

Rapport de projet

Dynamique Forestière



Réalisé par

Jules DIAZ - Kévin GIORDANI - Antoine NGUYEN - Clara REBEJAC

Sous la direction de

Marc JOANNIDES

Pour l'obtention du DUT Informatique

Année universitaire 2015 - 2016

Remerciements

Nous tenons à remercier notre enseignant tuteur, Monsieur Joannides, pour son aide et son temps tout au long de ce projet.

Sommaire

<u>Introduction</u>	Page 6
<u>1 Analyse</u>	Page 7
<u>1.1 Analyse du sujet et de son contexte</u>	Page 7
<u>1.1.1 Analyse de l'existant</u>	Page 8
<u>1.1.2 Analyse de l'environnement dans lequel le logiciel va être utilisé</u>	Page 10
<u>1.2 Analyse des besoins fonctionnels</u>	Page 10
<u>1.2.1 Spécifications fonctionnelles</u>	Page 10
<u>1.2.2 Diagrammes d'analyse</u>	Page 11
<u>1.3 Analyse des besoins non fonctionnels</u>	Page 14
<u>2 Rapport technique</u>	Page 15
<u>2.1 Conception</u>	Page 15
<u>2.2 Résultats</u>	Page 17
<u>2.3 Perspectives</u>	Page 18
<u>3 Manuel d'utilisation</u>	Page 19
<u>4 Rapport d'activité</u>	Page 23
<u>4.1 Méthode de développement</u>	Page 23
<u>4.2 Planification</u>	Page 23
<u>4.3 Méthodes et outils de travail</u>	Page 23
<u>Conclusion</u>	Page 24
<u>Les références bibliographiques et sitographiques</u>	Page 25
<u>Annexes</u>	Page 26

Table des Figures

Figure 1 - Maquette de l'interface graphique.....	Page 10
Figure 2 - Diagramme de cas d'utilisation.....	Page 11
Figure 3 - Diagramme de classes.....	Page 11
Figure 4 - Diagramme de séquence lorsque l'utilisateur lance la simulation.....	Page 12
Figure 5 - Diagramme de séquence lorsque le moteur gère la forêt.....	Page 13
Figure 6 - Diagramme de séquence lorsque l'écran actualise l'affichage.....	Page 14
Figure 7 - Arborescence des fichiers.....	Page 16
Figure 8 - Exemple modèle logistique.....	Page 17
Figure 9 - Exemple modèle malthusien.....	Page 18
Figure 10 - Menu de lancement.....	Page 20
Figure 11 - Capture simulation lancée.....	Page 20
Figure 12 - Anatomie de l'interface.....	Page 21
Figure 13 - Carte de la forêt.....	Page 21
Figure 14 - Graphique de la population en fonction du temps.....	Page 21
Figure 15 - Interface d'entrées des paramètres.....	Page 22
Figure 16 - Boutons de l'interface.....	Page 22
Figure 17 - Boutons de l'interface.....	Page 22
Figure 18 - Fichier data.txt.....	Page 23
Figure 19 - Fichier conf.txt.....	Page 23
Figure 20 - Planning prévisionnel.....	Page 24
Figure 21 - Planning réel.....	Page 24
Figure 22 - Diagramme de séquence de l'algorithme.....	Page 27

Glossaire

agronomie* : étude des relations entre les plantes cultivées, le sol, le climat et les techniques de culture, dont les principes régissent la pratique de l'agriculture.

implémenter* (un algorithme): installer en réalisant les adaptations nécessaires à son fonctionnement.

java.awt* : AWT propose un ensemble de composants et de fonctionnalités pour créer des interfaces graphiques.

javax.swing* : Swing fait partie de la bibliothèque Java Foundation Classes (JFC). C'est une API dont le but est similaire à celui de l'API AWT mais dont les modes de fonctionnement et d'utilisation sont complètement différents.

Modèle individu-centré* : processus de naissance, mort, compétition, mutation (Metz et al. 1996, Bolker et Pacala 1997, Kisdi 1999, Dieckmann et Law 2000, Doebeli et Dieckmann 2001, Fournier et Méléard 2004, C., Ferrière et Méléard 2006...).

modèles numériques* : un modèle numérique est une représentation par ordinateur qui simule un algorithme mathématique.

stochastique* : se dit de phénomènes qui, partiellement, relèvent du hasard et qui font l'objet d'une analyse statistique.

modèle malthusien* : modélisation déterministe de l'évolution d'une population proposée par Thomas Malthus en 1798. Les seuls paramètres pris en compte sont le taux de natalité et le taux de mortalité.

modèle logistique* : modélisation déterministe de l'évolution d'une population proposée par Verhulst en 1936. Ce modèle ajoute un facteur de compétition au modèle malthusien : la croissance est limitée par la proximité d'autres individus (dispute pour les ressources).

capacité biotique* : la capacité biotique, ou capacité de charge, est la taille maximale de la population qu'un milieu peut supporter. Dans ce projet, elle sera définie par le facteur de compétition inter-individus.

Introduction

La modélisation des dynamiques des populations d'arbres vise à expliquer, et éventuellement à prévoir, les évolutions d'une population dans un cadre écologique ou géographique donné. Cette démarche, qui s'est à l'origine concentrée sur la dynamique des populations humaines, s'applique aussi à la dynamique de ressources végétales (telles que les forêts) ou animales et est généralement couplée à des outils de gestion visant à optimiser l'exploitation durable de ces ressources naturelles.

Le projet consiste à **implémenter*** le modèle numérique de Bolker et Pacala dans une simulation de population d'arbres. Ce programme permettra à l'utilisateur de configurer des paramètres et de lancer la simulation gérée par un "moteur", ainsi que d'afficher l'interface graphique pour la visualisation des résultats. Les paramètres fondamentaux de la simulation sont la durée de vie moyenne d'un arbre, son taux (*ses chances*) de reproduction et l'intensité de compétition qui agit entre deux arbres voisins.

Notre problématique sera de réaliser un programme qui simule l'évolution d'une population d'arbres afin de parvenir à un outil de gestion optimal d'une "forêt".

Après avoir mis au point la définition du sujet de notre projet, l'ensemble des besoins fonctionnels et les contraintes techniques définies avec le tuteur, nous présenterons, dans le rapport technique, le fonctionnement de la simulation par le fonctionnement de l'algorithme fourni, pour ensuite nous concentrer sur toutes les fonctionnalités implémentées dans le programme. Vous trouverez également une partie du manuel d'utilisation indiquant toutes les fonctionnalités dont l'utilisateur dispose, ainsi qu'un rapport d'activité rendant compte des méthodes de travail mises en œuvre dans ce projet.

1 Analyse

1.1 Analyse du sujet et de son contexte

1.1.1 Analyse de l'existant

Dès le *XXème*, le développement des premiers ordinateurs a permis aux scientifiques de développer des **modèles numériques***. Les premiers algorithmes de simulation sont des systèmes dynamiques **stochastiques***, qui portent sur des quantités évoluant au cours du temps et dépendant du hasard. On s'intéressera ici au modèle de Bolker et de Pacala dans lequel l'algorithme est implémenté dans notre simulation.

Cela fait donc appel à des notions d'**agronomie*** et permet aux biologistes d'étudier l'effet des paramètres individuels sur l'évolution de la population d'arbres. Mais afin d'étudier le modèle de Bolker et de Pacala, les biologistes utilisent des programmes pour tester leurs résultats. Il n'est pas vraiment possible de faire l'analyse de l'existant puisque ces programmes sont souvent des petits projets des biologistes qu'ils se partagent entre eux mais qu'ils ne publient pas.

Ces petits projets sont principalement des simulations de populations de paramètre individu-centré. Elle diffère par son graphisme, par l'affichage des individus, que l'on ne mettra pas si l'on veut accélérer les calculs, et par la précision des calculs. Bien sûr, chaque simulation se fait de façon aléatoire mais certains programmes peuvent sauvegarder les simulations que l'on veut revoir.

Algorithme :

$F \leftarrow$ Population d'arbres initiale

$t \leftarrow 0$

Tant que 1

$$\lambda g \leftarrow |F| (\lambda b + \lambda d) + \sum_{A \in F} \lambda c(A) \quad (\text{Taux Global})$$

$$t \leftarrow t + \frac{-\ln(\text{rand}())}{\lambda g} \quad (\text{Avancée du temps : inversion de la fonction de répartition})$$

$e \leftarrow$ Tirage

x_i	B	D	C
p_i	$ F \lambda b$	$ F \lambda d$	$\sum_{A \in F} \lambda c(A)$

switch (e)

cas B : $a \leftarrow$ choix de l'arbre uniforme

$a' \leftarrow$ tirage $D(a)$

$F = F \cup \{a'\}$

cas D : $a \leftarrow$ choix de l'arbre uniforme

$F = F \setminus \{a\}$

cas C : tirage d'un arbre en compétition

a_1	...	$a(i * i)$
$\lambda c(a_1)$...	$\lambda c(a(n))$

$F = F \setminus \{a\}$

Fin switch

Mettre à jour

Fin Tant que

1.1.2 Analyse de l'environnement dans lequel le logiciel va être utilisé

Le logiciel de la simulation du modèle numérique de Bolker et Pacala des populations d'arbres visera à expliquer, et éventuellement à prévoir, les évolutions d'une population d'arbres selon les paramètres donnés pour un individu dans un cadre écologique ou géographique donné. On pourra donc essayer de varier les paramètres entrés afin de voir quels sont les impacts des individus dans l'évolution de la population d'arbres.

1.2 Analyse des besoins fonctionnels

1.2.1 Spécifications fonctionnelles

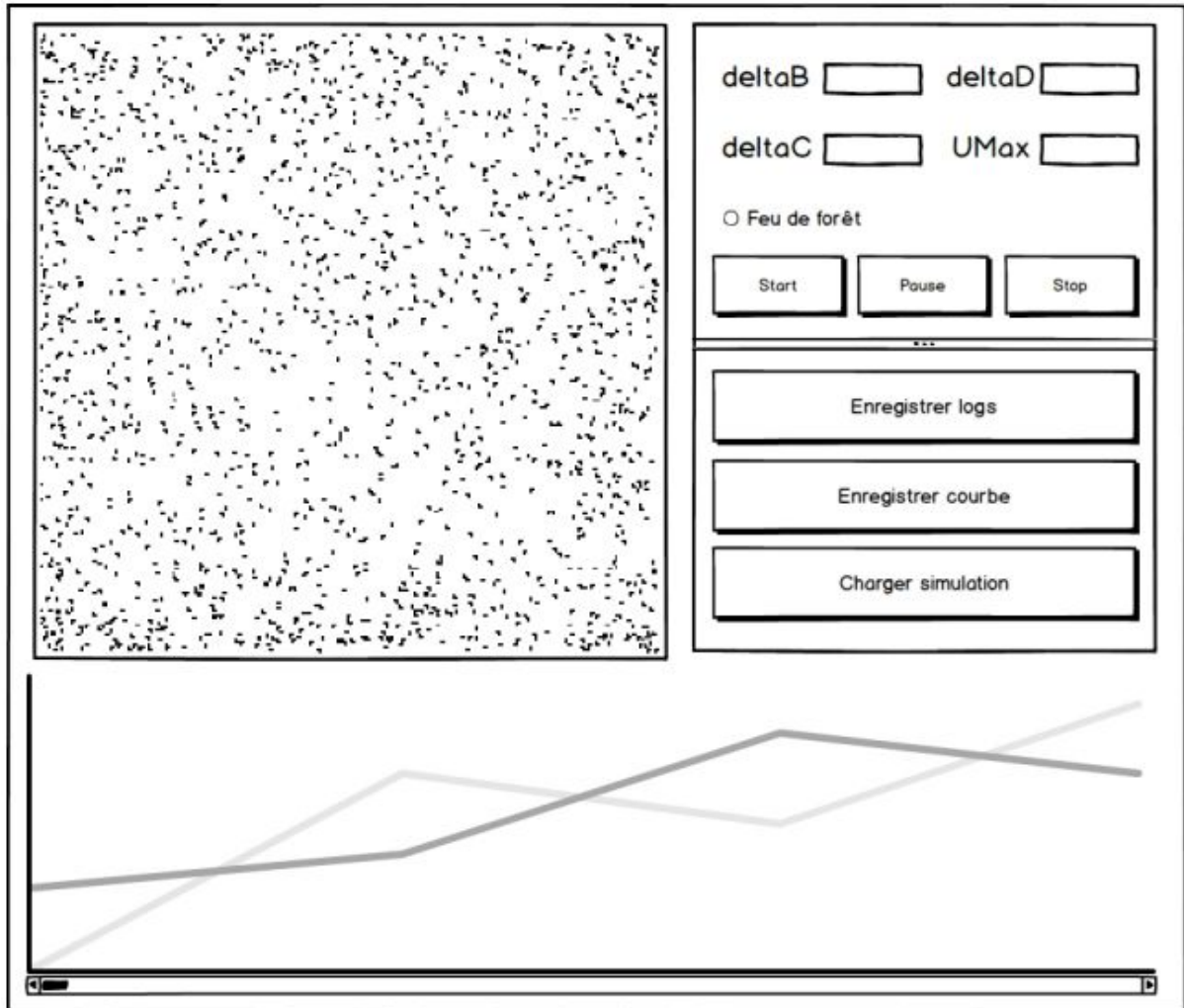


Figure 1 - Maquette de l'interface graphique

La réalisation de la maquette a été faite selon les critères du cahier des charges.

- Il y aura une fenêtre dans laquelle :
 - on pourra observer l'évolution des arbres au cours du temps sur une carte.
 - on pourra observer une courbe qui renseignera sur la population d'arbres en fonction du temps avec la moyenne donnée sur les 100 dernières unités de temps.
- On pourra lancer la simulation à partir d'un bouton "Start", après avoir rentré les paramètres.
- On pourra mettre en pause la simulation à partir d'un bouton "Pause".
- Il sera aussi possible d'enregistrer la simulation ou bien de charger une autre simulation.

1.2.2 Diagrammes d'analyse

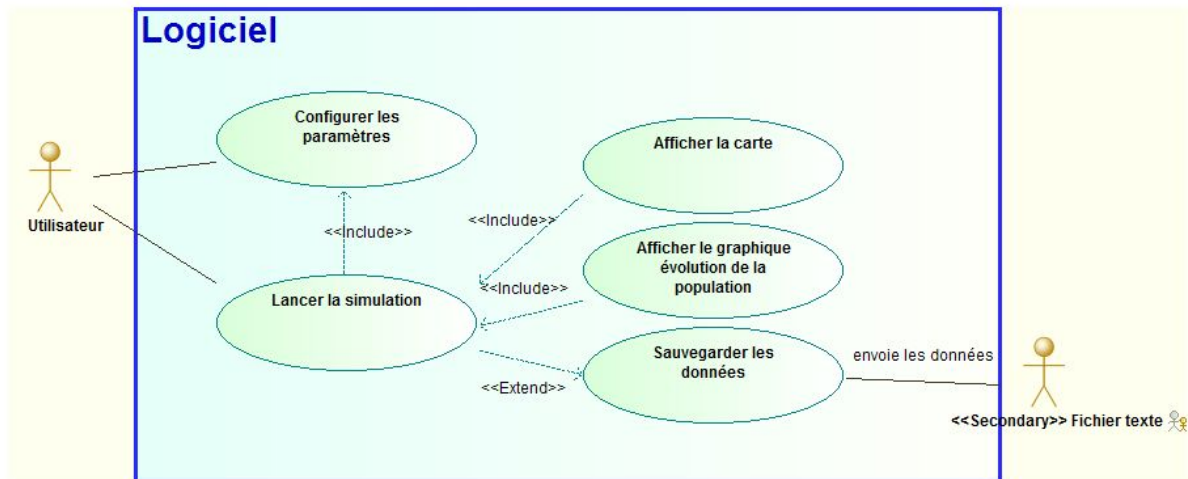


Figure 2 - Diagramme de cas d'utilisation

L'utilisateur devra configurer les paramètres pour lancer la simulation. Lorsqu'il lancera la simulation, le logiciel affichera la carte de la population d'arbres en temps réel et aussi le graphique de l'évolution de la population d'arbres en fonction du temps. De plus, tant que la simulation tourne, il pourra sauvegarder les données de la simulation.

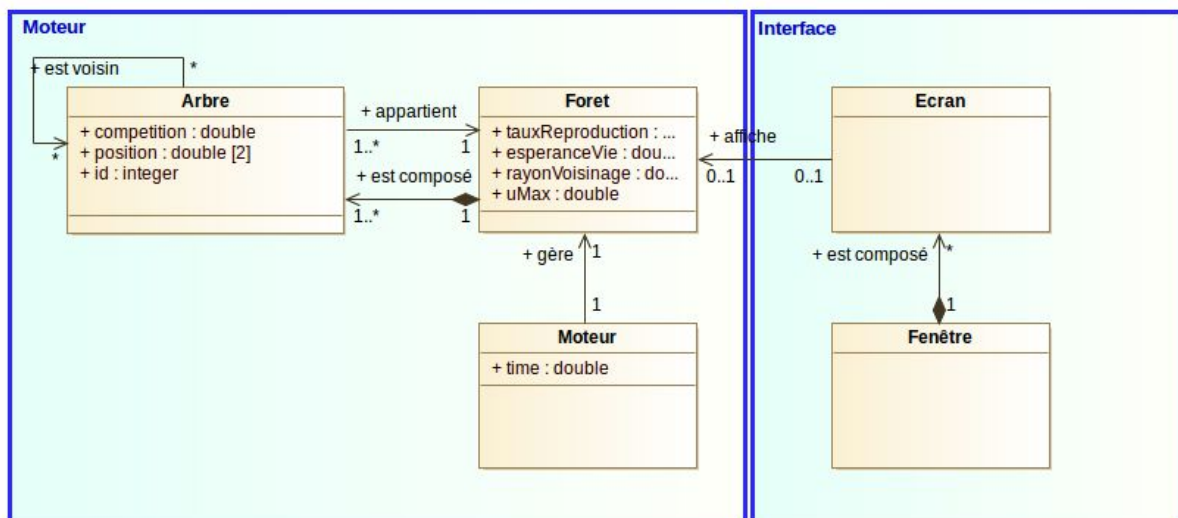


Figure 3 - Diagramme de classe

Une forêt est composée d'arbres, un arbre appartient à une forêt, un arbre est voisin d'un autre arbre, le moteur gère la forêt, la fenêtre est composée d'un écran qui affiche la forêt. La forêt a un taux de reproduction, la durée de vie moyenne d'un arbre, un rayon de voisinage, un rayon de dispersion. Un arbre a une position en x et en y, une compétition avec ses voisins.

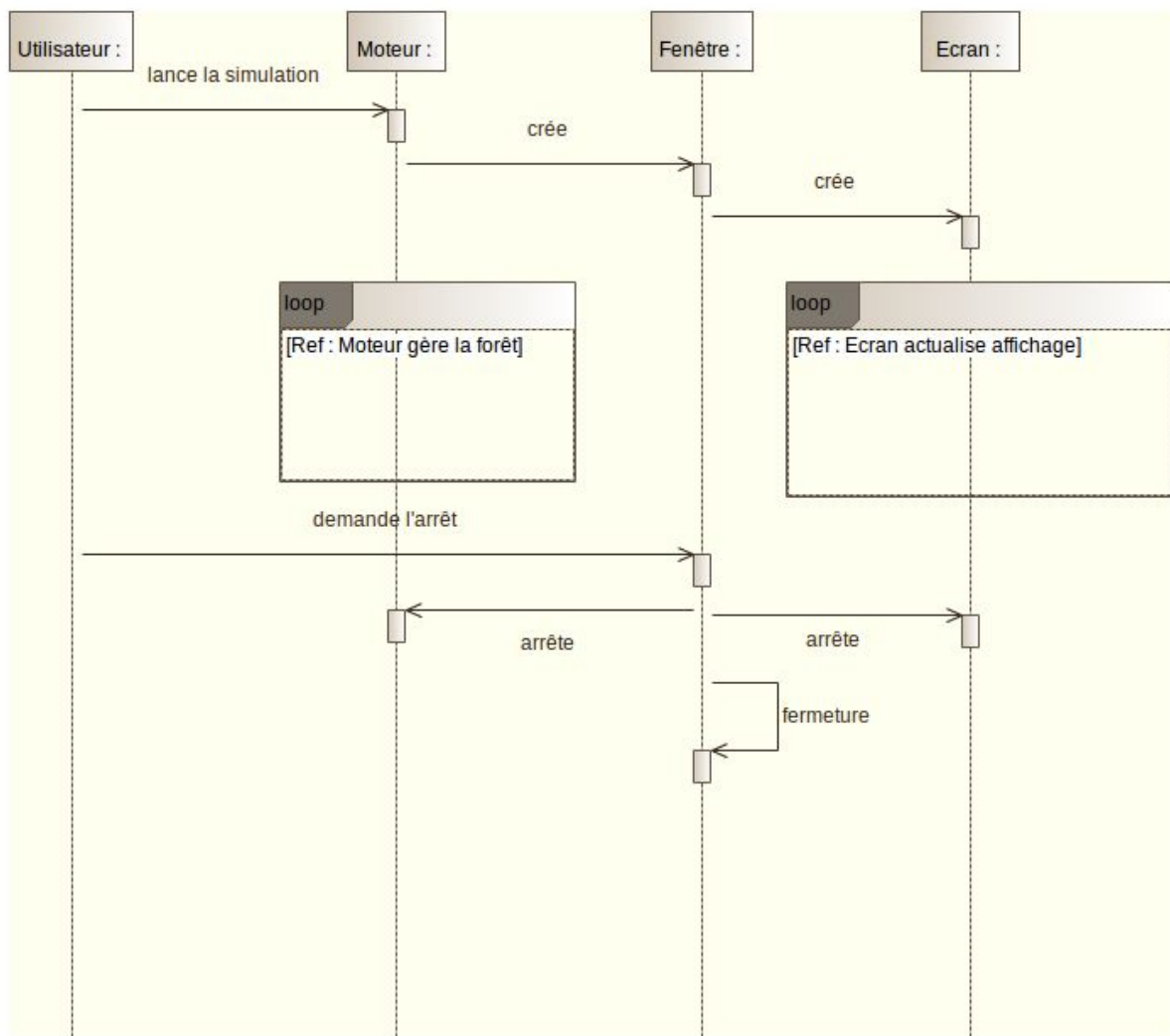


Figure 4 - Diagramme de séquence lorsque l'utilisateur lance la simulation

Lorsque l'utilisateur lance la simulation, le moteur crée une fenêtre qui va créer un écran. Le moteur va donc gérer la forêt et l'écran va afficher ce que le moteur lui envoie comme données. Lorsque l'utilisateur demande l'arrêt, la fenêtre demande au moteur et à l'écran d'arrêter et la fenêtre se ferme.

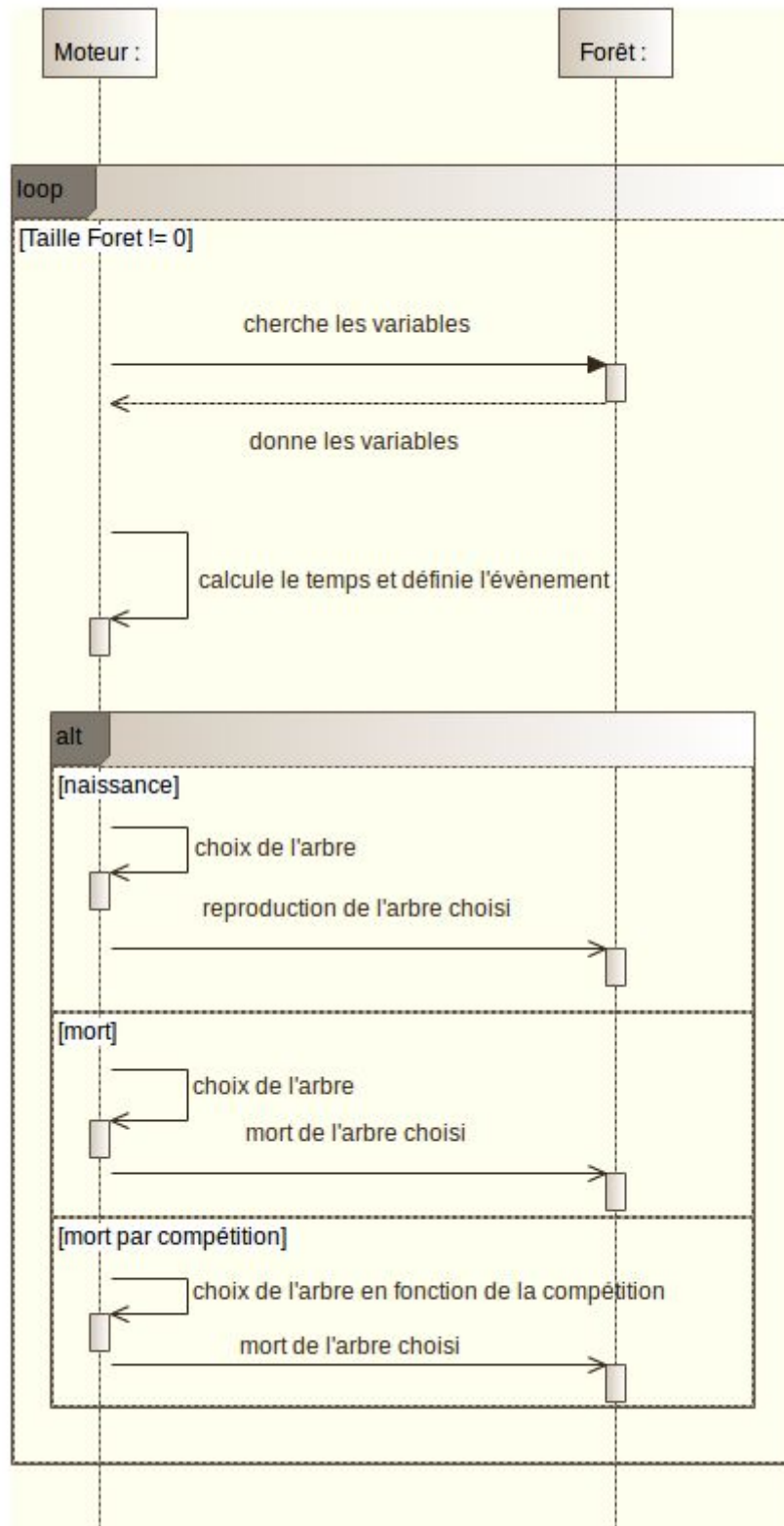


Figure 5 - Diagramme de séquence lorsque le moteur gère la forêt

Tant qu'il y a encore des arbres, le moteur fait tourner l'algorithme qui va choisir si un arbre va naître **ou** si un arbre va mourir **ou** si un arbre meurt par compétition, et cela pour chaque unité de temps.

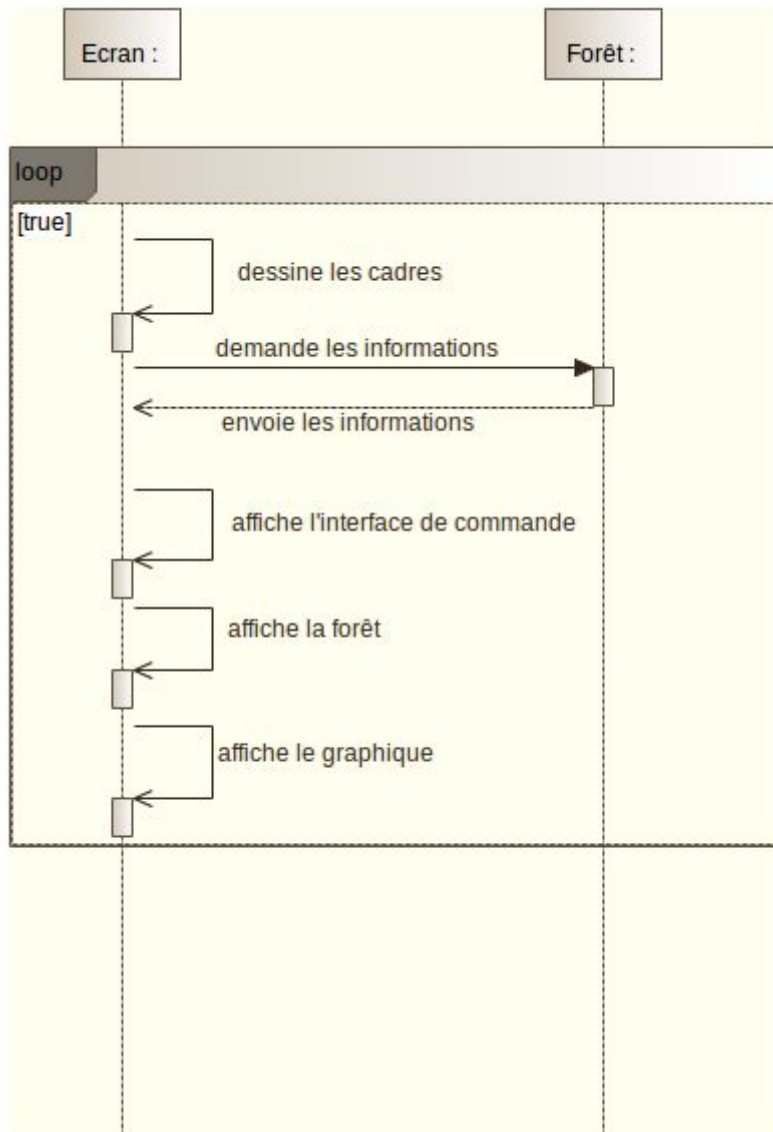


Figure 6 - Diagramme de séquence lorsque l'écran actualise l'affichage

Tant que la simulation tourne, l'écran dessine les cadres et demande les informations, la forêt envoie les informations, l'écran affiche l'interface de commande, la forêt et le graphique de l'évolution de la population d'arbres.

1.3 Analyse des besoins non fonctionnels

La réalisation du programme devra être faite soit en python soit en java soit en python et en java. De plus, on pourra éventuellement recourir à l'utilisation d'une librairie scientifique et d'une librairie graphique. Le moteur devra implémenter l'algorithme fourni.

2 Rapport technique

2.1 Conception

Après le premier rendez-vous avec notre tuteur, qui a permis une compréhension plus claire du sujet et de ses limites, la phase d'analyse a débuté. Elle a compris la réalisation de diagrammes de cas d'utilisation, de classes, de séquence pour l'écran, le moteur et la simulation.

Nous nous sommes ensuite consacrés à la programmation en elle-même. C'est le langage Java qui a été choisi car il était connu de tous les membres du groupe. De plus, l'utilisation du package **javax.swing*** a permis de tirer le maximum du potentiel du langage. Pour créer une interface, correspondant à notre idée initiale, nous avons utilisé le package **java.awt***.

Concernant le travail en groupe, la répartition des tâches et les échanges entre les membres se sont respectivement faits sur Skype. Les fichiers et les sources ont été mis en communs sur Google Drive, puis sur le SVN pour ceux à présenter à notre tuteur.

L'algorithme `run()` de la classe Moteur: il s'agit ici de l'algorithme principal, celui qui gère la simulation. Il va créer une forêt à partir d'une configuration prédéfinie ou attendre celle de l'utilisateur. Une fois la simulation lancée, il va créer les arbres puis gérer la disparition ou l'apparition d'arbre, en fonction des contraintes apportées par les paramètres de la simulation. Cette fonction va aussi limiter le nombre d'action par secondes à celui indiqué par l'utilisateur. Enfin, c'est `run()` qui va permettre de redémarrer la simulation (`reset`), de la mettre en pause (`pause`) ou de sortir les résultats sur un fichier texte (`getData`). (cf. Annexes)

Lors du lancement du programme, c'est d'abord l'objet moteur qui est créé. Ce dernier entraîne le lancement de `moteur.run()`. Ce sera le coeur du programme. Cette méthode va créer une forêt et la configurer en fonction des paramètres de base/entrés par l'utilisateur. La classe `Frame` est ensuite appelée et construit la fenêtre qui accueillera l'interface. Cette interface est créée par la classe `Screen`, disposant elle aussi de sa méthode `run()`. A ce stade, la simulation peut démarrer et l'interface affiche l'évolution de la population au cours du temps.

Une fois démarrée, la simulation se déroule automatiquement, attendant d'être mise en pause ou arrêtée. C'est la classe `Screen` qui actualise l'affichage à l'aide de sa méthode `repaint()` appelant elle-même la méthode `paintComponent()`.

L'arborescence de fichiers du programme se découpe en deux principaux dossiers : "bin" et "src". Le premier contient les exécutables `.class`. Le second, quant à lui, contient un

dossier “main” et “test” contenant les dossiers “resources” et “java” pour l’un, et les dossiers “resources” et “test” pour l’autre. Dans “/src/main/resources” comme dans “/src/test/resources”, se trouvent les dossiers “img” et “conf” contenant respectivement les images nécessaires à l’interface graphique et le fichier de configuration initiale. Les dossiers finaux “test” et “java” contiennent quant à eux les sources des tests et du programme en lui-même.



Figure 7 - Arborescence des fichiers

2.2 Résultats

La simulation affichera des résultats variant avec les paramètres entrés par l'utilisateur. Lors de l'utilisation, s'il a été décidé d'un taux de compétition non nul, l'évolution de la population suivra le **modèle logistique***, sa croissance étant limitée par une **capacité biotique***. Capture ci-dessous :

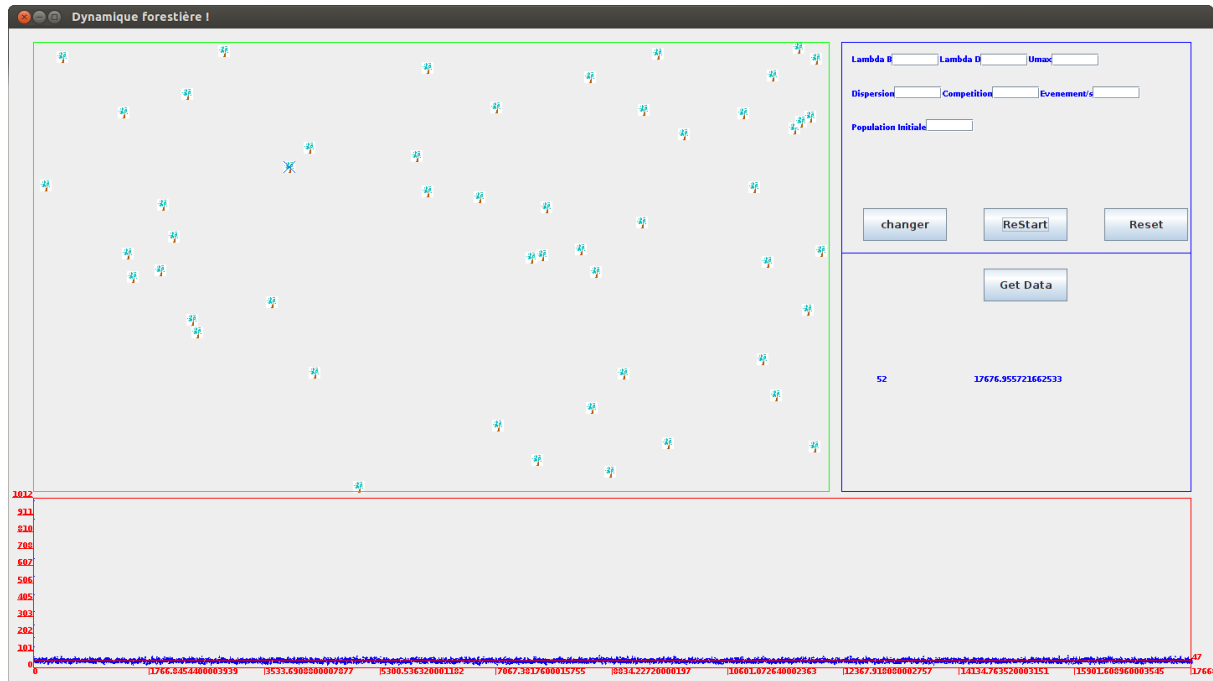


Figure 8 - Exemple modèle logistique

Si l'utilisateur souhaite l'absence de compétition, la croissance (ou dépression) démographique suivra le **modèle malthusien***, ne s'arrêtant jamais de croître (ou s'arrêtant lorsque tous les individus ont disparus). Capture ci-dessous :

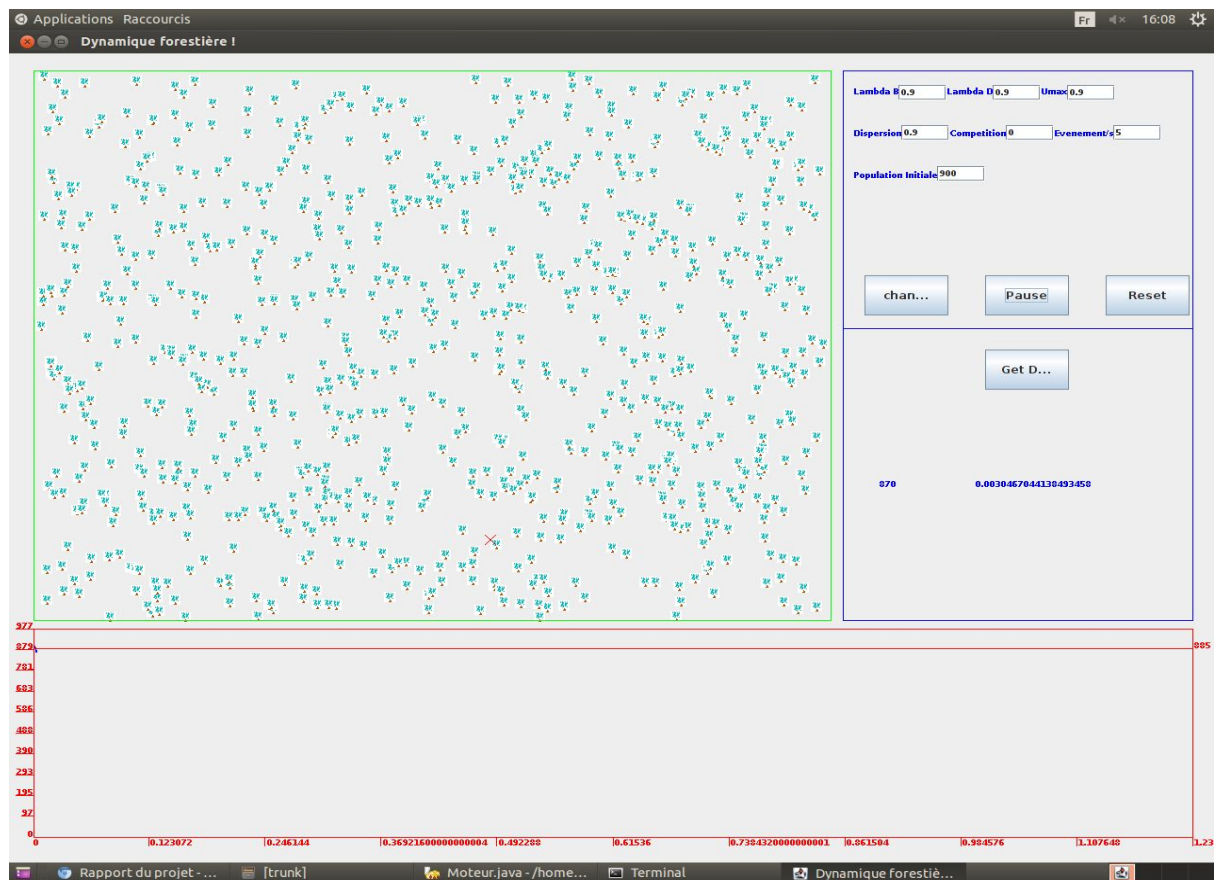


Figure 9 - Exemple modèle malthusien

2.3 Perspectives

Notre application étant fonctionnelle, il reste néanmoins des imperfections et des points à améliorer. Graphiquement, l'interface pourrait être plus attrayante et ergonomique. En effet, les champs de modification de paramètres ne sont pas alignés et les boutons ne contiennent pas l'ensemble du texte action. Sur la partie de l'affichage de la forêt en temps réel, les images d'arbre font perdre en précision, un choix qui a été basé sur l'idée que cet affichage n'était que peu utile à l'exploitation des données, son but étant uniquement une visualisation de l'effet de niche.

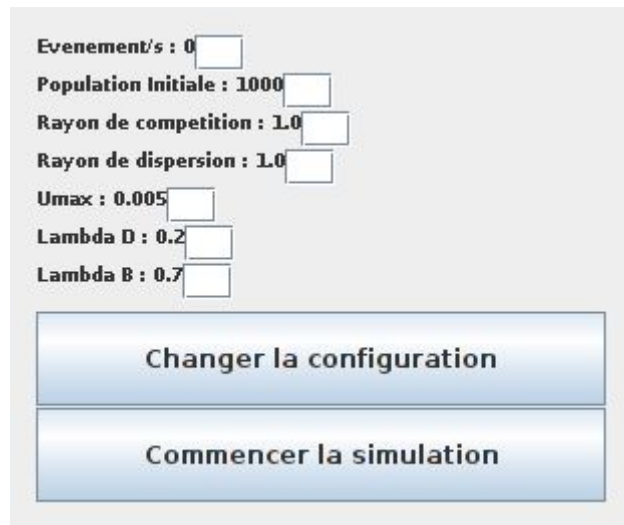
Il serait aussi possible de préciser encore les interactions entre les individus : les facteurs de compétition, naissance et mort pourraient être évolutifs, éventuellement suivant un cycle (phénomène saisonnier). La compétitivité pourrait aussi être comprise entre deux bornes : une compétitivité minimale entre tout individu et une maximale, atteinte avec la proximité de deux arbres. Ainsi, cette idée de compétition deviendrait moins binaire (soit l'arbre est trop loin pour être affecté, soit il est affecté par un taux fixe).

La naissance d'un arbre est actuellement immédiate : si la simulation provoque la naissance d'un nouvel individu, celui-ci apparaîtra à l'endroit prévu, autour de l'arbre ayant prévu la graine, et aura déjà atteint l'âge adulte. Une évolution possible serait de faire varier les chances de naissance de l'individu après émission de la graine et de lui faire suivre un cycle de croissance faisant varier son impact sur la compétitivité et ses taux de natalité/mortalité.

Ces axes sont des exemples de modifications du programme, approchant un peu plus la simulation du cas réel, mais dépassant le cadre des objectifs du projet initial.

3 Manuel d'utilisation

Voici le menu de lancement :



Evenement/s : 0

Population Initiale : 1000

Rayon de competition : 1.0

Rayon de dispersion : 1.0

Umax : 0.005

Lambda D : 0.2

Lambda B : 0.7

Changer la configuration

Commencer la simulation

Figure 10 - Menu de lancement

On peut rentrer des paramètres, ou directement lancer la simulation avec des paramètres déjà présents à gauche des cases. Ces valeurs se trouvent dans le fichier conf.txt.

Quand on lance la simulation on obtient :

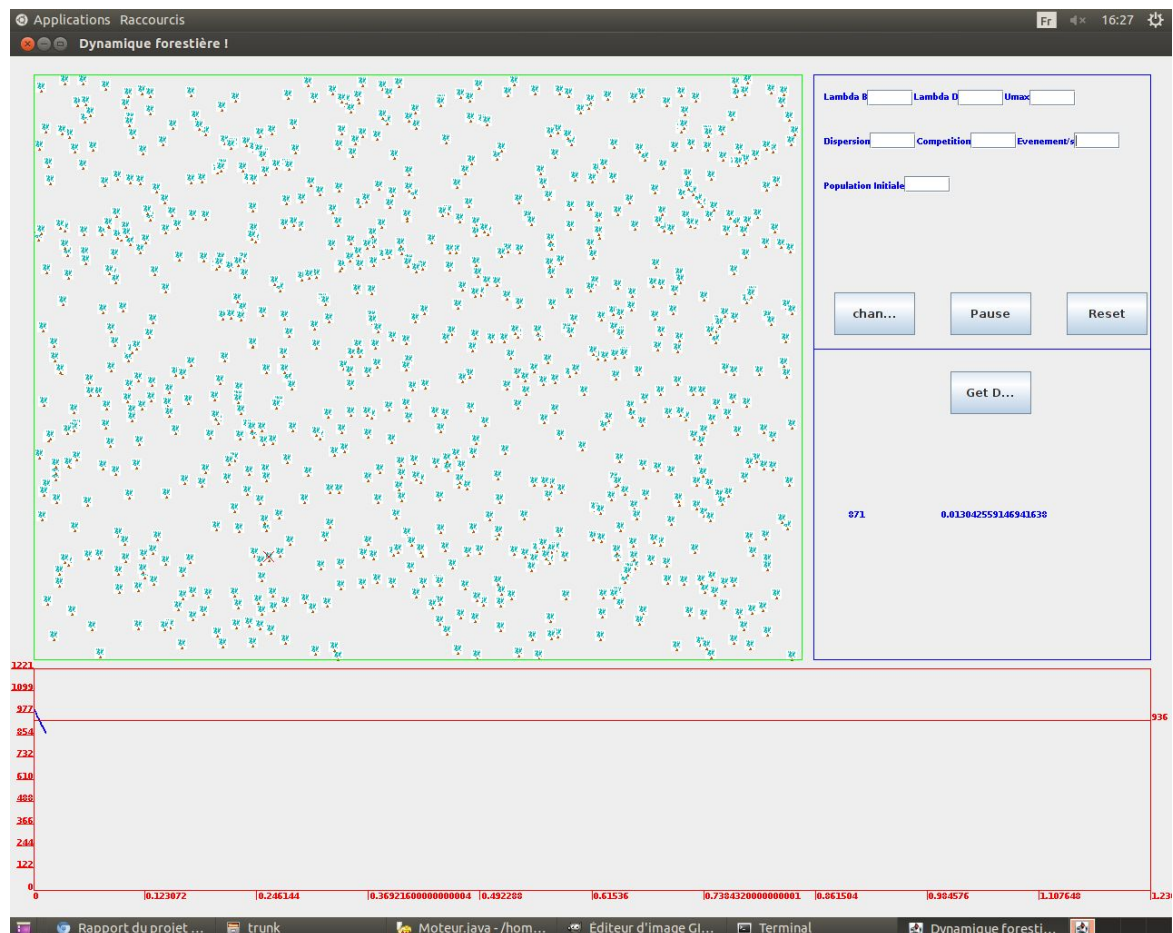


Figure 11 - Capture simulation lancée

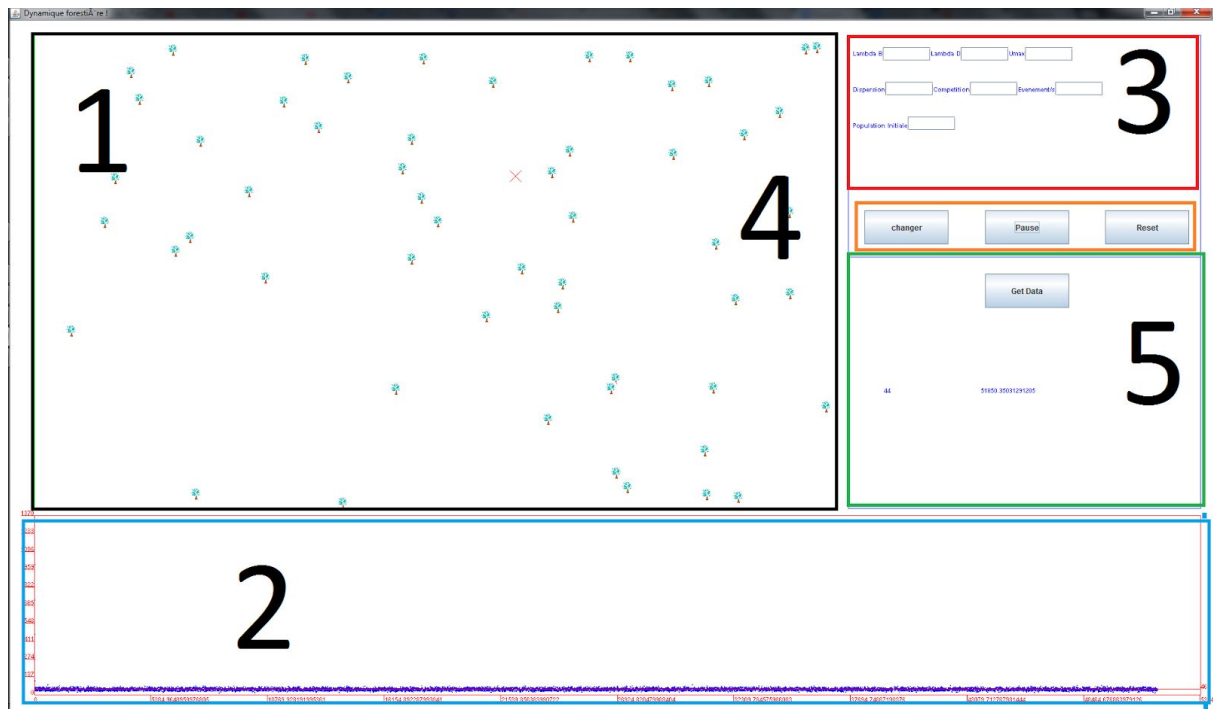


Figure 12 - Anatomie de l'interface

On peut distinguer 5 parties lorsque la simulation est lancée :

- 1) La carte représentant la population d'arbres :

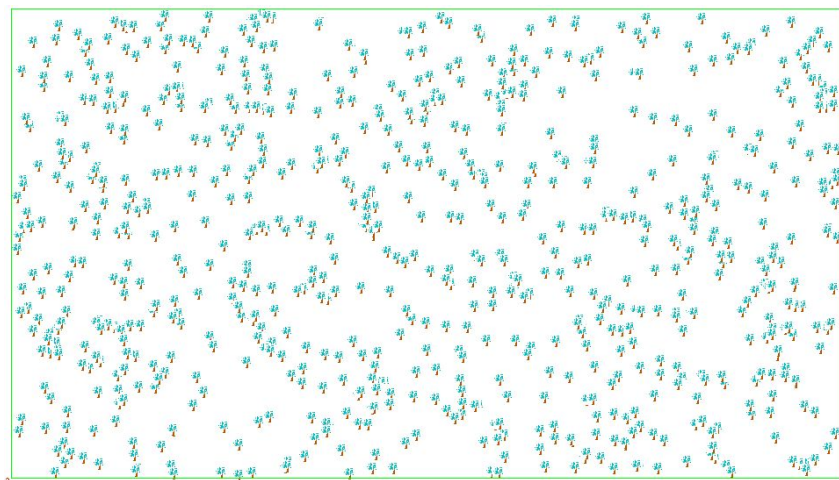


Figure 13 - Carte de la forêt

- 2) La courbe représentant la population d'arbres ainsi que la moyenne :



Figure 14 - Graphique de la population en fonction du temps

3) Des cases pour modifier les paramètres ;

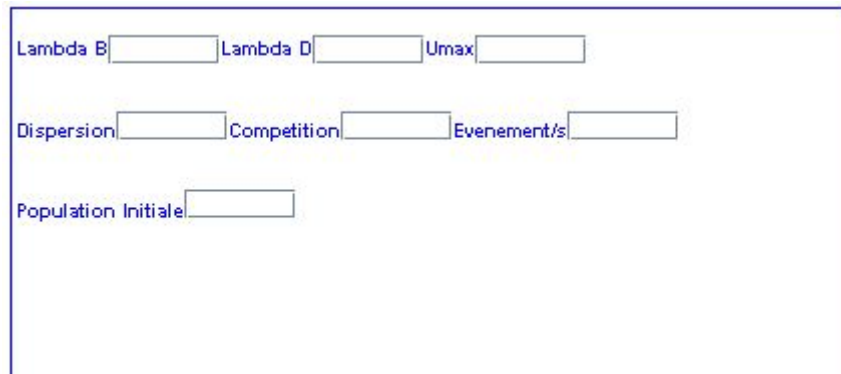


Figure 15 - Interface d'entrées des paramètres

4) Une série de 3 boutons : Changer, Pause, Reset, le bouton Pause est remplacé par Re Start quand la simulation est en pause.



Figure 16 - Boutons de l'interface

5) Une zone avec un bouton : GetData, qui permet de récupérer l'historique de la population à tous les pas de temps. Le nombre d'arbres présents sur la carte est affiché en bas à gauche et le temps qui est écoulé est affiché en bas à droite.



Figure 17 - Boutons de l'interface

Si on veut changer les paramètres pendant que la simulation est lancée, il suffit de rentrer les nouvelles valeurs dans les cases appropriées puis de cliquer sur “CHANGER”. Le programme changera les valeurs des paramètres, mais sur la courbe on pourra toujours voir les anciens résultats (avant la modification des valeurs) mais les nouveaux points qui apparaîtront proviendront de la simulation avec les nouveaux paramètres.

Pour faire une nouvelle simulation, il suffit de mettre le programme en pause avec le bouton “PAUSE”, faire le changement de valeurs, cliquer sur “CHANGER” puis “RESET” et relancer en cliquant sur “RE START”.

Pour pouvoir récupérer les données de la courbe, un bouton “GET DATA” est présent. Si vous cliquez dessus, le fichier data.txt existant sera écrasé avec les nouvelles valeurs, s’il n’existe pas alors un fichier data.txt sera créé dans “/src/test/resources”.

Voici à quoi ressemble le fichier data.txt :

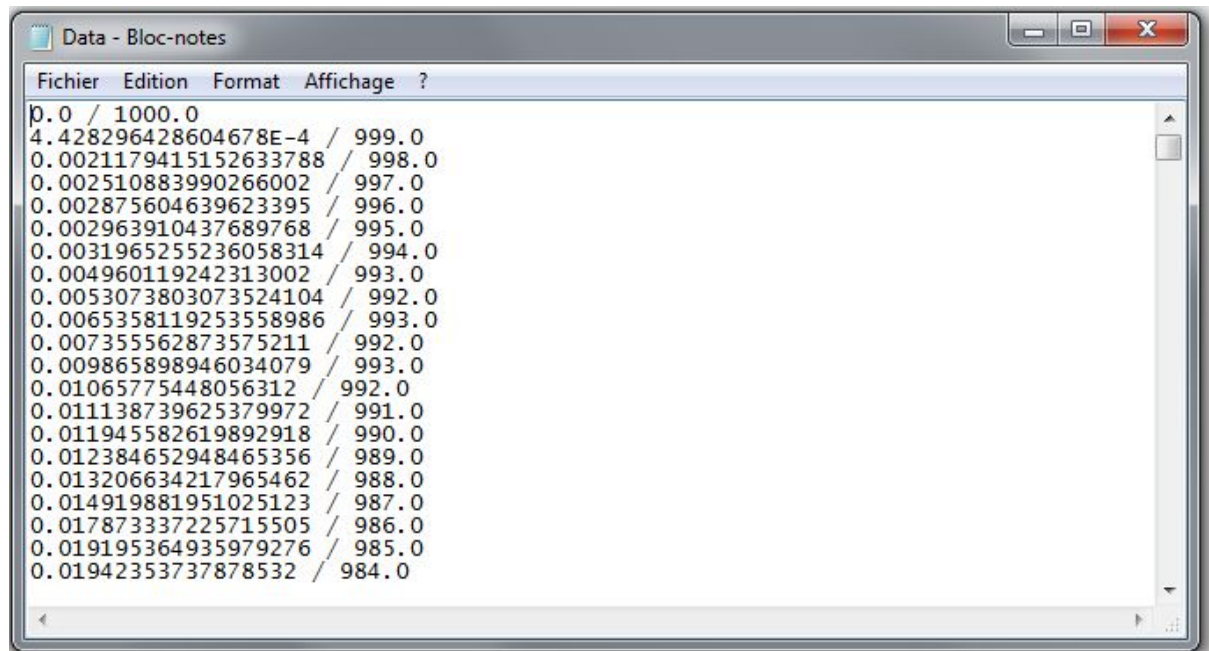


Figure 18 - Fichier data.txt

Pour changer les paramètres qui sont enregistrés de base lorsqu’ on lance le programme, il faut ouvrir le fichier conf.txt se trouvant dans “/src/main/resources/conf”. Ensuite il faut modifier le paramètre désiré tout en suivant l’ordre indiqué en commentaire.

Voici le fichier conf.txt :

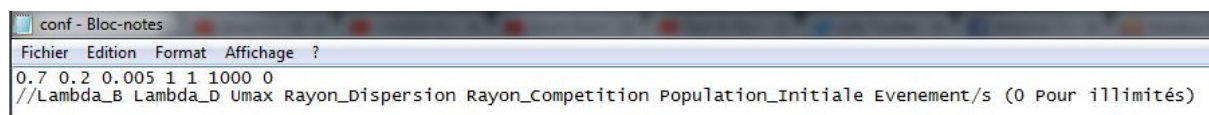


Figure 19 - Fichier conf.txt

4 Rapport d'activité

4.1 Méthode de développement

Le tuteur avait un projet bien défini sur les objectifs à faire, nous avons donc décidé de prendre une méthode de développement en cascade car cela semblait plus approprié dans le cadre de ce projet. Nous avons donc établi, dès la fin du premier rendez-vous, un cahier des charges afin de voir quelles étaient les principales contraintes de ce projet.

4.2 Planification

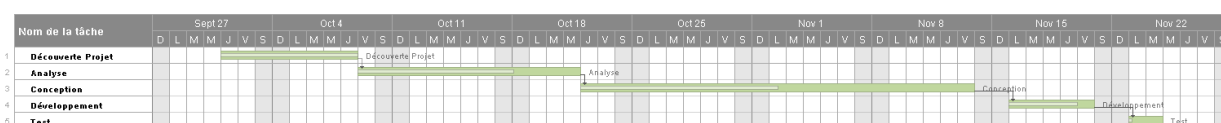


Figure 20 - Planning prévisionnel

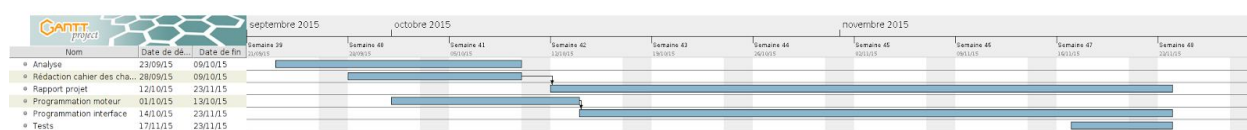


Figure 21 - Planning réel

Nous pensions au début qu'il fallait faire étape par étape mais nous avons d'abord pris conscience des objectifs et du contexte du projet afin de pouvoir commencer l'analyse. Mais comme il fallait commencer à programmer pour pouvoir comprendre quelles étaient les possibilités des solutions, nous avons débuté la programmation en parallèle la partie moteur qui était la base du code source. Lorsque nous avons fini la phase d'analyse, le tuteur nous a demandé de commencer le rapport tout en continuant de programmer l'interface graphique. Nous avons aussi commencer les tests des fonctions et du programme après avoir presque fini tout ce qui était interface graphique.

4.3 Méthodes et outils de travail

Le SVN nous a aidé à contrôler l'accès à des systèmes de gestion de code source mais aussi à faciliter la collaboration au sein du groupe de développement. Grâce au Google Drive, nous avons pu partager les documents ainsi que les différents diagrammes UML et surtout travailler ensemble, en même temps, sur le rapport. Et dans le but de communiquer, nous avons utilisé des messageries instantanées comme Skype et Facebook. Nous organisions aussi des réunions de travail avec le tuteur afin de faire le point sur ce qui allait ou bien n'allait pas. Cela permettait au tuteur de suivre notre avancée mais aussi de nous donner des conseils.

Conclusion

Les objectifs de ce projet de semestre 3 “Dynamique Forestière” été de développer un programme de simulation avec un moteur et une interface graphique ainsi que de fournir un outil de gestion qui simule l’évolution d’une population d’arbres.

Les résultats obtenus au regard des attentes initiales ne sont pas si différents de ceux attendus. Nous avons respecté la maquette que nous avons faite lors de la conception. La programmation de l’outil de gestion et la mise en place de l’algorithme de Bolker et Pacala du projet nous amènent tout de même à penser qu’il reste des perspectives d’évolution comme l’amélioration de l’interface graphique ou encore le changement du modèle de “dissémination de graines” ou encore une compétition calculée différemment entre les arbres.

Malgré une méthode de développement en cascade, ce projet nous a tout de même permis d’acquérir plus d’expérience en termes de méthodes de travail. Nous avons pu tous travailler ensemble grâce au SVN, dans lequel ce système de gestion de code source nous a permis de diviser les tâches de travail. Les documents comme les différents diagrammes UML et le rapport ont été partagés grâce au Google Drive. Cela nous a permis de travailler ensemble simultanément sur le rapport. Mais nous avons aussi compris qu’il fallait commencer à coder un minimum afin de pouvoir comprendre quelles étaient les solutions pour atteindre l’objectif principal.

Cela nous a aussi permis d’acquérir des connaissances techniques surtout pour la réalisation de l’interface graphique avec **java.awt**. Sinon, nous avons pu avoir la relation entre la modélisation mathématique et la programmation. D’une part pour l’implémentation de l’algorithme de Bolker et Pacala mais aussi d’un autre côté pour l’observation des données, avec la cohérence entre les paramètres entrés pour un individu et l’évolution de la population d’arbres.

Nous concluons sur le fait que le projet a été intéressant non seulement sur la manière dont nous avons pris en main le sujet mais aussi sur l’aspect de la programmation d’un outil de gestion avec une interface graphique.

Les références bibliographiques et sitographiques

- <https://www.agroparistech.fr/IMG/pdf/Desassis.pdf>
- <http://halieutique.agrocampus-ouest.fr/afh/forum11/Presentation/Lambert.pdf>
- <http://liris.cnrs.fr/Documents/Liris-3717.pdf>
- http://www.cmap.polytechnique.fr/~ecolemathbio2012/Notes/La_Londe_09_2012.pdf
- <http://inra.dam.front.pad.brainsonic.com/ressources/afile/231489-a146d-resource-caq12-10.html>
- <http://www.futura-sciences.com/magazines/matiere/infos/dico/d/matiere-modelisation-11321/>
- <ftp://ftp-sop.inria.fr/members/Fabien.Campillo/publications/campillo2013d.pdf>
- <https://tel.archives-ouvertes.fr/inria-00502537/document>
- <https://perso.univ-rennes1.fr/jean-sebastien.pierre/cours/DDP%20M1%202010.pdf>

Annexes

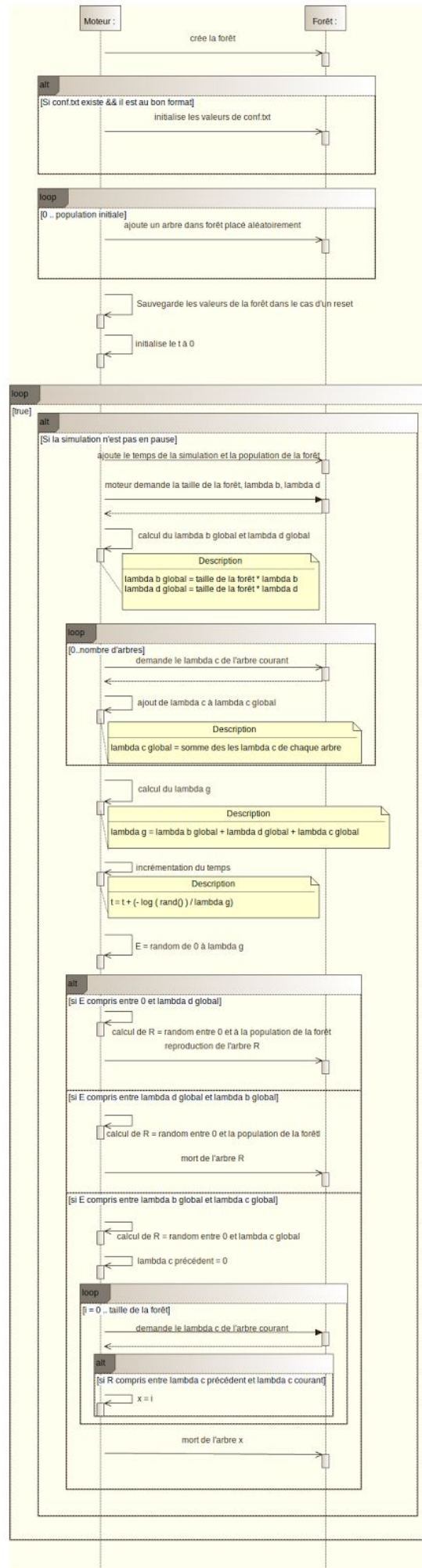


Figure 22 - Diagramme de séquence de l'algorithme

Quatrième de couverture

Dynamique forestière est un projet qui implique d'implémenter un algorithme dans un outil de gestion qui simule l'évolution d'une population d'arbres. L'algorithme met en oeuvre un modèle mathématique qui se concentre sur un individu d'une population, il a été fait par Bolker et Pacala. La simulation commence lorsque l'utilisateur entre les paramètres appliqués pour un seul arbre, elle affiche la population d'arbres sur un Tore, l'évolution de la population à l'aide d'un graphique. De là, l'utilisateur pourra décider : de changer les paramètres, de mettre en pause la simulation ou d'enregistrer les données de la simulation. Le projet utilise principalement le langage Java.

Mots clés : Java, Modèle mathématique, Algorithme, Simulation, Arbres, Bolker et Pacala, Individu, Population

“Forest dynamics” is a project involving the implementation of an algorithm within an interface which simulate the evolution of the population of a forest. The algorithm uses a mathematical model, which focuses on an individual in a population, it was originally made by Bolker and Pacala. The simulation starts when the user enters the different parameters for a single tree, it displays the population of the forest in a toroid and the evolution of the population in a graph. From there, the user is able to decide if he wants to change the parameters, to put the simulation on pause or to save the data collected by the simulation. The project was made in the programming language Java.

Keywords : Java, Mathematical Model, Algorithm, Simulation, Tree, Bolker and Pacala, Individual, Population