Lite / Java 2D	Formation SQL Lite / Java 2D	Formation SQL Lite / Java 2D
Lundi	Matin	Classe Grille / Cellule	Création du package settings	Création du package settings	Création Enumémration / Interface
	Après-Midi	Classe Grille / Cellule	Création du package settings	Création du package settings	Développement IHM
Mardi	Matin	Création Grille en Java 2D	Création Grille en Java 2D	Paramétrage SQL Lite	Paramétrage SQL Lite
	Après-Midi	Création Grille en Java 2D	Création Grille en Java 2D	Package DataBase	Package DataBase
Mercredi	Matin	Intégration Classe G/C dans l'affichage	Intégration Classe G/C dans l'affichage	Package DataBase	Package DataBase
	Après-Midi	Intégration Classe G/C dans l'affichage	Intégration Classe G/C dans l'affichage	Changement UML	Changement UML
Jeudi 31	Matin	Développement Mode Incendie / Insecte	Développement Mode Incendie / Insecte	Intégration Java 2D	Documents du Projet
	Après-Midi	Développement Mode Incendie / Insecte	Développement Mode Incendie / Insecte	Package DataBase	Documents du Projet
Vendredi	Matin	Correction Bugs	Correction Bugs	Correction Bugs	Documents du Projet
	Après-Midi	Finalisation de l'Automate	Finalisation de l'Automate	Finalisation de l'Automate	Finalisation du dossier de Projet
Lundi	Matin	SOUTENANCES			
	Après-Midi				

C. Synthèse sur l'organisation et le déroulement du projet (dont la répartition de la charge de travail)

Le Projet J-Sim Forrest a été l'occasion de réunir une nouvelle équipe de développeur, qui tout au long de la durée de ce projet a fait preuve d'initiative et de coordination exemplaire pour mettre à bien les attentes fixées par l'ONF. De plus, la bonne répartition des tâches nous a permis de rendre actif chaque membre groupe sur toute la durée du projet. Nous avons mis en œuvre une analyse fonctionnelle et UML complète avant de passer à la phase de développement de l'application.

III. Partie 3 : Développement

A. Analyse fonctionnelle (les diagrammes de cas d'utilisation seront présentés dans cette partie)

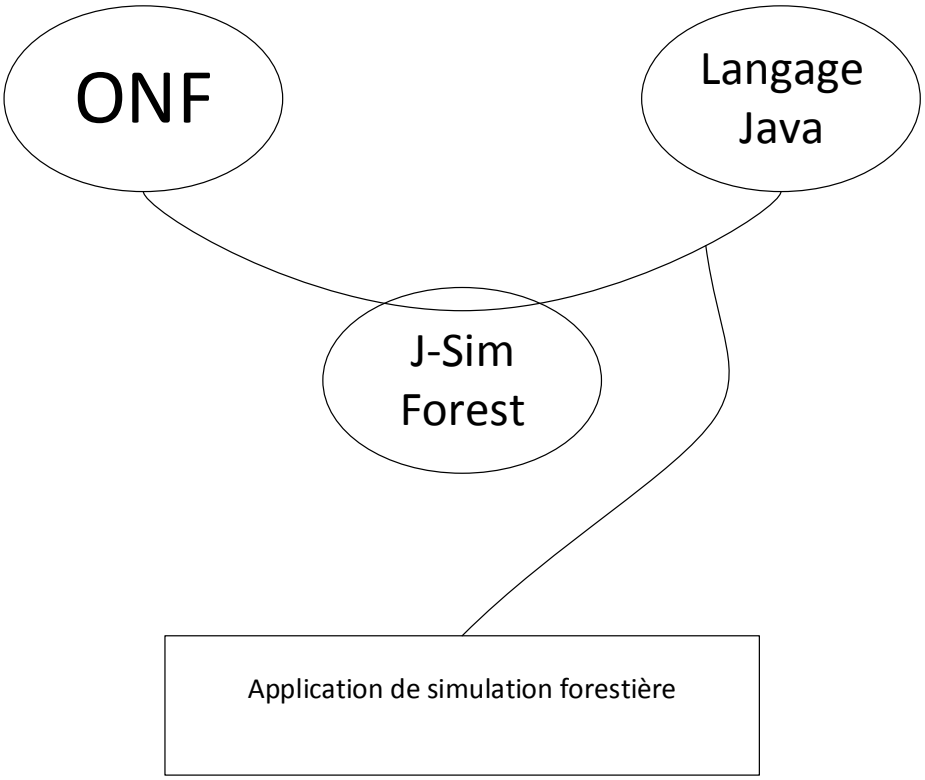
Ce diagramme nous a permis d'identifier les possibilités d'interaction entre le système et les acteurs.

L'utilisateur peut, dans le cadre de l'utilisation de cette application, exécuter une simulation Forestière. Pour ce faire le choix des paramètres doit avoir été réalisé. Pour fonctionner, le choix des paramètres inclus les modules de choix de la taille, du choix du nombre de pas et du choix de la vitesse d'exécution.

L'utilisateur peut également importer et/ou exporter des paramètres et aura besoin pour cela du module de choix des paramètres.

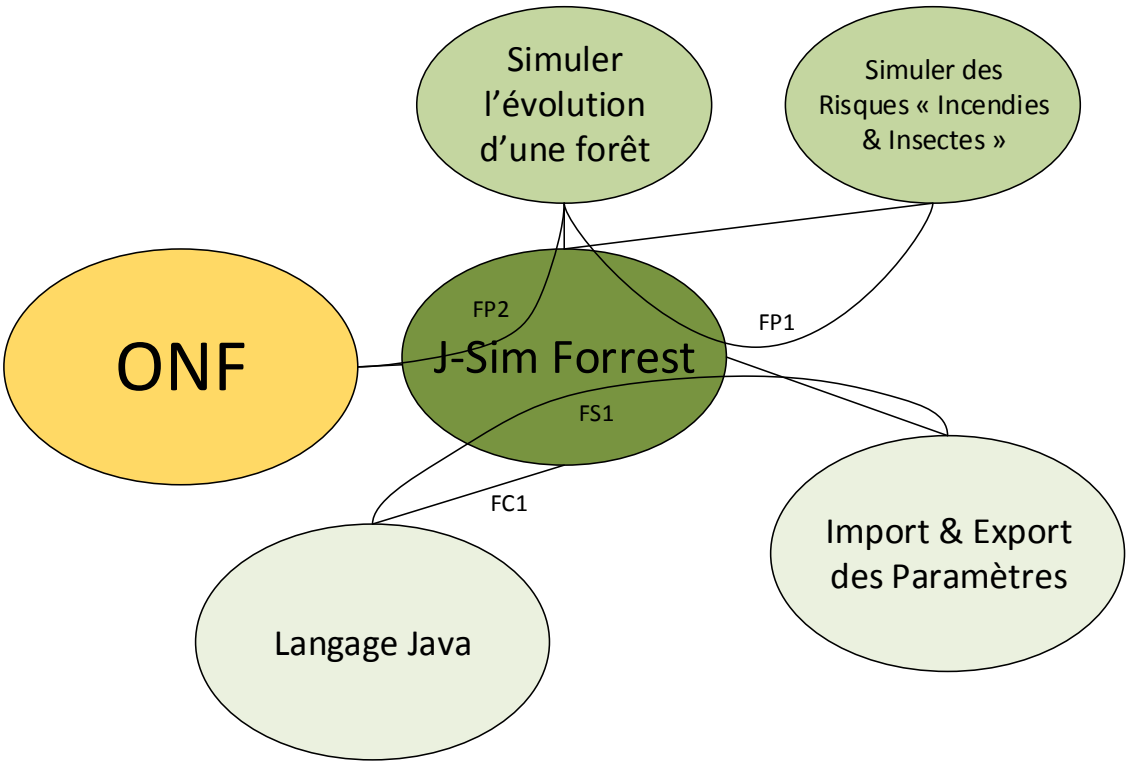
Pour exécuter une simulation des risques, l'utilisateur peut également effectuer une simulation des risques. Il aura cependant besoin de la simulation forestière et inclura un module de choix du risque et offrira le choix entre les risques d'incendie et d'insectes.

- Bête à corne

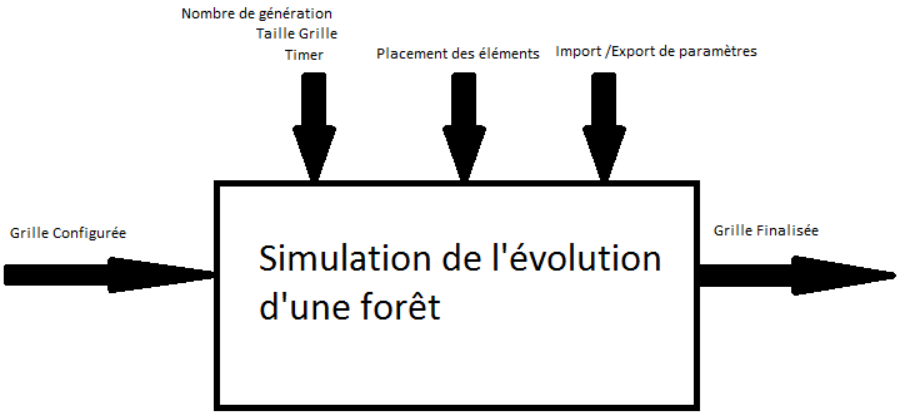


Le diagramme bête à Cornes nous permet de nous resituer dans le contexte. A savoir : que l'ONF nous demande un produit permettant de réaliser une simulation forestière et le tout en Java.

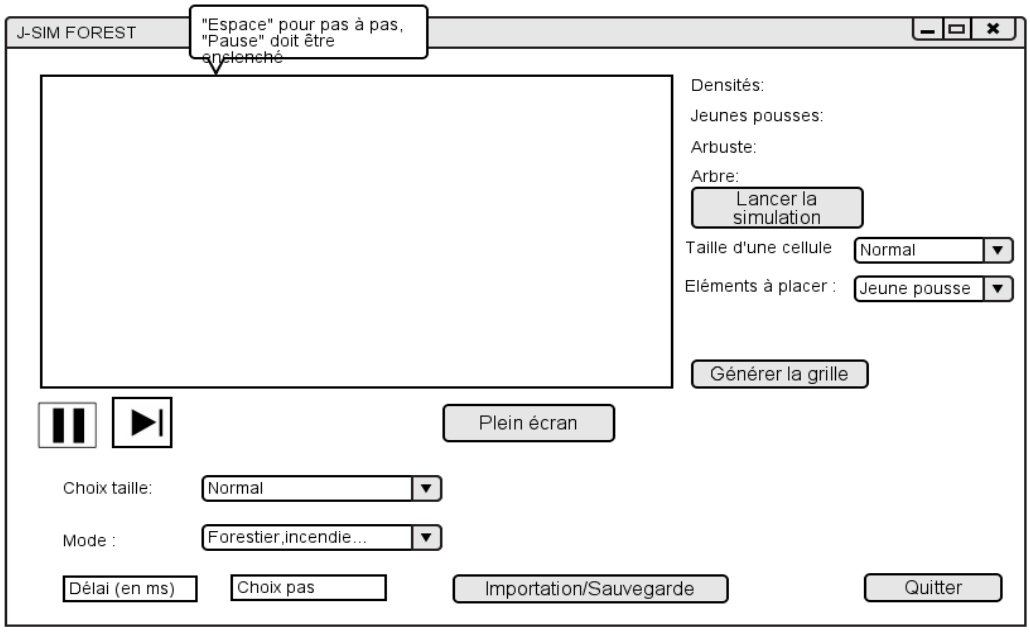
- Diagramme Pieuvre



- SADT



B. Maquette de l'IHM



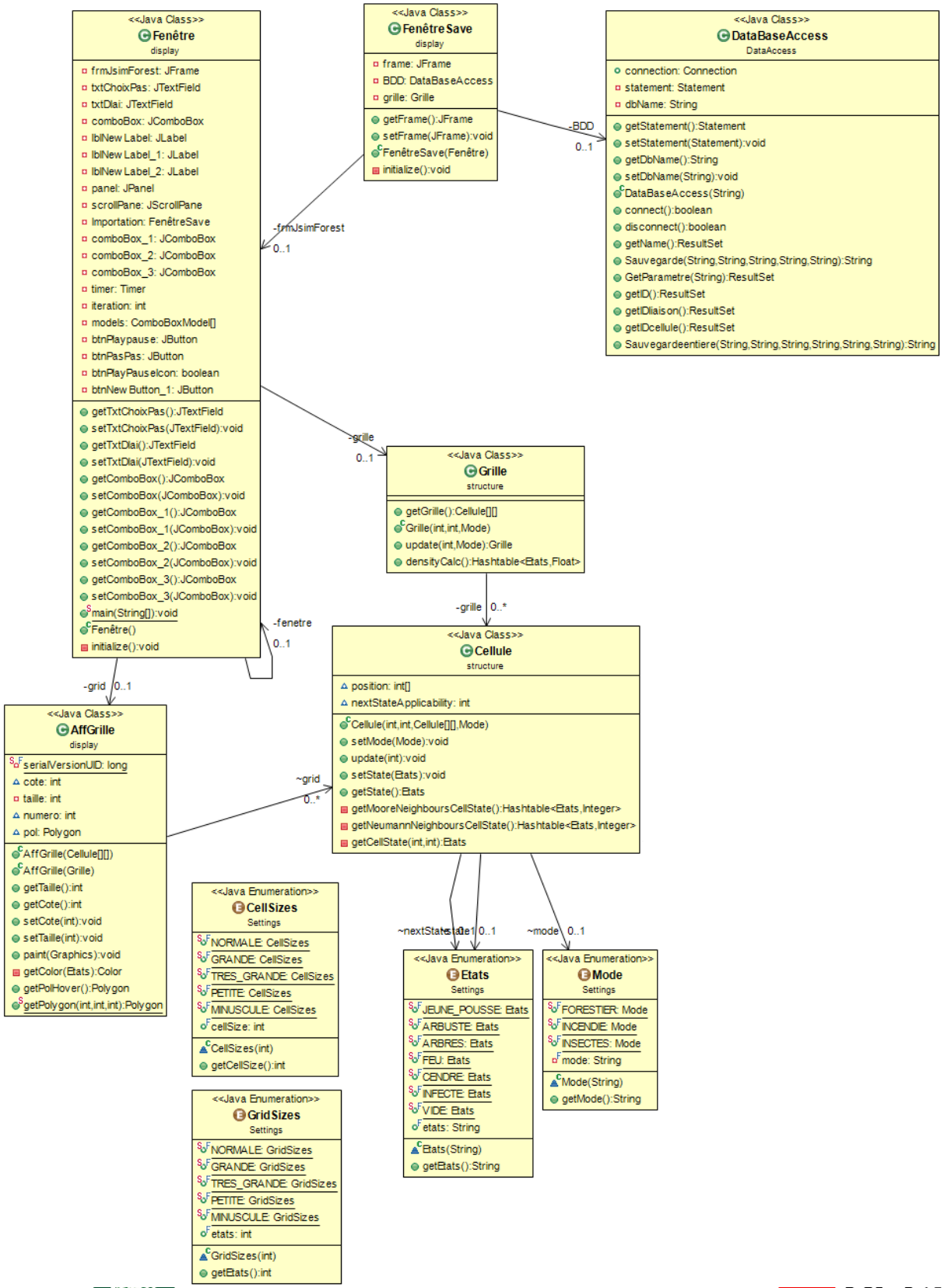
C. Conception UML commentée. Vos choix conceptuels doivent être justifiés.

- Diagramme de Classe

Il nous sert à représenter les classes, les interfaces et les énumérations intervenantes dans le système. Nous avions pensé travailler sur trois packages : Settings, Display et Data base.

Dans le package display, nous pensions que chaque classe aurait ses propres méthodes pour s'afficher d'où l'implémentation des classes grilles et cellules de l'interface affichable. Les cellules composent une grille. Cette grille composée d'un objet stockant les paramètres. La fenêtre élément central de l'application étant une agrégation de la grille, d'un Time (ayant pour but de rafraichir l'affichage à intervalles réguliers), et héritant de la classe de paramètre dans le but de stocker facilement les informations.

Les énumérations ont pour but de convertir des valeurs facilement lisibles pour le développeur en un code numérique plus facile à stocker et à manipuler.


<<Java Class>>
Fenêtre
display
frmJsimForest: JFrame
txtChoixPas: JTextField
txtDlai: JTextField
comboBox: JComboBox
lblNew Label: JLabel
lblNew Label_1: JLabel
lblNew Label_2: JLabel
panel: JPanel
scrollPane: JScrollPane
Importation: FenêtreSave
comboBox_1: JComboBox
comboBox_2: JComboBox
comboBox_3: JComboBox
timer: Timer
iteration: int
models: ComboBoxModel[]
btnPlaypause: JButton
btnPasPas: JButton
btnPlayPauseIcon: boolean
btnNew Button_1: JButton
getTxtChoixPas():JTextField
setTxtChoixPas(JTextField):void
getTxtDlai():JTextField
setTxtDlai(JTextField):void
getComboBox():JComboBox
setComboBox(JComboBox):void
getComboBox_1():JComboBox
setComboBox_1(JComboBox):void
getComboBox_2():JComboBox
setComboBox_2(JComboBox):void
getComboBox_3():JComboBox
setComboBox_3(JComboBox):void
main(String[]):void
Fenêtre()
initialize():void
<<Java Class>>
FenêtreSave
display
frame: JFrame
BDD: DataBaseAccess
grille: Grille
getFrame():JFrame
setFrame(JFrame):void
FenêtreSave(Fenêtre)
initialize():void
<<Java Class>>
DataBaseAccess
DataAccess
connection: Connection
statement: Statement
dbName: String
getStatement():Statement
setStatement(Statement):void
getDbName():String
setDbName(String):void
DataBaseAccess(String)
connect():boolean
disconnect():boolean
getName():ResultSet
Sauvegarde(String,String,String,String,String):String
GetParametre(String):ResultSet
getID():ResultSet
getIDliaison():ResultSet
getIDcellule():ResultSet
Sauvegardeentiere(String,String,String,String,String,String):String
-BDD
0..1
-frmJsimForest
0..1
-grille
0..1
<<Java Class>>
Grille
structure
getGrille():Cellule[][]
Grille(int,int,Mode)
update(int,Mode):Grille
densityCalc():Hashtable<Etats,Float>
-grille
0..*
-fenetre
0..1
-grid
0..1
<<Java Class>>
Cellule
structure
position: int[]
nextStateApplicability: int
Cellule(int,int,Cellule[][],Mode)
setMode(Mode):void
update(int):void
setState(Etats):void
getState():Etats
getMooreNeighboursCellState():Hashtable<Etats,Integer>
getNeumannNeighboursCellState():Hashtable<Etats,Integer>
getCellState(int,int):Etats
<<Java Class>>
AffGrille
display
serialVersionUID: long
cote: int
taille: int
numero: int
pol: Polygon
AffGrille(Cellule[][])
AffGrille(Grille)
getTaille():int
getCote():int
setCote(int):void
setTaille(int):void
paint(Graphics):void
getColor(Etats):Color
getPolHover():Polygon
getPolygon(int,int,int):Polygon
~grid
0..*
<<Java Enumeration>>
CellSizes
Settings
NORMALE: CellSizes
GRANDE: CellSizes
TRES_GRANDE: CellSizes
PETITE: CellSizes
MINUSCULE: CellSizes
cellSize: int
CellSizes(int)
getCellSize():int
<<Java Enumeration>>
GridSizes
Settings
NORMALE: GridSizes
GRANDE: GridSizes
TRES_GRANDE: GridSizes
PETITE: GridSizes
MINUSCULE: GridSizes
etats: int
GridSizes(int)
getEtats():int
~nextStateAppli
0..1
~mode
0..1
<<Java Enumeration>>
Etats
Settings
JEUNE_POUSSE: Etats
ARBUSTE: Etats
ARBRES: Etats
FEU: Etats
CENDRE: Etats
INFECTE: Etats
VIDE: Etats
etats: String
Etats(String)
getEtats():String
<<Java Enumeration>>
Mode
Settings
FORESTIER: Mode
INCENDIE: Mode
INSECTES: Mode
mode: String
Mode(String)
getMode():String

Modification finale du diagramme UML. La classe de fenêtre a gagné des attributs pour permettre d'accéder aux différents objets depuis les objets de gestion d'évènements. Les énumérations supplémentaires sont présentes pour permettre de remplir de manière plus pratique les listes déroulantes contenant les tailles de cellules et de grille.

Nous nous sommes ensuite rendus compte que en utilisant Java2D, il nous était plus simple de créer une classe gérant l'affichage de la grille et des cellules, nous avons donc crée cette classe et déplacé nos classes grille et cellule dans un nouveau package que nous avons nommé structure.

De manière générale, l'ensemble de nos classes ont gagné des méthodes et attributs dont nous n'avions pas pensé avoir besoin à la première analyse.

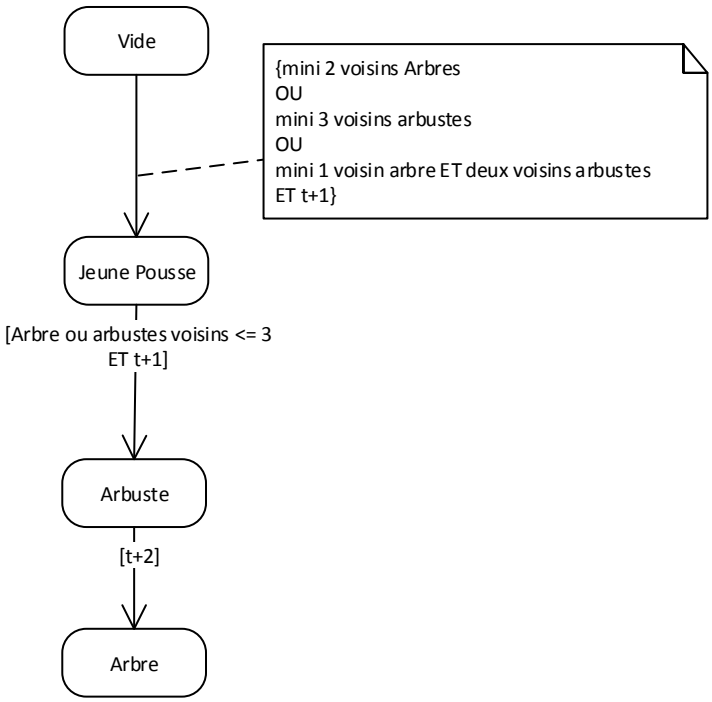
De plus, la classe Paramètres a été supprimée car puisque les objets Grille et cellule ne servaient plus à l'affichage, il paraissait plus logique de stocker les paramètres des objets dans les objets eux-mêmes et d'éviter un maximum les doublons.

Ce diagramme n'est probablement pas final et est peut-être amené à évoluer pour par exemple l'implémentation du module d'export CSV, le renommage de certaines méthodes ou autre…

- Diagramme d'activité

Nous avons choisi ce diagramme UML, afin de décrire sous forme d'enchainement d'activités le comportement du système et de ses composants. Il nous permet de décrire l'enchainement des actions à réaliser pendant la réalisation d'une simulation

- Diagramme d'état de transition (Mode Forestier)



Le choix de ce diagramme est justifié par le fait qu'il représente le fonctionnement de l'automate en mode forestier et nous a permis de respecter les conditions de passage d'un état à l'autre.

IV. Partie 4 : Conclusion & perspective

A. Analyse des problèmes rencontrés et solutions apportées.

Tout au long de l'avancée de ce projet, ayant bien réparti les tâche à chaque membre de l'équipe nous avons malgré tout eu quelques problèmes sur l'affichage de la grille en Java 2D dans un JScrollPane.

B. Evolution proposée pour le système

Malgré une application relativement complète, nous pensons qu'il serait possible d'ajouter les fonctionnalités suivantes :

- Revenir aux états précédents
- Remplacer les couleurs unies par des images
- Nouveau Look and Feel

C. Bilan personnel & groupe

Groupe :

Au cours du projet, la bonne coordination entre les membres du groupe, nous a permis de mener à bien les demandes du client.

Gwilherm BERNARD :

Ce projet fut très intéressant pour moi. Ce fut encore une fois l'occasion de mettre à profit mes connaissances. Il m'a également permit de travailler en groupe, même si quelques tensions ont pu apparaitre lors de cette semaine. Au premier abord le sujet pouvait paraitre compliqué mais après une bonne analyse il nous est apparu bien plus simple. Le Java est un langage qui n'est pas évident à maitriser mais j'ai bien réussi à le prendre en main. Pour résumer ce fut un bon projet qui m'a beaucoup intéressé.

Arnaud GRASTIEN :

Le JAVA n'est pas mon langage préféré, pourtant j'ai aidé à la réalisation d'un projet dans ce langage. J'ai appris à utiliser les logiciels de versioning pour travailler en groupe sur le projet. C'est la 1ère fois qu'on abordait l'UML dans le cadre d'un projet, ce qui a amélioré la mise en pratique des connaissances sur l'UML ainsi que le JAVA. Quelques moments de panique lors de "merges" boiteux qui font perdre plus de temps que de code. En conclusion, ça a été un très bon projet.

Benjamin BRAYE :

Projet très intéressant, j'ai appris débloquer certaine situation sur le Java durant ce projet, d'autre part, je souhaitai revenir sur la bonne ambiance dans le groupe, sans laquelle nous n'aurions pu atteindre nos objectifs.

Antoine-Ali ZARROUK :

Projet très intéressant. M'a permis de prendre un peu mieux le Java en main. Il est cependant dommage que nous n'ayons pas eu plus de temps pour pouvoir réaliser en temps réel les modifications sur les diagrammes UML et les commenter.

V. Annexe :

JScrollPane : JScrollPane est un conteneur permettant de munir un composant de barres de défilement. Ceci permet de visualiser des composants plus grands que l'espace dans lequel ils sont visualisés. Le composant scrollé doit implémenter l'interface Scrollable.