Conception d'un module de visualisation pour un simulateur parcelle/troupeau

Lucie Masson, Alice Villéger

Institut Supérieur d'Informatique de Modélisation et de leurs Applications

COMPLEXE DES CEZEAUX
BP 125 – 63170 AUBIERE CEDEX

Institut National de la Recherche Agronomique

Unité de recherches sur les herbivores
THEIX
63122 ST-GENES-CHAMPANELLE

Rapport de projet 2eme année

Conception d'un module de visualisation pour un simulateur parcelle/troupeau

Présenté par : Lucie Masson et Alice Villéger

De septembre à mars 2001

Responsables du projet :
Laurent Perochon
Claude Mazel
David Hill

Institut Supérieur d'Informatique de Modélisation et de leurs Applications

COMPLEXE DES CEZEAUX
BP 125 – 63170 AUBIERE CEDEX

Institut National de la Recherche Agronomique

Unité de recherches sur les herbivores
THEIX
63122 ST-GENES-CHAMPANELLE

Rapport de projet 2eme année

Conception d'un module de visualisation pour un simulateur parcelle/troupeau

Présenté par : Lucie Masson et Alice Villéger

De septembre à mars 2001

Responsables du projet :
Laurent Perochon
Claude Mazel
David Hill

REMERCIEMENTS

Nous tenons à remercier David Hill pour ses précieux conseils et son aide dans la réalisation du projet, Laurent Pérochon pour son encadrement rigoureux, Christine Force pour sa bonne humeur, Claude Mazel pour ses idées originales, et toute l'équipe de l'INRA pour le temps qu'ils ont bien voulu consacrer à répondre à nos questions.

TABLE DES ABBREVIATIONS

AIX	: Système d'exploitation de type UNIX pour les serveurs IBM
C++	: Langage de programmation orienté objet, dérivé du C
HP-UX	: Système d'exploitation de type UNIX pour les serveurs HP
INRA	: Institut National de la Recherche Agronomique
ISIMA	: Institut Supérieur d'Informatique, de Modélisation et de leurs Applications
LINUX	: Système d'exploitation de type UNIX libre et gratuit
STL	: Standard Template Library
UML	: Unified Modelling Language
UP	: Unified Process
Xlib	: Librairie graphique standard des systèmes d'exploitation de type UNIX

TABLES DES FIGURES ET ILLUSTRATIONS

FIGURE 1 : CE QUE VOIT L'ANIMAL (VUE DETAILLEE) 15

FIGURE 2 : VUE INTERMEDIAIRE CENTREE SUR UN ANIMAL 15

FIGURE 3 : VUE INTERMEDIAIRE CENTREE SUR UNE ZONE 16

FIGURE 4 : VUE INTERMEDIAIRE DES DIFFERENTS CHOIX POSSIBLES 16

FIGURE 5 : DIAGRAMME DE CLASSES REPRESENTANT LA HIERARCHIE DES VUES 19

FIGURE 6 : DIAGRAMME DE CLASSES REPRESENTANT LA HIERARCHIE DES OBJETS GRAPHIQUES 20

FIGURE 7 : LA LEGENDE 20

FIGURE 8 : DIAGRAMME DE CLASSES DES LECTEURS DE FICHIERS DE CONFIGURATION 21

FIGURE 9 : DIAGRAMME DES CLASSES DES LECTEURS DE FICHIERS TRACE 22

FIGURE 10 : DIAGRAMME DE CLASSES REPRESENTANT LA VUE DETAILLEE 23

FIGURE 11 : DIAGRAMME EN SEQUENCE DE LA PHASE D'INITIALISATION DE VUEDETAILLEE 24

FIGURE 12 : DIAGRAMME D'ACTIVITES DE LA GESTION DU CLAVIER 25

FIGURE 13 : COPIE D'ECRAN 26

RESUME

Le but de ce projet est de réaliser un logiciel de visualisation des résultats fournis par le simulateur parcelle/troupeau. Ce simulateur, encore en phase de conception à l'INRA, doit permettre de simuler le comportement d'un troupeau de ruminants sur des parcelles en conditions extensives. Sa structure peut être décomposée en trois parties : le module A, ou module animal, qui s'occupe du comportement individuel d'un ruminant, le module V, ou module végétal, qui s'occupe de l'évolution de la végétation et enfin le module S, ou module spatial/social, qui gère le comportement grégaire des animaux.

Notre logiciel de visualisation, appelé *visionneuse*, est un module externe de ce simulateur n'ayant à effectuer aucun calcul, statistique ou autre, mais à afficher de façon facilement compréhensible pour un être humain et agréable à l'œil les résultats que lui fournit le simulateur.

La première étape a été de réaliser le cahier des charges de ce programme. Après discussion avec les biologistes sur ce qu'ils désiraient voir apparaître à l'écran, il a été décidé de diviser le module visionneuse en trois sous modules :

- échelle 1 ou vue détaillée : pour étudier le comportement d'un animal particulier pendant une phase d'ingestion.
- échelle 2 ou vue intermédiaire : pour étudier le comportement d'un ou plusieurs animaux sur une partie restreinte de la parcelle, voire pour étudier les différentes possibilités offertes à un unique animal à partir d'un état donné du système.
- échelle 3 : pour étudier l'évolution de la végétation sur la totalité de la parcelle et le comportement du troupeau de ruminants.

La conception de l'échelle trois, très ambitieuse pour un projet de 100 heures et dont la réalisation dépend de concepts non encore implémentés dans le simulateur, a été remise à plus tard.

La communication entre le simulateur et la visionneuse se fait par l'intermédiaire de fichiers trace que le simulateur écrit sur le disque au cours de ses calculs. L'idée de faire fonctionner en parallèle simulateur et visionneuse en les faisant communiquer grâce à des buffers a été abandonnée du fait que le simulateur n'est pas encore en état de fonctionner.

La visionneuse, comme le simulateur, est réalisée en C++ sous Linux, elle utilise uniquement la librairie graphique Xlib, pour des raisons de portabilité et de pérennité du code.

A l'heure où sont rédigées ces lignes, la conception du sous module «échelle 1 » est terminée, il reste quelques perfectionnements à faire et quelques erreurs à corriger mais cela devrait être fini fin mars. L'échelle 2, encore en phase d'analyse, n'est pas encore codée, mais elle sera terminée, ainsi que l'échelle 3 si sa réalisation est encore d'actualité, lors du stage de Lucie d'avril à septembre 2001.

Mots clefs : simulateur parcelle/troupeau, visionneuse, Linux, Xlib, C++

ABSTRACT

This project is aimed at building a software displaying results provided by a pasture/herd simulator. The simulator, currently unfinished, is still being designed at INRA, and should be able to simulate the behaviour of a herd of ruminants on a pasture in extensive conditions. Its structure can be decomposed into three parts : the A module, or animal module, managing the individual behaviour of each animal, the V module, or vegetal module, managing the evolution of vegetation, and finally the S module, or spatial/social module, managing group behaviour.

Our visualisation software, named *visionneuse* (viewer), is an external module of the simulator, and doesn't have to perform any statistic operations, or any other kind of calculations : its sole purpose would be to display the results provided by the simulator in an easy to understand and aesthetically pleasing way.

The first step consisted in drawing up the specifications of the program. After some discussions with the biologists about what they wished to see appear on the screen, it was decided to divide the viewer module into three sub-modules :

- scale 1 or detailed view : studying the behaviour of a particular animal during an ingestion phase.
- scale 2 or in-between view : studying the behaviour of one or several animals on a limited portion of the pasture, or even studying the various possibilities offered to a single animal from a given situation.
- scale 3 : studying the evolution of vegetation on the whole pasture and the behaviour of a herd of ruminant.

Designing the third scale, quite an ambitious goal in a 100 hours project, would depend on concepts not even implemented in the simulator yet, so this part of the work had to be postponed.

Communication between simulator and viewer is performed through trace files written on disk by the simulator during its calculations. The idea of having the simulator and viewer running simultaneously and communicating through buffers had to be given up due to the fact the simulator isn't functional yet.

Following the example of the simulator, the viewer has been implemented in C++, running under Linux ; it uses the Xlib graphics library only in order to improve the portability and permanence of the source code.

Currently, the scale 1 sub module is fully designed and operational; a few points remains to get perfected, and a few mistakes to be corrected, but this should be finished by late March. Scale 2, still in analysis phase, hasn't been coded yet, but it will be completed (and so will scale 3 if its realisation is still to be considered) during Lucie's placement from April to September 2001.

Key words : pasture/herd simulator, viewer, Linux, Xlib, C++

TABLE DES MATIERES

REMERCIEMENTS **3**

TABLE DES ABBREVIATIONS **4**

TABLES DES FIGURES ET ILLUSTRATIONS **5**

RESUME **6**

ABSTRACT **7**

TABLE DES MATIERES **8**

INTRODUCTION **10**

CHAPITRE 1: CONTEXTE DU PROJET **11**

I. LE SIMULATEUR PARCELLE/TROUPEAU 11
1. Le module animal (Module A) *11*
2. Le module végétal (Module V) *11*
3. Le module spatial/social (Module S) *12*
II. L'ETAT ACTUEL DU SIMULATEUR 12
III. BESOIN D'UNE VISIONNEUSE 12
1. Validation du modèle *12*
2. Interface graphique *12*
3. Mise en valeur du simulateur *13*

CHAPITRE 2: CONCEPTION DE LA VISIONNEUSE **14**

I. REDACTION DU CAHIER DES CHARGES 14
1. Introduction *14*
2. Les trois vues *14*
a. Echelle 1 14
b. Echelle 2 15
c. Echelle 3 16
3. Dialogue Visionneuse/Simulateur *17*
II. IMPLEMENTATION 18
1. Description globale *18*
a. Organisation des vues 18
b. Objets graphiques 19
c. Les lecteurs de fichiers 21
2. La vue détaillée (échelle 1) *23*
a. Description 23
b. Fonctionnement 24

CHAPITRE 3: RESULTATS ET DISCUSSION **26**

I. RESULTATS 26
1. Plate-forme de travail *26*
2. Tests, copies d'écrans *26*

II. DISCUSSION 27
1. Déroulement du travail *27*
2. Raison des choix de limitation *27*
3. Avenir du projet *27*
CONCLUSION **28**
REFERENCES BIBLIOGRAPHIQUES **29**

INTRODUCTION

L'INRA, en collaboration avec l'ISIMA, est actuellement en train de concevoir un simulateur dit «simulateur parcelle/troupeau » dont le but est de reproduire de la façon la plus réaliste possible le comportement d'un troupeau de ruminants, ovins ou bovins, sur une parcelle.

La conception de ce simulateur très complexe, qui regroupe des connaissances à la fois sur le comportement individuel des ruminants, leur comportement grégaire et l'évolution de la végétation d'une parcelle en fonction du climat et des actions des animaux, a été divisée en plusieurs modules plus ou moins indépendants dont les actions seront synchronisées par un programme maître appelé noyau de simulation.

Notre objectif, dans le cadre de ce projet, est de réaliser un programme de visualisation graphique des résultats donnés par le simulateur. Ce logiciel, qui est un module totalement indépendant du simulateur, doit permettre de voir sous forme graphique, donc plus facile à comprendre pour un être humain, ce que fait le simulateur et les résultats de ses calculs, afin de pouvoir s'assurer du bon fonctionnement du simulateur et à terme afin de pouvoir plus facilement utiliser les résultats qu'il fournit. Le travail consiste à réaliser le cahier des charges de ce programme, puis à le concevoir.

Nous allons vous présenter de façon plus précise le fonctionnement et les objectifs du simulateur avant d'expliciter les raisons pour lesquelles il est nécessaire de concevoir un programme de visualisation, programme qui a pris le nom de *Visionneuse*. Ceci servira d'introduction à la description du travail effectué, qui comporte donc deux parties : l'une pour le cahier des charges, l'autre pour la conception de la visionneuse. Enfin, nous exposerons l'état d'avancement du projet et les résultats obtenus.

Chapitre 1: CONTEXTE DU PROJET

I. LE SIMULATEUR PARCELLE/TROUPEAU

A cause notamment de l'exode rural et d'une mauvaise gestion des prairies, des parcelles autrefois entretenues se dégradent et deviennent broussailleuses. Afin d'empêcher ce processus, il convient de trouver des méthodes d'exploitation efficaces, qui utilisent au mieux ces prairies en fonction du comportement des troupeaux de ruminants.

C'est pour résoudre ce problème que l'INRA a entamé la réalisation d'un simulateur parcelle/troupeau [1]. Celui-ci à pour but de modéliser le comportement de ruminants sur des parcelles extensives et hétérogènes, et ce sur de longues périodes de temps, afin d'en déduire les interactions entre les animaux et leur lieu de pâturage. A terme, l'étude des résultats fournis par ce simulateur devrait permettre d'anticiper l'évolution des parcelles et d'empêcher leur dégradation.

Les concepts utilisés dans la création de ce simulateur couvrent des domaines très divers, qui vont de l'étude de la végétation au comportement social d'un groupe de ruminant. C'est pourquoi il a été décidé de diviser le simulateur en trois grands modules distincts, s'occupant chacun d'un aspect spécifique du simulateur.

Voici une description très succincte de ces différents modules :

1. Le module animal (Module A)

Cette partie s'occupe de modéliser le comportement individuel de l'animal. Le bovin ou l'ovin peut en effet boire, manger, ruminer, se déplacer ou se reposer, l'animal simulé devant être aussi proche que possible qu'un animal réel. Ce module reprend pour une bonne part le modèle de Sauvant et al [2], étendu de façon à prendre en compte les spécificités du simulateur, et notamment la spacialisation du problème.

On distinguera, dans le comportement de l'animal, le repos long du repos court. Le repos long consiste en une période de repos prise en troupeau où l'animal se couche pour dormir, le repos court étant uniquement une brève période d'inactivité de l'animal pendant laquelle il ne fait rien ou se demande ce qu'il fera ensuite. On distinguera de même le déplacement court du déplacement long, ce dernier se faisant en troupeau et, comme son nom l'indique, sur de grandes distances, tandis que le déplacement court est un déplacement individuel entre deux endroits de défoliation.

2. Le module végétal (Module V)

C'est la partie qui simule l'évolution de la parcelle. Les végétaux qui forment la prairie croissent en fonction du climat, et sont défoliés par les animaux. La parcelle est divisée en cellules de forme hexagonales regroupées en sites et faciès, cellules mesurant un dixième de mètre carré (place occupée par un animal en train de brouter).

On appelle faciès un ensemble de cellules, pas forcément contiguës, ayant des caractéristiques communes de végétation. Des cellules appartenant au même faciès contiendront les mêmes espèces végétales, dans les mêmes proportions.

On appelle un site un ensemble de cellules contiguës appartenant au même faciès. On peut bien entendu trouver sur une parcelle plusieurs sites appartenant au même faciès.

3. Le module spatial/social (Module S)

Dans cette troisième partie, c'est de l'aspect social qu'il est question. Ce module s'occupe des interactions entre animaux, de leurs relations sociales et de leurs déplacements de groupes. En effet les ruminants ont un comportement grégaire, ils tendent à rester groupés autour d'un ou plusieurs animaux dominants et ne se déplacent qu'en troupeau, suivant le comportement d'un «leader».

II. L'ETAT ACTUEL DU SIMULATEUR

Comme on peut l'imaginer, la conception d'un tel simulateur est un projet long et ambitieux, qui demande de longues années d'étude et doit se baser sur des recherches scientifiques solides.

Sa réalisation n'en est qu'à ses débuts, et les différents modules, dont aucun n'est encore fini, sont à des stades de conception différents. En effet, si le module A, implémenté par François Guerry en 2000 [1], est déjà presque opérationnel, le modèle V est encore en conception, et l'implémentation du modèle S ne commencera que dans le courant de l'année 2001.

Enfin, lorsque ces trois modules, ainsi que le noyau de simulation qui coordonne le tout, seront développés, il faudra encore faire des tests afin de vérifier la validité des hypothèses utilisées et probablement modifier les modèles en conséquence.

III. BESOIN D'UNE VISIONNEUSE

Etant donné la complexité du simulateur, le faire s'occuper aussi de l'affichage ne ferait que l'alourdir et risquerait de provoquer des erreurs. De plus la gestion graphique ne fait pas à proprement parler partie de la simulation, et pour des raisons de maintenance il est préférable de diviser les tâches en différents modules plus ou moins indépendants. C'est pourquoi la visionneuse est un module à part, totalement indépendant du simulateur qui lui fourni les informations à afficher.

1. Validation du modèle

A court terme, il apparaît comme indispensable de disposer d'une série d'outils assez spécifiques de visualisation pour procéder à la validation du modèle de simulation.

De tels outils devront permettre de visualiser pas à pas le comportement individuel de l'animal, afin de vérifier instantanément la cohérence de ses choix alimentaires, mais aussi, dans un cadre spatio-temporel un peu plus large, de donner un aperçu de la trajectoire suivie par un ou plusieurs animaux, afin d'évaluer l'influence de différents facteurs, tels que la végétation, la présence d'autres membres du troupeau ou encore l'impact du facteur aléatoire sur les choix de l'individu.

Ces outils permettront de vérifier que le simulateur fournit des résultats réalistes au niveau du comportement animal à court terme, et, dans le cas contraire, de recalibrer par le biais d'une série de tests comparatifs les paramètres qui influencent ce comportement.

2. Interface graphique

Une fois la validation effectuée, il y aura sans doute également un intérêt à disposer d'un outil offrant une vue globale de la parcelle, qui retracerait de manière synthétique l'évolution de la simulation au cours du temps. Afin d'être la plus complète possible, une telle vue doit permettre une certaine souplesse au niveau du réglage des différents paramètres visuels

(afficher ou bien masquer les animaux, répercuter ou non les modifications du couvert végétal, selon l'aspect de la simulation auquel s'intéressera l'utilisateur), et temporels (vitesse d'affichage, pas de mise à jour, durée considérée,...). Les choix de l'utilisateur concernant ses options s'effectueraient alors par le biais d'une interface graphique à laquelle l'outil «visionneuse » servirait de support. Pour un meilleur confort d'utilisation, cette interface pourrait également permettre d'effectuer des zooms depuis cette vue globale vers une des vues plus détaillées évoquées en 1 ou encore d'avancer ou reculer rapidement dans le temps (comme pourrait le faire un magnétoscope).

3. Mise en valeur du simulateur

Dans un cadre plus ambitieux où l'on envisagerait la perspective d'une publication (papier ou électronique) présentant à un public plus large le travail du simulateur, des échantillons de résultats présentés sous forme d'images (éventuellement animées) apparaîtraient sans doute plus parlants (et plus attractifs) qu'une série de chiffres dans un tableau. La visionneuse, loin d'être un simple gadget, conférerait à l'ensemble du projet de simulateur une dimension supplémentaire.

Chapitre 2: CONCEPTION DE LA VISIONNEUSE

I. REDACTION DU CAHIER DES CHARGES

1. Introduction

La rédaction du cahier des charges a pris, sur la durée du projet, près de la moitié du temps. Il s'agissait de définir exactement ce que désiraient les biologistes de l'INRA en matière de visualisation. La définition de ces besoins n'était pas une chose aisée car, comme il a déjà été expliqué, le simulateur est encore en phase de conception, ce qui fait que certaines des fonctions graphiques doivent afficher des opérations du simulateur qui ne sont pas encore implémentées.

Il a de plus été bien précisé que la visionneuse devait être un module du simulateur «sans intelligence ». C'est le simulateur qui fait les calculs et les hypothèses, la visionneuse doit seulement représenter ses résultats à l'écran. En aucun cas la visionneuse ne devra réaliser de calculs statistiques ou autre, ou extrapoler les données fournies par le simulateur.

Le simulateur étant codé en C++ [3] sous le système d'exploitation Linux, ce sera aussi le cas de la visionneuse. De plus, pour des raisons de compatibilité et de pérennité du code, il a été décidé que la visionneuse n'utiliserait que la librairie Xlib [4] pour les fonctions graphiques, cette librairie étant la librairie standard de base pour l'affichage en mode fenêtré sous UNIX.

Toutefois, il est prévu de pouvoir faire fonctionner ultérieurement cette visionneuse sous Windows, en convertissant le code pour qu'il puisse être compilé en C++ sous Windows ou le traduire en Java.

Les objectifs que doit atteindre la visionneuse sont multiples, et par certains points incompatibles : il n'est par exemple pas possible de visualiser en même temps le comportement individuel d'un ruminant et l'évolution des végétaux sur la totalité de la parcelle. C'est pourquoi la visionneuse devra être composée de trois modules distincts, que l'on appellera par la suite *vue* ou *échelle*, qui s'occuperont chacun d'un type de visualisation différent.

2. Les trois vues

a. Echelle 1

Le rôle de l'échelle 1 est de visualiser pas à pas les déplacements et les choix d'un animal. A chaque pas de déplacement il faudra faire figurer l'animal et ses états (faim, soif, fatigue…), les 15 cellules de son champ de vision, et son choix définitif (c'est à dire la cellule où il ira).

L'écran sera composé de 2 éléments distincts : un schéma figurant son champ de vision, chaque cellule étant colorée suivant la valeur d'un paramètre donné, et un tableau récapitulatif indiquant de façon exhaustive toutes les informations disponibles sur les cellules ainsi que les états de l'animal.

Jusqu'à présent il a été question d'un champ de vision de l'animal de 15 cellules (soit 3 couronnes), mais le nombre de couronnes sera paramétrable par l'utilisateur. L'orientation des cellules correspondra à la vision subjective de l'animal : elle restera fixe au cours du temps.

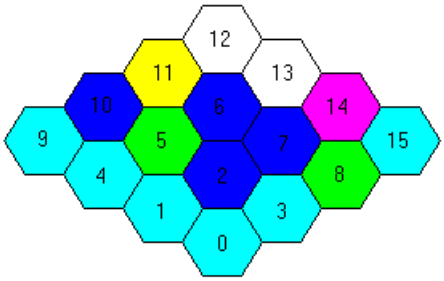


Figure 1 : ce que voit l'animal (vue détaillée)

Le fait de savoir si les coordonnées et les caractéristiques des quinze cellules du champ de vision de l'animal devaient être ou non fournies par le simulateur a été longuement discuté. Après confrontation des arguments, nous nous sommes entendus sur le fait que le simulateur possédait ces informations, nécessaires à ses propres calculs, et qu'il serait bien plus simple qu'il les fournisse à la visionneuse.

On peut voir dans la Figure 1 le champ de vision de l'animal, celui-ci se trouvant à la cellule 0. Dans cette figure le champ de vision comporte 3 couronnes, chaque couronne correspondant à une profondeur supplémentaire à ce que voit la vache (couronne 1 : cellules 1, 2, 3 ; couronne 2 : cellules 4, 5, 6, 7, 8 ; couronne 3 : cellules 9, 10, 11, 12, 13, 14, 15).

b. Echelle 2

Le but de cette échelle est de visualiser l'activité d'un ou plusieurs animaux, sur une durée équivalente à celle d'une séquence d'ingestion (400 déplacements, 20 minutes), et sur une surface de l'ordre du millier de cellules. Les états internes des animaux seront représentés de façon bien plus sommaire que dans l'échelle 1.

Dans les différents schémas qui suivent, la croix représente l'animal étudié, les ronds les autres animaux et les traits des trajectoires.

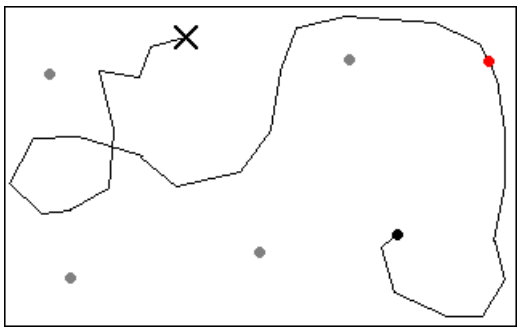


Figure 2 : vue intermédiaire centrée sur un animal

Deux perspectives sont envisagées. La première se focaliserait sur le parcours d'un animal sur une durée d'environ 20 minutes, la zone de la parcelle à afficher étant déterminée à partir des coordonnées maximales et minimales des déplacements de l'animal. Afin de visualiser le comportement social de l'animal, ses congénères situés sur la partie visible de la parcelle devront apparaître (cf. Figure 2).

La seconde se focaliserait sur une zone prédéterminée, sur une durée équivalente, centrée sur un élément remarquable (point d'eau, point d'ombre, frontière entre 2 sites, …), on y

afficherait les déplacements des animaux qui traversent la zone ou éventuellement les différents choix de déplacement d'un même animal (cf. Figure 3).
Il faut colorer les cellules suivant la valeur du paramètre choisi par l'utilisateur et mettre à jour les modifications des cellules par défoliation.

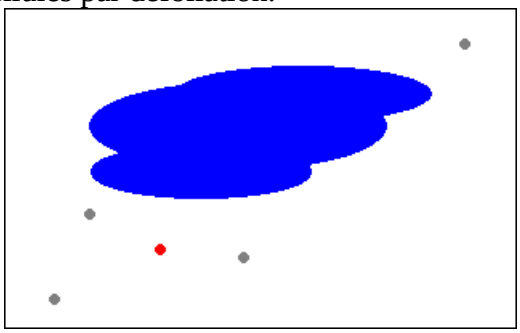


Figure 3 : vue intermédiaire centrée sur une zone

L'animal devra évoluer de façon réaliste, non saccadée, la trajectoire s'affichant de façon progressive en fonction de sa vitesse. On pourra choisir dans un groupe d'animaux de distinguer le leader par une couleur particulière, par exemple.

A ces deux types de visualisation s'en ajoute une troisième dont le but est de voir les différentes trajectoires possibles d'un unique animal à partir d'un état initial donné du système (cf. Figure 4). Les différentes trajectoires choisies par l'animal doivent apparaître l'une après l'autre. Bien évidemment il n'est pas possible de voir la défoliation que provoque l'animal. Cette visualisation est intéressante car le simulateur effectue de nombreux tirages aléatoires. Pour voir la diversité des réponses possibles, à un état initial constant, il faut donc réaliser plusieurs simulation.

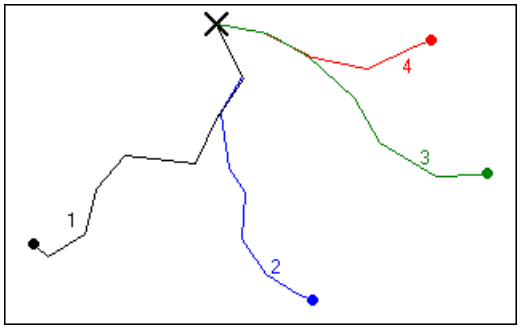


Figure 4 : vue intermédiaire des différents choix possibles

c. Echelle 3

Cette échelle doit permettre de visualiser la totalité de la parcelle, et donc d'observer l'évolution de la végétation sur une période longue, pouvant aller jusqu'à l'année. A cette échelle il n'est plus question de voir le comportement d'un unique animal, mais celui du troupeau dans son ensemble. De même la notion de cellule disparaît au profit de la notion d'agrégation de cellules, un ensemble de cellules ayant des caractéristiques communes.

L'idée de concevoir cette échelle est apparue du fait des limitations à la fois spatiales et temporelles des deux autres échelles, qui ne satisfaisaient pas les biologistes pour l'observation de l'évolution des végétaux. Néanmoins, la réalisation de cette échelle nécessite des concepts (l'agrégation de cellules notamment) dont on ne sait pas encore comment elles seront implémentées dans le simulateur, et elle est surtout trop complexe pour un projet de 100 heures.

La réalisation de cette échelle a donc été remise à plus tard.

3. Dialogue Visionneuse/Simulateur

Il y avait deux façons différentes de faire communiquer la visionneuse avec le simulateur. La première, abandonnée assez vite, consistait à les faire dialoguer en temps réel grâce à des flux de données. Dans cette hypothèse, le simulateur et la visionneuse auraient fonctionné en même temps, et cela aurait permis, par exemple, de pouvoir arrêter une simulation dont le résultat se serait trop écarter du schéma voulu. C'était séduisant, mais il y avait deux problèmes majeurs : le premier est que le simulateur ne fonctionne pas encore… Impossible donc le lancer en même temps que la visionneuse. Le second est qu'après discussion avec les utilisateurs, il s'est avéré qu'ils désiraient pouvoir revenir en arrière dans la visualisation de la simulation, afin de pouvoir mieux comprendre l'évolution des diverses entités et aussi tout simplement pouvoir mettre en pause l'animation.

C'est pourquoi c'est une deuxième hypothèse qui a été envisagée : les faire communiquer par l'intermédiaire de fichiers. Pour revenir en arrière il suffit alors de revenir en arrière dans les fichiers, et l'absence du simulateur peut-être palliée par la création de fichiers «à la main » dont la lecture par la visionneuse peut permettre de tester son bon fonctionnement.

Ces fichiers sont de deux types : les fichiers dits de configuration, qui sont lus lors du lancement de la visionneuse ou le lancement d'une des vues et permettent de connaître le nombre de paramètres, les noms des fichiers utilisés… et les fichiers dits de trace, qui donne chronologiquement l'évolution des états des animaux et des végétaux.

Ces derniers seront écrits au format CSV, qui est directement éditable avec Excel, ce qui leur permet d'être directement lisible pour un être humain. De plus ce format à l'avantage d'être simple d'emploi (c'est un format «texte brut » avec un caractère particulier configurable comme délimiteurs de colonne).

On notera que les fichiers de configuration, pratiques mais peu conviviaux, pourront être remplacés par une interface graphique qui permettra de configurer la visionneuse de façon plus aisée. Cette amélioration, qui n'est pas essentielle à la réalisation du logiciel, fera l'objet d'un travail ultérieur.

Voici une description du contenu de ces fichiers pour les vues détaillée et intermédiaire :

a. Vue détaillée

Fichiers de configuration

- Fichier initialisation : c'est le fichier qui indique le nombre de couronnes, le nombre de déplacements de la vache et les noms des fichiers utiles

Ex :
```
NBE_COURONNES     3
NBE_ETAPES        500
FICHIER_LEGENDE   legende.txt
FICHIER_CELLULES  cellules1.txt
FICHIER_ANIMAL    vache.txt
```

- Fichier légende : indique les noms des paramètres et leurs valeurs min et max

Ex :
```
VV          0     50
VS          10    60.325
RV          0.521 2.365
RS          10    20
VICell      40    10
TpsCell     5     12
```

- Fichier choix : ce fichier comporte une suite de chaînes de caractères correspondant à une suite de raisons de choix de cellule pour l'animal, une ligne correspondant à une raison

Fichiers trace

- Fichier animal : contient la suite des positions d'un animal et les valeurs de ses paramètres. *Choix* indique le numéro de la cellule suivante, *raison* le numéro de la ligne de raison dans le fichier choix, *leader* est un booléen qui indique si l'animal est leader ou non, *paramX* correspond à des paramètres que l'on pourrait ajouter par la suite. Contenu :

```
date abscisse ordonnée direction choix raison leader param1 param2 …
```

- Fichier cellules : ce fichier contient des blocs de K lignes au format suivant :

```
date param1 param2 param3 param4 param5 param6
```

chaque bloc donne les nouvelles valeurs des paramètres des K cellules du champ de vision de l'animal.

b. Vue intermédiaire

Fichiers de configuration

- Fichier d'initialisation : Il comporte à peu près les mêmes éléments que celui de la vue détaillée, avec en plus un paramètre `ECH` qui indique quel sous module de la vue intermédiaire est utilisé et les noms des différents fichiers des animaux (1 par animal, identique à celui de la vue précédente)
- Fichier légende : identique à celui de la vue détaillée

Fichiers de trace

- Fichiers animaux : 1 fichier par animal observé
- Fichier modification des cellules : donne pour toute la parcelle les modifications des cellules par ordre chronologique. Chaque cellule est identifiée par ses coordonnées.

```
date abscisse ordonnée param1 param2 …
```

II. IMPLEMENTATION

1. Description globale

Pour l'analyse et la conception des objets, on utilise UML [5]. C'est un langage de modélisation qui définit un méta-modèle ainsi qu'un ensemble de notations graphiques permettant d'exprimer une analyse et une conception.

a. Organisation des vues

Comme on l'a vu, la visionneuse est divisée en trois parties distinctes, les trois vues (détaillée, intermédiaire et large), la vue intermédiaire comportant elle-même trois modules distincts. Cette division en modules se retrouve dans la hiérarchie de classe de la visionneuse (voir la Figure 5 pour une description de la hiérarchie des classes).

La classe *Visionneuse* est le noyau du programme, lorsque celui-ci sera fini, c'est elle qui s'occupera du passage d'une vue à l'autre et du lancement des vues. Pour l'instant son travail est assez réduit, puisqu'elle lance uniquement la vue détaillée, mais elle est appelée à évoluer lorsque les autres vues seront elles aussi implémentées.

La classe *VueAbstraite* a été définie afin de factoriser les attributs et les méthodes identiques pour toutes les vues. Il s'agit de méthodes telle que l'affichage de la fenêtre, la gestion du clavier et le lancement de l'animation. En ce qui concerne les attributs, toutes les vues possèdent une légende (décrite plus loin dans la section des objets graphiques), et des fichiers de lecture (décrits plus loin). C'est aussi cette classe qui ouvre le serveur X, dans son constructeur, et initialise les objets de classe d'*ObjetGraphique* (décrits plus loin)

En matière de gestion des interruptions clavier, la classe *VueAbstraite* traite l'appui sur la touche [FIN], et si la touche enfoncée n'est pas une touche de fin de visualisation, elle envoie le code de cette touche à la fonction *AppuiTouche,* redéfinie dans chacune de ses sous-classes.

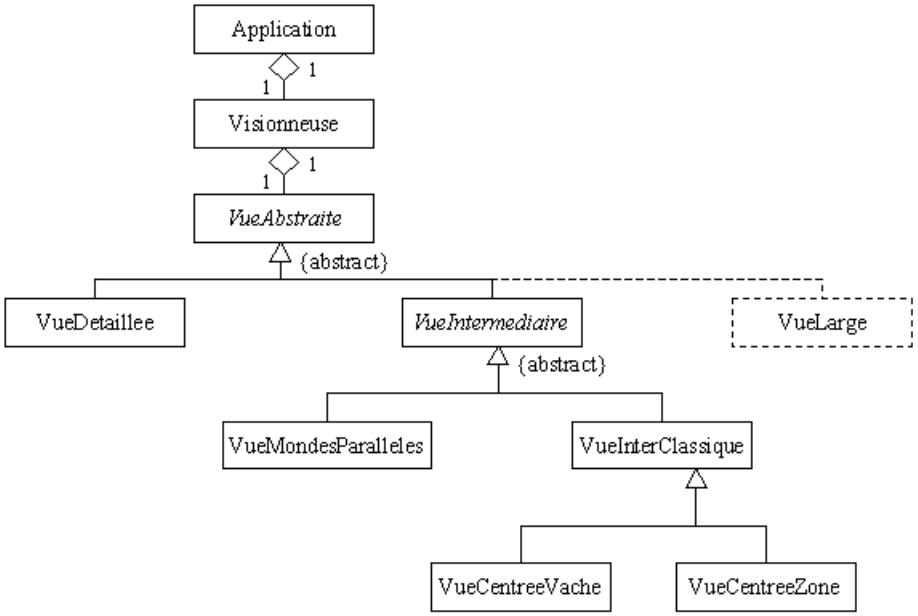


Figure 5 : diagramme de classes représentant la hiérarchie des vues

De la même façon, toutes les variations de la vue intermédiaire dérivent d'une sous-classe de VueAbstraite, *VueIntermédiaire*. La vue que pour l'instant on appelle «VueMondesParallèles », nom qui correspond assez bien à sa particularité de faire coexister les différents choix qu'aurait pu faire un animal à partir d'un moment donné, a été mise seule d'un côté tandis que les deux autres vue, qui ont des caractéristiques communes assez évidentes, ont été regroupées et descendent toutes deux de la classes abstraite *VueInterClassique*.

Ces deux vues, *VueCentreeZone* et *VueCentreeVache,* sont en effet très proches et ne diffèrent que sur le fait que la première vise plutôt à observer le comportement des animaux autour d'un point stratégique, tandis que la seconde tend à observer le comportement d'un animal particulier, mais tout en regardant ce qui ce passe autour. C'est en quelque sorte comme si cet animal était lui-même un point stratégique.

b. Objets graphiques

Bien évidemment, puisqu'il faut afficher des dessins à l'écran, on ne pouvait se passer d'une classe d'objets graphiques. Tous les objets dessinables dérivent d'une seule et même classe abstraite : *ObjetGraphique,* leur hiérarchie est représentée Figure 6.

En plus des classiques méthodes d'affichage et d'effacement de l'objet, ainsi que des attributs de coordonnées (x et y), la classe *ObjetGraphique* possède des attributs de classe spécifiques à la Xlib. En effet, toutes les fonctions d'affichage de cette librairie demandent en paramètre le périphérique d'affichage utilisé (display), le contexte graphique (gc), et la fenêtre (win), variables qui indiquent où doit se faire l'affichage. Ces valeurs, initialisées dans le constructeur de *VueAbstraite* lors de l'ouverture du serveur X, ne varient ensuite plus et sont bien entendu identiques pour tous les objets affichés dans la même fenêtre.

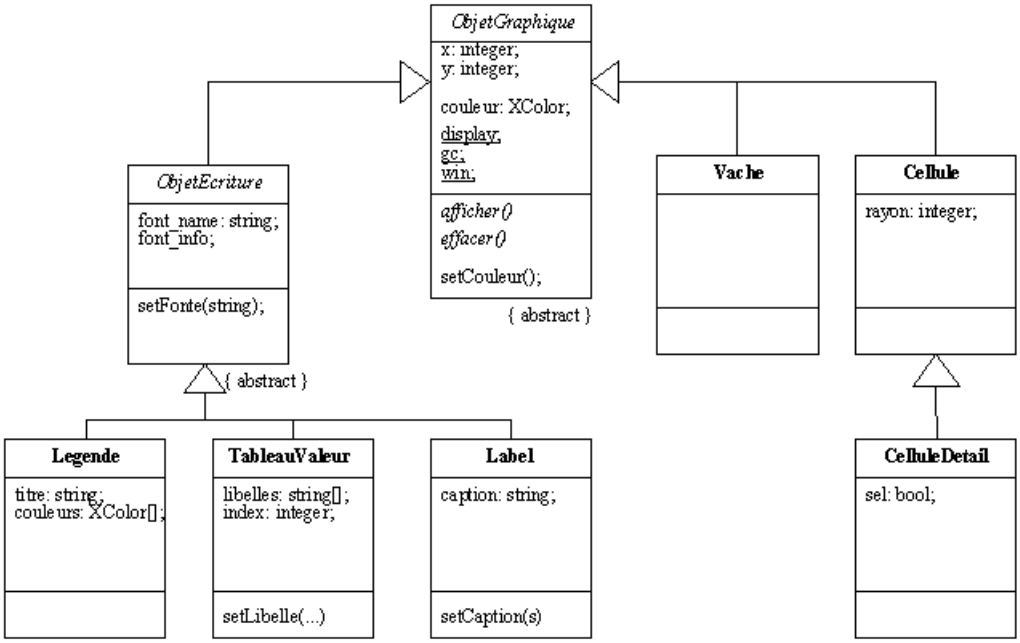


Figure 6 : diagramme de classes représentant la hiérarchie des objets graphiques

De plus, comme toutes ses sous-classes ont besoin d'un attribut et de méthodes pour leur couleur, ceci a été factorisé dans *ObjetGraphique*.

La classe *ObjetEcriture*, qui dérive d'*ObjetGraphique*, est une classe abstraite qui implémente des fonctions de gestion de police de caractères. Les classes dérivées d'*ObjetEcriture* sont des classes qui ont à écrire des chaînes de caractères à l'écran.

Les classes *Vache* et *Cellule* ne font que redéfinir la fonction afficher() héritée d'*ObjetGraphique*, la première pour dessiner un rond représentant la vache, la seconde pour dessiner un hexagone coloré. La classe *CelluleDetail* ajoute un attribut de sélection utilisé dans la vue détaillée.

La classe *Legende*, qui sera utilisée dans toutes les vues, est un objet graphique qui, comme son nom l'indique, sert à afficher une légende à l'écran. L'apparence de la légende est telle qu'elle apparaît à la Figure 7.

Elle est totalement configurable : on peut régler le nombre d'items, leur disposition (en colonne, en ligne, en tableau), leurs couleurs. Comme elle dérive d'*ObjetEcriture*, on peut changer la police utilisée et sa couleur.

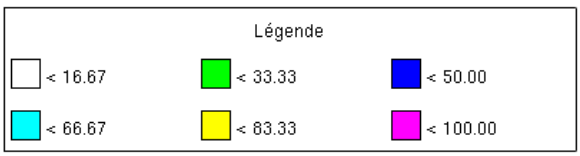


Figure 7 : La légende

La légende peut être de deux types : soit on ne connaît que les valeurs minimum et maximum que peut prendre le paramètre observé, et dans ce cas c'est la légende qui découpe cet intervalle en «nombre d'items » intervalles égaux représentés chacun par une couleur, soit on connaît une liste de valeurs qui servent de délimiteurs.

c. Les lecteurs de fichiers

Pour l'instant, le simulateur ne peut communiquer d'informations à la visionneuse que par l'intermédiaire de fichiers plus ou moins spécifiques à chaque vue. Ceux-ci sont globalement de 2 types : fichiers de configuration et fichiers trace.

Les fichiers de configuration servent à indiquer la valeur d'un certain nombre de paramètres qui resteront constants au cours de l'exécution. Chaque objet lecteur n'accède donc ici qu'une seule fois au fichier, lors dans la construction, pour stocker les valeurs de ces paramètres dans l'attribut correspondant. Ils fournissent ensuite les méthodes publiques permettant d'accéder en lecture aux valeurs de ces attributs privés.

Pour l'échelle 1, ces fichiers de configuration sont de 3 types : fichier d'initialisation, fichier légende et fichier choix.

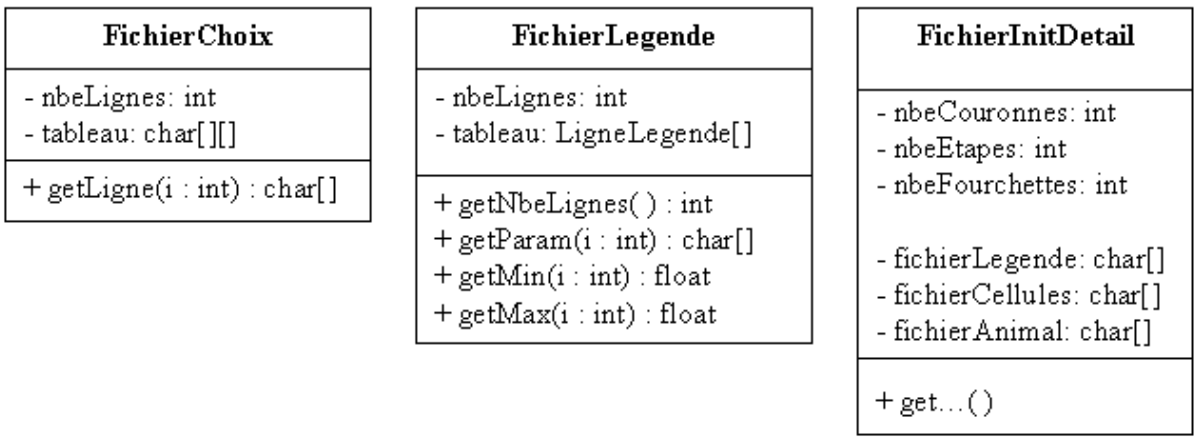


Figure 8 : diagramme de classes représentant les lecteurs de fichiers de configuration

Le fichier d'initialisation de la vue détaillée peut fournir, par le biais d'associations de type variable/valeur, des informations concernant la simulation elle-même, telles que le nom des fichiers trace, le nombre total d'étapes à considérer, et le nombre de couronnes du champ de vision de l'animal. Ces informations servent à initialiser les lecteurs des fichiers trace par l'intermédiaire de leur constructeur. Le fichier d'initialisation sert également à préciser le nom de fichiers auxiliaires utilisés par la visionneuse, tels que le fichier légende.

Le lecteur se charge de parcourir en une fois le fichier d'initialisation, recherchant des noms de variables correspondant à ses attributs, pour y stocker la valeur associée. Les paramètres qui ne figureraient pas dans le fichier reçoivent une valeur par défaut, tandis que les noms de variables inconnus du lecteur sont ignorés.

Le fichier légende contient une série de lignes indiquant le nom d'un paramètre et les valeurs minimum et maximum que celui-ci peut prendre en théorie. Le lecteur se charge de stocker ces informations dans un tableau à la structure adaptée. A partir des informations contenues dans ce tableau (et du paramètre *nbeFourchettes* fourni par le fichier d'initialisation) on peut ainsi construire une légende appropriée en divisant l'intervalle [min, max] de chaque paramètre en un nombre fixe de sous-intervalles réguliers.

Enfin le fichier choix contient une série de lignes correspondant aux raisons potentielles des choix effectuées par un animal. Son lecteur se charge de stocker ces lignes dans un tableau de chaînes de caractères. Ce lecteur sera a priori employé exclusivement par la classe *FichierAnimalDetail*, définie ci-dessous, dont les instances obtiennent, par l'intermédiaire du fichier trace de l'animal, un indice indiquant à la raison du choix, l'explication correspondante étant stockée dans le tableau de l'objet *FichierChoix* associé par défaut à la classe *FichierAnimalDetail*.

Les fichiers trace contiennent toutes les informations relatives aux résultats de la simulation elle-même. Assez volumineux, ils décrivent chronologiquement, ligne par ligne, l'évolution de la situation.

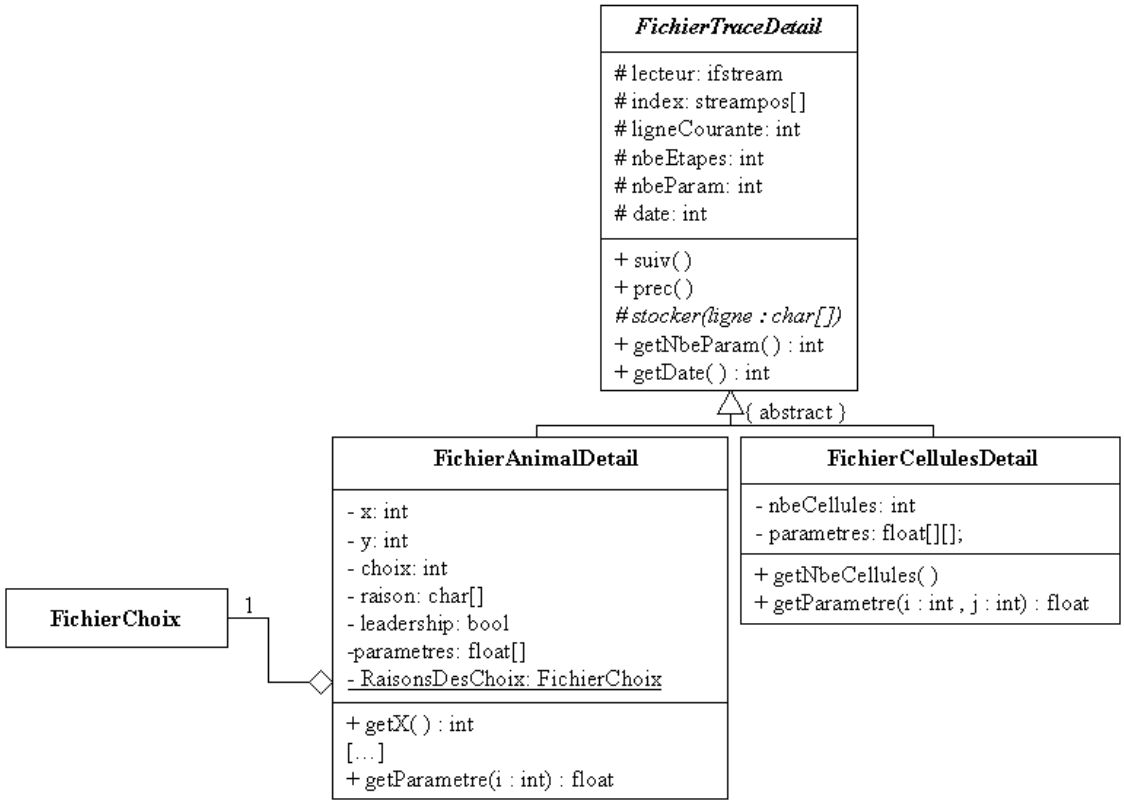


Figure 9 : diagramme des classes représentant les lecteurs de fichiers trace

Pour l'échelle 1, on dispose de deux types de fichier trace : l'un correspond aux informations concernant les déplacements, les choix, et l'état interne d'un animal donné, et l'autre aux paramètres relatifs au groupe de cellules parmi lesquelles cet animal effectue son choix à chaque étape.

Etant donné la structure assez similaire des deux types de fichier trace, on peut définir une classe abstraite de lecteur, appelée *FichierTraceDetail*, qui stocke dans ses attributs les types d'informations pouvant concerner à la fois un animal et un groupe de cellules, tels que le nombre de paramètres, ou la date, et fournit les méthodes publiques d'accès en lecture à ces attributs privés. Mais cette classe gère surtout les problèmes d'ouverture, de parcours, par l'intermédiaire des méthodes *suiv* et *prec*, qui permettent d'avancer ou de reculer dans le fichier (c'est à dire dans le temps), avec gestion des débordements, et enfin de fermeture du fichier. Les attributs cachés tels que *lecteur*, *index* ou *ligneCourante*, sont utilisés par ces méthodes pour se repérer au sein du fichier et accéder à son contenu.

Les classes dérivées *FichierAnimalDetail* et *FichierCellulesDetail* possèdent des attributs propres correspondant à divers paramètres (d'un nombre variable) spécifiques à ces entités, et implémentent les méthodes pour y accéder. Elles redéfinissent également en privé la méthode virtuelle *stocker* héritée de la classe mère, appelée par les méthodes *suiv* et *prec*, et qui correspond à la mise à jour des paramètres relatifs à nos entités à partir d'une position donnée dans le fichier trace.

L'échelle 2 devrait comporter un type supplémentaire de fichier trace, correspondant à l'état initial de la parcelle en début d'exécution. Le fichier trace relatif à l'évolution de chaque animal ne devrait pas être différent de celui de l'échelle 1. Concernant l'évolution du couvert végétal, on ne disposera par contre que d'un unique fichier retraçant les modifications subies par toutes les cellules de la parcelle, classées par ordre chronologique. Contrairement aux lecteurs de la vue détaillée, pour des raisons de vitesse d'exécution, ceux de la vue intermédiaire n'accéderont qu'une seule fois à leur fichier trace respectif, lors de leur création. Ils stockeront dans leurs attributs les valeurs initiales de leurs différents paramètres, suivies de toutes les modifications successives.

2. La vue détaillée (échelle 1)

a. Description

Maintenant que l'on a vu le fonctionnement global de la visionneuse, il est temps de passer à une description plus précise de la vue détaillée. Comme on l'a vu, à cette échelle il n'y a pas à proprement parler d'animation : c'est l'utilisateur qui fait avancer pas à pas le résultat de la simulation. Comme le précisait le cahier des charges, il y a deux éléments importants dans cette vue : un ensemble d'hexagones symbolisant les cellules que voit l'animal et un tableau indiquant pour chaque cellule les valeurs de ses paramètres. On ajoute à cela une légende afin de rendre compréhensible la signification des couleurs des hexagones, et un certain nombre de labels indiquant la position de l'animal, sa direction, la raison de son choix et la date par rapport au début de la simulation.

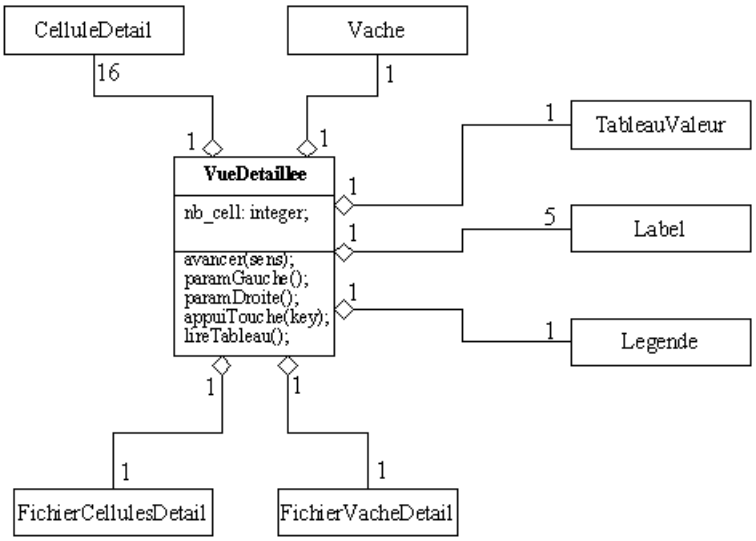


Figure 10 : diagramme de classes représentant la vue détaillée

L'utilisateur peut effectuer quatre actions différentes :

- avancer dans la simulation, en appuyant sur [Espace] ou [Flèche Haut]
- reculer, en appuyant sur [Backspace] ou [Flèche Bas]
- observer un autre paramètre, en appuyant sur [Flèche Gauche] ou [Flèche Droite], qui lui permettent respectivement de passer au paramètre de la colonne de gauche ou de droite dans le tableau de valeur. Le paramètre observé est représenté par une colonne rouge, contrairement aux autres qui sont noires.
- Quitter cette échelle en appuyant sur [Fin] ou [Q]

Une instance de la classe *VueDetaillee* coordonne le fonctionnement des différents objets graphiques (voir Figure 10).

C'est dans le constructeur de *VueDetaillee* que sont crées les instances des objets graphiques et des fichiers qui seront utilisés. L'appel aux constructeurs se fait dans un ordre bien précis, qui découle des informations contenues dans les fichiers d'initialisation (voir Figure 11).

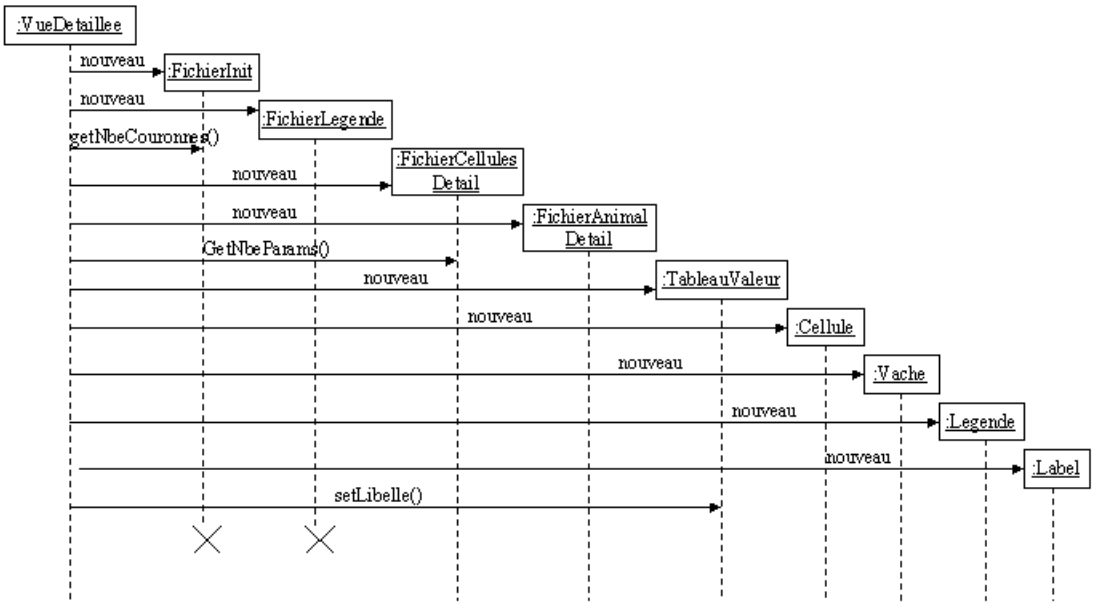


Figure 11 : diagramme en séquence de la phase d'initialisation de VueDetaillee

b. Fonctionnement

La gestion des événements clavier représente bien évidemment la partie la plus importante à cette échelle (voir Figure 12).

Lorsqu'une touche est pressée, il y a un appel à la fonction *appuiTouche(key)*, dans lequel se trouve un test sur la valeur de la touche. En fonction de la valeur de key, une des fonctions suivantes est appelée :

- *avancer(sens : booléen)* : avancement dans la simulation, que ce soit vers l'avant ou l'arrière, en fonction de la valeur de sens. Si sens=true, on avance dans la simulation.
- *ParamGauche()* : afficher les valeurs du paramètre de gauche dans le tableau

- *ParamDroite() :* idem mais à droite

La gestion des événements du clavier se fait dans une boucle infinie. Lorsque l'utilisateur décide de quitter la vue, en appuyant sur [Fin], le programme sort de cette boucle infinie, sort de la fonction *lancer()*, et le contrôle est rendu à la classe *Visionneuse*.

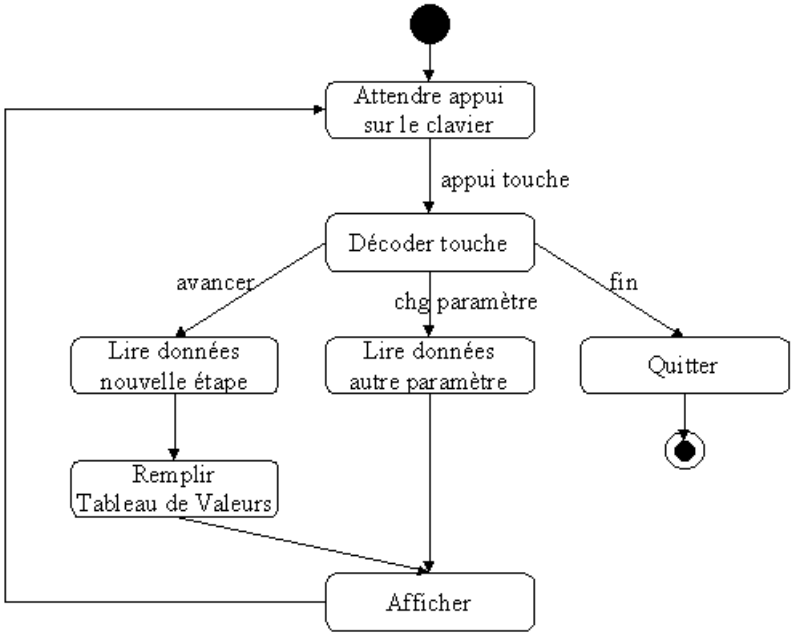


Figure 12 : diagramme d'activités de la gestion du clavier

Dans le tableau, qui a deux dimensions, les paramètres se trouvent en colonne et les cellules en ligne : la valeur du troisième paramètre dans la sixième cellule se trouve à la sixième ligne, troisième colonne.

Le nombre de paramètres affichés dans le tableau de valeurs pouvant être élevé (jusqu'à 30), ce tableau a la particularité de pouvoir défiler : seuls 8 paramètres au maximum peuvent être visibles en même temps, et la case en haut à gauche indique s'il y a possibilité de faire défiler à gauche ou à droite le tableau.

Par exemple, si cette case à cette apparence [← →], alors le tableau peut défiler à gauche et à droite. S'il ressemble à [→], alors le tableau ne peut défiler que vers la droite.

Chapitre 3: RESULTATS ET DISCUSSION

I. RESULTATS

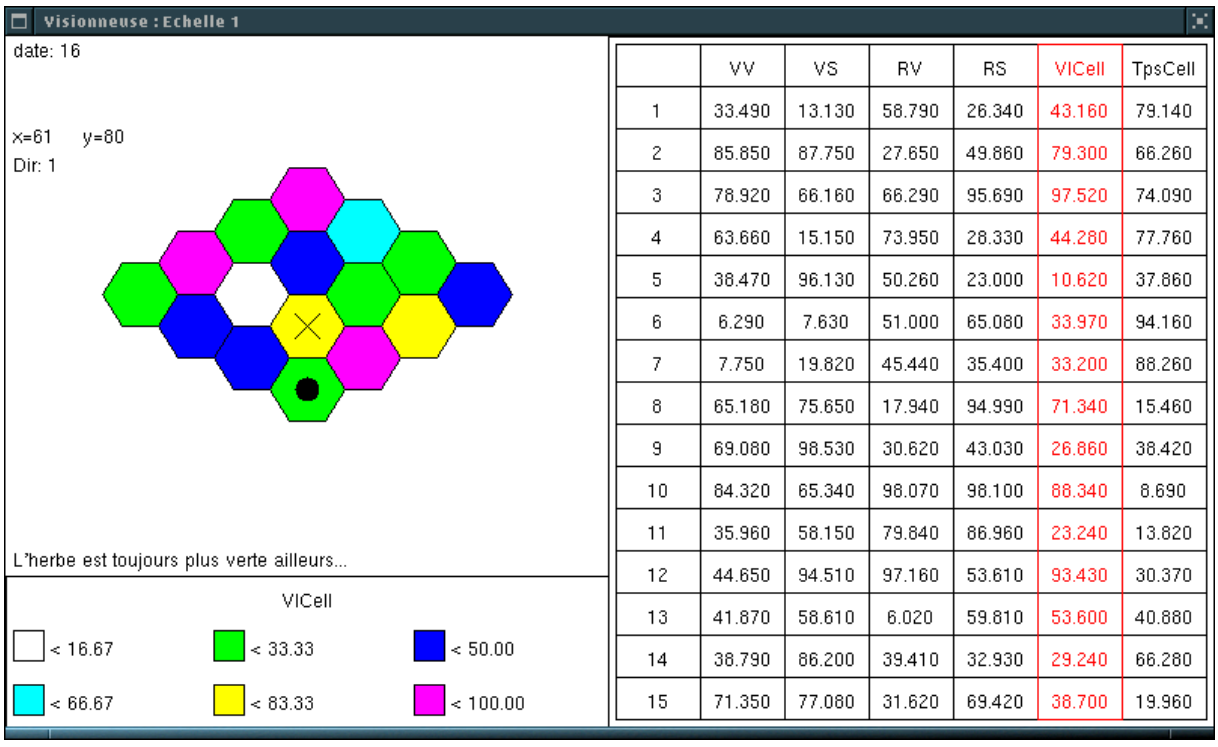
1. Plate-forme de travail

Comme prévu étant donné que nous avions travaillé avec la Xlib en respectant toutes les contraintes ANSI et en utilisant uniquement des objets de la STL [3], notre programme est portable. Il fonctionne ainsi parfaitement non seulement sur sa plate-forme de développement (un PC sous Linux, distribution Mandrake 7.2) mais aussi sur HP-UX et AIX, deux des systèmes d'exploitation utilisés à l'ISIMA. . La visionneuse est probablement compatible avec tout système UNIX, moyennant de légères modifications telles que le changement de police, celles-ci n'étant pas identiques partout.

2. Tests, copies d'écrans

Comme on peut le constater sur la Figure 13, le résultat est graphiquement très correct. Le paramètre observé voit sa colonne mise en rouge dans le tableau de gauche, et son nom est inscrit dans la légende.

La raison du choix de la prochaine cellule est indiquée en texte au-dessus de la légende.



	VV	VS	RV	RS	VICell	TpsCell
1	33.490	13.130	58.790	26.340	43.160	79.140
2	85.850	87.750	27.650	49.860	79.300	66.260
3	78.920	66.160	66.290	95.690	97.520	74.090
4	63.660	15.150	73.950	28.330	44.280	77.760
5	38.470	96.130	50.260	23.000	10.620	37.860
6	6.290	7.630	51.000	65.080	33.970	94.160
7	7.750	19.820	45.440	35.400	33.200	88.260
8	65.180	75.650	17.940	94.990	71.340	15.460
9	69.080	98.530	30.620	43.030	26.860	38.420
10	84.320	65.340	98.070	98.100	88.340	8.690
11	35.960	58.150	79.840	86.960	23.240	13.820
12	44.650	94.510	97.160	53.610	93.430	30.370
13	41.870	58.610	6.020	59.810	53.600	40.880
14	38.790	86.200	39.410	32.930	29.240	66.280
15	71.350	77.080	31.620	69.420	38.700	19.960

Figure 13 : Copie d'écran

Les tests ont été réalisés avec un nombre de paramètres allant de 1 à 30. Le résultat obtenu reste agréable à l'œil, notamment grâce au défilement du tableau qui permet d'éviter d'afficher trop de colonnes de nombres rébarbatives.

II. DISCUSSION

1. Déroulement du travail

Dans un premier temps, il a fallut se concentrer sur l'analyse des besoins des utilisateurs, afin de mener à bien la rédaction définitive du cahier des charges.

Celle-ci n'a pu aboutir qu'à l'issue de nombreuses réunions avec les intéressés, réunions au cours desquelles ont été longuement discutés les objectifs de la visionneuse, et les contraintes qui en découleraient, en particulier au niveau du format des fichiers trace qui lui seraient fournis par le simulateur.

Une fois ces points éclaircis, nous avons pu aborder, courant janvier, l'aspect technique de la réalisation de la visionneuse, à travers, tout d'abord, une étude théorique du problème, qui a donné lieu à la conception de schémas UML synthétisant le travail à accomplir en pratique au niveau de l'organisation des différents objets/classes.

En dernier lieu, nous nous sommes enfin tournées vers l'aspect pratique de la réalisation du projet. Le travail accompli précédemment a ainsi pu se concrétiser à travers l'élaboration d'un prototype fonctionnel de "l'échelle 1", ou "vue détaillée".

2. Raison des choix de limitation

Le projet initial, nourri des divers désirs et exigences des futurs utilisateurs, apparaît comme très ambitieux, non pas tant du point de vue technique, qu'au niveau du temps nécessaire à l'élaboration du modèle et son implémentation. C'est pourquoi à l'issue de l'élaboration du cahier des charges, il nous a été explicitement demandé de nous concentrer en priorité sur l'implémentation des aspects de la visionneuse qui apparaissaient comme les plus importants dans un proche avenir, c'est à dire ceux concernant la validation du modèle (échelle 1 et 2).

L'échelle 3 a donc été écartée, d'autant plus que certains concepts entrant en compte dans son élaboration n'avaient pas encore été définis au niveau du simulateur lui-même. L'idée, pourtant séduisante, d'une interface graphique complète a du, elle aussi, être repoussée à plus tard, l'élaboration d'une telle interface exigeant un travail considérable, pouvant en lui-même constituer l'objet d'un autre projet de 100 heures.

Une fois ces points éclaircis, il nous a encore fallut choisir, faute de temps, entre l'implémentation de l'échelle 1 et celle de l'échelle 2. Il aurait en effet été impossible de mener à bien l'élaboration des deux échelles dans le temps qui nous restait. Nous avons donc préféré nous concentrer exclusivement sur le sous-module de l'échelle 1, afin de pouvoir présenter à la fin du temps imparti un logiciel, sinon complet, du moins entièrement fonctionnel.

3. Avenir du projet

Le sous-module correspondant à l'échelle 1 ne devrait plus nécessiter que de légères corrections afin d'optimiser son intégration au sein de la visionneuse une fois celle-ci intégralement terminée. Il restera bien sûr également envisageable d'y intégrer de nouvelles fonctionnalités, suivant l'évolution des besoins des utilisateurs.

L'échelle 2 devrait pouvoir être implémentée assez rapidement, au cours d’un prochain stage. Les bases théoriques ont déjà été vaguement esquissées.

L'échelle 3 et l'interface graphique représentent deux objectifs à long terme, dont l'élaboration dépendra de l'évolution du projet du simulateur lui-même.

CONCLUSION

Nous sommes parvenues à produire un prototype fonctionnel qui répond aux besoins des utilisateurs. Il reste encore du travail à accomplir, la vue intermédiaire n'étant pas encore implémentée, mais les bases théoriques sont là.

Après coup on constate qu'une grande partie du temps à été consacrée à recueillir les besoins des biologistes afin d'établir le cahier des charges. Cette période formatrice nous a permis de mieux prendre conscience des difficultés à concilier les besoins des futurs utilisateurs et les contraintes inhérentes à un projet de grande envergure tel que ce simulateur.

La deuxième partie du projet, la réalisation de la visionneuse, nous a permis de nous familiariser avec la conception et la programmation objet et de réaliser ainsi l'importance de l'analyse théorique précédant la phase d'implémentation.

Dans un futur proche, d'autres modules viendront s'ajouter au module déjà existant : la vue intermédiaire, peut-être aussi la vue large. Pour plus de convivialité, une interface graphique devra remplacer le système des fichiers de configuration. Ces améliorations feront partie des objectifs d'un stage de 6 mois réalisé en 2001.

On peut envisager de nouvelles améliorations mettant en valeurs d'autres aspects de la simulation. Mais les concepts envisagés, tels que l'agrégation de cellules, n'ont pas encore été implémentés, et l'avenir de la visionneuse dépendra donc de l'évolution du simulateur. N'oublions pas que le simulateur est un projet ambitieux, étalé sur plusieurs années, et dont la validation est appelée à prendre du temps.

REFERENCES BIBLIOGRAPHIQUES

[1] F GUERRY, *Conception d'un simulateur multi-agents de l'interaction troupeau/parcelle pâturée,* rapport de stage ISIMA avril/septembre 2000

[2] D SAUVANT, R BAUMONT, P FAVERDIN, *Development of a Mechanistic Model of Intake and Chewing Activities of Sheep,* Journal of Animal Science, 74, 1996, pp 2785-2802

[3] B GARCIA, *La Librairie standard du C++*, cours sur la STL

[4] J GETTYS, R W SCHEIFLER, *Xlib – C Language X Interface*, 1996

[5] C FORCE, *Méthodes de génie logiciel UML et UP*, support de cours disponible à l'ISIMA