Analyse et conception du couplage entre les modules animal et végétal du simulateur troupeau/parcelle

Guillaume Martin, René Baumont, Laurent Pérochon, Claude Mazel

INSTITUT **S**UPERIEUR
D'**I**NFORMATIQUE
DE **M**ODELISATION
ET DE LEURS **A**PPLICATIONS

Complexe des Cézeaux
BP 125-63170 Aubière CEDEX

INSTITUT **N**ATIONAL
DE LA **R**ECHERCHE
AGRONOMIQUE

Unité de Recherches sur les Herbivores
Theix
63122 St-Genès-Champanelle

Rapport de stage 2ème année

Analyse et conception du couplage entre les modules animal et végétal du simulateur troupeau/parcelle

Tome I

Présenté par : Guillaume Martin
Lieu de stage : INRA Theix
Responsables de stage :
René Baumont, Laurent Pérochon
Tuteur ISIMA : Claude Mazel
Tuteur INRA : Laurent Pérochon

Du 8 avril au 29 septembre 2002

INSTITUT **S**UPERIEUR
D'**I**NFORMATIQUE
DE **M**ODELISATION
ET DE LEURS **A**PPLICATIONS

Complexe des Cézeaux
BP 125-63170 Aubière CEDEX

INSTITUT **N**ATIONAL
DE LA **R**ECHERCHE
AGRONOMIQUE

Unité de Recherches sur les Herbivores
Theix
63122 St-Genès-Champanelle

Rapport de stage 2ème année

Analyse et conception du couplage entre les modules animal et végétal du simulateur troupeau/parcelle

Tome I

Présenté par : Guillaume Martin
Lieu de stage : INRA Theix
Responsables de stage :
René Baumont, Laurent Pérochon
Tuteur ISIMA : Claude Mazel
Tuteur INRA : Laurent Pérochon

Du 8 avril au 29 septembre 2002

Remerciements

Je tiens à remercier M Laurent Pérochon pour tous les conseils et toutes les réponses qu'il m'a apportés ainsi que pour son encadrement dans ce stage. Je remercie aussi M René Baumont pour le temps qu'il a bien voulu m'accorder pour effectuer les validations et les corrections qui s'imposaient dans mon travail.

Merci également à M David Hill pour ses précieux conseils et solutions en programmation ainsi que pour les connaissances techniques qu'il m'a apportées. Je tiens encore à remercier M Claude Mazel pour ses commentaires et ses observations judicieuses, et Mme Christine Force pour son engagement dans ce projet et sa participation à l'avancement des prototypes.

En fin, je tiens à remercier toute l'équipe RAP pour son accueil.

Table des abréviations

AIX : Système d'exploitation de type UNIX pour les serveurs IBM
C++ : Langage de programmation orienté objet, dérivé du langage C
DNDF : Digestibilité des parois cellulaires
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
NDF : Taux de fibres dans la plante
RAP : Relation Animal / Plantes
UML : Unified Modelling Language
UP : Unified Process
URH : Unité de Recherches sur les Herbivores
XDM : X Display Manager
Xlib : Librairie graphique standard des systèmes d'exploitation de type UNIX

Table des figures et illustrations

Résumé

Un simulateur Parcelle / Troupeau, visant à aider à mieux comprendre le comportement d'un troupeau pâturant une parcelle, est en cours de développement. Il doit pouvoir modéliser plusieurs animaux sur une parcelle en conditions extensives.

Mon rôle consiste à relier et à faire communiquer entre eux des applications déjà existantes. Pour cela, il a été nécessaire de faire plusieurs prototypes en augmentant petit à petit la difficulté pour arriver à un animal capable d'effectuer ses propres choix sur une parcelle donnée.

Une phase d'analyse et de reverse engineering a permis de comprendre exactement quel était le rôle des différents programmes déjà existant. J'ai pu en déduire le premier prototype en mettant en relation les applications déjà existantes pour simuler et visualiser un animal se déplaçant sur une parcelle possédant une végétation sans dynamique. La deuxième phase consiste à simuler les choix alimentaires d'un animal sur une parcelle. Enfin, le troisième prototype inclut, en plus du deuxième, la dynamique physiologique de l'animal.

Les trois prototypes fonctionnent et peuvent être utilisés pour effectuer les nombreux tests nécessaires à la validation du modèle. Certains programmes, utilisés auparavant, ont été modifiés pour s'adapter aux différentes versions et de nouvelles applications ont été créées comme le générateur de statistiques.

Les différentes applications sont développées en C++ sous Linux et pour des raisons de portabilité du code, aucune librairie non standard autre que la Xlib n'a été utilisée. On peut donc envisager dans l'avenir de faire fonctionner ces programmes sous d'autres plates-formes de travail.

Mots-clef : simulateur ; prototype ; C++ ; parcelle ; troupeau

Abstract

A pasture/herd simulator is being developed to allow better understanding of the herd behaviour. This simulator will model several animals on a pasture in extensive conditions.

My role consists of connecting already existing applications together to enable them to communicate. I had to create several prototypes of increasing complexity to obtain an animal able to make its own choices on a given pasture.

The first phase of analysis and reverse engineering enable me to understand the role of the already existing programs. I created the first prototype by establishing interface between those applications. This prototype simulates and allows visualisation of an animal moving on a fixed pasture. The second prototype consists of simulating the animal's food choices on a pasture. Finally the last prototype is an improvement of the second one and includes the physiological dynamics of the animal.

The three prototypes are working and are currently being validated. Previous programs have been modified to be used with the three prototypes and new applications have been created, such as a statistics generator.

The various applications are developed in C++ under Linux and for portability reasons of the code, only standards libraries, but Xlib, were used. In the future these programs will be ported to other operating systems.

Keywords: simulator ; prototype ; C++ ; pasture ; herd

Table des matières

Introduction

L'équipe RAP de l'Unité de recherches sur les Herbivores de l'I.N.R.A. de Clermont Ferrand Theix développe actuellement un simulateur parcelle/troupeau en collaboration avec l'I.S.I.M.A. Celui-ci a pour but de modéliser les interactions d'un troupeau en condition d'élevage extensif avec une parcelle pâturée. Ainsi, il devrait être possible de prévoir le comportement d'un troupeau sur une prairie afin de mieux le gérer. Ce comportement provoquant une évolution de la parcelle, il sera aussi possible de prédire les modifications de celle-ci.

Des stages ont déjà été effectués dans différentes équipes relativement indépendantes. En 2000, François Guerry a commencé à modéliser et à programmer la partie animale et en 2001, Lucie Masson a poursuivi ce projet en développant la partie mémorisation. La partie végétale a été développée au centre INRA de Crouël.

Mon stage en 2002 consiste à interconnecter les différentes parties du simulateur et à les améliorer pour les rendre plus homogènes et compatibles entre elles. Il faut en particulier que l'animal puisse défolier la parcelle et que le modèle prenne en compte ces défoliations sur la végétation. Une série de test sera nécessaire pour régler le comportement de l'animal et faire en sorte que celui-ci soit le plus proche possible de la réalité. Dans un deuxième temps, il consiste à compléter le simulateur en faisant interagir un, puis plusieurs animaux sur la parcelle. Il faudra tenir compte de leurs choix et permettre une visualisation de leurs mouvements.

Chapitre 1: Contexte scientifique et technique

1.1 L'existant

1.1.1 Le simulateur parcelle/troupeau

La simulation d'un troupeau de ruminants pâturant une parcelle est complexe. Elle doit représenter les comportements individuels et collectifs des animaux ainsi que leurs interactions avec la végétation en prenant en compte l'aspect spatial. L'objectif étant de modéliser des situations réelles, il a été nécessaire de représenter plusieurs niveaux de perception avec de nombreuses variables.

Tout d'abord, il faut modéliser l'animal. C'est un ruminant qui se nourrit au pâturage et qui prend des décisions en fonction de son état physiologique. Ceci représente le module A ou module animal du simulateur.

Ensuite, il y a la parcelle qui représente à la fois la végétation avec sa dynamique temporelle et son hétérogénéité spatiale. C'est le module V ou module végétal. L'espace est découpé en cellules hexagonales de 0.1 m² qui contiennent chacune différents paramètres, comme la quantité de biomasse des tissus ou le taux de digestibilité des parois d'une cellule. Les cellules appartiennent à un faciès, caractérisé par l'association en proportion constante de différentes espèces de végétation. Des équations modélisant leur croissance et leur sénescence leur sont également associées.

Le troupeau quant à lui est vu comme le résultat des interactions et compétitions entre individus, et c'est pour cela que nous avons choisi l'approche multi-agents. C'est le module S ou Social et Spatial. Il gère les interactions sociales. Elles sont de deux types. Tout d'abord il y a la proximité entre animaux, et ensuite les déplacements longs. Ces derniers sont générés par un animal qui décidera d'aller sur un autre emplacement de la parcelle. A ce moment, l'ensemble du troupeau peut décider de le suivre.

1.1.2 Programmes existants

Une approche multi-modèles a été adoptée pour développer ce simulateur. Le principe consiste à décomposer le simulateur en modules pour avoir un plus haut niveau d'abstraction et simplifier le modèle comme le propose Paul A. Fishwick [9] . Chaque modèle peut ainsi être décomposé en d'autres sous modèles et ainsi de suite jusqu'à obtenir le niveau de détail désiré. L'avantage est que, quelle que soit la méthode utilisée dans chaque module, les connexions avec les autres modules restent inchangées. Ainsi, il est possible de développer chaque module individuellement et de les interconnecter plus tard. Le simulateur a été décomposé en trois modules qui sont à des stades d'avancements différents.

1.1.2.1 Le Module A

Cette première version était centrée sur la motivation à ingérer, l'ingestion et l'évolution de l'état physiologique de l'animal. Il s'agit principalement de l'adaptation et de la simplification d'un modèle précédemment développé (Sauvant et al 1996) [10] . Une partie décisionnelle, concernant les actions que l'animal choisit d'effectuer, a été implémentée mais elle est très simplifiée. L'animal

choisi de manger, de boire, de se reposer ou de se déplacer. S'il choisi de manger, il mangera toujours le même type de cellule, et cela modifiera son état physiologique ainsi que sa motivation à ingérer. Le temps consacré à ces différentes activités est modélisé (Figure 1).

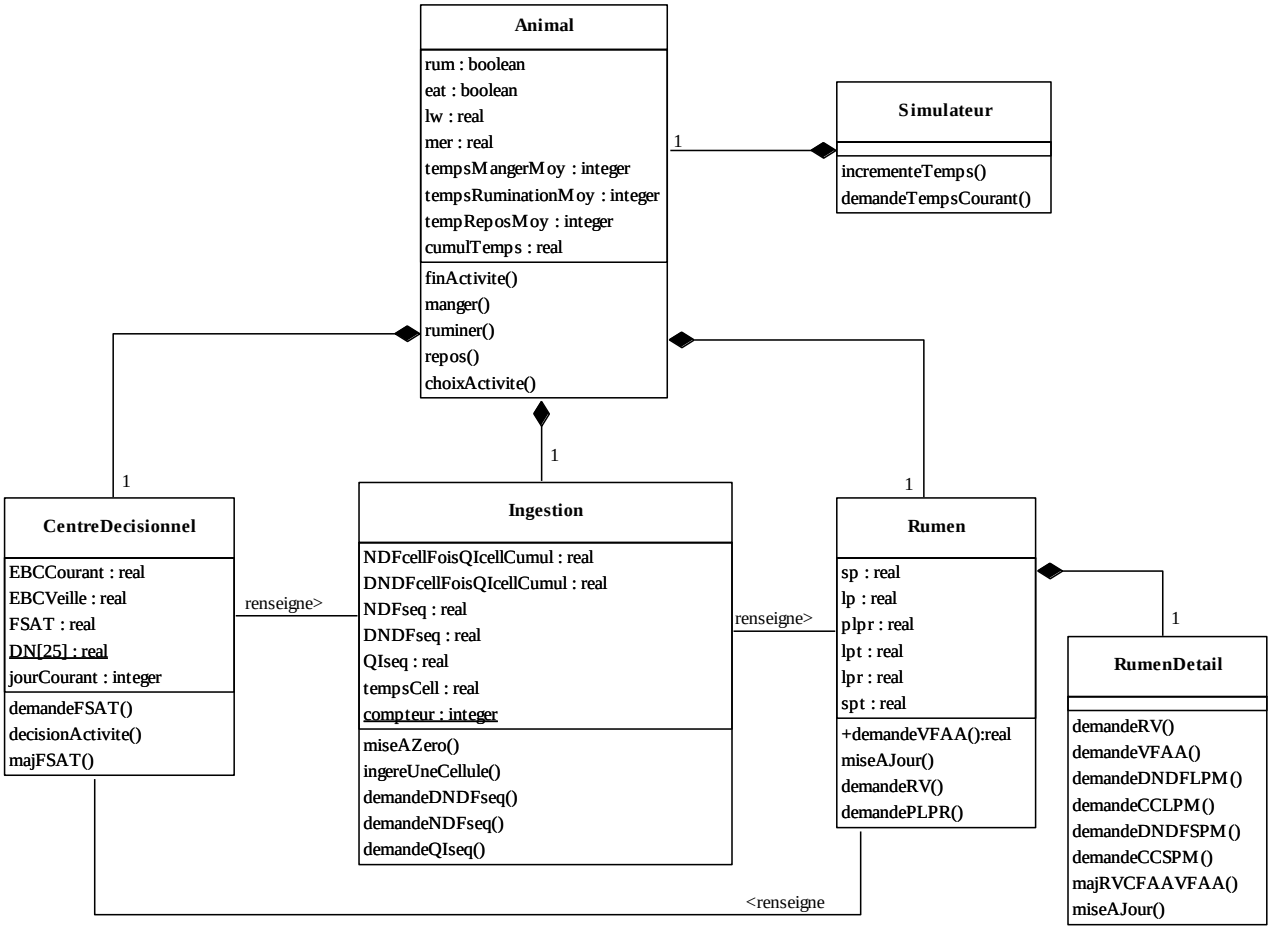


Figure 1 : Diagramme de classe de la biologie interne de l'animal

D'autres programmes font partie du module A mais n'y sont pas encore intégrés. C'est le cas de l'automate décisionnel. Il a été réalisé lors du stage de Lucie Masson. Cet automate permet de choisir parmi toutes les activités que le ruminant peut accomplir. Il a été réalisé de manière à pouvoir simplement rajouter de nouvelles activités si nécessaires. Le principe est simple (Figure 2). Chaque fois que l'animal doit prendre une décision, il utilisera cet automate. Il vérifiera d'abord s'il respecte toujours les règles sociales (éloignement par rapport à ces congénères, action d'un meneur). Si tel n'est pas le cas, sa prochaine action sera de faire en sorte que ces règles soient respectées.

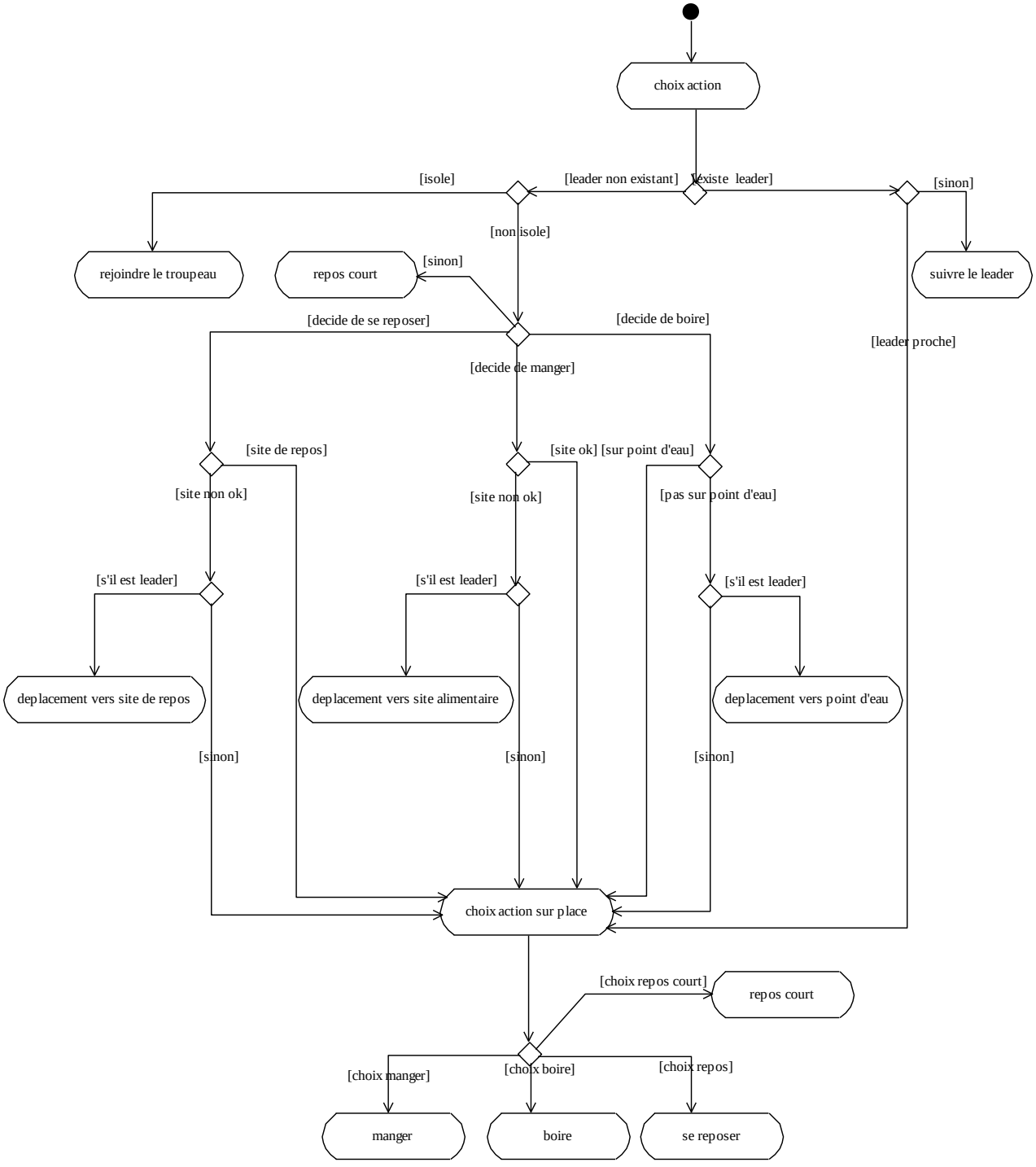


Figure 2 : Diagramme d'activité de l'automate décisionnel

Dans le cas contraire, il choisira entre les différentes activités possibles. En fonction de l'état interne de l'animal, une probabilité plus ou moins importante est attribuée à chaque activité potentielle. Un tirage aléatoire sélectionne le choix final. Des tests ont été réalisés et ont validé son bon fonctionnement.

Un autre programme, le générateur de trajectoires, a aussi été développé lors du stage de Lucie Masson, pour simuler les déplacements d'un animal. Ce programme consiste à faire choisir une

cellule à l'animal parmi celles qui se trouvent devant lui afin de la défolier. Pour cela, il prend en compte la qualité alimentaire des cellules, leur distance et leur direction (Figure 3).

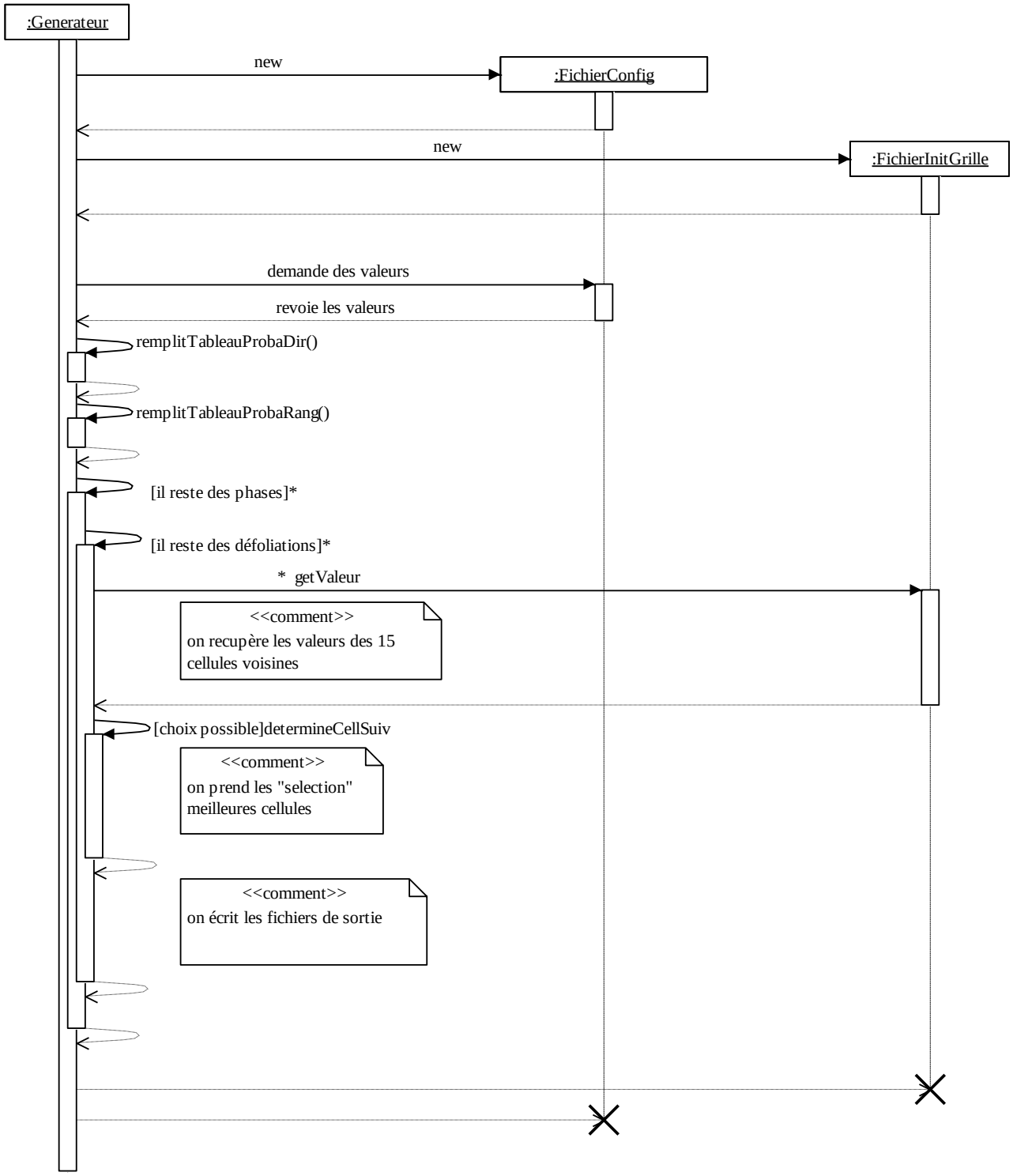


Figure 3 : Diagramme de séquence du générateur de trajectoire

Le fichier de sortie de ce programme est utilisé par la visionneuse pour représenter la trajectoire.

1.1.2.2 Le module V

Il est terminé. Il possède trois fonctionnalités :

- La mise à jour de la parcelle
- La défoliation de la parcelle
- La mise à disposition des valeurs de chaque cellule

La première fonctionnalité représente la simulation de la pousse et de la sénescence de la végétation sur la parcelle. L'herbe a sa propre dynamique et, en dehors de toute interaction, elle doit être capable d'évoluer en fonction des aléas climatologiques. La mise à jour de la parcelle est quotidienne et s'effectue la nuit.

La deuxième fonctionnalité permet de prendre en compte les actions effectuées par les ruminants sur la parcelle. Ces informations sont fournies par l'animal qui indique quelles cellules ont été défoliées et la quantité de matière prélevée sur chacune d'elles. La gestion de la parcelle se fait donc ainsi, par une succession de défoliations suivie d'une mise à jour quotidienne. Bien que la mise à jour soit faite quotidiennement, les défoliations sont prises en compte ponctuellement sur chaque cellule.

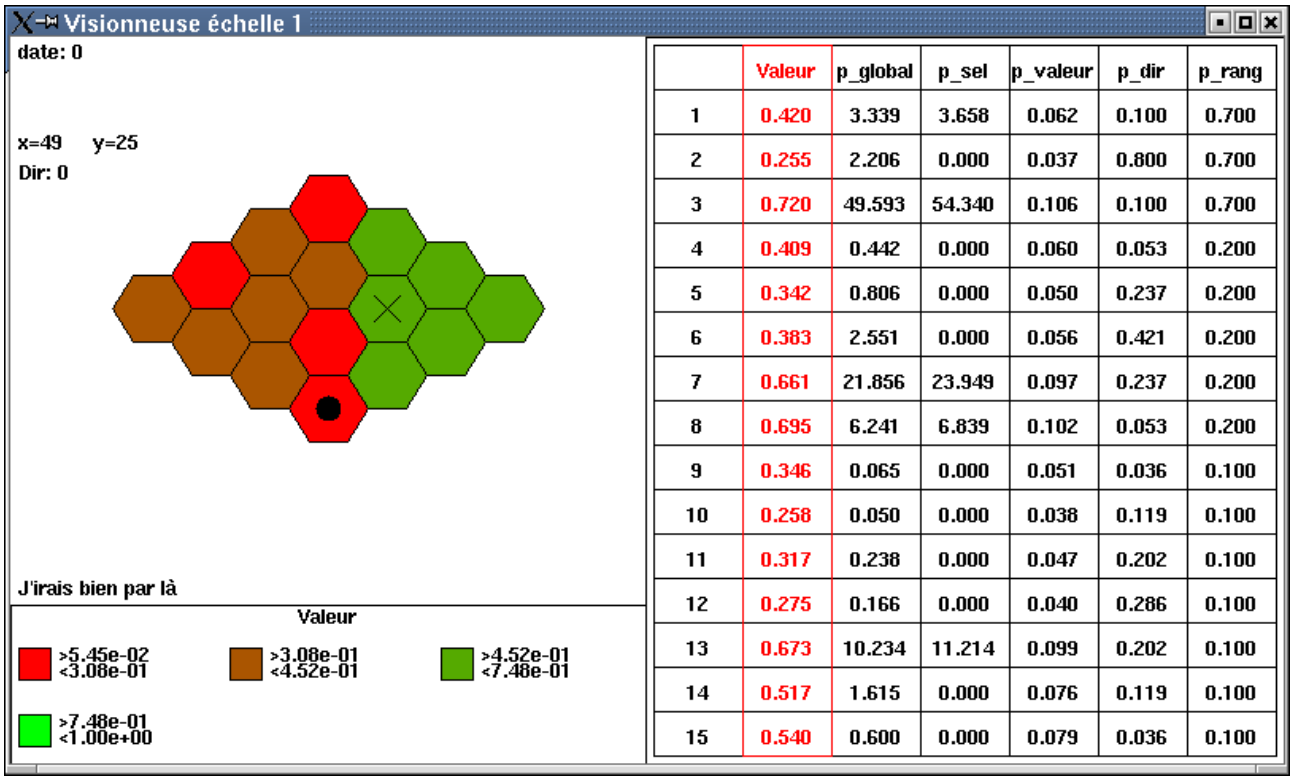
La dernière fonctionnalité permet à l'animal de demander quelles sont les valeurs des différentes cellules qui se trouvent devant lui. Pour cela, il fournit au module sa position sur la carte ainsi que la direction selon laquelle il regarde. La réponse est récupérée dans un fichier de sortie spécifié dans la demande.

1.1.2.3 La visionneuse

Cet outil est un programme technique et n'est inclus dans aucun module.

Il sert à visualiser l'espace, la végétation et les déplacements des animaux. Il a été développée lors d'un projet de cent heures par Lucie Masson et Alice Villéger [2] , puis légèrement modifiée lors de mon projet avec Pierre Chatelier [5] . Il gère plusieurs échelles de perception.

L'échelle 1 donne la vision du point de vue du ruminant (Figure 4). La représentation graphique montre les quinze cellules qui se trouvent devant lui, ainsi que leurs valeurs. Le rond noir représente la position de l'animal avant que son choix ne soit fait. La croix montre la cellule choisie.



	Valeur	p_global	p_sel	p_valeur	p_dir	p_rang
1	0.420	3.339	3.658	0.062	0.100	0.700
2	0.255	2.206	0.000	0.037	0.800	0.700
3	0.720	49.593	54.340	0.106	0.100	0.700
4	0.409	0.442	0.000	0.060	0.053	0.200
5	0.342	0.806	0.000	0.050	0.237	0.200
6	0.383	2.551	0.000	0.056	0.421	0.200
7	0.661	21.856	23.949	0.097	0.237	0.200
8	0.695	6.241	6.839	0.102	0.053	0.200
9	0.346	0.065	0.000	0.051	0.036	0.100
10	0.258	0.050	0.000	0.038	0.119	0.100
11	0.317	0.238	0.000	0.047	0.202	0.100
12	0.275	0.166	0.000	0.040	0.286	0.100
13	0.673	10.234	11.214	0.099	0.202	0.100
14	0.517	1.615	0.000	0.076	0.119	0.100
15	0.540	0.600	0.000	0.079	0.036	0.100

Figure 4 : Echelle 1 de la visionneuse

Le tableau permet de voir des informations sur chacune des cellules. Les valeurs des cellules après perception sont sur la première colonne. Les colonnes « p_valeur » « p_dir » et « p_rang » correspondent à la probabilité qu'a l'animal de se rendre sur la cellule respectivement en fonction de la valeur, de la direction et du rang. La colonne « p_global » représente la probabilité globale qui va lui permettre d'effectuer un choix. La colonne « p_sel » correspond aux probabilités recalculées après sélection des N meilleures cellules, N étant un paramètre du fichier de configuration.

L'échelle 2 représente une partie de la parcelle (Figure 5). Cette échelle permet d'observer les déplacements du ruminant grâce aux fichiers de sortie du générateur de trajectoires.

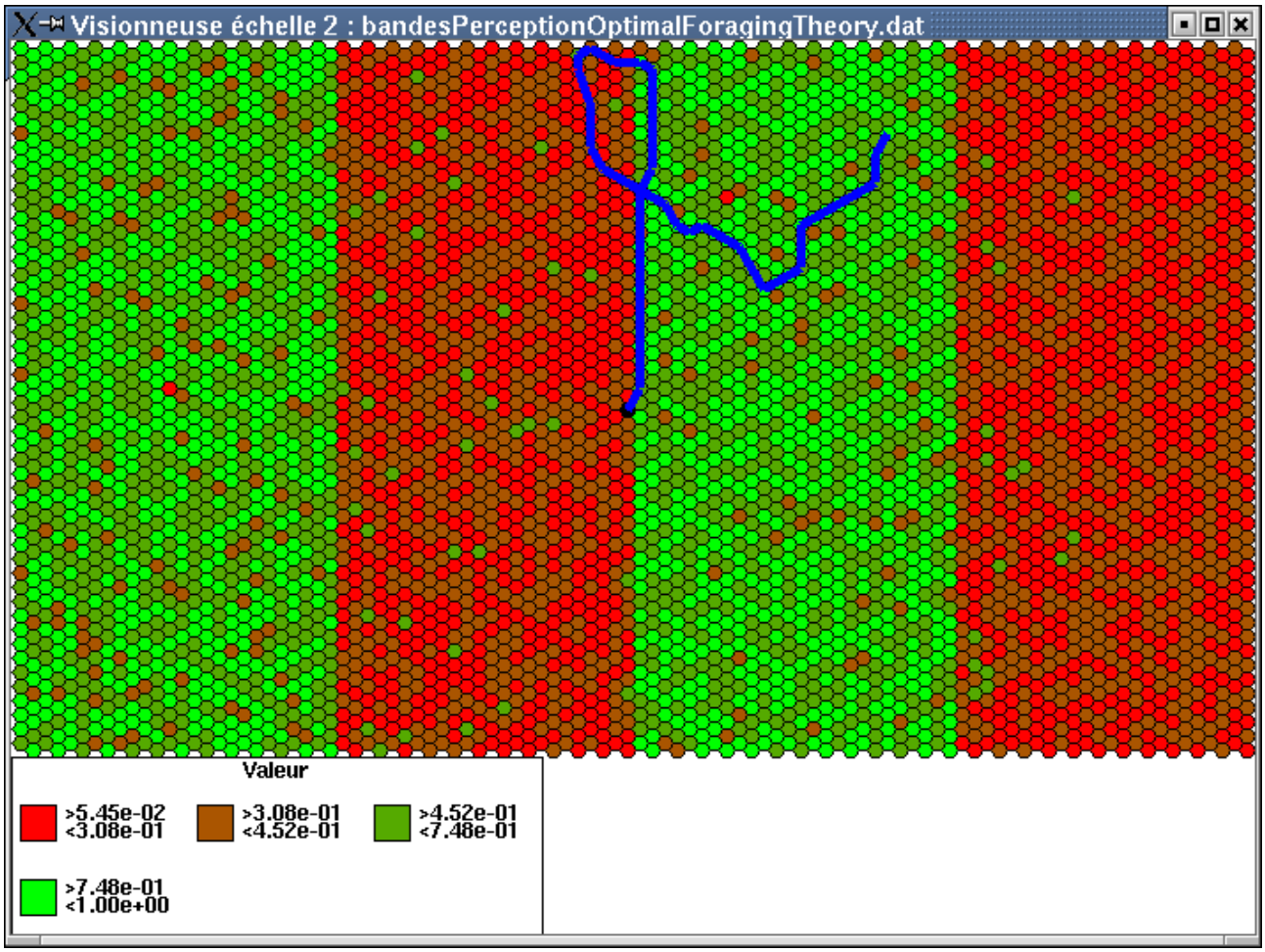


Figure 5 : Echelle 2 de la visionneuse

L'échelle 3 n'est pas encore implémentée mais était destinée à représenter des parcelles de grande taille.

1.1.2.4 Le générateur de carte

Son rôle est de créer une carte de végétation vue par l'animal. Il a été développé lors de mon projet avec Pierre Chatelier [5] . Il se décompose en deux parties. La première consiste à construire une parcelle à partir d'une définition de site en forme de rectangle. Chaque ligne du fichier représente un site, pour chaque site, on donne les coordonnées du point supérieur gauche et celles du point inférieur droit. On précise aussi la quantité de biomasse et la répartition de celle-ci parmi les quatre compartiments. A chaque valeur est affectée une perturbation gaussienne pour rendre la parcelle plus ou moins hétérogène. Un programme appelé « créateur » se charge de cette fonction. Cette carte est constituée de cellules qui possèdent des paramètres caractéristiques appelés compartiment. La seconde partie, appelée « percepteur », consiste à prendre la carte ainsi générée et à appliquer dessus une fonction de perception. En sortie du « percepteur », on obtient une carte ou chaque cellule est représentée par une valeur correspondant à la représentation que se fait l'animal de la qualité de la cellule. L'organisation des fichiers est décrite Figure 6.

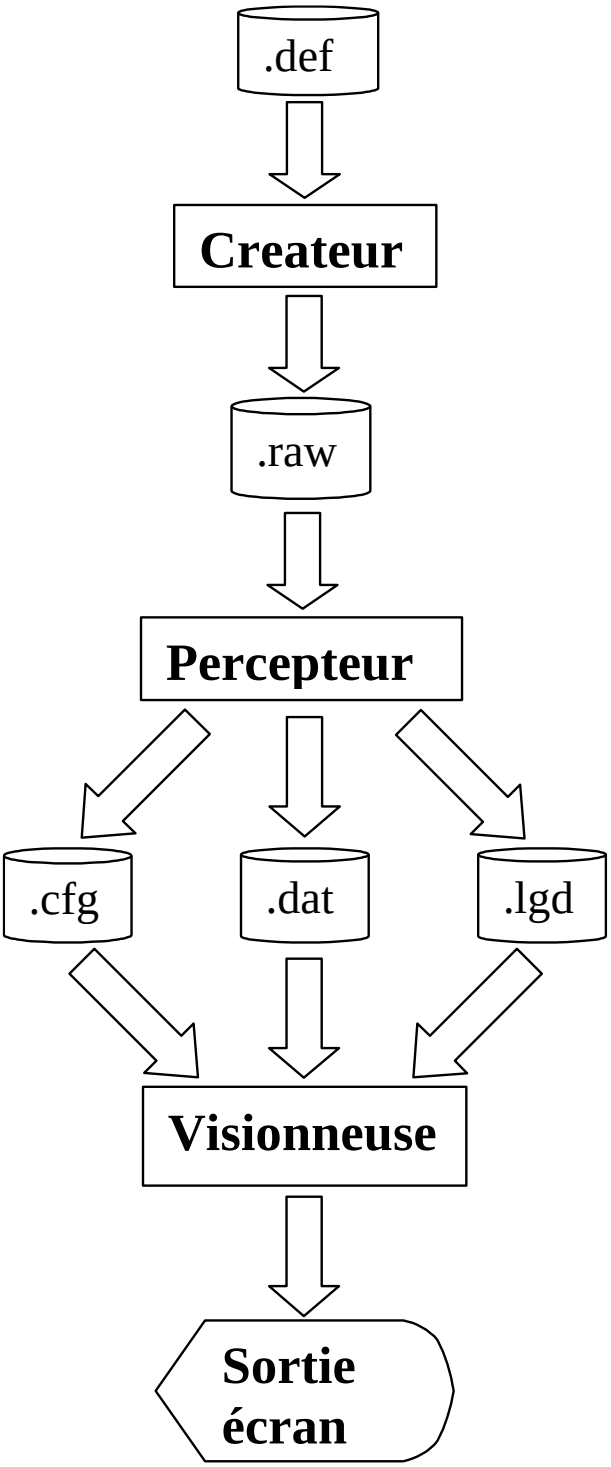


Figure 6 : Organisation des fichiers du générateur de carte

1.1.2.5 Le module S

C'est le module social et spatial du troupeau. Il se charge de gérer le comportement des animaux avec le reste du troupeau.

Il prend en compte la mémoire spatiale du ruminant. La modélisation de celle-ci a été développée par Lucie Masson [3] . Pour le ruminant, c'est la méthode de mémorisation d'un site déjà visité. Il pourra ainsi effectuer des déplacements longs pour aller d'un site à l'autre.

Ce programme ne sera pas utilisé lors de mon stage.

1.2 Outils de développement utilisés

Pour réaliser ce simulateur, on a choisi d'effectuer le développement du programme sous linux. La version utilisée auparavant était la version Mandrake 7.2, mais pour un souci de sécurité et pour avoir un meilleur compilateur, on est passé à la version Mandrake 8.1. Le langage choisi pour coder le simulateur est le C++ avec le compilateur gcc version 2.96 de la version actuelle de linux. Dans un souci de portabilité du code aucune bibliothèque non standard autre que la Xlib (utilisée pour la représentation graphique) n'a été utilisée. Par conséquent, le simulateur qui ne tourne actuellement que sous linux pourra facilement être porté sous Windows en modifiant les chemins d'accès aux fichiers.

1.3 Travail demandé

On s'aperçoit que de nombreux programmes existent déjà sans que leurs connexions ne soient réalisées. Une analyse est nécessaire pour relier et interconnecter toutes ces parties.

Pour permettre des étapes de validation progressives, nous allons créer différents prototypes de plus en plus complexes.

Le premier objectif sera de créer une connexion avec une carte de végétation n'évoluant pas pour vérifier et valider les paramètres de choix et de perception.

Le second objectif sera de valider les déplacements d'un animal sur la parcelle, indépendamment de son état physiologique. Pour cela, nous interconnecterons le générateur de trajectoire avec le module V. C'est donc le simulateur végétal qui fournira à l'animal les paramètres des cellules qu'il voit à proximité.

Le troisième objectif sera de créer un modèle associant la dynamique interne à un animal avec son comportement dans son environnement. Pour cela, il faudra ajouter au prototype précédent le module A précédemment développé ainsi que l'automate décisionnel.

Afin de pouvoir aider à la validation, les différents prototypes devront fournir des sorties utilisables par la visionneuse.

Chapitre 2: Travail effectué

2.1 Amélioration et création d'outils techniques

2.1.1 Connexion entre le générateur de trajectoire et la visionneuse

Le principe était de connecter la carte en sortie du « percepteur », c'est à dire une carte ne contenant qu'une seule valeur pour chaque cellule, au générateur de trajectoires. Ce dernier est capable, à partir d'une position de départ donnée, de simuler les choix qu'aurait fait un animal en observant les différentes cellules qui se trouvent devant lui. Enfin, il fallait que l'on puisse observer le résultat grâce à la visionneuse.

La difficulté réside dans la compatibilité entre les différents fichiers de configuration souvent longs et difficiles à utiliser. C'est pour cela que maintenant, le « percepteur » peut construire des fichiers de configuration nécessaires aux autres programmes avec des valeurs par défaut. Il ne reste ainsi plus qu'à les éditer et à modifier leurs contenus.

Pour faciliter ces opérations, un programme permettant d'automatiser les différentes tâches est maintenant disponible (voir Annexe II). Son rôle est de construire les fichiers de configuration qui ne sont pas déjà générés par les autres programmes. Il peut aussi lancer toutes les applications nécessaires avec le bon nombre de paramètres et les bons fichiers de configuration. Il est décomposé en deux parties, la première sert uniquement pour configurer, générer et percevoir une carte de végétation, la seconde permet de générer une trajectoire et de visualiser celle-ci sur la carte précédemment créée. L'avantage de cette décomposition est qu'on n'est pas obligé de recréer la carte de végétation chaque fois. Pour générer plusieurs trajectoires avec différents scénarios de perception, il suffit de passer directement à la deuxième étape. Le choix des différentes fonctionnalités se fait grâce à des menus selon le diagramme d'activité en Annexe II. De plus, le nom des fichiers de sortie est organisé de sorte que l'on puisse facilement retrouver quelle application est configurée par un fichier donné. Ils sont formés à partir d'un nom de base auquel on rajoute le nom de la fonction de perception et la bonne extension. Ainsi, chaque nom est associé à un nom de base et à une fonction de perception de façon à pouvoir les réutiliser facilement par la suite et à éviter de les effacer par mégarde. La Figure 7 montre comment, et par quelles applications, les différents fichiers de sorties sont utilisés.

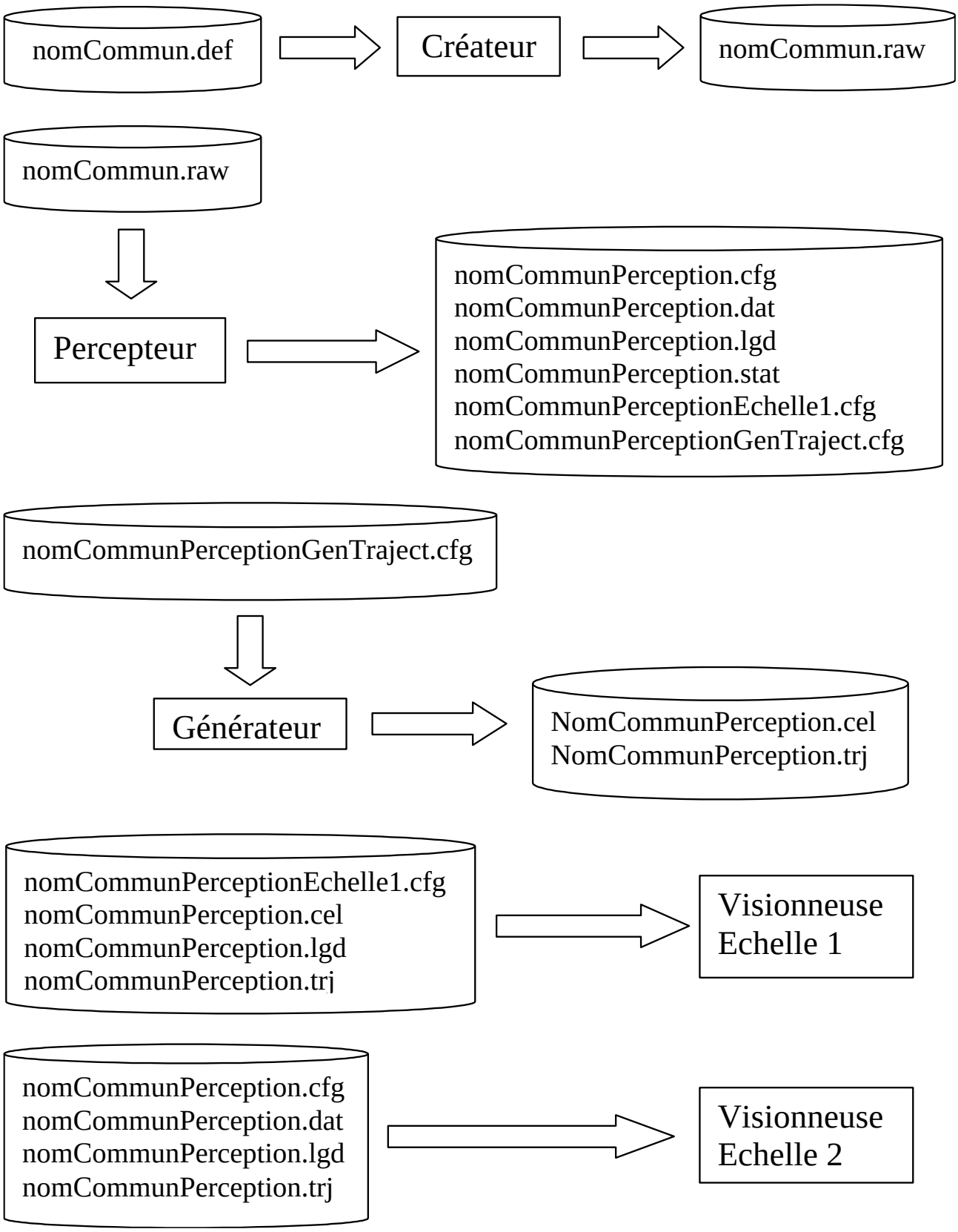


Figure 7 : Gestion des fichiers de sortie

2.1.2 Le générateur de statistiques

Pour pouvoir effectuer des tests et calibrer les différents paramètres, il faut avoir des outils statistiques permettant de comprendre les choix effectués par l'animal. Les informations brutes sont contenues dans deux fichiers de sortie des différents programmes. Le fichier portant l'extension « .cel » contient des informations sur toutes les cellules qui ont été dans le champ de vision du ruminant lors de la défoliation. Celui avec l'extension « .trj » regroupe les coordonnées des cellules visitées par l'animal. Ces fichiers étant assez volumineux, et pour éviter de faire manuellement des recoupements entre ces fichiers, un générateur de statistiques a été créé pour ne garder que les informations utiles aux tests.

Celles-ci sont recueillies et stockées dans un fichier avec un format « .csv » que peut éditer l'application Microsoft Excel. Dans ce fichier, on retrouve sur chaque ligne, les cellules défoliées par l'animal avec leurs probabilités d'être choisies ainsi que les coordonnées de l'animal au moment où il a fait ce choix. Au final, on obtient les répartitions des choix en fonction du rang, de la direction vers laquelle se trouve la cellule ou encore du site auquel appartient la cellule. La Figure 8 montre un exemple de fichier de sortie.

nombre de choix = 149
rang 1 = 104 (69.7987%)
rang 2 = 34 (22.8188%)
rang 3 = 11 (7.38255%)
direction millieu = 102.333 (68.6801%)
direction droite = 25.1667 (16.8904%)
direction gauche = 21.5 (14.4295%)
site 1 visité 82 fois (55.0336%)
site 2 visité 67 fois (44.9664%)

Figure 8 : Exemple de fichier de statistique

Pour chaque rang, on a le nombre de fois où il a choisi une cellule appartenant à ce rang ainsi que la fréquence de ce choix. De même pour les directions, sauf que certaines cellules se trouvent en partie à droite ou à gauche et en partie au milieu. Par conséquent, on observe des chiffres qui ne sont pas des entiers. Pour chaque site, on sait combien de fois le ruminant a choisi une cellule appartenant à ce site.

Il faut donner en paramètre à cette application, les noms de trois fichiers d'entrée. Le premier est le fichier de cellule, le suivant est le fichier de trajectoire et le dernier est le fichier de définition de la parcelle. J'ai intégré ce programme dans les différents prototypes en rajoutant un choix dans le menu et la possibilité d'éditer le fichier de sortie.

2.1.3 Méthodes du calcul des probabilités

Le scénario utilisé pour calculer la probabilité en fonction de la qualité, de la direction et du rang a évolué. Pour la qualité de chaque cellule, on prend les valeurs fournies par le simulateur végétal et on y applique une fonction de perception. Le tout est ensuite remis entre 0 et 1 en divisant par la somme des valeurs des cellules vues par l'animal. Les modifications se situent sur le calcul des probabilités du rang et de la direction. Avant, pour la direction, chaque cellule de chaque couronne

était initialisée selon les valeurs de « dir 1 » et « dir 2 ». Puis, pour chaque couronne, on faisait la somme de toutes les cellules pour ramener ces valeurs entre 0 et 1. Maintenant, la méthode de calcul est plus homogène puisque l'on répartit les valeurs initiales comme avant mais, on fait en sorte que la somme des probabilités de l'ensemble des cellules soit égale à 1. De même pour le calcul de probabilité du rang, on procède maintenant à une répartition homogène sur les cellules d'un même rang pour que la somme des cellules soit égale à 1.

Pour ce qui concerne la qualité de la cellule, l'approche heuristique donnée par les fonctions de perception semble correcte. Au final, ceci correspond à une détermination heuristique d'une probabilité de choix basée sur la qualité.

Par contre, bien que les probabilités de rang et de direction soient maintenant homogènes à celle de la qualité, il n'empêche qu'elles sont fausses et ne correspondent pas à ce que l'on attendait. En faisant des tests statistiques sur un nombre important de cellules et en partant avec des valeurs initiales déterminées, l'analyse des résultats donne une répartition statistique différente. Par exemple, avec une répartition initiale de 70% pour le rang1, 20% pour le rang2 et 10% pour le rang3. Après les tests, on observe une répartition de 67.98% pour le rang1, 21.5% pour le rang2 et 10.52% pour le rang3 sur un test statistique de 5000 cellules.

Ceci nous a amené à nous poser des questions sur ce calcul. L'idée de décomposer le choix en combinaison de deux probabilités différentes était mal appropriée. Les probabilités de rang et de direction sont liées et le choix d'une cellule doit se faire avec des probabilités conditionnelles.

Claude Mazel nous a proposé une solution qui consiste à créer une matrice de probabilité, en mettant les probabilités de rangs sur les colonnes et les probabilités de direction sur les lignes. En faisant le produit des lignes et des colonnes, on récupère dans chaque case de la matrice les probabilités concernant chaque cellule (Figure 9).

		Rang			
		1	**2**	**3**	**Total**
Direction	**Droite**	0,07	0,02	0,01	**0,10**
	Millieu	0,56	0,16	0,08	**0,80**
	Gauche	0,07	0,02	0,01	**0,10**
	Total	**0,70**	**0,20**	**0,10**	

Figure 9 : Matrice de probabilité

Contrairement à la méthode utilisée auparavant, on distingue maintenant la direction droite de la gauche. Ainsi, il est maintenant possible de donner des poids différents à ces deux probabilités, ce qui peut être utile pour faire des tests. Dans la version finale, ces deux valeurs seront les même car l'animal n'a pas de raison de choisir la droite plutôt que la gauche.

Par contre, cette méthode nous fourni neuf valeurs alors que l'animal voit quinze cellules devant lui (Figure 10).

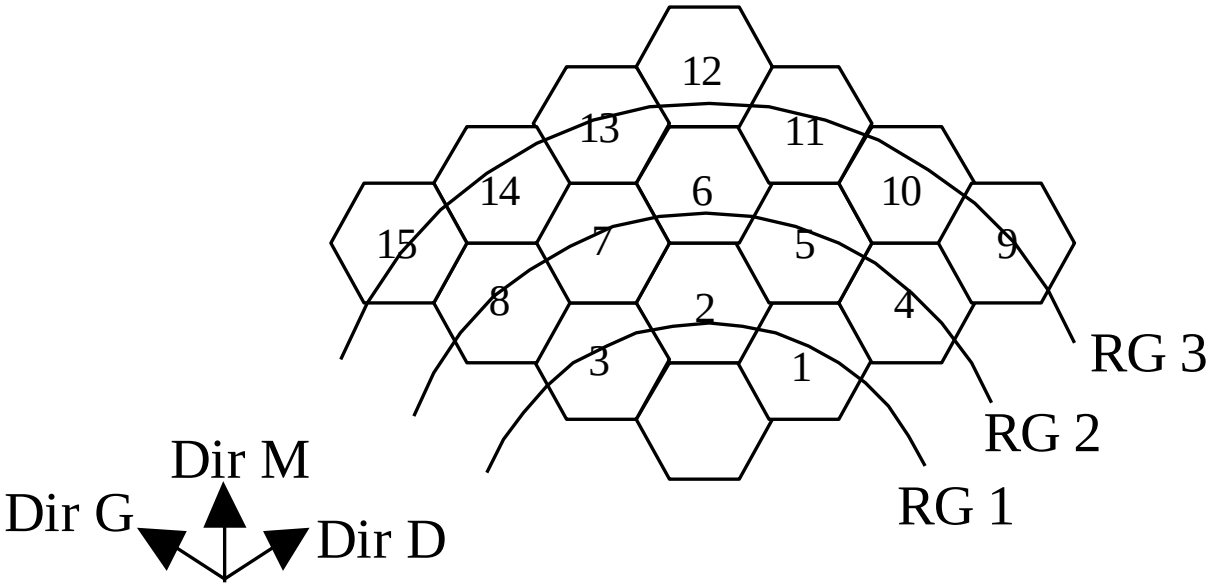


Figure 10 : Les quinze cellules

Cela vient du fait que le nombre de cellules par rang n'est pas constant et que cette méthode fonctionne bien pour un nombre constant de cellules. Donc, pour le premier rang, il suffit de prendre les probabilités de la première colonne. Pour le second rang, la cellule du milieu et celles des extrémités sont bien selon les directions du départ. Pour les cellules 5 et 7, on effectue le calcul comme suit.

dir cellule 5 = ½(dir cellule 6) + ½(dir cellule 4)
dir cellule 7 = ½(dir cellule 6) + ½(dir cellule 8)

Les probabilités globales sont les suivantes pour le rang2 :

Probabilité cellule 4 = 2/3(dir D2)
Probabilité cellule 5 = 1/2(1/2 dir M2) + 1/2(2/3 dir D2)
= 1/4(dir M2) + 1/3(dir D2)
Probabilité cellule 6 = 1/2(dir M2)
Probabilité cellule 7 = 1/2(1/2 dir M2) + 1/2(2/3 dir G2)
= 1/4(dir M2) + 1/3(dir G2)
Probabilité cellule 8 = 2/3(dir G2)

Avec :

- dir M2, la valeur obtenue dans la matrice à la ligne 2 et à la colonne 2
- dir D2, la valeur obtenue dans la matrice à la ligne 1 et à la colonne 2
- dir G2, la valeur obtenue dans la matrice à la ligne 3 et à la colonne 2

Les coefficients sont calculés de tel sorte que la somme des cinq probabilités soit égale à la probabilité du rang 2 (20%).

Pour le rang3, le calcul est similaire, mais, les coefficients sont différents.
On obtient par conséquent le tableau suivant (Figure 11)

	numéro des cellules	qualité cellules	probabilité qualité	probabilité dir-rang	valeur globale	choix finaux		choix sélection
Rang 1	1	2,327	6,67%	7,00%	9,218E-08	**7,00%**	70,00%	**8,47%**
	2	2,327	6,67%	56,00%	7,374E-07	**56,00%**		**67,74%**
	3	2,327	6,67%	7,00%	9,218E-08	**7,00%**		**8,47%**
Rang 2	4	2,327	6,67%	1,33%	1,756E-08	1,33%	20,00%	0,00%
	5	2,327	6,67%	4,67%	6,145E-08	4,67%		0,00%
	6	2,327	6,67%	8,00%	1,053E-07	**8,00%**		**9,68%**
	7	2,327	6,67%	4,67%	6,145E-08	**4,67%**		**5,65%**
	8	2,327	6,67%	1,33%	1,756E-08	1,33%		0,00%
Rang 3	9	2,327	6,67%	0,50%	6,584E-09	0,50%	10,00%	0,00%
	10	2,327	6,67%	1,22%	1,610E-08	1,22%		0,00%
	11	2,327	6,67%	1,94%	2,561E-08	1,94%		0,00%
	12	2,327	6,67%	2,67%	3,512E-08	2,67%		0,00%
	13	2,327	6,67%	1,94%	2,561E-08	1,94%		0,00%
	14	2,327	6,67%	1,22%	1,610E-08	1,22%		0,00%
	15	2,327	6,67%	0,50%	6,584E-09	0,50%		0,00%
	TOTAL	**34,905**	**100,00%**	**100,00%**	**1,317E-06**	**100,00%**		**100,00%**

Figure 11 : Tableau des probabilités

2.2 Analyse du domaine

Les méthodes utilisées pour effectuer les différentes analyses sont toutes basées sur le langage UML (Unified Modeling Language). Ce langage est basé sur la méthode UP (Unified Process). Elle consiste à modéliser les concepts abstraits comme le langage objet en se plaçant à un haut niveau d'abstraction pour être le plus général possible dans la description du modèle. Tous les diagrammes d'analyse respectent au mieux ce concept.

L'objectif du projet, prévu depuis le début, est de regrouper les modules A et V pour qu'ils puissent communiquer ensemble. Ceci est maintenant possible puisqu'ils sont tous les deux quasiment terminés.

L'analyse du domaine nous permet de voir quelles classes communiquent entre elles dans les différents modules. L'acteur principal du module A est l'animal. C'est lui qui dans l'avenir devra dialoguer avec les autres modules. Les autres classes du module A sont toutes reliées à l'animal et représentent généralement des compositions de l'animal. Du coté du simulateur végétal, c'est la parcelle qui tient le rôle principal et qui sera en relation avec l'animal. Le diagramme (Figure 12) suivant vient donc logiquement représenter cette interconnexion.

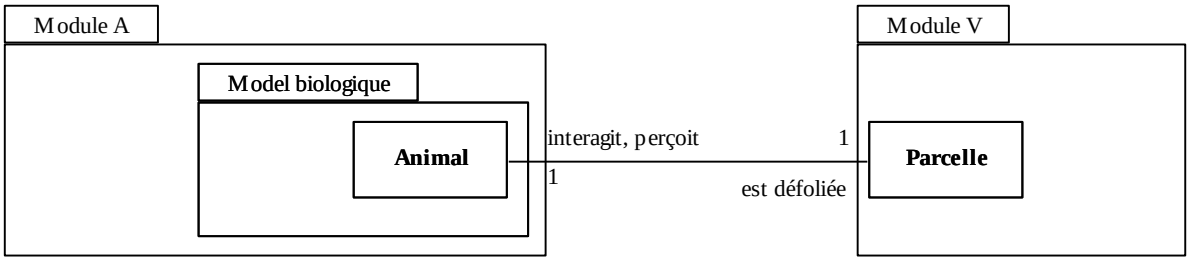


Figure 12 : Analyse du domaine, relation entre les modules A et V

L'animal n'est pas directement placé dans le module A. Il est dans le paquetage « model biologique » car, il est modélisé comme un individu au sens biologique. Il est par conséquent normal de le mettre dans un paquetage spécifique. Cette analyse montre les relations entre le module A et le module V. Par contre la connexion ne peut se faire directement avec l'animal pour l'instant.

2.3 Interconnexion entre le générateur de trajectoire et le simulateur végétal

2.3.1 Principe

Le module V simule la croissance végétale et a besoin de plusieurs fichiers de configuration. En particulier, le simulateur végétal utilise une carte sur laquelle on indique à quel site appartient chaque cellule. Ce fichier d'initialisation porte l'extension « .csv » et peut être donc visualisé ou créé directement sous Excel. Par contre, pour éviter les erreurs, j'ai modifié le « créateur » pour qu'il puisse générer une carte du type précédent à partir d'une définition de carte par site. La différence réside dans le fait qu'on garde uniquement les coordonnées du site et le paramètre associé à la biomasse est remplacé par un numéro de site. Tous les autres paramètres ne sont plus nécessaires. L'avantage est que maintenant, il est possible de connaître exactement à quel site appartient chaque cellule dans le générateur de statistiques. Il suffit de récupérer le numéro du site contenu dans ce fichier. Ainsi, on pourra beaucoup plus facilement reconnaître les sites visités par l'animal.

Le générateur de trajectoires doit être modifié pour ne s'adresser qu'au simulateur végétal au lieu de récupérer les valeurs des cellules grâce à un fichier carte. Les simulateurs animal et végétal doivent rester indépendants et par conséquent, ils ne doivent communiquer directement entre eux (Figure 13), il faut donc un intermédiaire entre ces simulateurs pour communiquer.

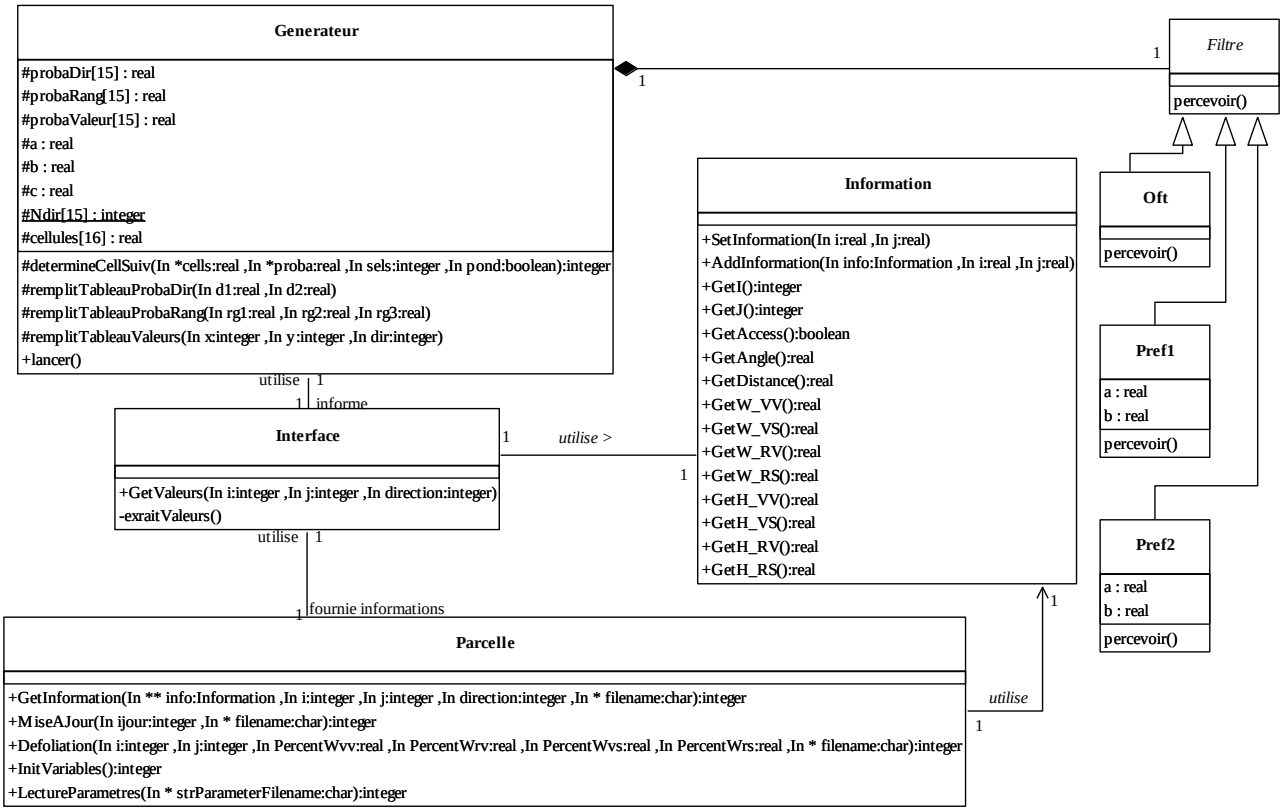


Figure 13 : Diagramme de classe de l'interconnexion entre les modules A et V

Ce diagramme a été réalisé après la phase d'analyse, il est susceptible d'évoluer par la suite. La classe information est utilisée par le simulateur végétal pour y stocker les données concernant la cellule. Il est donc préférable de la réutiliser dans le générateur de trajectoire, plutôt que de créer un autre type de conteneur.

Les différentes fonctions de perception comme OptimalForagingTheory ou les scénarios de préférence 1 et 2 dérivent d'une classe Filtre (Figure 21). Ces fonctions sont ensuite utilisées pour percevoir les données fournies par le simulateur végétal (Figure 14).

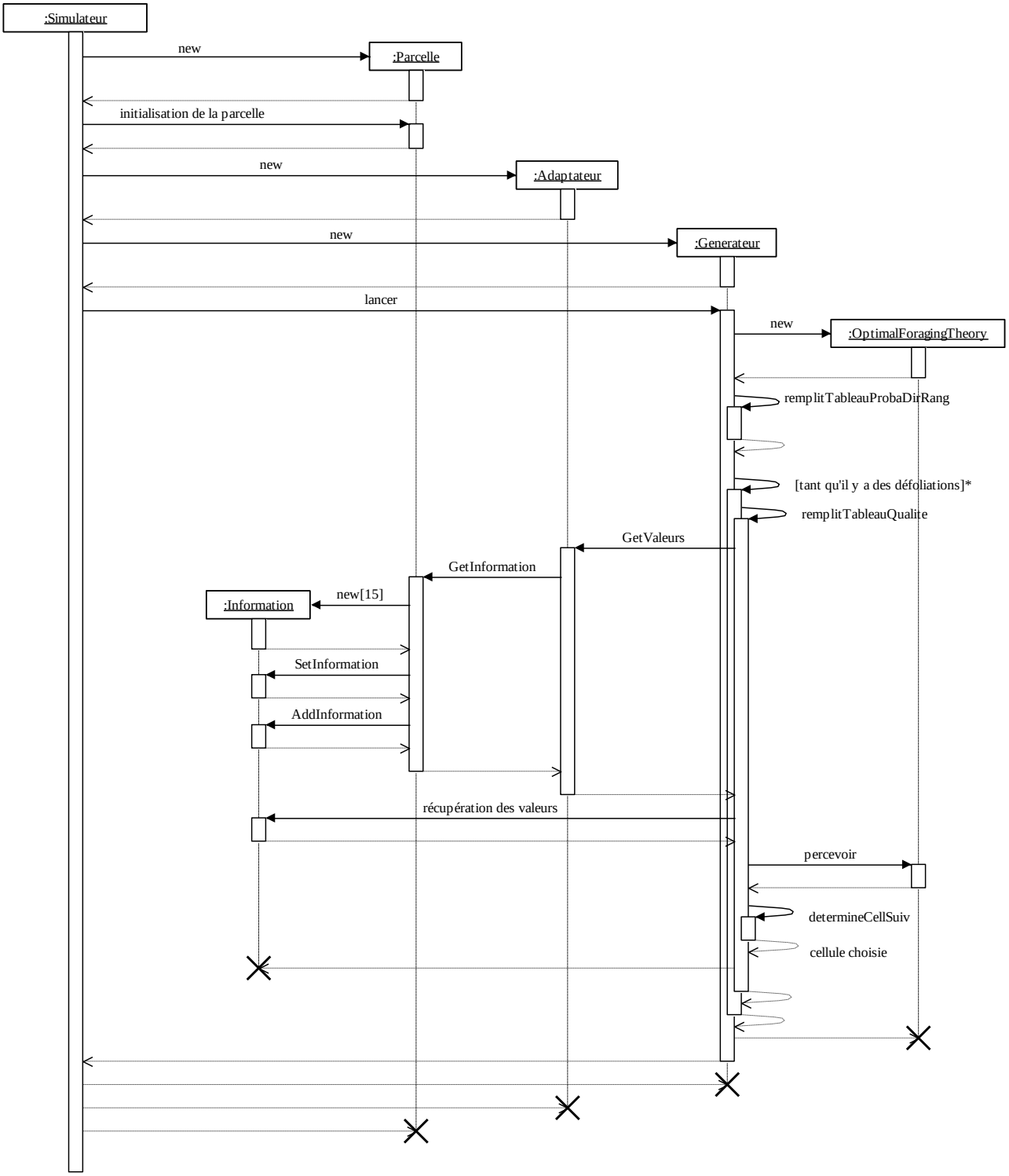


Figure 14 : Diagramme de séquence de l'interconnexion entre les modules A et V

Ici, la classe « OptimalForagingTheory » a été prise comme exemple mais, n'importe quelle autre fonction de perception peut être utilisée par le générateur de trajectoires. Par contre, pour chaque instance du générateur, on ne peut utiliser qu'une et une seule fonction de perception à la fois. La classe adaptateur correspond à l'interface entre les deux modules.

En comparant ce diagramme avec celui de la Figure 3, le générateur de trajectoires ne gère plus que les fonctions de perception, toutes les informations sont maintenant fournies par le simulateur végétal.

2.3.2 Besoin d'intermédiaire entre les deux modules

Comme on peut le voir sur la Figure 14, un adaptateur a été rajouté pour permettre la communication entre les différents modules. Ceci correspond à un patron adapter [6] . On utilise ce concept quand on veut mettre en relation deux classes qui n'ont pas les mêmes interfaces. Ainsi, on n'a pas à modifier les classes existantes. La Figure 15 montre l'aspect de ce type de patron.

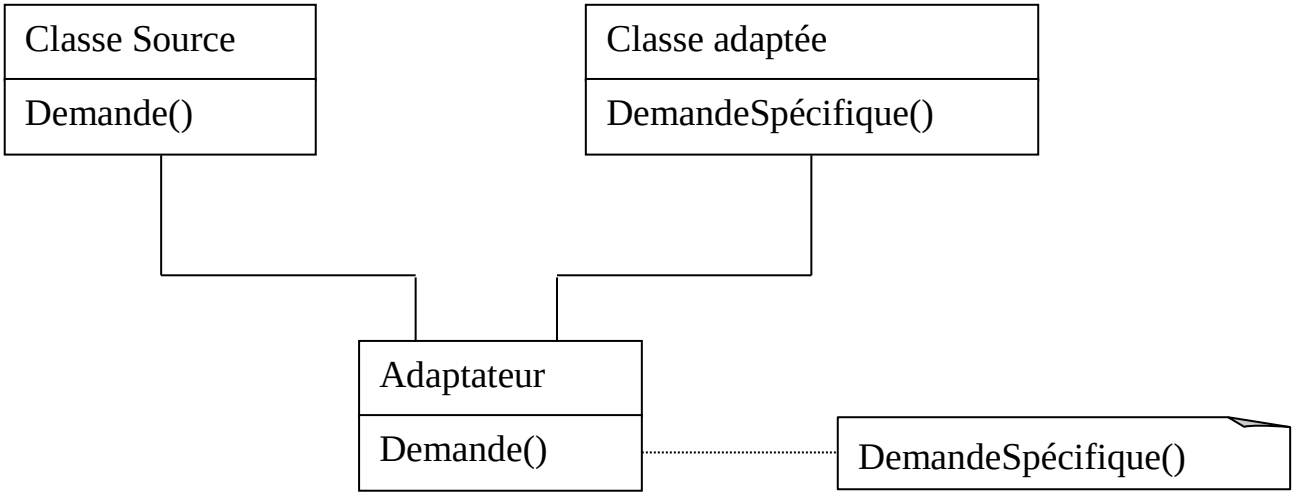


Figure 15 : Patron adapter

Cette classe adaptateur comporte des méthodes destinées à être utilisées par le générateur de trajectoires pour éviter que celui-ci n'ait à communiquer directement avec le module V. Elle possède entre autres une méthode pour demander des informations au simulateur végétal ou encore une méthode pour informer ce dernier du pourcentage de biomasse prélevé lors d'une défoliation. Elle sert aussi à mettre à jour la parcelle et donc peut être utilisée plus tard par d'autres applications ou modules du simulateur.

Le simulateur végétal garde aussi des traces des différentes actions effectuées. L'interface a pour rôle de masquer ceci à l'animal en évitant à l'utilisateur de devoir préciser le fichier de sortie. Les fonctions du simulateur végétal ont besoin de connaître une chaîne de caractères représentant le nom du fichier de sortie. Cette chaîne est indispensable, sinon une erreur apparaît. En mettant la gestion de ces fichiers dans l'adaptateur, on peut choisir d'incrémenter un chiffre dans le nom des différents fichiers de sortie ou de concaténer ces fichiers pour n'en garder qu'un seul et éviter d'avoir un fichier de sortie par défoliation.

Enfin, si une modification est apportée au simulateur végétal, il n'est pas nécessaire de modifier le générateur de trajectoires, il suffit de modifier et de recompiler la classe adaptateur.

De ceci, on obtient la solution suivante lors de la mise en œuvre (Figure 16).

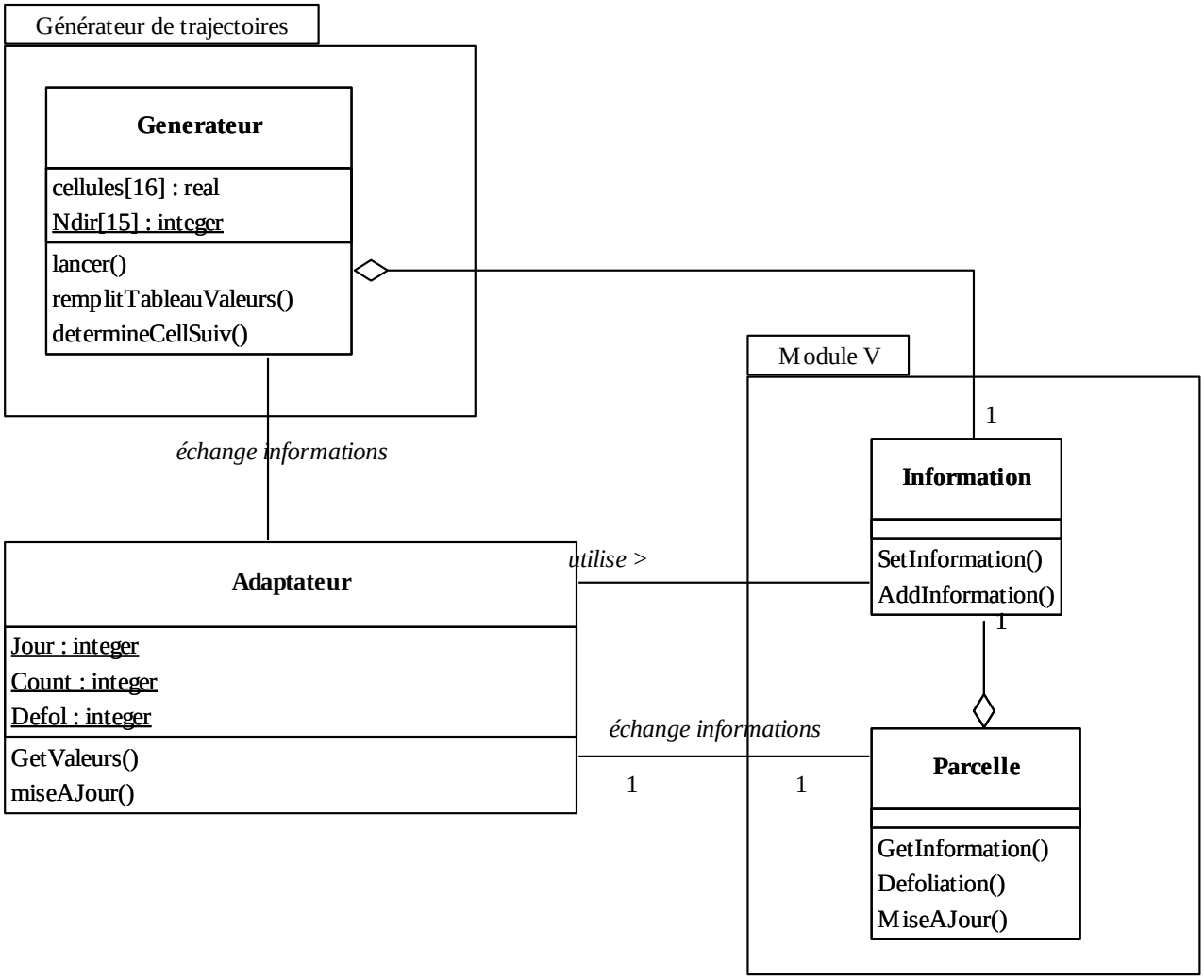


Figure 16 : Mise en œuvre de l'interconnexion

Les relations entre l'animal et l'adaptateur ne sont pas encore faites puisque le but de cette interconnexion est pour le moment de ne connecter que le générateur de trajectoires au module végétal. On constate aussi que le générateur ne communique qu'avec l'adaptateur qui se charge de transmettre les informations au simulateur végétal.

2.3.3 Détails d'implémentations

Certaines modifications ont du être effectuées et certains détails techniques méritent une explication.

2.3.3.1 Patron stratégie

Tout d'abord, pour pouvoir utiliser des fonctions de perception différentes tout en évitant de toutes les mélanger, un choix doit être effectué dans le programme principal avant l'instanciation de

la classe « Générateur ». Par conséquent, cette classe est devenue une classe template qui prend comme argument la classe correspondant à la fonction de perception que l'on veut utiliser. Toutes les fonctions de perception dérivent d'une classe mère abstraite « Filtre » qui contient une méthode virtuelle pure « percevoir(vecDouble& biomasse) ». Même s'il est possible de mettre une autre classe quelconque en paramètre du template, il est fortement conseillé de faire auparavant dériver cette classe de Filtre et de spécifier la méthode « percevoir ».

L'inconvénient des templates est qu'on est obligé de mettre le code dans le fichier d'en-tête, ce qui est logique car un template décrit du code qui sera généré lors de la compilation. Donc, le compilateur a besoin de connaître tout le code pour pouvoir créer la classe. Le désavantage est qu'on perd de la lisibilité et qu'il est par conséquent plus difficile de différencier ce qui correspond à un en-tête standard de ce qui correspond aux méthodes traditionnelles. Donc, la classe Générateur a quand même été décomposée en deux fichiers. Le premier est un fichier portant l'extension « .hpp », il décrit la classe et c'est lui qu'il faut inclure dans les autres classes. Le second fichier portant l'extension « .hxx », il décrit les méthodes de classe qui seront générées par la suite mais, pour respecter la contrainte ci-dessus, il est inclu dans le fichier Generateur.hpp et ne doit pas être inclu ailleurs. Cela permet de décomposer le code tout en gardant quelque chose de propre du point de vu langage objet.

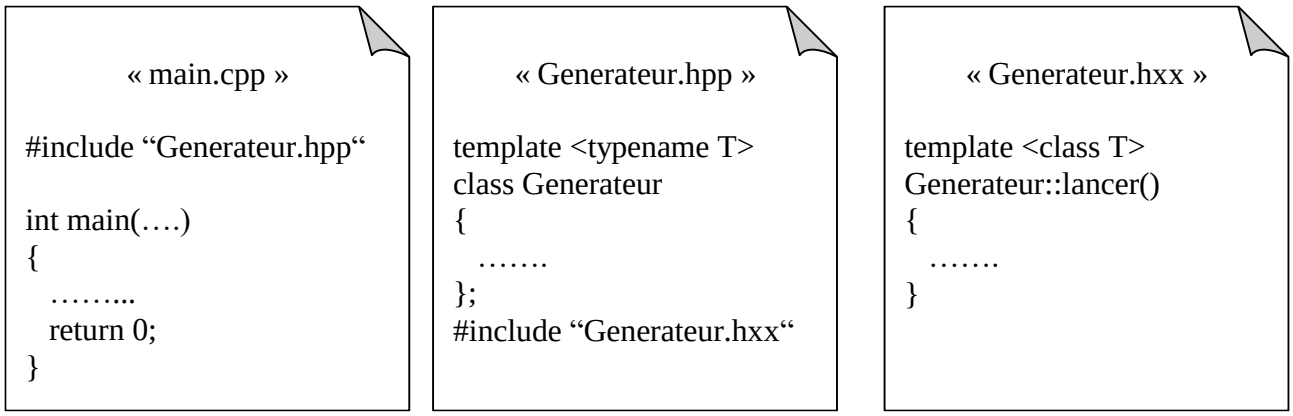


Figure 17 : Aperçu des différents fichiers pour la première solution

Une autre solution aurait été de décomposer comme d'habitude en deux fichiers (.hpp et .cpp) et d'inclure le fichier Générateur.cpp (qui contient les même données que mon fichier Générateur.hxx) au bon moment, c'est-à-dire à l'instant ou on instancie pour la première fois un objet de la classe Générateur. L'avantage est un gain de temps lors de la compilation. Le code étant inclus au dernier moment, il ne surcharge pas le compilateur de données dont il n'a pas besoin actuellement et donc, la compilation est plus rapide. Par contre, le résultat, bien que tout à fait correct du point de vu programmation, reste assez étrange et difficile à comprendre, ce qui pose problème pour le maintien du code par une autre personne. Il faut rappeler que ce projet a débuté deux ans plus tôt, qu'il ne sera pas terminé à la fin de mon stage et qu'une autre personne devra reprendre ce que j'ai fait. Donc, j'ai préféré garder la première solution car elle me semble plus simple et plus claire et on n'a pas besoin de gagner du temps lors de la compilation.

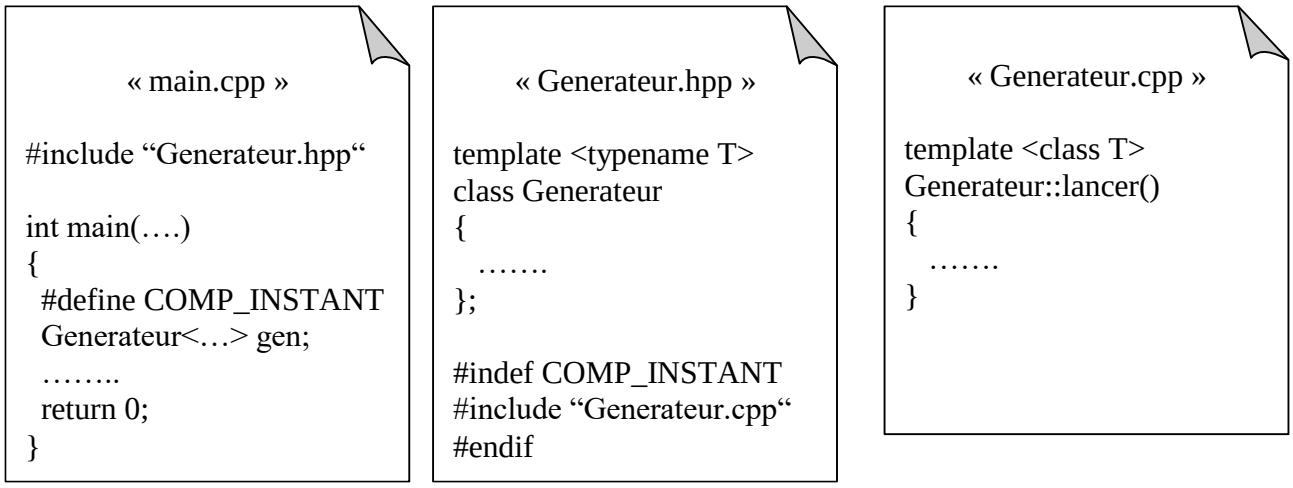


Figure 18 : Aperçu des différents fichiers pour l’autre solution

La solution adoptée ici correspond en analyse à un patron stratégie [6] . On utilise ce type de patron quand on veut utiliser un algorithme précis parmi plusieurs. On définit donc une classe par algorithme qui vont différer l’une de l’autre uniquement par leur comportement. Elles auront les même prototypes de méthodes qui coderont un algorithme différent. Deux solutions sont envisageables pour la classe qui va agréger l’ensemble du patron stratégie. Soit on passe en argument au constructeur une instance de classe du patron, soit on utilise une classe template de la même manière que ci-dessus. La Figure 19 montre l’aspect d’un patron stratégie.

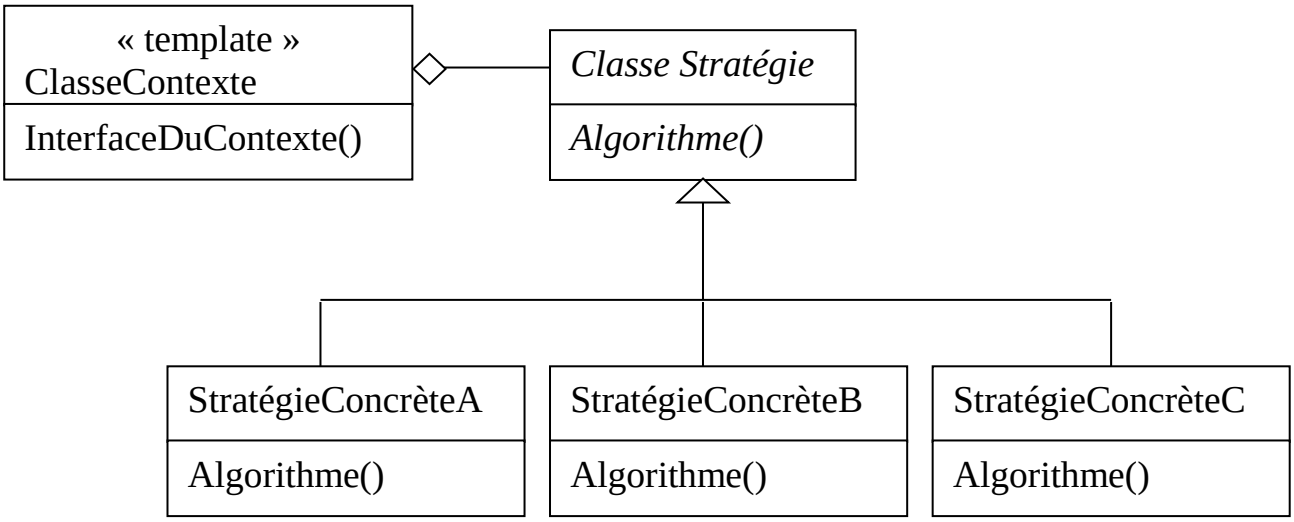


Figure 19 : Patron stratégie

La classe stratégie est abstraite tout comme les méthodes associées à l’algorithme. Celles-ci sont implémentées concrètement dans les classes filles. La classe contexte est déclarée template.

2.3.3.2 Paramètres des fonctions de perception

Pour revenir aux fonctions de perception, on remarque qu'elles sont différentes et qu'elles peuvent prendre plusieurs paramètres ou ne pas en prendre du tout (Figure 20). Ceci pose un problème dans l'instanciation d'une de ces classes dans le générateur de trajectoires.

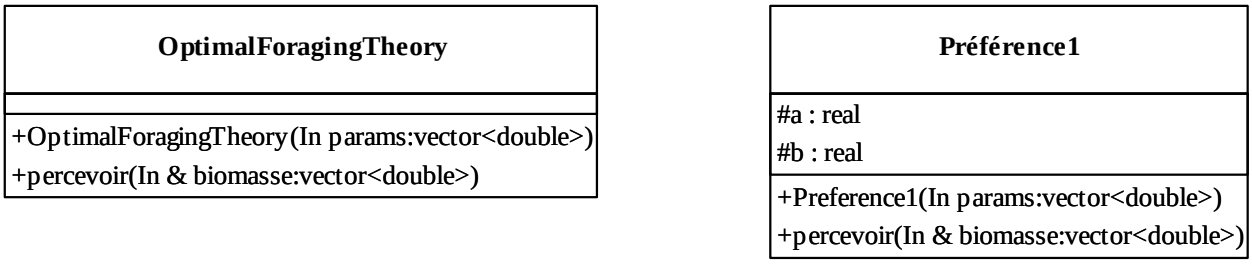


Figure 20 : Différentes fonctions de perception

Comment savoir sans connaître la classe qu'il faut un, deux, plus de deux ou aucun paramètre(s). Ceux-ci peuvent être disponibles ou pas et dans ce dernier cas prendre une valeur par défaut ou provoquer une erreur d'exécution. Dans cet exemple, la classe OptimalForagingTheory n'a aucun attribut alors que la classe Preference1 en a deux.

Une solution consiste à mettre la gestion des erreurs dans les classes dérivées de la classe Filtre. Pour l'initialisation, un vecteur de paramètres est donné en argument au constructeur (Figure 20), ainsi, dans la classe fille, on n'a plus qu'à gérer le nombre de paramètres que l'on souhaite utiliser et les valeurs par défaut que l'on souhaite prendre si des paramètres sont absents. Dans le générateur, il suffit de prendre les paramètres du fichier de configuration (on peut par conséquent mettre ces paramètres facultatifs) et les mettre dans un ordre bien défini dans notre vecteur avant de le passer en paramètre au constructeur de la classe. Dans ce cas, la règle est la suivante, on place dans l'ordre la valeur du paramètre « a » puis celle du paramètre « b ». Si le paramètre « b » est absent, on suppose que c'est volontaire et on ne met rien d'autre dans le vecteur. Les classes dérivées de « Filtre » prendront une valeur par défaut si possible ou provoqueront une erreur grâce à la classe d'erreur contenue dans Filtre. Si le paramètre « a » est absent, alors on suppose qu'il y a eu une erreur ou un oubli. Par conséquent, le paramètre « b » est sans doute erroné lui aussi, donc on ne met rien dans le vecteur et les classes Filtre prendront une valeur par défaut pour les deux paramètres. Cette solution a été choisie car théoriquement, on peut considérer qu'il n'y aura jamais d'erreur dans le fichier de configuration et donc tous les paramètres nécessaires seront présents, s'il en manque un, il doit y avoir une erreur. Or, la logique qui se cache derrière cet algorithme est que puisque l'on ne connaît pas quelles fonctions de perception on utilise, alors peu importe le nom du premier paramètre, il doit être présent. S'il ne l'est pas, il n'y a pas de raison que le second paramètre y soit, donc on considère qu'il n'y est pas. Ainsi, on obtient le même comportement que des fonctions à nombre d'argument variable et à valeur par défaut : seul les derniers arguments peuvent être mis par défaut. Par conséquent, si une future fonction a besoin de plus de paramètres, il suffit de les rajouter dans le vecteur du constructeur et de prendre les paramètres déjà existant comme premiers paramètres pour cette fonction. Il en est de même pour la fonction « percevoir ». Celle-ci accepte en entrée un vecteur (Figure 21) qui contient une valeur pour chaque compartiment (RS, VS, RV, VV). Rien n'empêche de rajouter par la suite les valeurs de NDF ou de DNDF.

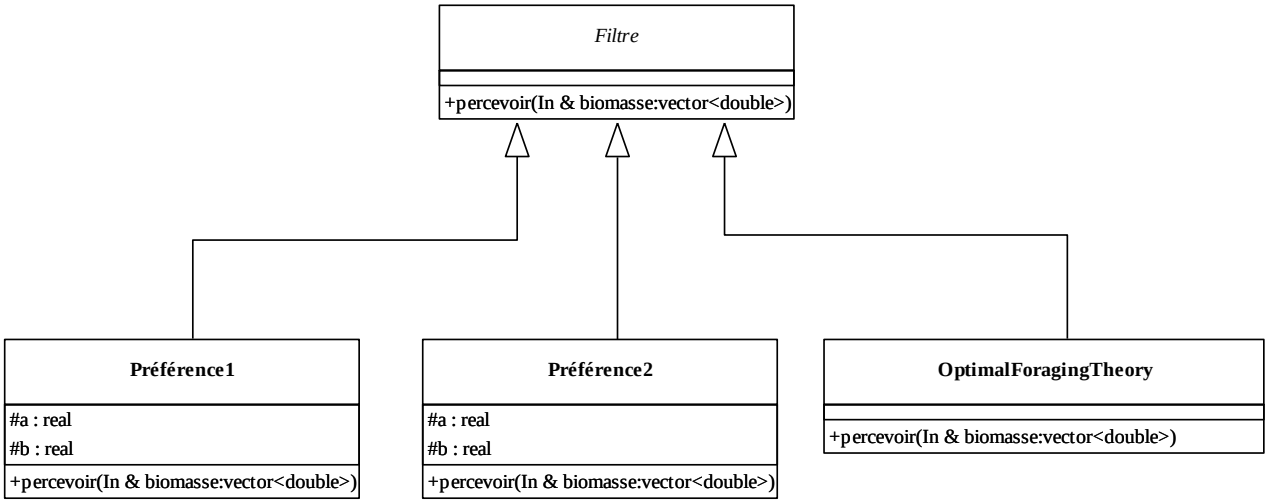


Figure 21 : Dérivation de la classe Filtre

2.3.3.3 Compatibilité des applications

L'interconnexion entre deux modules pose souvent des problèmes de compatibilité. Ici, les cellules sont numérotées pour pouvoir facilement les ranger dans un tableau. Il en est de même pour les directions : il est plus facile de manipuler un chiffre entre 0 et 5 qu'un nom. Mais, le sens utilisé pour compter les cellules et les directions était différent dans le simulateur végétal et le générateur de trajectoires (Figure 22 et Figure 23). Il a fallu harmoniser cela et choisir une seule solution plutôt que d'utiliser des astuces pour occulter le problème sans le résoudre et à long terme risquer de provoquer des erreurs. Finalement, on a choisi de conserver la numérotation utilisée par le simulateur végétal (Figure 22) et donc modifier en conséquence le générateur de trajectoire ainsi que la visionneuse.

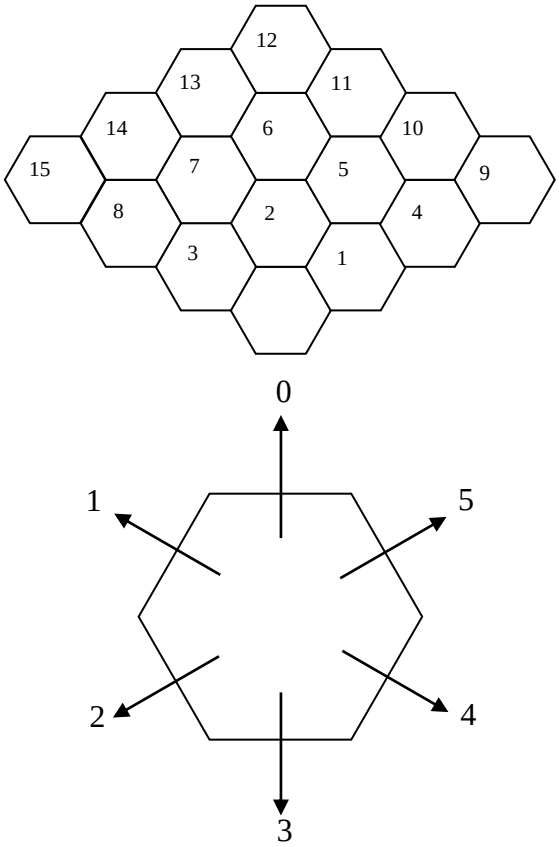


Figure 22 : Numérotation du simulateur végétal

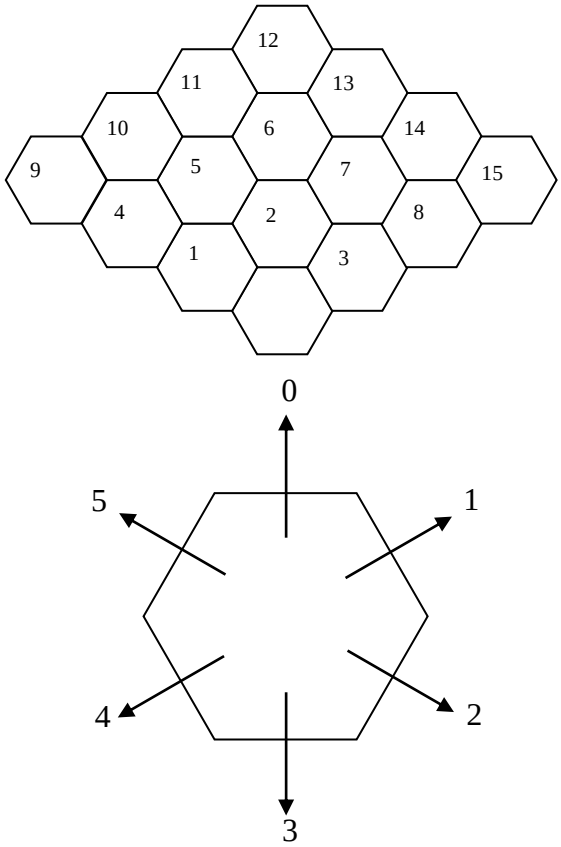


Figure 23 : Numérotation du générateur de trajectoires

Ici, la direction est absolue, c'est-à-dire orientée par rapport à la carte et non par rapport à l'animal. Ainsi, la direction 0 correspond au haut de la carte et 3 correspond au bas de la carte. Puis, pour le simulateur végétal, on a choisi 1 pour en haut à gauche, 2 pour en bas à gauche, etc. Cette numérotation est indispensable pour déterminer les coordonnées des cellules situées devant l'animal.

2.3.4 Tests et fichiers d'entrée-sortie

On garde une trace des exécutions des différentes applications grâce à des fichiers de sortie. Ceux-ci vont nous permettre, en fonction des fichiers d'entrée d'effectuer des tests et de valider le bon fonctionnement de l'interconnexion.

Tout d'abord, le simulateur végétal a besoin d'une carte représentant les types de végétation. Cette carte peut être fournie par le créateur de carte de manière simple. Pour mes tests, la végétation est composée de deux faciès répartis sur trois sites différents. Les paramètres de configuration de chaque faciès correspondent aux valeurs utilisées pour les tests du simulateur végétal. Puisque tout doit passer par le simulateur végétal et que celui-ci est opérationnel, les paramètres liés aux des faciès peuvent être quelconques dans un premier temps. Il faut juste vérifier que l'interconnexion s'est bien passé. Plus tard, ces valeurs seront à regarder de près car, comme c'est le point de départ de toute la simulation, si une erreur est commise ici, toute la simulation est faussée. D'autres fichiers sont nécessaires pour configurer le simulateur végétal, mais, ils concernent uniquement l'évolution de la végétation au cours du temps.

Le simulateur végétal est donc maintenant créé, il ne reste plus qu'à voir le générateur de trajectoires. Celui-ci prend un fichier de configuration en paramètre. Il a été un peu modifié pour y faire figurer les valeurs des paramètres utiles dans les fonctions de perception, alors que d'autres paramètres sont devenus inutiles.

En sortie, il faut récupérer les valeurs renvoyées par le simulateur végétal pour vérifier les fonctions de perception et récupérer les tableaux des différentes probabilités pour vérifier si le choix effectué par l'animal est correct.

Les tests ont été effectués sur quelques cas particuliers dans un premier temps, puis sur des cas plus généraux par la suite. Pour les fonctions de perception, seules les perceptions préférence 1 et OptimalForagingTheory ont été testées car préférence 2 est similaire à préférence 1. Les résultats sont conformes à ce que l'on attendait. On peut remarquer que toutes les cellules appartenant à un même faciès ont des valeurs après perception identiques. Ceci est normal puisque ces valeurs représentent le même type de végétation qui a poussé en même temps sous les même conditions climatiques. Pour avoir des valeurs différentes, il faudra donc effectuer les mêmes tests mais cette fois sur une parcelle où il y a déjà eu des interactions avec un troupeau et de nombreuses défoliations. C'est pour cela que les tests ont été effectués sur la limite entre deux faciès. Ainsi, on a des valeurs de cellules différentes, sinon, le choix de l'animal s'effectue toujours entre des cellules de même qualité.

La suite revient juste à vérifier que les calculs effectués par le générateur de trajectoires sont bons. Pour cela, la Figure 24 montre les valeurs calculées « à la main » alors que la Figure 25 montre les mêmes résultats calculés par le générateur de trajectoires et visualisés par la visionneuse. Ces résultats ont été obtenus avec la dernière version du prototype. Les modifications apportées sur le calcul des probabilités de direction et de rang ont donc été prises en compte.

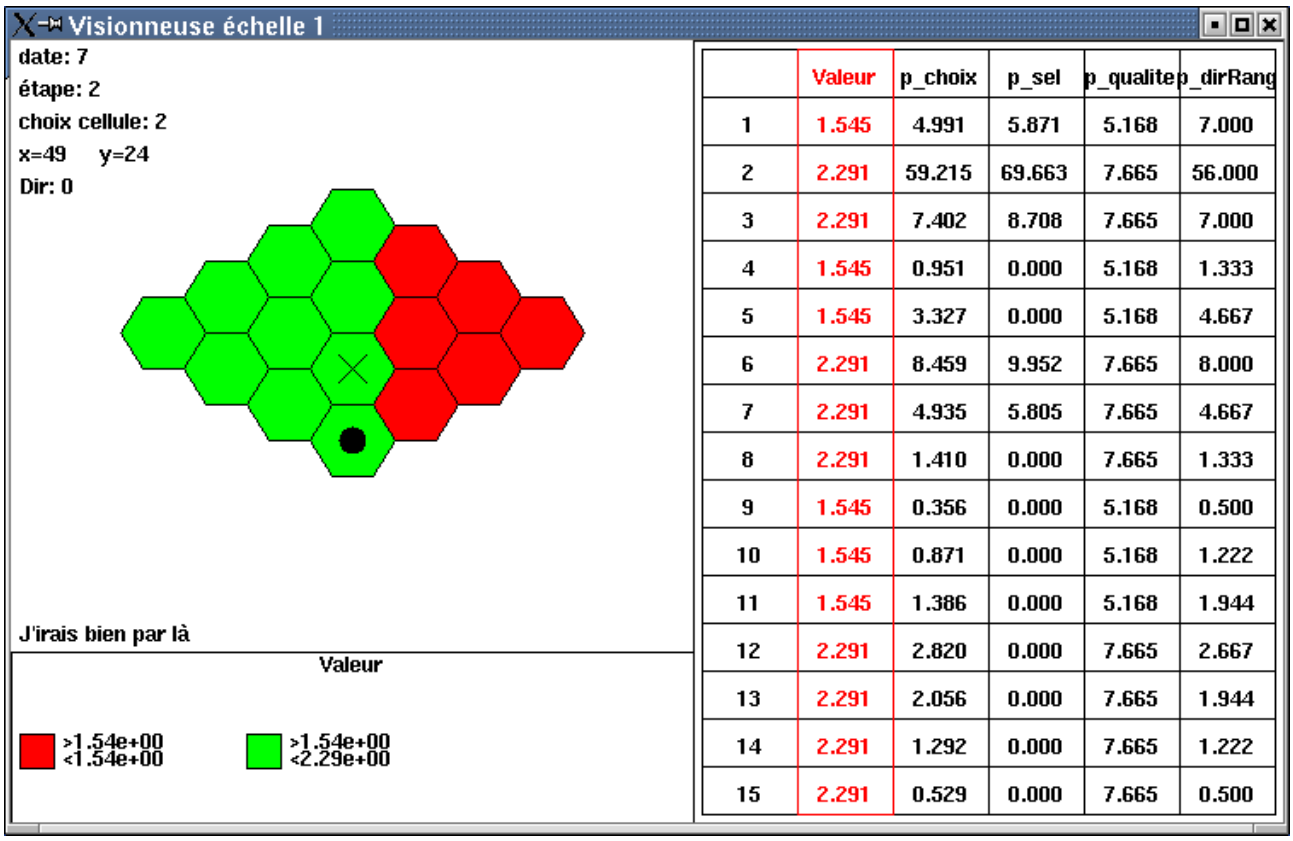
	numéro des cellules	qualité cellules	probabilité qualité	probabilité dir-rang	valeur globale	choix finaux	choix sélection
Rang 1	1	1,545	5,17%	7,00%	3,618E-03	**4,99%**	**5,87%**
	2	2,291	7,67%	56,00%	4,292E-02	**59,21%**	**69,66%**
	3	2,291	7,67%	7,00%	5,366E-03	**7,40%**	**8,71%**
Rang 2	4	1,545	5,17%	1,33%	6,892E-04	0,95%	0,00%
	5	1,545	5,17%	4,67%	2,412E-03	3,33%	0,00%
	6	2,291	7,67%	8,00%	6,132E-03	**8,46%**	**9,95%**
	7	2,291	7,67%	4,67%	3,577E-03	**4,93%**	**5,81%**
	8	2,291	7,67%	1,33%	1,022E-03	1,41%	0,00%
Rang 3	9	1,545	5,17%	0,50%	2,585E-04	0,36%	0,00%
	10	1,545	5,17%	1,22%	6,318E-04	0,87%	0,00%
	11	1,545	5,17%	1,94%	1,005E-03	1,39%	0,00%
	12	2,291	7,67%	2,67%	2,044E-03	2,82%	0,00%
	13	2,291	7,67%	1,94%	1,490E-03	2,06%	0,00%
	14	2,291	7,67%	1,22%	9,368E-04	1,29%	0,00%
	15	2,291	7,67%	0,50%	3,833E-04	0,53%	0,00%
	TOTAL	**29,889**	**100,00%**	**100,00%**	**7,249E-02**	**100,00%**	**100,00%**

Figure 24 : Tableau récapitulatif du calcul des probabilités

Pour faire la sélection des cinq meilleures cellules, une modification a été apportée. Le générateur de trajectoires ne prend pas systématiquement la même cellule en cas d'égalité entre la

cinquième cellule et la sixième meilleure cellule. Il effectue un tirage aléatoire pour effectuer la sélection.

La Figure 10 donne la correspondance des numéros avec les cellules.



	Valeur	p_choix	p_sel	p_qualite	p_dirRang
1	1.545	4.991	5.871	5.168	7.000
2	2.291	59.215	69.663	7.665	56.000
3	2.291	7.402	8.708	7.665	7.000
4	1.545	0.951	0.000	5.168	1.333
5	1.545	3.327	0.000	5.168	4.667
6	2.291	8.459	9.952	7.665	8.000
7	2.291	4.935	5.805	7.665	4.667
8	2.291	1.410	0.000	7.665	1.333
9	1.545	0.356	0.000	5.168	0.500
10	1.545	0.871	0.000	5.168	1.222
11	1.545	1.386	0.000	5.168	1.944
12	2.291	2.820	0.000	7.665	2.667
13	2.291	2.056	0.000	7.665	1.944
14	2.291	1.292	0.000	7.665	1.222
15	2.291	0.529	0.000	7.665	0.500

Figure 25 : Visionneuse Echelle 1, sortie du générateur de trajectoires

On obtient les mêmes valeurs en sortie du générateur que celles calculées ci-dessus avec les colonnes dans un ordre inverse et les dernières modifications sont effectives.

2.4 Interconnexion avec le simulateur animal

2.4.1 Principe

L'animal peut maintenant effectuer une séquence complète de défoliation sur la parcelle. Par contre, celle-ci s'effectue sans interaction avec l'animal, c'est-à-dire que le nombre de cellules défoliées est fixe et ce n'est pas l'animal qui décide de s'arrêter de manger. De plus, on ne sait pas quelle quantité de nourriture est prélevée sur la parcelle et donc, on ne peut ni faire évoluer le rumen de l'animal, ni faire de mise à jour sur la parcelle.

Par conséquent, il faut relier le modèle biologique de l'animal au générateur de trajectoires et au simulateur végétal. Ce modèle contient les fonctions permettant de connaître les valeurs de biomasse prélevées et donc, il nous sera possible de mettre à jour les différents modules.

Le générateur de trajectoires correspond en fait à un sous module de l'animal et doit être appelé quand celui-ci décide de manger. Par conséquent, il faut placer l'appel de ce module après la prise de décision. Une grande différence par rapport au fonctionnement du prototype précédent est qu'il

faut qu'il y ait une interaction permanente entre le générateur et le modèle biologique. La durée d'une phase de défoliation est variable et dépend du temps mis par l'animal sur chaque cellule pour la défolier. Elle peut aussi dépendre d'interactions avec l'extérieur, même si pour l'instant on considère que chaque phase de défoliation ne peut être interrompue.

Une analyse préliminaire m'a amené à envisager l'interconnexion comme sur le diagramme de classe de la Figure 26.

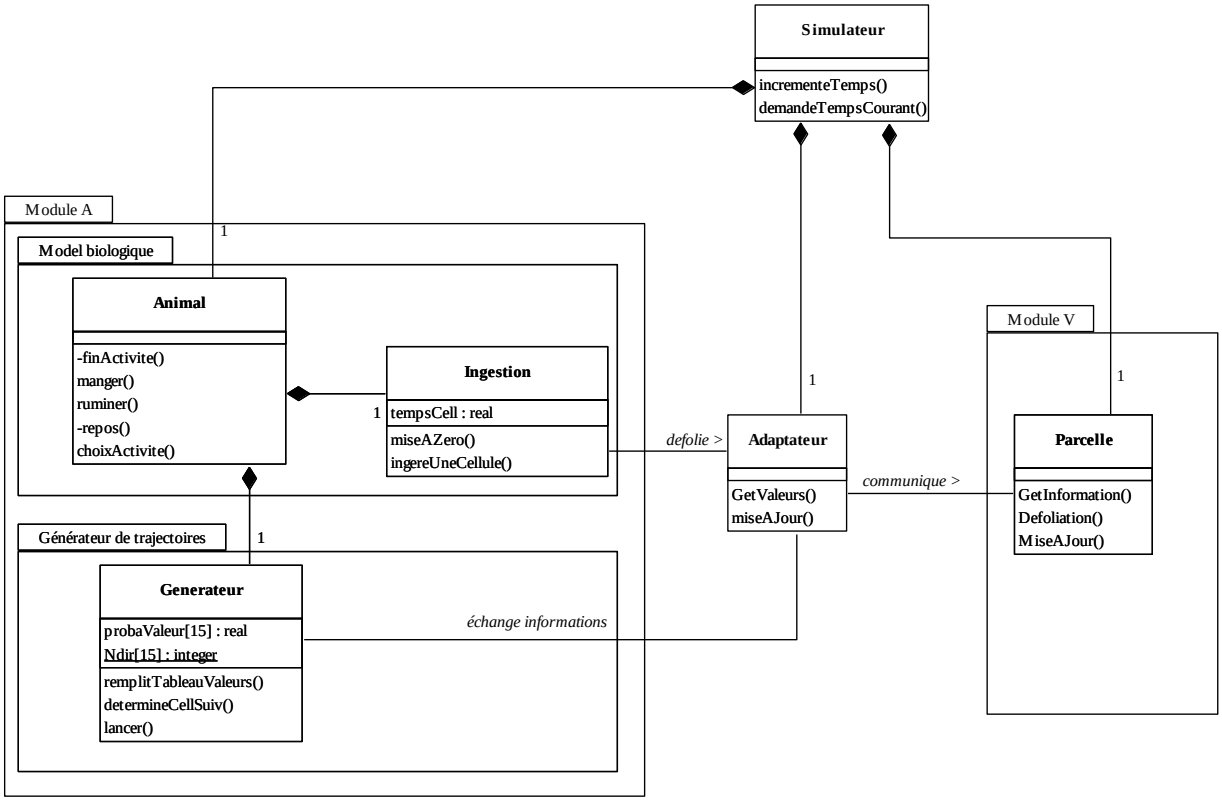


Figure 26 : Diagramme de classe de l'interconnexion avec le simulateur animal

Une première remarque concerne la communication entre les différents modules. Comme on l'a vu plus tôt, un adaptateur est nécessaire, mais on constate que l'animal ne communique plus directement avec l'adaptateur mais par l'intermédiaire d'éléments qui le compose comme le système d'ingestion ou le générateur de trajectoires. Ces derniers vont faire appel à des services différents proposé par l'adaptateur.

2.4.2 Récupération des informations pour le Rumen

Pour l'instant, les informations sont récupérées par le générateur de trajectoires. Avant de chercher à récupérer absolument les valeurs de la cellule à défolier, il faut modifier le générateur de trajectoires. Celui-ci effectue des séquences de défoliations alors qu'on souhaite que ce soit un nombre variable de choix qui soit effectué. Par conséquent, le générateur de trajectoires ne doit effectuer qu'un seul choix et le communiquer au simulateur animal qui choisit de continuer ou non la séquence. Au final on peut conclure que le générateur doit faire partie de l'animal au même titre que le rumen ou le centre décisionnel.

Notre générateur va fournir à l'animal les informations consernant la cellule choisie. Celui-ci va transmettre les données au système d'ingestion des cellules pour déterminer la quantité de

nourriture prélevée et calculer le temps passé sur la cellule. Le rumen est informé et mis à jour à la fin de la séquence.

2.4.3 Mise à jour de la parcelle

Le comportement du simulateur après toutes ces étapes est rigoureusement le même. On observe un comportement similaire de l'animal quand on visionne les fichiers de sorties, ce qui était attendu. L'étape suivante va logiquement provoquer des nuances. Il s'agit d'informer le simulateur végétal des différentes défoliations effectuées sur la parcelle afin de la mettre à jour. L'objectif est de tenir compte de tous les passages de l'animal sur chaque cellule. Si l'animal veut défolier la même cellule plusieurs fois lors d'une même simulation, le simulateur végétal va tenir compte des défoliations successives et mettre à jour sa valeur.

Le principal changement est qu'au lieu d'avoir comme avant une parcelle bien homogène et uniforme, on va avoir une parcelle avec des cellules de plus ou moins bonne qualité à cause des défoliations.

On a vu précédemment que c'est le système d'ingestion qui détermine la quantité de biomasse prélevée sur la cellule ainsi que celle prélevée dans chaque compartiment. Il est donc logique que ce soit cette partie de l'animal qui communique ces résultats à l'adaptateur et donc au simulateur végétal. L'ordre des actions effectuées est décrit dans le diagramme de séquence ci-dessous (Figure 27).

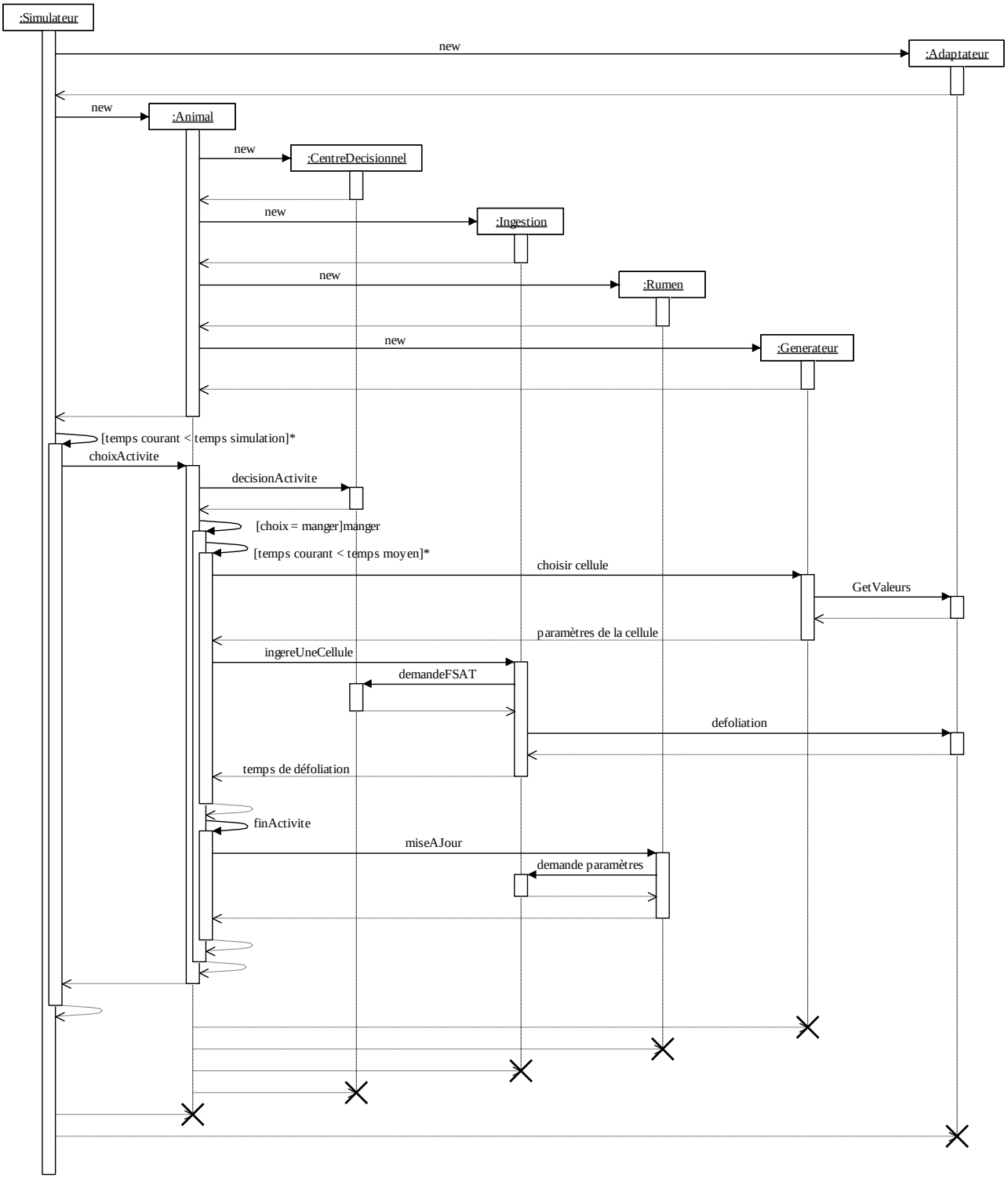


Figure 27 : Diagramme de séquence de l'interconnexion avec le simulateur animal

Par contre, on doit renvoyer le pourcentage de biomasse prélevé dans chaque compartiment et non la quantité de biomasse prélevée. De plus, ce pourcentage est une valeur comprise entre 0 et 100, donc il faut bien faire attention aux valeurs renvoyées au simulateur végétal.

2.4.4 Difficultés rencontrées

Certaines améliorations doivent encore être apportées car le modèle n'est pas encore opérationnel.

Un problème se situe au niveau des informations renvoyées par le simulateur végétal. Elles contiennent les quantités de biomasse présentes dans les quatre compartiments ainsi que la hauteur de la végétation. Par contre, on n'a pas les valeurs de NDF et DNDF qui sont indispensables pour mettre à jour le rumen. Des modifications sont à apporter pour que le simulateur végétal puisse fournir toutes les données dont on a besoin.

Les fichiers de sortie du simulateur végétal sont assez intéressants et il serait dommage de ne pas les conserver. Pour éviter de se retrouver avec un fichier différent pour chaque défoliation comme je l'ai fait au départ, on a choisi de concaténer les fichiers de sortie, au niveau de l'adaptateur, pour n'avoir plus qu'un seul fichier à la fin de la simulation. Ce fichier pourra plus tard remplacer le fichier de cellules utilisé par la visionneuse échelle 1.

Chapitre 3: Résultats et discussion

3.1 Trois versions

Actuellement, trois prototypes, à différents stades d'évolutions, sont disponibles et peuvent servir pour faire des tests. Les réunions réalisées ont permis de voir quels étaient les besoins des utilisateurs et on a conclu qu'il fallait pouvoir tester différents comportements. J'ai donc réalisé l'interconnexion en faisant plusieurs étapes et en conservant les différents prototypes. Ils sont facilement paramétrables et sont associés à une application permettant d'exécuter séquentiellement les actions associées aux simulateurs. Ces différentes versions offrent plus ou moins de souplesse à l'utilisateur.

La première version correspond au prototype décrit en Annexe II. Il consiste relier les programmes qui existaient déjà et à masquer à l'utilisateur les manipulations des fichiers d'entrée/sortie. Ce programme a été mis à jour plusieurs fois au court de ce stage pour qu'il puisse prendre en compte toutes les modifications apportées au simulateur. La carte de végétation est créer grâce au générateur de carte. Après perception de celle-ci par l'animal, le générateur de trajectoires va décrire le parcours de l'animal qu'il sera possible de voir avec l'échelle 1 et 2 de la visionneuse. Pour l'instant on ne fait ni appelle au module végétal ni au simulateur animal. Ce prototype est très souple car toutes les étapes sont indépendantes et peuvent être exécutées séquentiellement. Il est ainsi possible de régler les paramètres et de modifier les fichiers de sortie.

Le seconde version correspond à l'interconnexion avec le module végétal. Cette fois ci, la carte de végétation n'est plus générer par le générateur de carte, on a donc moins de contrôle sur la parcelle. Par contre, le simulateur végétal a été testé et en partie validé. La simulation va donc s'effectuer sur une vraie parcelle. Ce prototype a été adapté à partir du premier prototype pour pouvoir communiquer avec le simulateur végétal via un adaptateur. La visualisation se fait encore grâce à la visionneuse. Par contre, pour l'échelle 2, on ne dispose que de la carte de végétation avant perception, donc j'ai modifié le « le percepteur » pour qu'il accepte les cartes sorties du simulateur végétal.

La dernière version est la plus avancée des trois, mais dans celle-ci, moins de choses sont paramétrables et il est assez difficile de tester le comportement de l'animal. Ce prototype est basé sur le second auquel on a rajouté le simulateur animal.

Les 3 version sont utiles vues qu'elles permettent de calibrer les différents paramètres. On peut entre autres, sur la première version définir exactement quel type de végétation on souhaite avec plus ou moins de perturbation et plus ou moins d'homogénéité. Sur la deuxième version, la végétation n'a plus besoin d'être réglée et donc, on peut tester plus facilement les fonctions de choix de l'animal ou encore les paramètres de perception de celui-ci.

On peut aussi intervenir directement sur les fichiers de sortie en les modifiants si nécessaire. Sur le dernier prototype, il faut que tous les paramètres soient bien réglés si on veut pouvoir tester le comportement global de l'animal.

Tous ces tests sont à effectuer par les biologistes par comparaison avec des mesures relevées sur le terrain qui correspondent au comportement réel de l'animal, ceci afin d'avoir un simulateur qui reflète le plus possible la réalité.

3.2 Connexion XDM

Pour pouvoir effectuer les différents tests, il fallait qu'on puisse accéder facilement à mon ordinateur via le réseau informatique. Comme la plupart des utilisateurs sont sous Windows, il fallait qu'on puisse se connecter avec l'interface Exceed et donc configurer l'ordinateur sur lequel est implanté le simulateur pour qu'il réponde aux demandes de connexion de ce type. Exceed utilise le protocole XDM qui permet de demander une interface graphique pour se connecter en tant qu'utilisateur sur un ordinateur distant.

La configuration de ce protocole est assez simple mais a demandé quelques recherches avant d'obtenir le résultat désiré. La méthode permettant de configurer un ordinateur sous Linux avec la version Mandrake 8.1 est disponible en Annexe I.

3.3 Calcul des probabilités

Comme on l'a vu plus tôt, la répartition des probabilités avant modification était de 67.98% pour le rang1, 21.5% pour le rang2 et 10.52% pour le rang3 sur un test statistique de 5000 cellules.

Le même test réalisé sur 5000 cellules, après modification nous donne une répartition finale de 71.28% pour le rang1, 19.56% pour le rang2 et 9.16% pour le rang3. Ces répartitions ne sont pas encore celles attendues mais, se rapprochent de ce que l'on souhaite. Ces différences peuvent s'expliquer car l'animal ne fait pas toujours sur la parcelle un choix sur quinze cellules. Souvent, il se trouve sur les bords de la parcelle et ses possibilités de choix se trouvent restreinte. Ceci affecte nécessairement les probabilités qui sont censées marcher que s'il a quinze cellules.

3.4 Résultats, fonctionnement du programme

Actuellement, le programme est opérationnel. Il compile avec toutes les options (-ansi, -Wall, -pedantic) sans générer de « warning » et il s'exécute sans provoquer d'arrêt inopiné de l'application. Seul la génération automatique de fichier de configuration n'est pas encore installée mais ce sera fait dans la suite de mon stage.

Le fonctionnement du programme est le suivant. La classe « animal » contient tout l'animal et est autonome. La méthode « choisirActivité() » de cette classe lui fait faire un choix et en retour, on obtient le temps mis par l'animal pour faire ce choix. La classe « simulateur » se charge de la création des différentes instances comme la parcelle de terrain (simulateur végétal), l'adaptateur et l'animal. Sa méthode « lancer() » lance la simulation sur une période de temps prédéfini.

En entrée, pour l'instant, il n'y a que le fichier de configuration du générateur de trajectoires. Tous les paramètres nécessaires à l'animal ne sont pas modifiables via des fichiers de configuration et nécessitent la recompilation des fichiers si on désire les modifier. Un fichier de configuration sera donc mis en place dans la suite de mon stage.

En sortie, on obtient les fichiers habituels nécessaires à la visionneuse et au générateur de statistiques. En plus, on a des fichiers de sortie relatif à l'animal concernant son état lors des différents choix. Le simulateur végétal fournit lui aussi des fichiers de sortie relatifs aux différentes actions effectuées. Dans l'avenir, il est possible que ces sorties changent pour obtenir des fichiers plus proches de ceux utilisés par la visionneuse.

Les résultats devront être vérifiés pour voir s'ils sont conformes à ce que l'on attendait, en les comparant aux déplacements d'un animal réel sur une parcelle.

3.5 Avenir du projet

Le projet arrive presque à son terme. L'animal peut se déplacer tout seul sur une carte, effectuer ses propres actions et ses propres choix. Il peut manger, se reposer ou encore ruminer. Le simulateur végétal, également opérationnel, fournit les informations sur la parcelle utiles à l'animal et prend en compte les défoliations. La mise à jour de la parcelle permet de faire évoluer l'état de chaque cellule. Cependant, certaines équations sont encore en phase de validation et pourront être changées par la suite.

Le comportement social de l'animal reste encore à développer puisque seul la mémoire de l'animal est implémentée. Il faut encore voir comment se passent les interactions entre différents animaux présents sur une même parcelle. Pour l'instant l'automate décisionnel permet déjà de tenir compte du comportement social de l'animal et sera utilisé par la suite.

Par conséquent, il reste à placer plusieurs animaux sur une même parcelle de terrain et à les faire évoluer en même temps sans faire encore intervenir le modèle social de l'animal. Ceci va nécessiter un ordonnanceur pour gérer les actions de tous les animaux dans le temps et pour faire en sorte que les actions soient effectuées dans l'ordre chronologique, prenant en compte le temps passé pour chaque action. J'espère avoir le temps sur la fin de mon stage pour faire cette partie. Une amélioration serait de modifier la visionneuse pour pouvoir voir ensuite le déplacement des animaux en mode « pas à pas » afin de vérifier chacun de leurs mouvements.

Ensuite, une fois que le comportement social sera implémenté, il restera à interconnecter tous les modules entre eux pour obtenir le simulateur dans son intégralité.

Conclusion

Trois prototypes de l'interconnexion, représentant les différentes étapes de développement, sont disponibles. Ils sont tous opérationnels et conçus pour être simples à utiliser, même si certaines difficultés restent encore à être résolues. Les différents tests devraient donc ne pas être trop difficiles à réaliser.

Beaucoup de modules étaient déjà réalisés au début de mon stage et je n'ai donc eu à faire que des modifications sur ce qui existait déjà pour assurer l'interconnexion. Par conséquent, cela m'a permis de me rendre compte des difficultés que représente le développement d'applications sur plusieurs années et par plusieurs personnes. Chaque personne a une vision particulière des choses et une manière propre de coder (guide de style, méthodes employées …). Par conséquent, la difficulté est d'homogénéiser les différents travaux tout en gardant quelque chose de correct du point de vue programmation.

Ce stage très enrichissant m'a beaucoup apporté tant sur le plan relationnel que sur celui des techniques de programmation utilisées. Le développement d'un tel projet nécessite d'être en contact avec les principaux responsables pour connaître ce qu'ils désirent obtenir et faire en sorte d'avoir des résultats proches de ce qui est attendu. Du point de vue programmation, j'ai pu me rendre compte de ce qu'est le développement en multi-modèle et les difficultés qui y sont liées. En particulier, il a fallu, pour faciliter la compilation, créer des « make file » permettant de compiler chaque sous module et dossier, puis de créer le programme exécutable lors de l'éditions des liens. Ceci est assez facile à faire sur des cas d'école mais devient beaucoup plus difficile à gérer pour un plus grand projet.

Le projet touche presque à sa fin et devra faire encore l'objet d'un stage complet pour finir le module social de l'animal. Ensuite, il ne restera plus qu'une étape de mise au point pour terminer les tests de validations et corriger les dernières erreurs.

Pour ce simulateur une période d'analyse a été nécessaire pour comprendre ce qui avait déjà été implémenté et pour réaliser l'interconnexion plus facilement. Maintenant, l'analyse est presque terminée, il ne reste plus qu'à finir la mise en œuvre.

Références bibliographiques

[1] F. Guerry, Conception d'un simulateur multi-agents de l'interaction troupeau/parcelle pâturée, rapport de stage INRA 2000

[2] L. Masson, A. Villéger, Conception d'un module de visualisation pour un simulateur parcelle/troupeau, rapport de projet ISIMA-INRA 2000/2001

[3] L. Masson, Modélisation de la mémoire spatiale des animaux dans un simulateur parcelle/troupeau, rapport de stage INRA 2001

[4] N. Creplet, F. David, Modèle Prairie – Troupeau, Module Végétation, rapport de projet ISIMA-INRA 2000/2001

[5] P. Chatelier, G. Martin, Conception d'un générateur de cartes pour un simulateur parcelle/troupeau, rapport de projet ISIMA-INRA 2001/2002

[6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software

[7] P. Prados, La Qualité en C++ (1996) ISBN : 2-212-08917-1

[8] B. Stroustrup, Le langage C++, (1992)

[9] Paul A. Fishwick, Simulation Model Design And Execution, Chapter 8, Multimodeling, pages 285-289 (1995)

[10] D Sauvant, R Baumont, P Faverdin, Development of a Mechanistic Model of Intake and Chewing Activities of Sheep, Journal of Animal Science, 74, 1996, pages 2785-2802

INSTITUT **S**UPERIEUR
D'**I**NFORMATIQUE
DE **M**ODELISATION
ET DE LEURS **A**PPLICATIONS

Complexe des Cézeaux
BP 125-63170 Aubière CEDEX

INSTITUT **N**ATIONAL
DE LA **R**ECHERCHE
AGRONOMIQUE

Unité de Recherches sur les Herbivores
Theix
63122 St-Genès-Champanelle

Rapport de stage 2ème année

Analyse et conception du couplage entre les modules animal et végétal du simulateur troupeau/parcelle

Tome II

ANNEXES

Présenté par : Guillaume Martin
Lieu de stage : INRA Theix
Responsables de stage :
René Baumont, Laurent Pérochon
Tuteur ISIMA : Claude Mazel
Tuteur INRA : Laurent Pérochon

Du 8 avril au 29 septembre 2002

INSTITUT **S**UPERIEUR
D'**I**NFORMATIQUE
DE **M**ODELISATION
ET DE LEURS **A**PPLICATIONS

Complexe des Cézeaux
BP 125-63170 Aubière CEDEX

INSTITUT **N**ATIONAL
DE LA **R**ECHERCHE
AGRONOMIQUE

Unité de Recherches sur les Herbivores
Theix
63122 St-Genès-Champanelle

Rapport de stage 2ème année

Analyse et conception du couplage entre les modules animal et végétal du simulateur troupeau/parcelle

Tome II

ANNEXES

Présenté par : Guillaume Martin
Lieu de stage : INRA Theix
Responsables de stage :
René Baumont, Laurent Pérochon
Tuteur ISIMA : Claude Mazel
Tuteur INRA : Laurent Pérochon

Du 8 avril au 29 septembre 2002

Table des matières

Annexe I : Connexion avec XDM

1) Configuration par défaut de kdm :

Par défaut, quand tout est bien installé, il suffit de modifier le fichier *kdmrc* qui se trouve dans /usr/share/config/kdm/ et de remplacer la ligne « Enable=false » par « Enable=true » (à la fin du fichier après la ligne « [Xdmcp] »).

Si tout ce passe bien, il n'y a rien d'autre à faire.

2) Configuration de xdm :

Dans le dossier /etc/X11/xdm/, on trouve les fichiers de configuration de xdm (indispensable pour le bon fonctionnement de kdm).

Le fichier *Xaccess* contient 2 lignes par défaut :

- La première étoile signifie que tout le monde peut faire une demande directe (via xdmcp) à cet ordinateur. On peut donc spécifier quels ordinateurs peuvent avoir accès à celui-ci en supprimant cette ligne et en la remplaçant par le ou les noms des personnes autorisées.
- La deuxième étoile correspond à une demande indirecte. La ligne « * CHOOSER BROADCAST » signifie qu'en cas de demande de connexion à cet ordinateur, celui-ci va faire de même et chercher d'autres ordinateurs susceptibles de fournir une interface graphique.

Le fichier *Xserveurs* contient 1 ligne par défaut :

- « :0 local /usr/X11R6/bin/X –deferglyphs 16 » correspond au local host et indique qu'il faut fournir une interface graphique justement à cet ordinateur.
- Toutes les autres lignes permettent de rajouter des utilisateurs auxquels une interface X est proposée sans aucune demande de leur part (ceci est utile en cas de terminaux sans disque). Normalement, ceci n'est pas nécessaire dans le cas d'une connexion via exceed. Dans le doute, rajouter la ligne « * :0 foreign »

3) En cas de problèmes :

Si l'interface proposée par kdm ne fonctionne pas avec l'utilisation de exceed, est incompatible ou si on préfère ne pas l'utiliser, il est possible de démarrer le mode graphique en utilisant xdm au lieu de kdm. Pour cela, dans le dossier /etc/X11/, il faut modifier le fichier *prefdm* en supprimant le « if » final et en ne conservant que la ligne qui contient xdm (ou alors en modifiant l'ordre de lancement dans le « if »).

Note : pour l'instant sous Mandrake 8.1, xdm ne supporte pas l'option *–nodeamon* par conséquent il faut la supprimer dans le fichier /etc/inittab ou alors enlever le « $* » du fichier /etc/X11/prefdm (la ligne précédemment conservée).

Annexe II : Manuel d'utilisation du 1er prototype

1) Principe

Ce programme permet d'automatiser le traitement des différentes tâches du module animal comme la génération d'une carte, l'application d'une perception de l'animal sur cette carte ou encore la génération d'une trajectoire.

Ce programme peut servir d'interface entre l'utilisateur et la machine, en évitant de devoir effectuer toutes les opérations en ligne de commande.

Certaines fonctionnalités pourront être rajoutées en fonction des besoins de l'utilisateur et de l'avancement du simulateur.

2) Fonctionnement

Le programme se décompose en deux parties. La première concerne la génération d'une carte ainsi que la perception de celle-ci par l'animal, l'autre partie concerne la création d'une trajectoire pour l'animal et la visualisation de celle-ci via la visionneuse échelle 1 et 2. Ainsi, il est possible de créer plusieurs trajectoires à partir d'une même carte. Pour lancer le programme, il suffit de taper « simul » sur la ligne de commande dans le bon répertoire.

1ère étape : Au lancement du programme, il faut spécifier un nom qui sera commun à tous les fichiers qui seront générés par les différentes applications.
Par exemple, pour créer un fichier bandes.def qui sera nécessaire pour le créateur de carte, il faut spécifier « bandes » comme nom commun au départ.
Ce système permet par la suite de retrouver plus facilement les fichiers qui sont en rapport les uns avec les autres et d'éviter de supprimer un fichier en le remplaçant par un autre du même nom sans y prendre garde.
L'invite de commande à l'écran est comme suit :

```
Entrer un nom de fichier, il est nécessaire pour effectuer les
opérations suivantes.
Remarque :Inutile de spécifier d'extensions, il suffit de donner
          le nom générique par lequel commencera tout les noms des
          fichiers qui seront créer.
          Si par exemple, le fichier toto.def existe, il suffit de
          taper toto.
          S'il n'existe pas, il sera automatiquement créer avec
          l'aide nécessaire
>
```

2ème étape : Le menu suivant est ensuite proposé pour effectuer plusieurs opérations correspondant à la première partie du programme cité ci-dessus.

```
Menu de création
0) sortie du programme
1) configuration de la carte (avant création)
2) lancer la perception
3) voir les statistiques de la parcelle avant perception
4) continuer le traitement
>
```

- Pour configurer le fichier de définition de carte, il faut faire le choix 1. Un éditeur de texte est alors lancé. Si le fichier *nomCommun.def* (*nomCommun* représente le nom commun donné au début du programme) existe, alors ce fichier sera édité et pourra être modifié. S'il n'existe pas, alors il est automatiquement créé à partir d'un fichier d'exemple nommé exemple.def (ne pas supprimer ce fichier) basé sur une carte en forme de bandes de différentes compositions. Ce fichier permet de donner la localisation du site sur la parcelle en précisant les coordonnées du point supérieur gauche ainsi que celles du point inférieur droit (car tous les sites sont représentés sous forme de rectangles), de donner la valeur de la biomasse de ce site en tonne par hectare (cette unité peut changer par la suite s'il le faut) ainsi que les taux de répartition dans les différents compartiments de chaque cellule (VV, VS, RV, RS) et de donner les ratios de perturbation. Une aide plus précise se trouve au début de ce fichier.
  La génération de la carte se fait automatiquement avec en entrée le fichier *nomCommun.def*. Le générateur produit le fichier *nomCommun.raw* en sortie. Ce programme sert à créer la carte en donnant à chaque cellule des caractéristiques propres plus ou moins contrastées en fonction des perturbations effectuées.

- Le choix 2 sert à configurer et à lancer la perception sur cette carte ainsi produite. Pour chaque cellule, la fonction de perception choisie va générer une valeur unique par cellule en fonction des valeurs des caractéristiques de celle-ci. La valeur ainsi obtenue correspond au niveau de préférence que porte l'animal à cette cellule. Au cours de cette phase, un fichier percepteur.cfg est créé : il contient la perception choisie, les paramètres associés à cette perception, ainsi que le nombre de fractiles nécessaire lors de la création automatique de la légende. En sortie, les fichiers *nomCommunPerception.dat .cfg .lgd* et *.stat* sont générés, ils serviront à configurer la visionneuse. Deux nouveaux fichiers sont aussi générés : *nomCommunPerceptionGenTraject.cfg* qui sert à configurer le générateur de trajectoires selon la perception utilisée et *nomCommunPerceptionEchelle1.cfg* qui sert aussi à configurer la visionneuse mais cette fois-ci avec les paramètres nécessaires à l'échelle 1.
  Lors de cette étape le menu suivant apparaît :

```
Quelle perception voulez-vous utilisée
0) annuler
1) Préférence 1
2) Préférence 2
3) Optimal Foraging Theorie
>
```

  Ensuite, pour Préférence 1 et 2, les paramètres A et B seront demandés, ils sont indispensables pour ces fonctions de perception. Puis, il faut donner le nombre de fractiles nécessaire pour la légende.

- En-fin, le choix 3 permet de visualiser le fichier de statistiques de la carte avant perception afin de récupérer des informations utiles sur celle-ci. On

peut noter que celui-ci est généré uniquement quand on effectue ce choix et donc ce fichier sera mis à jour chaque fois que l'on voudra le visualiser.

3ème étape : En effectuant le choix 4, on a accès aux autres programmes comme le générateur de trajectoires et la visionneuse.
Si toutes les étapes précédentes ont été faites au court de la même exécution de l'application, alors il n'y a rien d'autres à faire, le programme garde en mémoire la perception. Sinon, il faut préciser quelle fonction de perception a été utilisée. Ce renseignement est indispensable si on veut pouvoir utiliser une carte déjà générée (par exemple un jour précédent) pour effectuer sur celle-ci plusieurs tests de génération de trajectoires. Il est aussi nécessaire pour pouvoir visualiser plusieurs cartes avec des perceptions différentes, générées à partir d'une même carte. Ainsi, toutes les trajectoires générées après seront appliquées sur la carte *nomCommunPerception.dat*.

```
Menu de visualisation
0) retour au menu de création
1) lancer la visionneuse echelle1
2) lancer la visionneuse echelle2
3) générer la trajectoire
4) éditer un fichier de configuration
5) générer le fichier de statistique
-----------------------------------------------------
Pour l'ancienne version du générateur de trajectoires
-----------------------------------------------------
6) lancer la visionneuse echelle1 (ancienne version)
7) lancer la visionneuse echelle2 (ancienne version)
8) générer la trajectoire avec l'ancienne version
9) générer le fichier de statistique de l'ancienne version
>
```

Avec le menu ci-dessus, on peut :

- Générer une trajectoire en effectuant le choix 3. Le programme de génération va ainsi être lancé en utilisant (si aucune modification n'a été faite) les paramètres par défaut utilisés lors de la création de ce fichier. Par exemple, pour l'instant, la trajectoire comporte 150 défoliations par défaut, il n'y a qu'une phase de défoliation, A=5, B=1, C=1 et les autres paramètres ont leurs valeurs habituelles ({0.7,0.2,0.1} pour la distance, {0.8,0.1} pour la direction …)
- Le choix 1 permet de lancer la visionneuse échelle 1 avec tous les fichiers requis.
- Le choix 2 permet de lancer la visionneuse échelle 2. Si le fichier contenant la trajectoire existe (c'est-à-dire que la trajectoire a été au moins lancée une fois sur ce type de fichier), alors la visionneuse est lancée avec ce fichier en paramètre (c'est-à-dire que l'on visionne aussi la trajectoire). Sinon, la visionneuse est lancée sans ce fichier et on observe la carte sans trajectoire. Remarque : la visionneuse échelle 1 et 2 ne bloque pas le fonctionnement du programme. Ainsi, on peut effectuer d'autres opérations tout en gardant la visualisation à l'écran.
- Le choix 4 permet d'éditer les différents fichiers de configuration ainsi que les fichiers de statistiques.

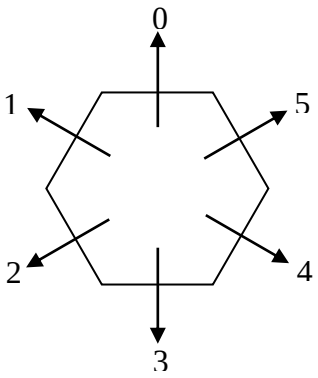
```
Quel fichier voulez-vous éditer ?
0) annuler
1) testPerceptionOptimalForagingTheoryGenTraject.cfg :
   configuration du générateur de trajectoire
2) testPerceptionOptimalForagingTheoryEchelle1.cfg :
   configuration de la visionneuse echelle1
3) testPerceptionOptimalForagingTheory.lgd : configuration
   de la légende (pour les intervalles)
4) testPerceptionOptimalForagingTheory.stat : statistiques
   de la parcelle après perception
5) testPerceptionOptimalForagingTheory.csv : statistiques de
   la trajectoire de l'animal
-------------------------------------------------------------
    fichiers de l'ancien générateur de trajectoires
-------------------------------------------------------------
6) testPerceptionOptimalForagingTheoryOldGenTraject.cfg :
   configuration du générateur de trajectoire
7) testPerceptionOptimalForagingTheoryOldEchelle1.cfg :
   configuration de la visionneuse echelle1
8) testPerceptionOptimalForagingTheoryOld.csv : statistiques
   de la trajectoire de l'animal
>
```

Ainsi, le choix 1 édite le fichier de configuration du générateur de trajectoires (on peut y régler le nombre de défoliation, les coordonnées de départ de l'animal, les différentes probabilités …). Le choix 2 permet d'éditer le fichier de configuration de la visionneuse échelle 1. Le choix 3 permet d'éditer le fichier de légende, pour pouvoir modifier le nombre d'items par exemple. Le choix 4 permet de visualiser le fichier de statistique de la carte après perception. Le choix 5 édite le fichier de statistique sur la trajectoire suivie par l'animal. Ce fichier est dans un format reconnu par Excel et il suffit donc de l'ouvrir avec Excel pour pouvoir le visualiser sous forme de tableau.

- Enfin, le choix 5 permet de lancer le générateur de statistique sur la trajectoire. Ce programme permet de visualiser quelles cellules ont été défoliées en notant leurs coordonnées ainsi que les différentes valeurs visualisables avec la visionneuse échelle 1. Remarque : les coordonnées sont en fait celles de la position actuelle de l'animal et donc pas celles de la cellule défoliée. Pour avoir les coordonnées de cette cellule, il suffit de prendre les coordonnées de l'animal à l'étape suivante (vu qu'il se déplace sur la cellule qu'il a choisie de défolier). Par conséquent, lors du résumé en fin de fichier, le site où se trouve l'animal en début de simulation est comptabilisé même s'il n'a effectué aucune défoliation dans celui-ci.

3) Remarques

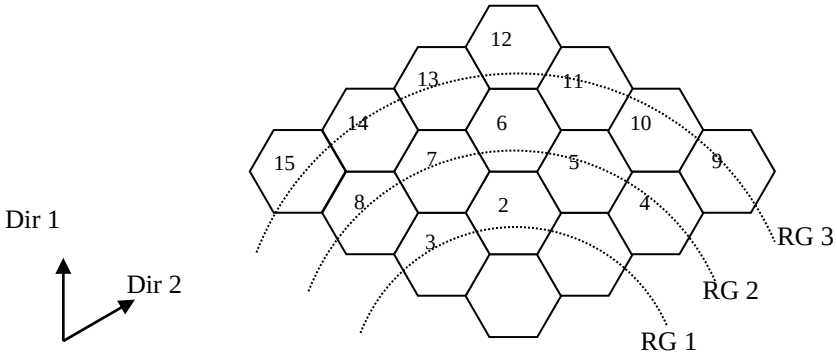
- Dans le fichier de configuration du générateur de carte, un paramètre permet de définir la direction dans laquelle regarde l'animal au départ de la simulation. Le code permettant de faire ce choix est un nombre compris entre 0 et 5 (Figure 1) avec, 0 pour que l'animal regarde vers le haut, 1 pour que l'animal regarde en haut à gauche, 2 pour en bas à gauche, 3 pour vers le bas, 4 pour en bas à droite et 5 pour en haut à droite.



Remarque : il y a un changement par rapport à l'ancienne version. Contrairement aux directions utilisées avant par le générateur de trajectoires, on prend la numérotation des directions utilisée par le simulateur végétal. Cela nous permet ainsi d'avoir un programme plus homogène.

Figure 1 : Numérotation des directions

- Si on a déjà effectué l'étape de création de carte et que l'on ne veut pas modifier la carte ainsi générée, on peut passer directement à l'étape 3 (entrer le *nomCommun* puis faire le choix 4).
- A tout moment, le choix 0 permet soit d'annuler, soit de sortir du programme. Lors de l'étape 2 ce choix fait quitter le programme, lors de l'étape 3, ce choix fait revenir à l'étape 2. Par contre, lors du choix de la perception ou du nombre de fractiles par exemple, on ne peut annuler l'opération. Donc, en cas d'erreur sur un des renseignements, il faut poursuivre la perception avec le paramètre erroné puis, relancer celle-ci avec cette fois-ci le bon paramètre.
- On peut constater que seul le fichier de configuration de la visionneuse échelle 1 peut être édité. Le fichier de configuration de l'échelle 2 ne sert uniquement qu'à renseigner les fichiers à utiliser, on n'a donc pas besoin de l'éditer.
- Par contre, il est utile de pouvoir éditer celui de l'échelle 1. Il contient un paramètre important qui est le nombre d'étape à visualiser. Par exemple, si la trajectoire a été générée avec 150 défoliations, et que le paramètre du fichier de configuration est fixé à 50, alors on ne peut visualiser que les 50 premières défoliations, les 100 restantes existent toujours, mais on ne peut les observer. Si au contraire, il y a 50 défoliations et que le paramètre est fixé à 150, on peut voir les 50 défoliations sans aucun problème, mais si on essaye de voir la 51ème (qui n'existe pas), cela provoque un arrêt brutal de la visionneuse.
- Pour ce qui est de la gestion des erreurs, des valeurs aberrantes et autre problème de ce genre, cette application ne vérifie pas, ni n'informe l'utilisateur. Pour l'instant cette application ne fait qu'appeler d'autres programmes qui eux sont capables de gérer tous les cas d'erreurs. Par contre aucun message n'est affiché à l'écran, il faut donc être prudent lors des différentes étapes, mais on peut noter qu'à ce jour, toutes les erreurs survenues dans un programme n'ont pas fait d'arrêt de cette application, ainsi, il est facile de corriger les erreurs.



Remarque : comme pour les directions, la numérotation des cellules a changé pour être compatible avec le simulateur végétal (voir ci contre pour la nouvelle numérotation)

Figure 2 : Les quinze cellules

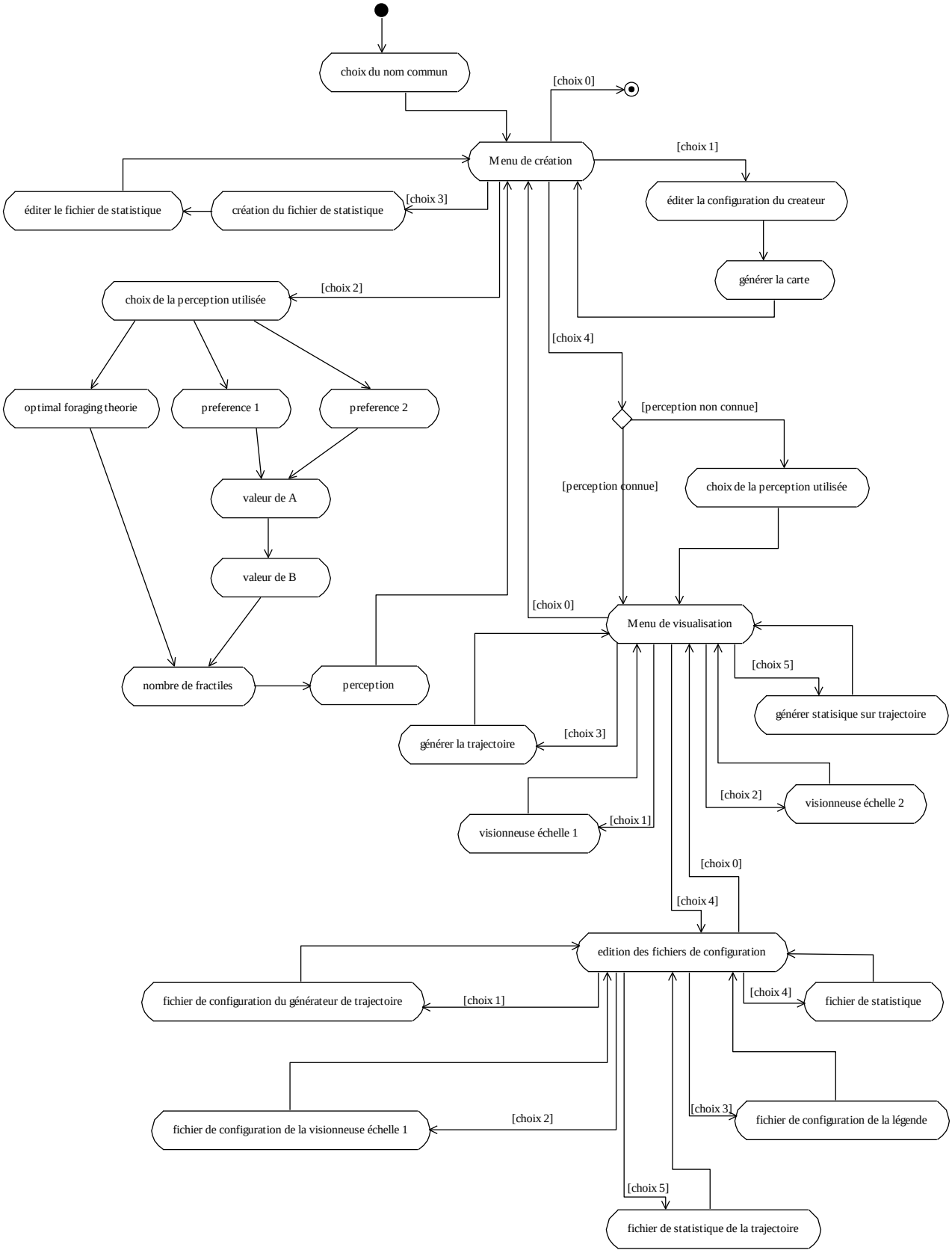


Figure 3 : Diagramme d'activité du premier prototype du module A

Annexe III : Fichiers d'entrée sortie

1) Le créateur :

Fichier d'entrée : Le fichier d'entrée permet de décrire une carte en construisant des sites en forme de rectangle. De préférence, le premier site sert à donner une valeur par défaut à toute la parcelle, sachant que les autres sites rajoutés par dessus le premier écrase celui-ci.

Les paramètres sont :

- La largeur de la parcelle
- La hauteur de celle-ci
- Une description de chaque site (une ligne par site dans ce fichier)

La description de chaque site se compose de :

- L'abscisse et l'ordonnée du premier point (x1,y1)
- L'abscisse et l'ordonnée du second point pour décrire un rectangle (x2,y2)
- La valeur de la biomasse totale en tonnes par hectare (B)
- La perturbation de cette biomasse selon une loi gaussienne (delta(b))
- Le taux de répartition de cette biomasse parmi les compartiments reproductif (on déduit le taux de répartition parmi les compartiments végétatifs en soustrayant ce taux à 1) (r)
- La variation de ce taux (delta(r)).
- Le taux de répartition de vert au sein des compartiments reproductifs et la variation qui va avec (vR,delta(vR))
- De même avec la répartition de vert au sein des compartiments végétatifs (vV,delta(vV))
- NDF et DNDF (ils ne sont pas utilisés actuelement)

Exemple de fichier :

```
#bandes.def
#Largeur hauteur
  100      50

#Site 1 : toute la parcelle
# x1 y1 x2 y2 B delta(b) r delta(r) vR delta(vR) vV  delta(vV) NDF DNDF
  0  0  99 49 8  0.25  0.55    0    0.8   0     0.99    0      12    34

#Site 2 : une bande
# x1 y1 x2 y2 B delta(b) r delta(r) vR delta(vR) vV  delta(vV) NDF DNDF
  0  0  25 49 2  0.25  0.10    0    0.99  0     0.99    0      12    34

#Site 3 : une bande
# x1 y1 x2 y2 B delta(b) r delta(r) vR delta(vR) vV  delta(vV) NDF  DNDF
  25 0  50 49 8  0.25  0.55    0    0.8   0     0.99    0      12    34

#Site 4 : une bande
# x1 y1 x2 y2 B delta(b) r delta(r) vR delta(vR) vV  delta(vV) NDF  DNDF
  50 0  75 49 2  0.25  0.10    0    0.99  0     0.99    0      12    34

#Site 5 : une bande
```

```
# x1 y1 x2 y2 B delta(b) r   delta(r) vR delta(vR) vV  delta(vV) NDF  DNDF
  75 0  99 49 8  0.25   0.55   0      0.8   0     0.99    0      12    34
```

Fichier de sortie : En sortie, on obtient une carte contenant les valeurs des différents compartiments comme décrit dans le fichier d'entrée.

Il contient :

- La valeur de la largeur et de la hauteur
- Les valeurs des compartiments (dans l'ordre) RV RS VV VS ainsi que 4 valeurs pour NDF (pour les 4 compartiments) et 4 valeurs pour DNDF.

Exemple :

```
100 50
3.67097568863e-06 3.70805625114e-08 3.30387811976e-05 3.33725062602e-07 1.188
0.012 10.692 0.108 3.366 0.034 30.294 0.306
```

Les valeurs correspondent à la première cellule du fichier.

2) L'analyseur

Il prend le fichier de sortie du créateur, il analyse la carte et fournit un fichier contenant des renseignements sur la carte générée.

Exemple de fichier de sortie :

```
Compartiment RV:
    Valeur Minimum :
          biomasse : 1.45901756599e-06

    Valeur Maximum :
          biomasse : 0.000620051938582

    Valeur Moyenne :
          biomasse : 0.000186400338696

    Ecart type :
          biomasse : 0.000177660598422

-----------------------------------------------
Compartiment RS:
    Valeur Minimum :
          biomasse : 1.47375511717e-08

    Valeur Maximum :
          biomasse : 0.000155012984646


    Valeur Moyenne :
          biomasse : 4.42255899654e-05

    Ecart type :
          biomasse : 4.66411769347e-05

-----------------------------------------------
Compartiment VV:
```

```
    Valeur Minimum :
          biomasse : 1.31311580939e-05

    Valeur Maximum :
          biomasse : 0.000627802587815

    Valeur Moyenne :
          biomasse : 0.000267789769118

    Ecart type :
          biomasse : 0.00011353481992

-----------------------------------------------
Compartiment VS:
    Valeur Minimum :
          biomasse : 1.32637960545e-07

    Valeur Maximum :
          biomasse : 6.34144028096e-06

    Valeur Moyenne :
          biomasse : 2.70494716281e-06

    Ecart type :
          biomasse : 1.14681636283e-06

-----------------------------------------------
```

3) Le percepteur

Il récupère la carte créée par le créateur de carte et applique dessus la fonction de perception choisie.

En entrée :

- Le fichier .raw
- Un fichier de configuration du percepteur (on ne le passe pas en paramètre mais il est nécessaire au programme)

Exemple de fichier de configuration du percepteur :

```
#Fonction de perception à choisir
#pour la lisibilité, on demande le nom
#comme les noms sont un peu compliqués, on les met tous
#et on commente ceux qui ne servent pas

#FONCTION_PERCEPTION PerceptionPreference1
FONCTION_PERCEPTION  PerceptionPreference2
#FONCTION_PERCEPTION PerceptionOptimalForagingTheory

#parametre 'a' utilisé par les fonctions de perception avec preference
#on peut ne pas le spécifier, il aura une valeur par défaut
#dependant justement de cette fonction
A   0.5

#parametre 'b' utilisé par les fonctions de perception avec preference
#on peut ne pas le spécifier, il aura une valeur par défaut
#dépendant justement de cette fonction
B   0.5
```

```
#le nombre de fractiles pour la légende . Exemple : 4 équivaut à la
répartition 25%, 50% 75% 100%
NB_FRACTILES 4
```

Fichiers de sortie :

- Le fichier .dat qui contient les valeurs de chaque cellule après perception (il ne reste qu'une valeur par cellules et donc on peut représenter cette carte grâce à la visionneuse)
- Le fichier .lgd qui est nécessaire pour la légende de la visionneuse
- Le fichier .cfg qui permet de configurer la visionneuse
- Le fichier .stat qui contient des renseignements sur la carte après perception.

Exemple de fichier de légende :

```
Valeur       [4.4825235e-07;5.5553849e-06]       [5.5553849e-06;6.7881806e-06]
[6.7881806e-06;8.0670544e-06] [8.0670544e-06;1.3414548e-05]
```

Exemple de fichier de configuration de la visionneuse échelle 2:

```
LARGEUR  100
HAUTEUR  50
ECHELLE  2
FICHIER_LEGENDE bandesPerceptionOptimalForagingTheory.lgd
NB_PARAM 1
#Pour compatibilité avec l'ancienne visionneuse (mais ce paramètre est
maintenant obsolète):
FICH_INIT_ECH  bandesPerceptionOptimalForagingTheory.cfg
```

Exemple de fichier de statistique :

```
Valeurs percues:
Min: 4.48252353989e-07
Max: 1.34145488597e-05
Moyenne: 6.8594724183e-06
Mediane: 6.78675408627e-06
Ecart-type: 1.87292751134e-06
Fractiles à 25% :
    [4.48252353989e-07;5.55538494942e-06]
    [5.55538494942e-06;6.78818069062e-06]
    [6.78818069062e-06;8.06705449948e-06]
    [8.06705449948e-06;1.34145488597e-05]
```

4) Le générateur de trajectoires

Il permet de créer une trajectoire représentant les déplacements de l'animal sur la parcelle.
Il prend en paramètre un fichier de configuration qui contient l'emplacement du fichier carte permettant d'obtenir les valeurs des différentes cellules de la parcelle.

Exemple de fichier de configuration :

```
# nombre de phases de défoliation
NB_PHASES 1
# nombre de défoliations par phase
NB_DEFOLIATIONS 50
```

```
# abscisse de départ
X_DEPART 48
# ordonnée de départ
Y_DEPART 25
# direction de départ
DIR_DEPART 0
# valeur de la proba de direction si l'animal va tout droit
DIRMILIEU 0.8
# valeur de la proba de direction si l'animal va tout à droite
DIRDROITE 0.1
# valeur de la proba de direction si l'animal va tout à gauche
DIRGAUCHE 0.1
# pareil pour la distance, ici sur la première couronne
RG1  0.7
# deuxième couronne
RG2  0.2
# troisième couronne
RG3  0.1
# coefficients : proba = qualite^A + rang/dir^B
A    1
B    1
# on choisit la cellule suivante parmi les SELECTIONS cellules
# ayant la plus forte probabilité. Si SELECTIONS=1, la cellule
# choisie est toujours celle ayant la probabilité max
SELECTIONS 5
# indique si on pondère ou non les probas
PONDERATION 0
# fichier contenant la parcelle où se déplace l'animal
FICHIER_GRILLE bandesPerceptionOptimalForagingTheory.dat
# fichier de sortie contenant la trajectoire
FICHIER_SORTIE bandesPerceptionOptimalForagingTheory.trj
# fichier de sortie contenant les cellules du champ de vision de
# l'animal (sert pour la visionneuse échelle 1)
FICHIER_PARAMS cellules.dat
```

Fichier de sortie :

- Le fichier de trajectoire
- Un fichier permettant de visualiser la carte avec la visionneuse échelle 1

5) La visionneuse

Fichiers en entrée :

- Le fichier de configuration de la visionneuse
- Le fichier de légende
- Le fichier carte
- Le fichier de trajectoire s'il existe

Le fichier de configuration de la visionneuse échelle 1 est un peu différent de celui de l'échelle 2.
Exemple de fichier de configuration de la visionneuse échelle1 :

```
LARGEUR  100
HAUTEUR  50
ECHELLE  1
FICHIER_LEGENDE bandesPerceptionOptimalForagingTheory.lgd
NB_PARAM 5
```

```
NBE_COURONNES 3
NBE_ETAPES 50
FICHIER_CELLULES cellules.dat
FICHIER_ANIMAL bandesPerceptionOptimalForagingTheory.trj
NBE_FOURCHETTES 6
FICHIER_CHOIX choix.txt
#Pour compatibilité avec l'ancienne visionneuse (mais ce paramètre est
maintenant obsolète):
FICH_INIT_ECH  bandesPerceptionOptimalForagingTheory.cfg
```

Le fichier choix sert à renseigner les motivations du choix de l'animal. Pour l'instant aucun choix précis n'est déterminé et cette fonctionnalité sera à rajouter par la suite.

Annexe IV : Le générateur de statistiques

Cette application sert à obtenir des informations d'ordre statistique sur le parcours de l'animal. Au cours du temps et des besoins, elle a beaucoup évolué et risque encore de subir des modifications dans l'avenir. Une fonctionnalité du générateur de trajectoires est la possibilité d'effectuer plusieurs phases de défoliation. Il est intéressant de pouvoir avoir des statistiques partielles sur chaque phase et d'autres globales pour toutes les phases. Par conséquent, si plusieurs phases de défoliation sont effectuées, on retrouve dans le fichier de sortie les statistiques attendues à la fin de chaque phase.

Fichiers d'entrée : trois fichiers doivent être fournis pour obtenir les statistiques attendues

Exemple de fichier « .def » :

```
#Largeur hauteur
  64      64

#Site 1 : toute la parcelle
# x1 y1  x2  y2   B   delta(b)  r   delta r  vR   delta(vR)  vV  delta(vV)   NDF  DNDF
  0  0   63  63  2.65  0.05   0.001  0.05   0.99     0      0.77     0       12    34

#Site 2 : une bande
# x1 y1  x2  y2   B   delta(b)  r   delta r  vR   delta(vR)  vV  delta(vV)   NDF  DNDF
  0  0   31  63  2.86  0.05   0.001  0.05   0.99     0      0.79     0       12    34
```

Exemple de fichier « .cel » :

```
date       param1       param2       param3       param4       param5       param6
0          0.741292     0            0            0            0            0
           0.446882     2.25894      2.25894      5.10748      7            7
           0.521361     39.0592      39.0592      5.95871      56           56
           0.511217     4.42556      4.42556      5.84277      7            7
           0.460088     0.497721     0.497721     5.25841      1.33333      1.33333
           0.4476       1.51809      1.51809      5.11568      4.66667      4.66667
           0.512635     5.12833      5.12833      5.85898      8            8
           0.523258     3.31458      3.31458      5.98039      4.66667      4.66667
           0.805332     8.17819      8.17819      9.20426      1.33333      1.33333
           0.713188     1.67044      1.67044      8.15112      0.5          0.5
           0.735183     4.75301      4.75301      8.4025       1.22222      1.22222
           0.800825     11.5965      11.5965      9.15274      1.94444      1.94444
           0.796342     15.4635      15.4635      9.1015       2.66667      2.66667
           0.521618     1.35956      1.35956      5.96164      1.94444      1.94444
           0.478801     0.556886     0.556886     5.47228      1.22222      1.22222
           0.475236     0.219462     0.219462     5.43154      0.5          0.5
……
```

Exemple de fichier « .trj » :

```
date       abs    ord    dir    choix  raison leader
0          31     63     0      10     1      0
9          33     61     5      2      1      0
9          34     61     5      3      1      0
15         34     60     0      2      1      0
……
```

Le fichier de sortie « .csv » obtenue est le suivant :

```
site;abscisse;ordonnée;choix cellule;qualité cellule;probabilité global;probabilité
sélection;probabilité qualité;probabilité direction;probabilité rang
2;31;63;10;0.735183;4.75301;4.75301;8.4025;1.22222;1.22222
……
1;51;12;6;0.808086;18.3468;18.3468;8.86562;8;8
nombre de choix = 149
rang 1 = 83 (55.7047%)
rang 2 = 41 (27.5168%)
rang 3 = 25 (16.7785%)
direction milieu = 94.8333 (63.6465%)
direction droite = 31.5 (21.1409%)
direction gauche = 22.6667 (15.2125%)
site 1 visité 118 fois (79.1946%)
site 2 visité 31 fois (20.8054%)
**********************************************
**********************************************
nombre total de choix = 5375
rang 1 = 3315 (61.6744%)
rang 2 = 1357 (25.2465%)
rang 3 = 703 (13.0791%)
direction milieu = 3621.33 (67.3736%)
direction droite = 910.833 (16.9457%)
direction gauche = 842.833 (15.6806%)
site 1 visité 2760 fois (51.3488%)
site 2 visité 2615 fois (48.6512%)
```

Ce fichier de sortie est décomposé en une succession de lignes représentant les paramètres de chaque cellule défoliée. Pour chaque phase de défoliation, on récupère des informations sur les répartitions en fonction du rang, de la direction et du site sur lequel se trouve la cellule. A la fin du fichier, on a des statistiques globales qui regroupent toutes les phases.

Annexe V : Diagrammes UML

1) Le simulateur animal

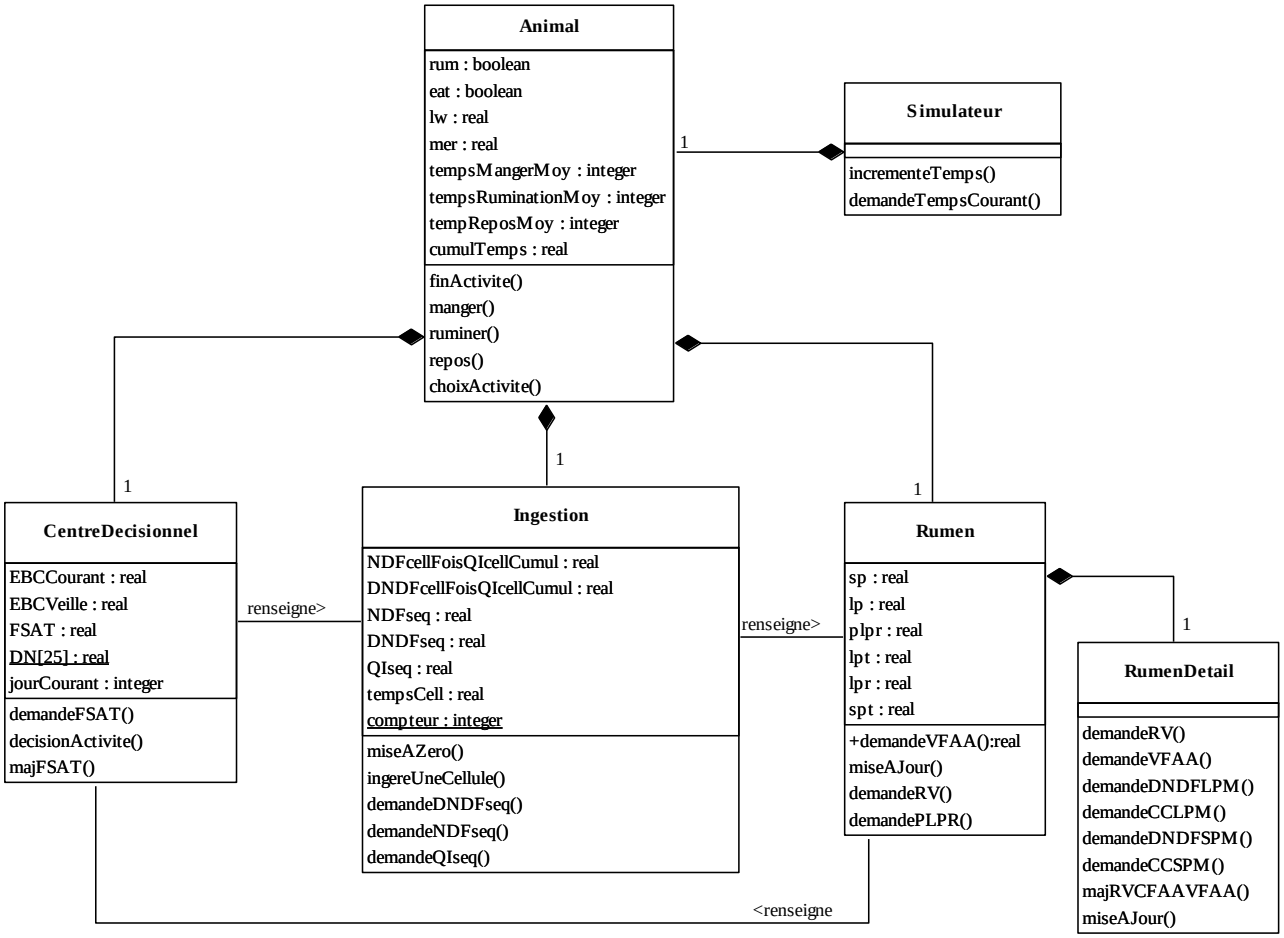


Figure 4 : Diagramme de classe du simulateur animal

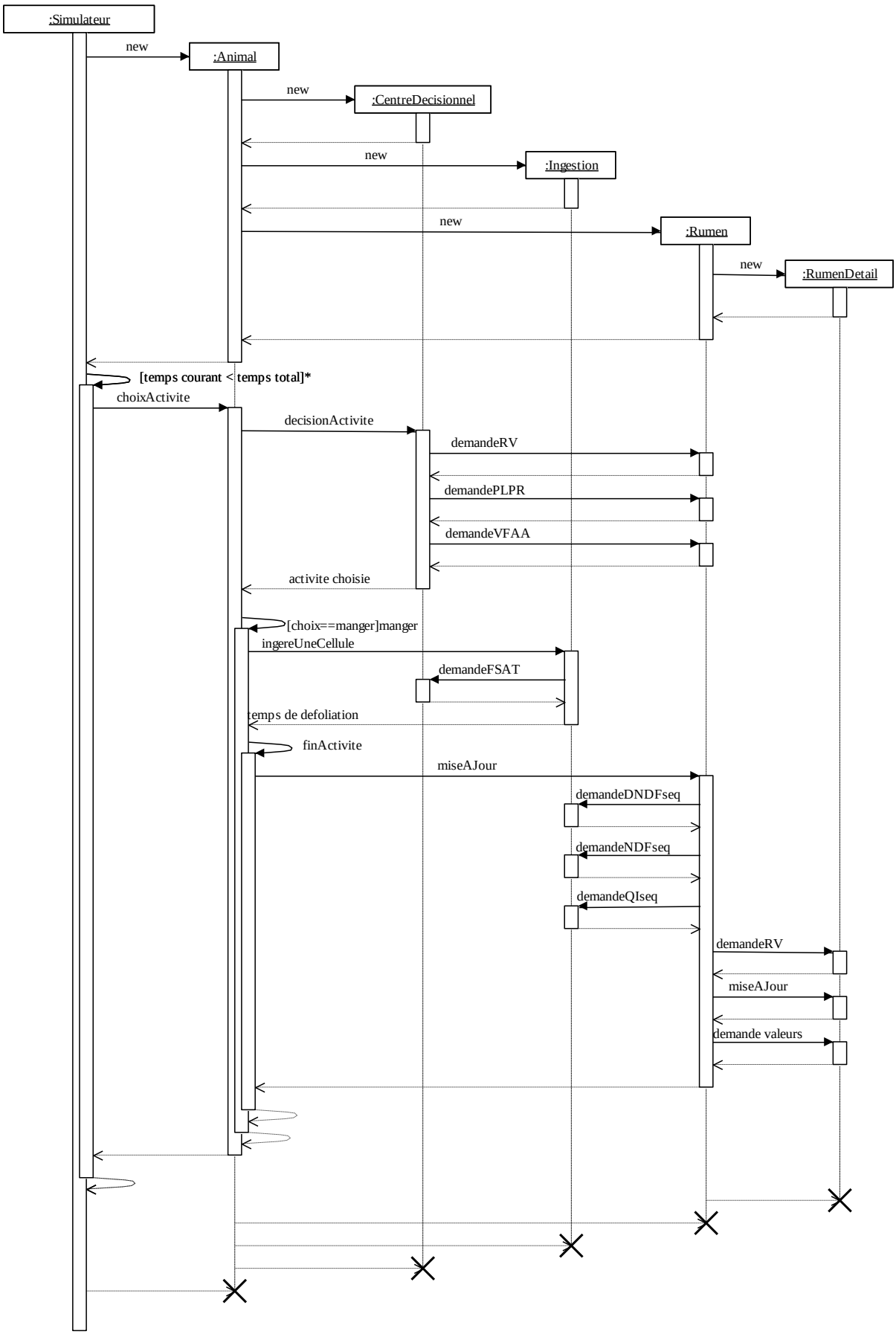


Figure 5 : Diagramme de séquence du simulateur animal

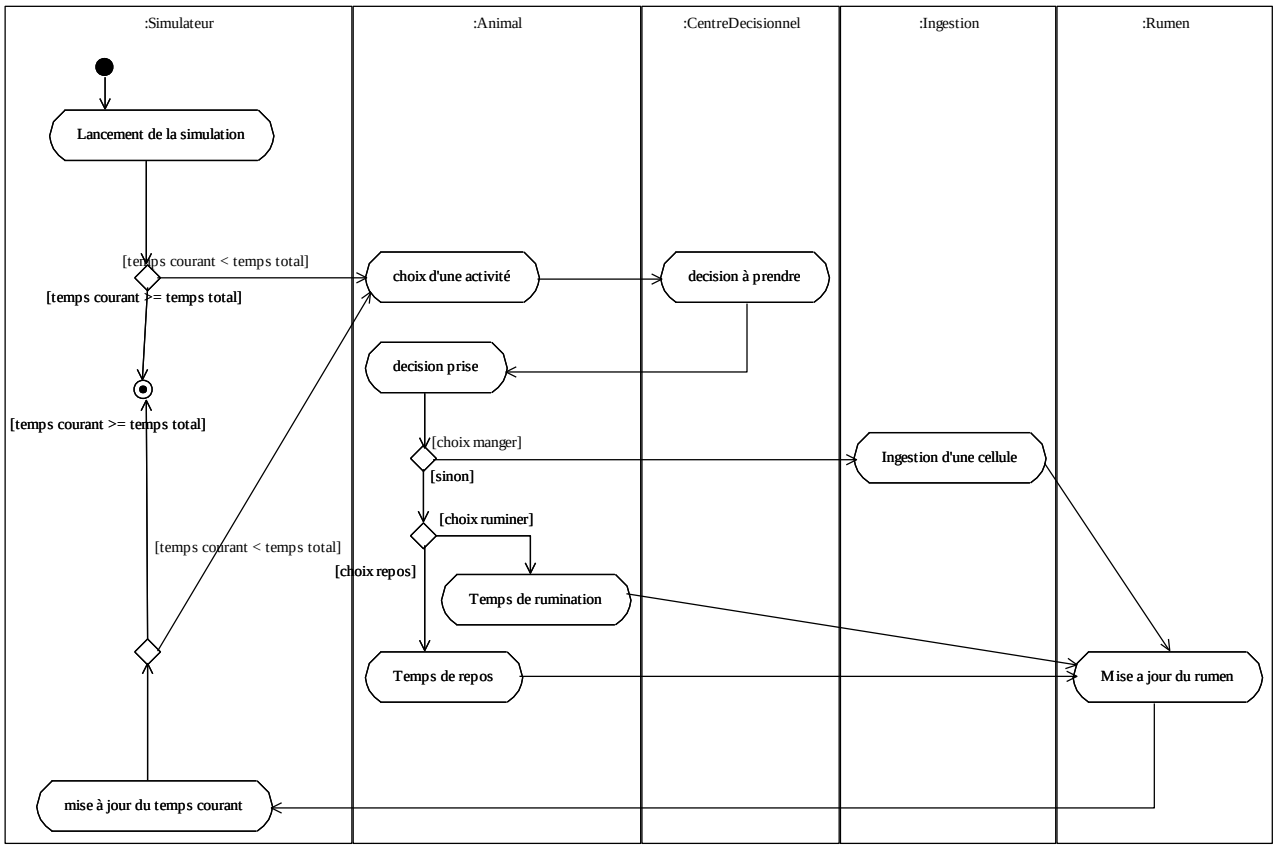


Figure 6 : Diagramme d'activité du simulateur animal

2) Le générateur de trajectoires

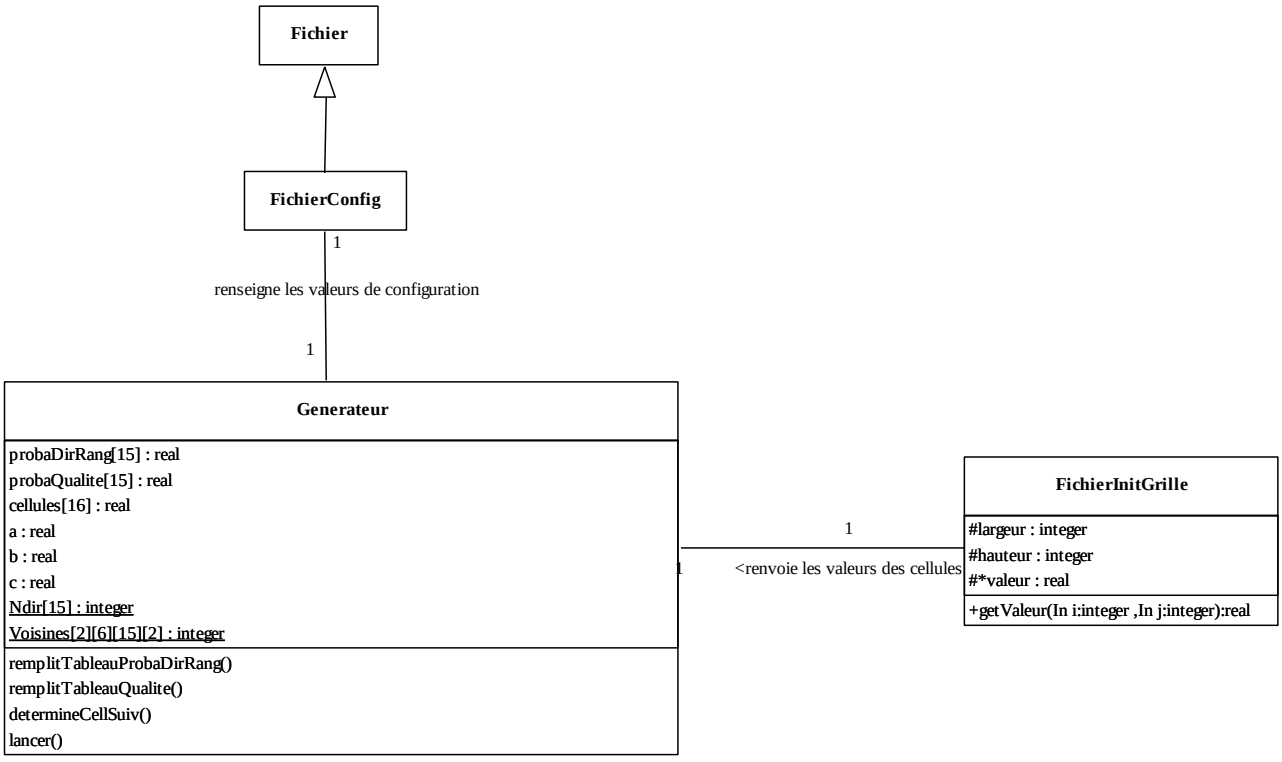


Figure 7 : Diagramme de classe du générateur de trajectoires

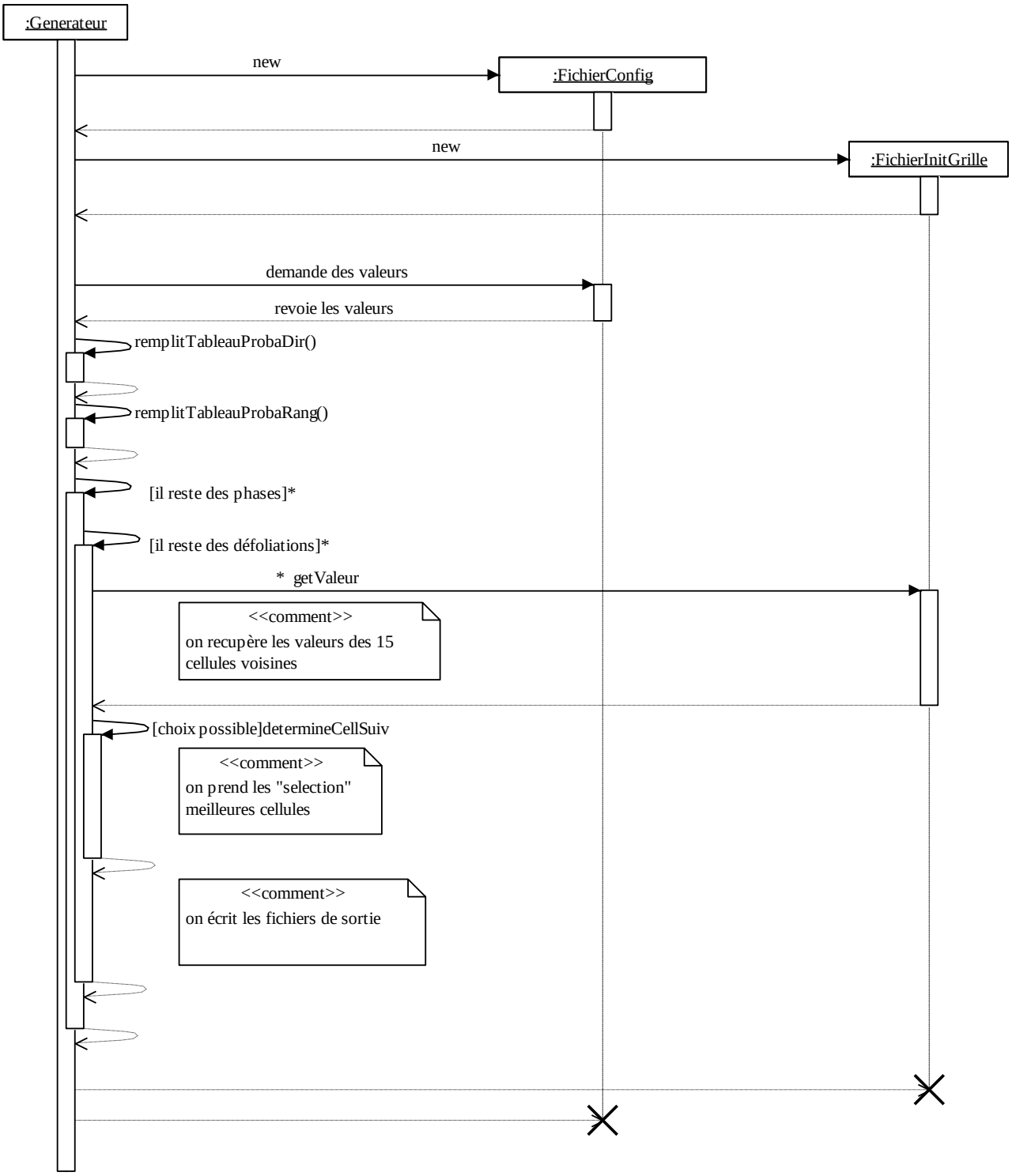


Figure 8 : Diagramme de séquence du générateur de trajectoires

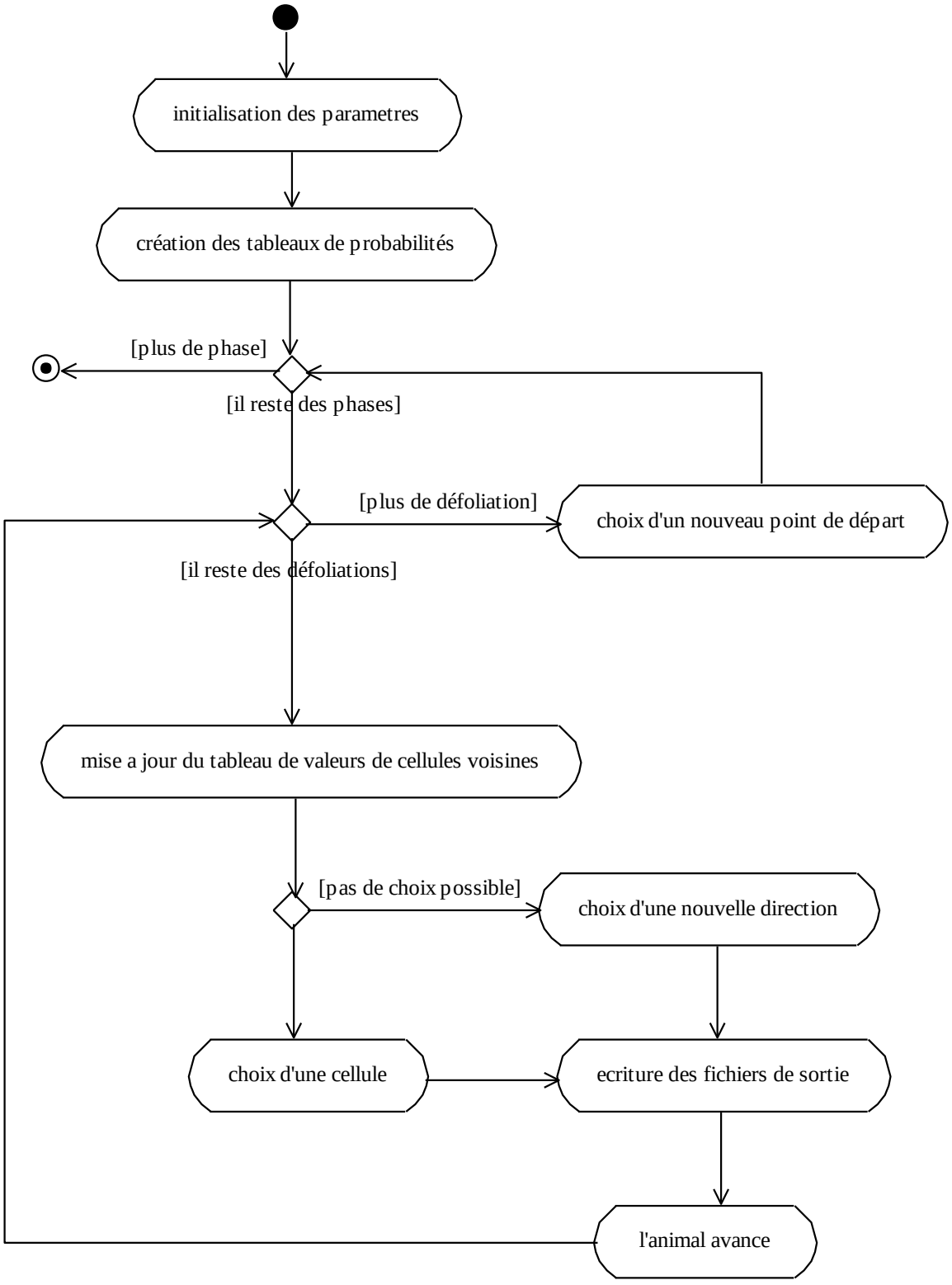


Figure 9 : Diagramme d'activité du générateur de trajectoires

3) L’automate décisionnel

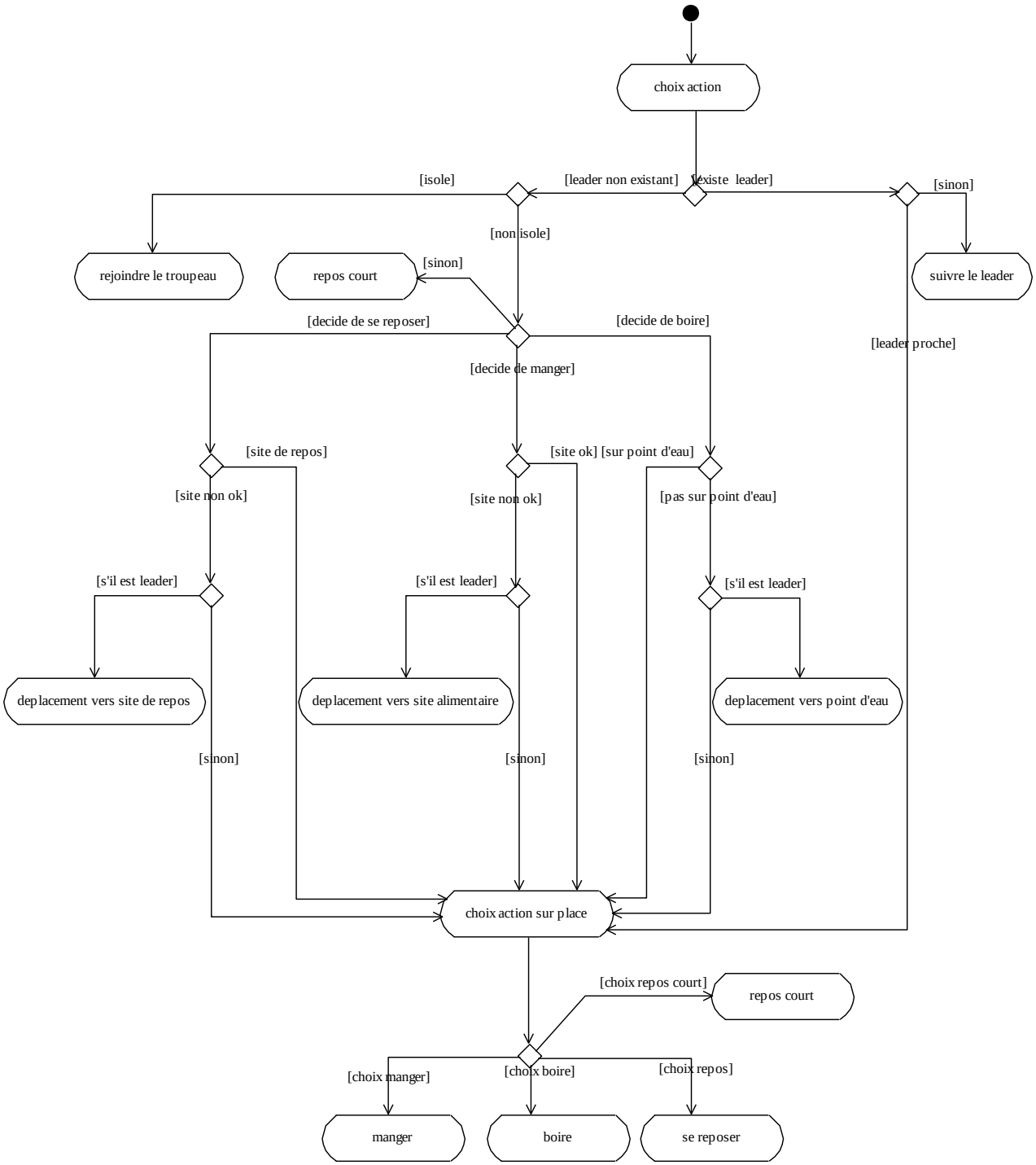


Figure 10 : Diagramme d’activité de l’automate décisionnel

4) Interconnexion entre le générateur et le simulateur végétal

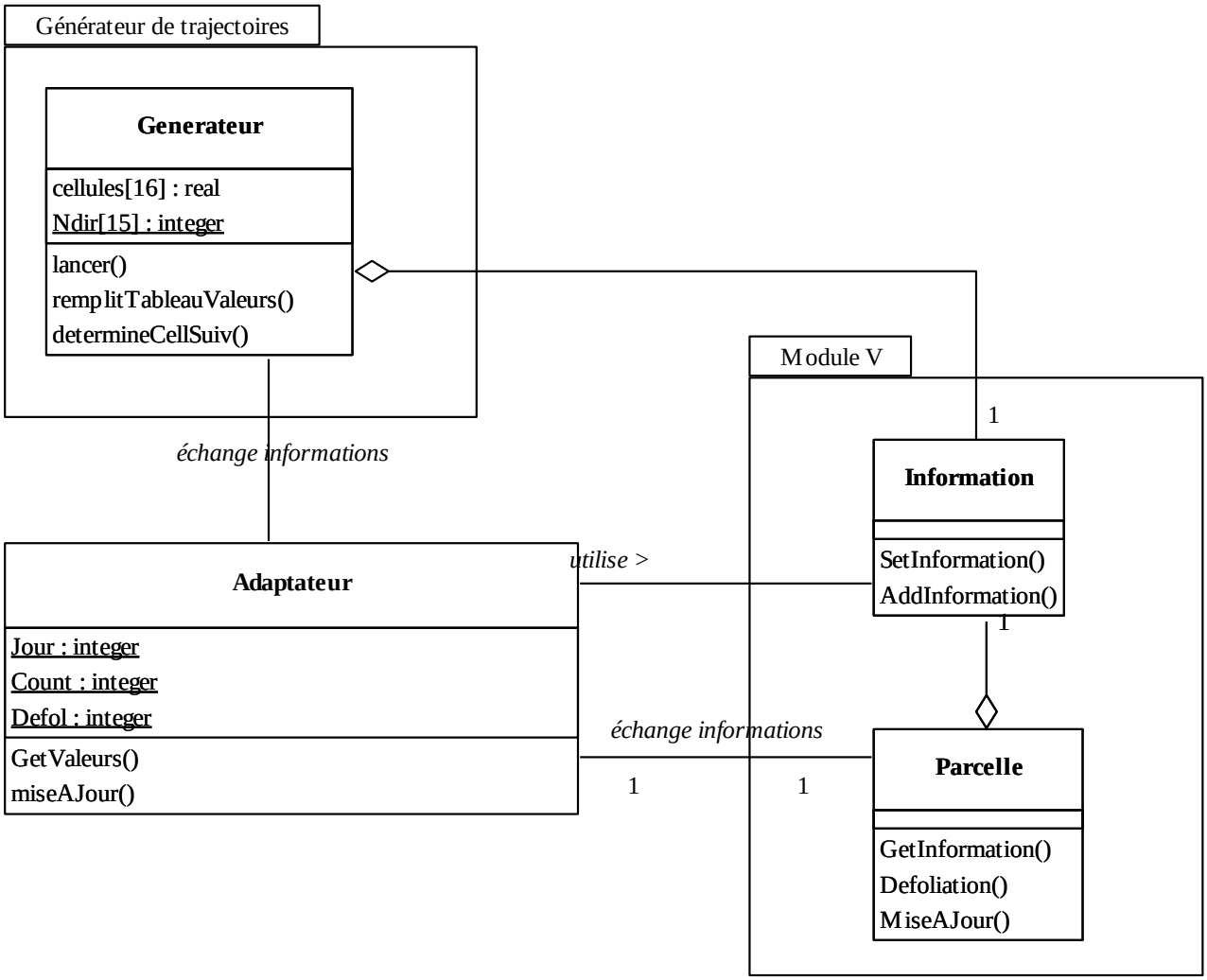


Figure 11 : Diagramme de classe d'interconnexion entre le générateur et le simulateur végétal

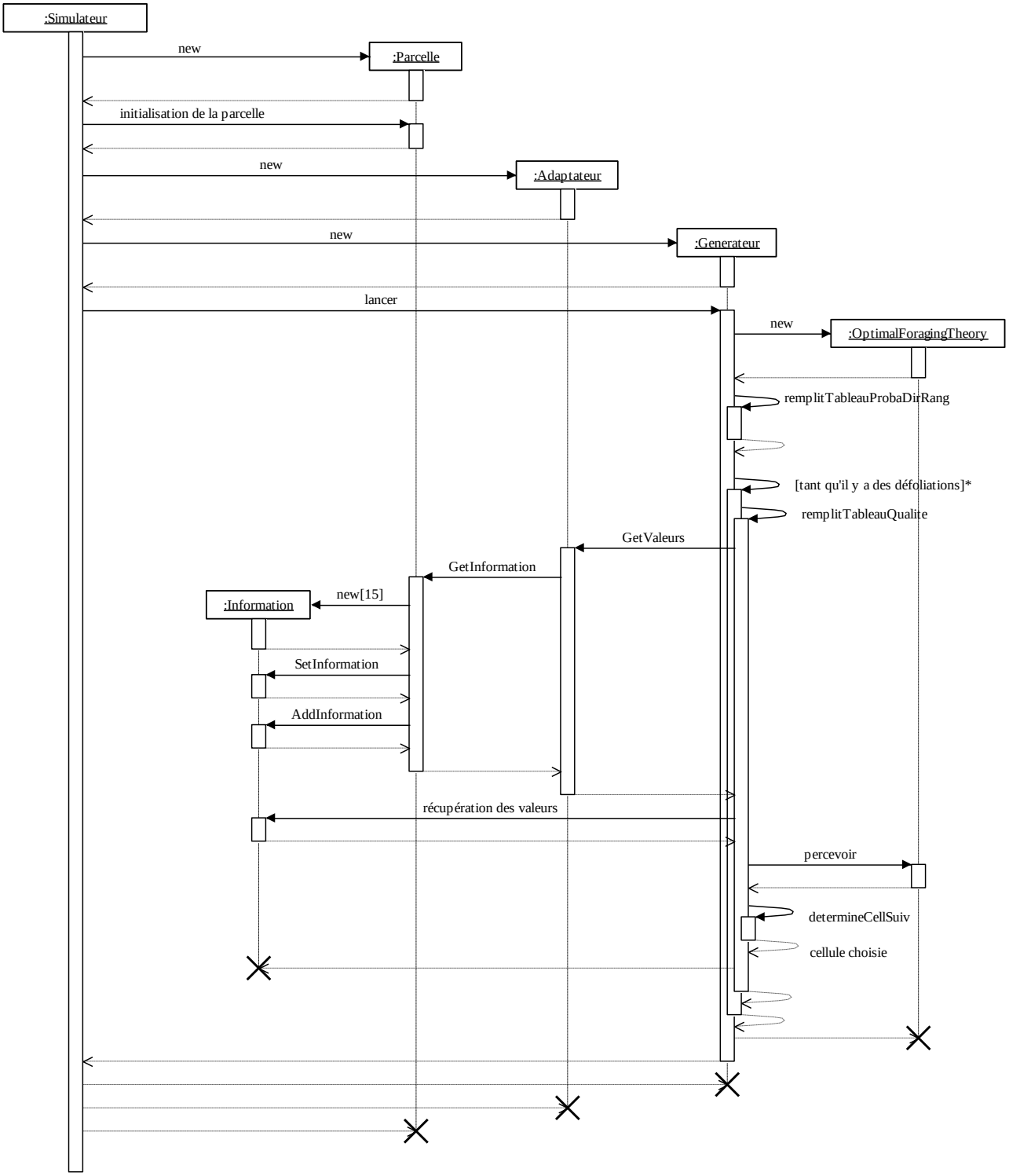


Figure 12 : Diagramme de séquence d'interconnexion entre le générateur et le simulateur végétal

5) Interconnexion avec le simulateur animal

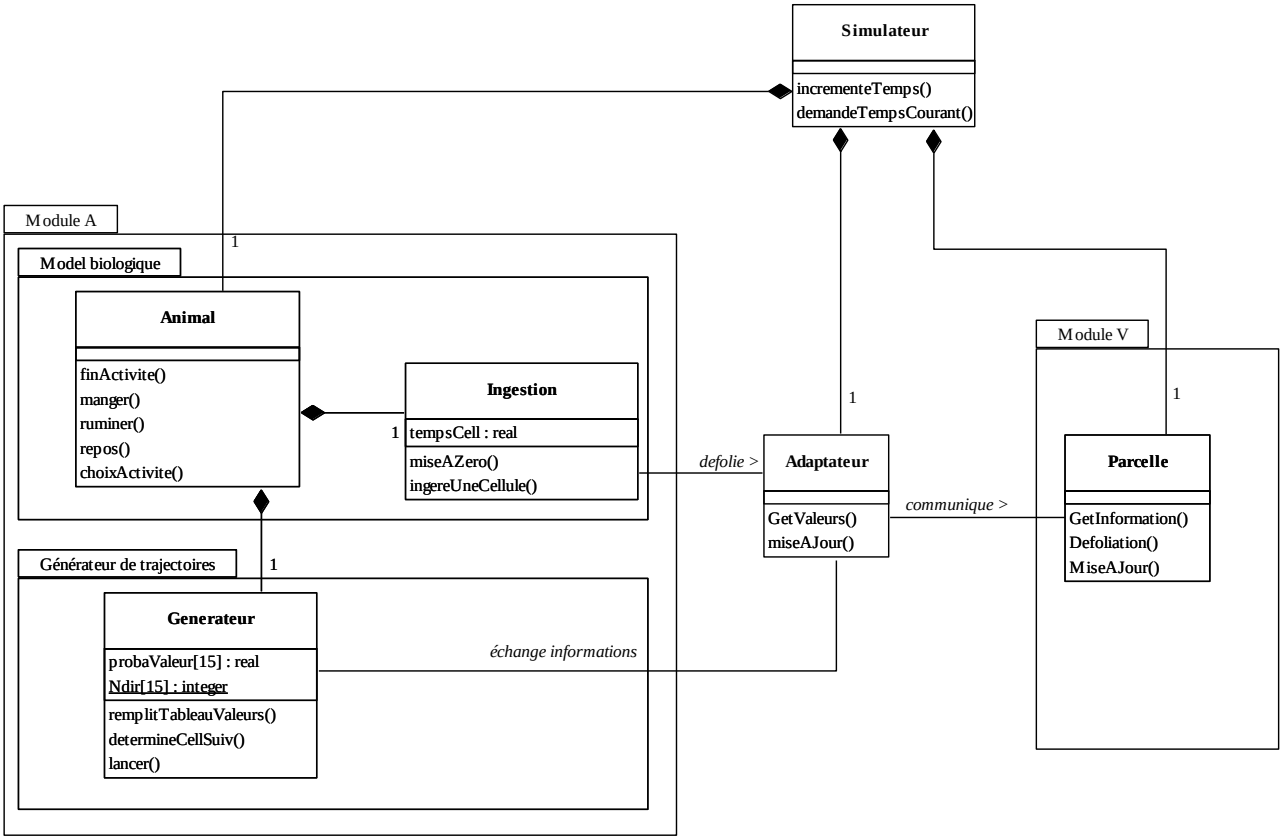


Figure 13 : Digramme de classe d'interconnexion avec le simulateur animal

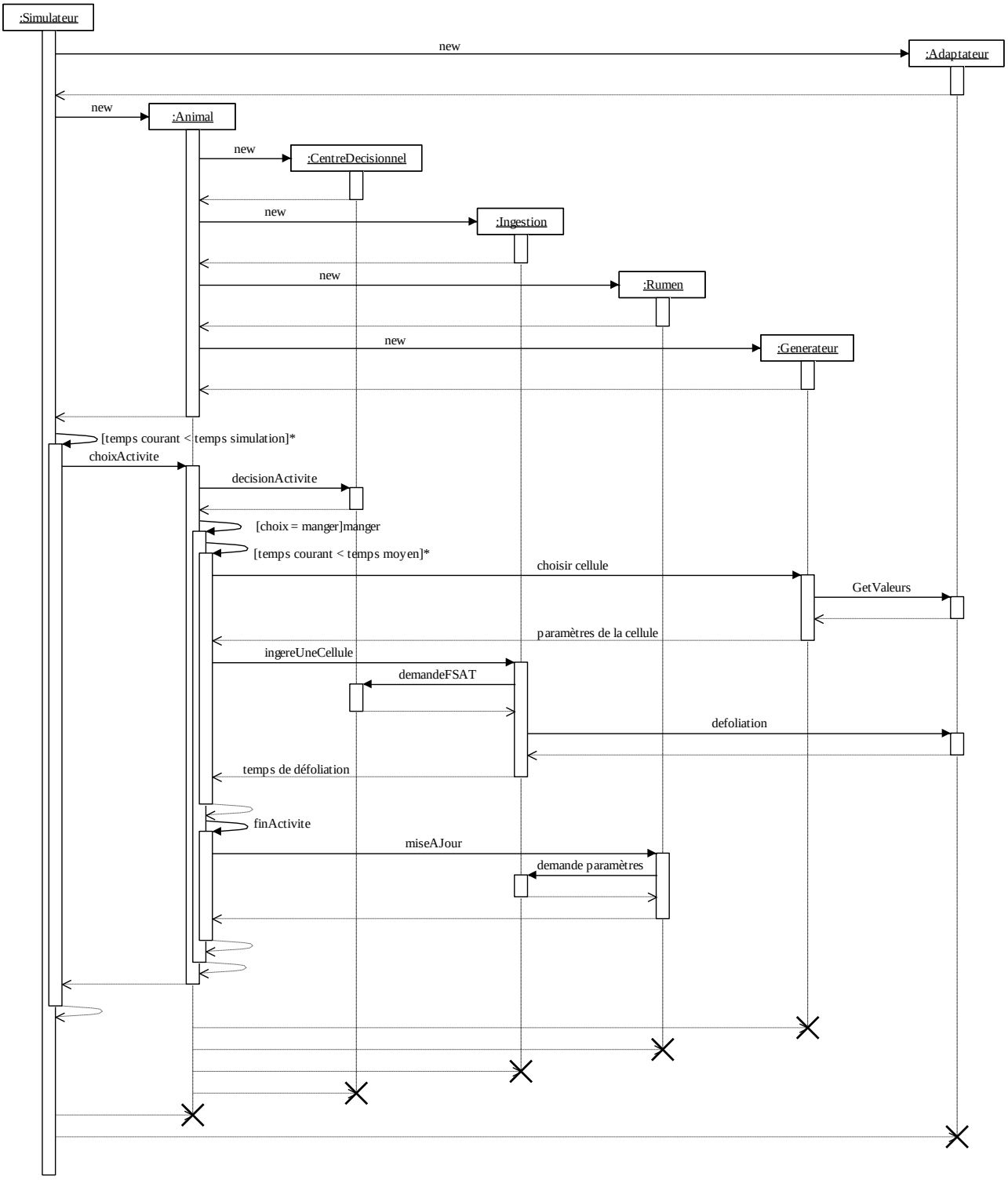


Figure 14 : Diagramme de séquence d'interconnexion avec le simulateur animal