# Migration de l'interface du simulateur FarmSim de SWING vers JavaFX

Barnabé Neyret

École Centrale de Marseille

Institut national de recherche pour
l'agriculture, l'alimentation et l'environnement

Unité mixte de Recherche sur l'Écosystème Prairial

Barnabé NEYRET - Promotion $\beta$
Stage de première année mené du 28/06 au 25/08/2021

# MIGRATION DE L'INTERFACE DU SIMULATEUR FARMSIM [10] DE SWING [35] VERS JAVAFX [24]

Tuteur à l'école : M. Bastien CHATELET
Maître de stage à l'INRAE [19] : M. Raphaël MARTIN

# Remerciements

Je tenais d'abord à remercier Guillaume CHIAVASSA de m'avoir aimablement conseillé pendant ma recherche de stage.

À mon tuteur Raphaël MARTIN, je suis reconnaissant de m'avoir épaulé de manière si proportionnée avec beaucoup de disponibilité.

À Laurence BENEDIT et Jean-Baptiste PAROISSIEN, un grand merci pour votre accueil chaleureux.

Merci encore à Solène TESSIORE pour la qualité de son travail amont, à Pierrick ARNAULT et Maël BAUDOUIN pour leurs critiques constructives.

# Résumé

J'ai passé mon stage dans l'unité chargée de l'étude des prairies, et travaillé sur le simulateur FarmSim [10], qui modélise le stockage de carbone et d'azote dans les parcelles d'une ferme. J'ai modernisé son interface graphique en assimilant les méthodes de l'unité comme le versionnement sur une forge logicielle. J'ai affiné ma conception des enjeux économiques et des relations de travail en entreprise au contact des autres stagiaires.

# Abstract

I spent my internship in the unit in charge of grasslands' study, and I worked on FarmSim [10], a model of carbon and nitrogen storage inside farm plots. I modernized its graphical interface, while putting into practice processes like software versioning. I refined my idea of economic issues and working relationships through contact with other trainees.

# Positionnement thématique

Cycle du carbone
Cycle de l'azote
Interface utilisateur
Gestion de versions

Carbon cycle
Nitrogen cycle
Graphical user interface
Versioning

# Glossaire

| | |
|---|---|
| [1] Anémométrique | Relatif au sens et à la vitesse du vent. |
| [2] ANR | Agence nationale de la recherche. |
| [3] Branche | Version parallèle du code en développement. |
| [4] Buron | Bâtiment traditionnel de passage des élevages pendant leur pâturage. |
| [5] CEMAGREF | Le centre d'étude du machinisme agricole et du génie rural des eaux et forêts est l'ancêtre de l'IRSTEA [21]. |
| [6] Ceres | Modèle élémentaire de FarmSim [10] consacré aux cultures. |
| [7] CHSCT | Le comité d'hygiène et de sécurité pour les conditions de travail est un acteur régional de la prévention des risques. |
| [8] ECODIV | Département de recherche chargé de l'écologie et de la biodiversité des milieux forestiers, prairiaux et aquatiques. |
| [9] Effluents | Fumiers ou lisiers. |
| [10] FarmSim | Simulateur des échanges de carbone et d'azote dans une ferme. |
| [11] Getter | Méthode générique très accessible pour connaître la valeur d'un attribut. |
| [12] GIEC | Groupe d'experts inter-gouvernemental sur l'évolution du climat. |
| [13] GNU | Un système d'exploitation libre pour s'affranchir des monopoles intellectuels. |
| [14] GUI | Graphical user interface. |
| [15] Hygrométrie | Mesure de l'humidité de l'air. |
| [16] IDE | Integrated development environment. |
| [17] IHM | Interface homme-machine. |
| [18] INRA | Institut national de la recherche agronomique. |
| [19] INRAE | Institut national de recherche pour l'agriculture, l'alimentation et l'environnement. |
| [20] IPCC | Modèle élémentaire de FarmSim [10] consacré aux bâtiments qui tire son nom des travaux du GIEC [12]. |

| | |
|---|---|
| [21] IRSTEA | Institut national de recherche en sciences et technologies pour l'environnement et l'agriculture. |
| [22] ISIMA | Institut supérieur d'informatique, de modélisation et de leurs applications. |
| [23] Java | Langage de programmation fortement typé orienté objet. |
| [24] JavaFX | Bibliothèque graphique plutôt récente. |
| [25] JDK | Java download kit. |
| [26] LPR | Loi de programmation de la recherche promulguée en 2020. |
| [27] Modalité | Relation de parentalité entre plusieurs fenêtres qui n'autorise la prise en main que d'une seule fenêtre à la fois. |
| [28] NIRS | Near infra-red spectroscopy. |
| [29] PaSim | Modèle élémentaire de FarmSim [10] consacré aux prairies. |
| [30] POO | Programmation orientée objet. |
| [31] Prions | Protéines infectieuses incriminées dans la maladie de la vache folle. |
| [32] Setter | Méthode générique très accessible pour l'affectation d'un attribut. |
| [33] Signature | Liste d'arguments attendue par une méthode au moment du passage de paramètres. |
| [34] SSH | Le Secure Shell est un protocole de communication informatique sécurisé qui repose sur le chiffrement cryptographique. |
| [35] Swing | Bibliothèque graphique un peu délaissée en terme de suivi. |
| [36] UML | Unified modeling language. |
| [37] UREP | Unité mixte de recherche sur l'écosystème prairial. |
| [38] Widget | Terme qui désigne indifféremment n'importe quelle brique élémentaire d'une interface graphique. |
| [39] Yearlings | Désigne les veaux de l'année passée. |
| [40] Zoonoses | Maladies transmises à l'homme par les animaux. |

# Ouverture à la diffusion

Les captures d'écran qui figurent dans ce rapport, à plus forte raison celles issues de mon travail, ne sont pas soumises à confidentialité. Mon tuteur et moi nous sommes entendus sur le fait que ces images puissent être librement diffusées, d'autant qu'elles ne constituent pas le moindre prototype de solution commerciale.

# Table des matières

# Introduction

Je souhaitais bien sûr découvrir l'environnement de travail typique d'un ingénieur mais pas forcément participer à de la production futile ou polluante à échelle industrielle pour rester en cohérence avec mes valeurs.

C'est pourquoi j'avais diversifié mes recherches autour de la conception mécanique pour la mobilité durable, de la performance énergétique des bâtiments, ou encore de la modélisation climatique. Mon stage se rapporte finalement davantage à ce dernier centre d'intérêt.

Bien qu'il soit d'abord un établissement de recherche, l'INRAE [19] prend position dans le secteur privé et le tissu économique avec des retombées nombreuses en terme de conseils et de procédés industriels pour l'agriculture du futur.

Du reste, mon stage s'inscrivait à sa manière dans la découverte de la production, dans la mesure où j'ai participé à la mise à jour d'un livrable logiciel en suivant les pratiques de développement déployées par l'INRAE [19].

# I - Présentation de l'entreprise

## I.1 - Un ancrage aux besoins

En tant qu'organisme de recherche, l'INRAE [19] se situe dans le secteur tertiaire. L'Institut réalise des prestations intellectuelles publiques mais prodigue également des conseils dans le privé et dans le secteur primaire auprès des agriculteurs.

Historiquement l'INRAE [19] a toujours été très attentif aux besoins agricoles nationaux, c'est au départ une institution de l'État et cette émanation politique se fait encore ressentir aujourd'hui. En 1946, l'INRA [18] est fondé sur décret pour que la France retrouve son autonomie alimentaire malgré les débours de la guerre. [6] Il participe alors à la diffusion des méthodes de traitement efficaces et au développement des procédés industriels dans les exploitations.

La perennité de l'Institut a été confortée par la mutation des enjeux alimentaires [11] et le besoin de préparer l'agriculture nationale aux changements climatiques. L'INRA [18] s'est chargé d'atténuer le réchauffement d'origine agricole par une gestion plus responsable des élevages notamment, et de sélectionner les pratiques et les cultures résilientes.

Dans ce contexte, l'UREP [37] fait des campagnes de mesures et de prélèvements sur de vraies parcelles, pour théoriser l'intérêt de la diversité dans la résistance des espèces aux environnements hostiles. Un grand pôle informatique (en proportions) prend le relais des données mesurées pour faire de la modélisation prédictive.

C'est finalement sur décision ministérielle que l'INRA [18] fusionne avec l'IRSTEA [21] en janvier 2020 pour former l'INRAE [19]. Ce rapprochement est certes motivé par la proximité des compétences mais c'est aussi la marque de la planification politique des travaux de recherche en agro-écologie.

# I.2 - Production et mise en concurrence

Le dépôt de brevets et la publication de papiers de recherche sont le coeur de métier à l'Institut. L'IRSTEA [21], anciennement le CEMAGREF [5], a notamment participé au développement de machines agricoles modulables pour les terrains pentus des alpages. Du reste, la production scientifique se quantifie par le nombre d'articles publiés par les chercheurs de l'unité. Le nombre de publications parues dans les magazines à comité de lecture montre encore davantage la reconnaissance internationale des travaux de l'unité.

Les bases de données constituées au gré des années dans les fermes expérimentales font évidemment partie de la production intellectuelle de l'INRAE [19], puisqu'elles nourrissent des études statistiques et des simulations. L'UREP [37] est aussi dynamique en terme de développement logiciel pour la simulation, et à terme ces outils pourraient se démocratiser dans les organismes de recherche et les entreprises agroalimentaires privées.

Les financements sont principalement de deux ordres à l'INRAE [19]: d'une part, les subventions directes et d'autre part, les ressources propres contractuelles. Les subventions prennent la forme de financements récurrents décorrélés du volume et de la qualité de la production. Mais les syndicats déplorent que l'État commence à conditionner ces ressources, parce que cela rend difficile d'assurer à la fois un bon niveau de recherche et la couverture des dépenses courantes incompressibles pour le maintien des postes et l'entretien des infrastructures. [2]

Les contrats publics sont des financements plus incitatifs. L'État octroie des fonds à l'ANR [2] et formule de grandes orientations pour la recherche, qui entrent souvent en résonance avec les besoins de l'actualité. La recherche sur les zoonoses [40] a par exemple retrouvé de l'intérêt dans le contexte pandémique.

L'Institut est alors soumis à concurrence dans l'écosystème foisonnant des organismes de recherche pour gagner l'affectation des projets de l'ANR [2], et percevoir les ressources attribuées. Dans ces conditions, les chercheurs doivent constituer un véritable dossier de démonstration de leur savoir-faire, de leur capacité à réaliser les missions demandées, de leur engagement à accorder du temps de travail au projet. L'ANR [2] consacre aussi une partie de ses fonds aux projets les plus matures à l'initiative des laboratoires, pour libérer l'innovation des directives de recherche.

Les détracteurs de ce mode d'allocation dénoncent les disparités de répartition, le "clientélisme" des recherches "court-termistes" [8] et les risques de conflits d'intérêt. En effet, la sélection des équipes scientifiques pour mener les projets de l'ANR [2] est assurée par des membres de l'Institut sous mandat particulier. [7] En tout état de cause, l'INRAE [19] est traversé par une injonction de productivité que ce soit par son but de sécuriser les rendements agricoles, ou par la transformation du métier de chercheur vers des standards plus jalonnés.

Le reste des ressources propres contractuelles provient des bureaux de recherche et développement de l'industrie, qui monnayent les prestations ou les droits d'exploitation des propriétés intellectuelles de l'Institut.

La stratégie de l'Institut repose d'abord sur l'orientation nationale et l'optimisation locale de la recherche. L'UREP [37] connait une hausse de publications et de contrats de recherches, mais les embauches restent limitées. Elles sont souvent concurrencées par l'achat ou le renouvellement de matériel coûteux : tout dernièrement, l'UREP [37] a fait l'acquisition d'un dispositif NIRS [28] pour scanner l'activité physiologique des plantes de manière non invasive. Les fournisseurs métrologiques qui sont en position de monopole sur un marché de niche, font également peser de fortes immobilisations sur l'unité. D'ailleurs, les services comptables d'appui à la recherche n'hésitent plus à responsabiliser les dépenses poste par poste pour compenser.

## I.3 - Implantation et structure

Comme c'est un établissement de recherche public, l'Institut n'a pas d'implantation à l'étranger. Mais les chercheurs sont régulièrement amenés à travailler avec des homologues de tous bords à l'occasion de collaborations internationales. Le découpage est d'une part géo-hiérarchique, d'autre part plus thématique.

La direction nationale régit l'organisation de la recherche à travers une vingtaine de centres régionaux. [11] Chacun regroupe plusieurs unités de recherche plus ou moins indépendantes. L'UREP [37] où j'ai passé mon stage est donc pour ainsi dire un des échelons les plus atomiques d'un centre régional. Cette décomposition hiérarchique est également suivie par les organes de prévention.

L'INRAE [19] dépend à la fois du Ministère de la Recherche et du Ministère de l'Agriculture, c'est d'ailleurs le ministre de l'Agriculture qui nomme son président. Les autres fonctions hiérarchiques comme la direction des centres régionaux, sont pourvues par mandat des chercheurs qui souhaitent consacrer plus de temps à l'animation administrative de la recherche.

Les directeurs d'unités sont quant à eux nommés par les directeurs de leur département de tutelle. Un département désigne un grand champ de recherche, c'est une subdivision thématique de toute la recherche menée par l'INRAE [19]. Les unités qui font partie du même département ont plus fréquemment des interactions privilégiées autour de projets communs. De son côté, l'UREP [37] appartient au département de recherche ECODIV [8]. Il y a en tout 14 départements [11] qui déclinent les missions générales de l'Institut.

Pour donner un ordre de grandeur, l'INRAE [19] emploie près de 12000 salariés [11] dont 73% de titulaires et le reste de contractuels. À lui seul, le centre Clermont/Auvergne/Rhône-Alpes compte pour 840 collaborateurs, tous agents confondus.

## I.4 - Valeurs partagées

J'ai toujours perçu un climat très convivial au sein de l'unité, que ce soit entre les générations de chercheurs ou les différents corps de métier. En effet l'INRAE [19] recrute à la fois des chercheurs, des ingénieurs d'étude, des ingénieurs de recherche et des techniciens. Sans compter les nombreux stagiaires, doctorants et post-doctorants qui sont les bienvenus pour apporter du dynamisme à la recherche.

De coutûme, les pauses sont partagées entre collègues dans le cadre champêtre très agréable de l'unité. Je ne peux pas non plus manquer de mentionner les *mardis de l'UREP* [37], des rencontres quasi-hebdomadaires qui relevaient presque de l'institution : ils rassemblent toute l'unité, pour un exposé par les pairs (y compris les stagiaires) des travaux qui sont en cours.

Outre un esprit de corps assez familial, j'ai trouvé de la sensibilité écologique à bien des égards : les infrastructures sont adaptées au transport à vélo, le personnel apprécie être au contact de la nature même en dehors des heures travaillées, certains sont engagés dans la vie associative et le bénévolat. La plupart est attentive à la cause des agriculteurs malmenés par les directives européennes et par la grande distribution.

# II - Rencontre avec la production Ma place et mes réalisations dans l'entreprise

## II.1 - Positionnement général

J'ai rejoint l'équipe de modélisation informatique, qui accueillait alors beaucoup de stagiaires. Je me retrouvais donc parmi d'autres étudiants généralement en cours de travaux de fin d'étude (fin de master ou d'école d'ingénieur). J'allais occuper un poste de bureau consacré au développement.

Swing [35] et JavaFX [24] sont deux bibliothèques graphiques associées au langage de programmation Java [23], c'est-à-dire des outils supplémentaires spécialisés pour réaliser des interfaces graphiques. JavaFX [24] est plus moderne que Swing [35], et là où le bât blesse, c'est que Swing [35] ne sera bientôt plus mise à jour, donc les chercheurs pourraient avoir des problèmes de compatibilité d'un jour à l'autre en continuant de l'utiliser.

FarmSim [10] est le simulateur des transferts de carbone et d'azote entre l'air et les parcelles agricoles développé par l'UREP [37]. En tant que projet d'assez longue date, l'affichage reposait complètement sur la bibliothèque Swing [35]. Ma mission consistait donc à reproduire le comportement visuel et interactif du programme dans la nouvelle bibliothèque, avec sporadiquement des changements dans la conception de la maquette pour que l'application métier devienne plus intuitive et plus agréable d'utilisation.

## II.2 - Processus opérationnel de l'unité

À travers la modélisation fine des prairies, l'UREP [37] cherche à démontrer les enjeux de prospérité de l'agriculture extensive. Pour ce faire, l'unité s'est dotée d'une procédure de développement logiciel normative, le versionnement. On parle de forge logicielle pour désigner une zone de dépôt délocalisé qui permet de maintenir les versions progressives d'une application, et de travailler ensemble sans risquer de se neutraliser l'un l'autre. Chacun agit séparément jusqu'à soumettre au tronc commun une amélioration fonctionnelle du code. Chaque développeur peut ainsi proposer des mises à jour individuelles et bénéficier des mises à jour collectives.

La chaîne d'approvisionnement est relativement réduite à l'Institut. La production intellectuelle se fonde sur les observations de terrain et les simulations. Les chercheurs eux-mêmes font des coupes régulières sur les prairies tutélaires pour étudier des échantillons. L'herbe fauchée est triée puis séchée dans des étuves parce qu'on ne réalise de mesures quantitatives que sur la matière sèche. Toutes les autres mesures prises dans les fermes expérimentales sont informatisées à la source.

Le contrôle de la qualité se fait aussi beaucoup en interne. L'équipe informatique détecte les anomalies dans les mesures automatisées. Surveiller qu'elles ne dépassent pas une limite raisonnable en moyenne et dans l'absolu a permis d'isoler un appareil défectueux qui enregistrait des records de précipitations invraisemblables. Les informaticiens tiennent aussi à présenter des résultats reproductibles même dans les procédés stochastiques : des simulations lancées avec les mêmes paramètres doivent retourner au moins statistiquement les mêmes résultats. L'intégrité du parc de spectrophotomètres garantie par l'assistante de prévention et le principe d'évaluation par les pairs, finissent d'asseoir la qualité des résultats de l'unité.

Pour ma part, j'avais grande latitude sur les choix de mise en oeuvre [C] mais tout au long de mon stage, j'ai consacré du temps à résoudre les bugs soulevés par la transition. [D] J'ai fait un état des lieux final de la qualité de ma production [F] pour faire connaître les points d'amélioration restants.

Les informaticiens travaillent beaucoup en relation avec les autres services de l'unité parce que FarmSim [10] a d'abord pour vocation d'éprouver les prédictions des autres chercheurs via des simulations reproductibles à loisir. Du reste, l'équipe informatique se mobilise continuellement pour perfectionner l'efficacité de la recherche. Les exemples qui suivent en attestent.

### La réduction du temps de simulation

Les équipes de chercheurs peuvent être amenées à faire des simulations très gourmandes avec un volume de données important (en nombre d'années et en nombre de parcelles étudiées). Les informaticiens essaient donc de mettre en place la parallélisation des programmes de simulation sur plusieurs cœurs (quand les ordinateurs ou serveurs en disposent). Il s'agit tout simplement de mobiliser simultanément toutes les capacités de mémoire de l'appareil pour effectuer plus de tâches en parallèle. Il en résulte un gain de temps considérable pour les grandes campagnes de simulation. À titre d'exemple, l'Institut participe à l'initiative *4 pour 1000* commandée par les *Accords de Paris*, qui consiste à évaluer les bienfaits sur l'effet de serre additionnel d'une augmentation du stockage de carbone de 0.4 % par an dans les premières couches du sol. [1]

### L'optimisation des jeux de paramètres

Le simulateur FarmSim [10] mobilise plusieurs modèles élémentaires, dont les plus remarquables sont PaSim [29] pour décrire les prairies non cultivées, Ceres [6] pour les cultures, IPCC [20] pour modéliser les transferts de carbone et d'azote avec les bâtiments. Avec eux ce sont autant de paramètres que l'on peut moduler pour simuler le comportement d'une ferme. Toutes les calibrations ne donnent pas de bons résultats en terme de minimisation de l'erreur entre les observations disponibles et les simulations. Des doctorants ont déjà formulés quatre sets de paramètres, qui sont chacun performant pour une restriction à certains observables mais globalement imprécis. L'équipe informatique essaie donc d'implémenter la proposition systématique du meilleur jeu de paramètres pour l'ensemble de sites en jeu et de variables mesurées.

### La liaison haut-débit avec la station de mesure

L'Institut a déployé une unité expérimentale autour d'un ancien buron [4] de Laqueuille (63). C'est devenu un véritable objet d'étude pour les chercheurs à propos des transferts de carbone et d'azote en milieu prairial. Le site est équipé d'une station de mesure en temps réel, dont les données sont quotidiennement transférées vers le serveur de l'UREP [37]. L'équipe informatique a fini par installer la fibre optique ainsi qu'un pont hertzien à force de devoir réparer les vices de format parmi les fichiers reçus. En effet, il arrivait souvent que les relevés dépassent la durée d'enregistrement attendue. Ce qui ne manquait pas de convoquer un effort de traitement assez artificiel pour scinder les mesures en bulletins journaliers.

## II.3 - Ma contribution

Mon travail est d'abord passé par la recherche des équivalents [E] les plus fidèles aux composants graphiques de Swing [35] en JavaFX [24]. J'ai de ce fait transposé tous les éléments de l'affichage comme les boutons, les champs de texte, les tableaux ou les titres de sections. Le transfert de la plupart de ces widgets [38] a pu se faire de manière assez intuitive avec l'auto-complétion de mon IDE [16].

J'avais pris le temps de prioriser mes tâches sur la base de trois critères fondamentaux : le caractère indispensable de chaque onglet pour un niveau minimal d'utilisation, une estimation de la quantité de travail au nombre de widgets [38] près, et une prévision de la difficulté en terme de réajustements.

Car je devais trouver des moyens de substitution quand il n'existait pas de correspondance directe entre les deux librairies. J'ai d'ailleurs fait une petite étude comparative pour étayer ces changements de paradigme.

Il n'y avait pas de modèles d'encarts titrés spontanément disponibles en JavaFX [24] donc j'ai mis au point une trame personnalisée, à partir d'un contenant à empilement vertical labellisé. Cela m'a tout de suite permis de donner à FarmSim [10] un cachet graphique uniforme de manière très factorisée.

Les boîtes de dialogue intégrées étaient elles aussi trop limitées pour proposer un formulaire de configuration varié. C'est pour cela que j'ai développé mes propres pop-ups modales [27], également plus homogènes avec le reste de mes intentions graphiques.

Le *CardLayout* de Swing [35] était un contenant de widgets [38] sans équivoque, qui permettait de commuter rapidement entre plusieurs contenus pour la même section à l'affichage. J'ai décidé de le remplacer par le *StackPane* de JavaFX [24] qui agit comme une sorte de pile et n'affiche que le dernier contenu ajouté. Je procédais alors par échange des statuts de visibilité pour reproduire la bascule graphique entre les contenus.

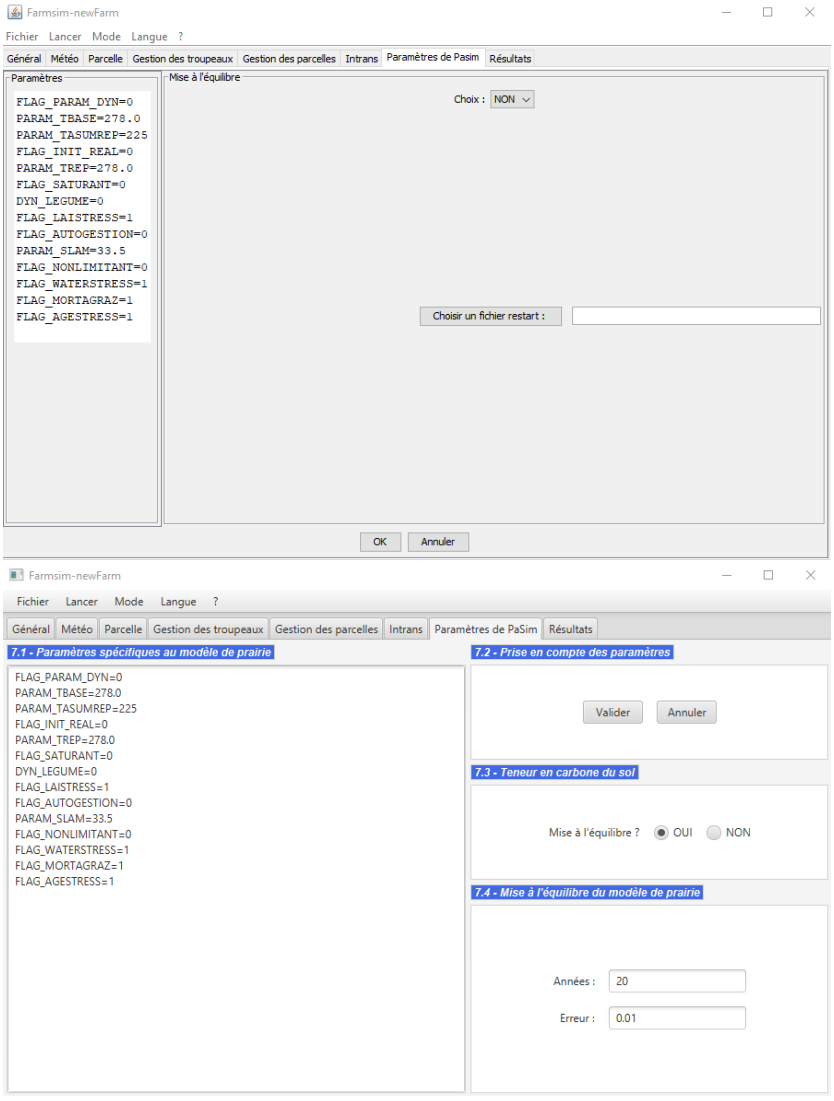



Fig. 1 Section alternative sur l'ancienne et la nouvelle maquette

Crédits : Raphaël MARTIN, FarmSim, UREP, INRAE

À la grande différence de Swing [35], JavaFX [24] reconnaît largement que les tableaux servent à la manipulation d'objets formatés comme les enregistrements d'une base de données. [12] Un tableau se définit naturellement par l'ensemble de ses colonnes, c'est-à-dire par le schéma du type d'enregistrement admissible. Une ligne en tant qu'objet se présente alors très bien comme une séquence d'attributs qui répondent aux en-têtes des colonnes. À partir de là, on ne raisonne plus sur les coordonnées des cellules pour les opérations courantes de notre application. On appelle plutôt tel attribut de tel enregistrement pour atteindre la valeur d'une cellule. Je vous propose de faire l'expérience de pensée sur le tableau ci-contre pour la valeur des précipitations à une date donnée. Comme vous le voyez, les méthodes de chargement, d'affichage et de sauvegarde sont à la fois bouleversées mais beaucoup plus personnalisées.

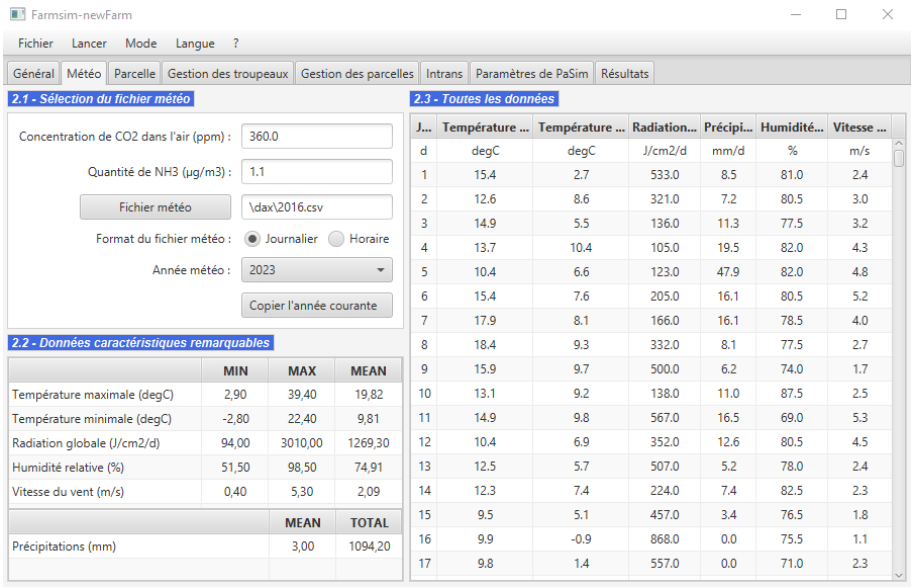



Fig. 2 Une nouvelle manière de penser les tableaux

J'étais aussi source de propositions pour imaginer un agencement plus ergonomique de l'interface, et profiter des fonctionnalités additionnelles de JavaFX [24]. J'ai notamment fait remarquer que l'alternance des rubriques mutuellement exclusives gaspillerait moins d'espace propre à l'affichage que leur persistance simultanée.

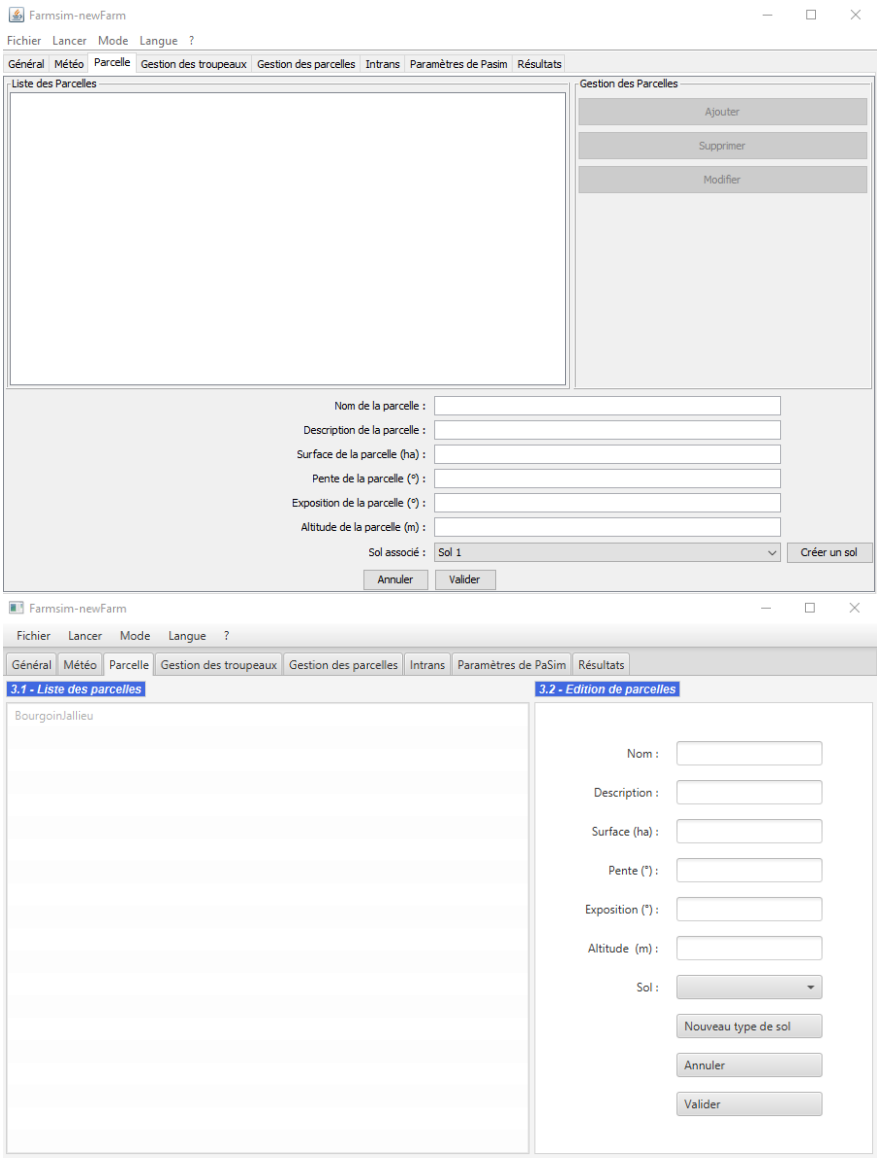



Fig. 3 Simplification de l'interface des parcelles

Crédits : Raphaël MARTIN, FarmSim, UREP, INRAE

Du reste, j'ai toujours pris soin de positionner mes widgets [38] relativement à la taille de la fenêtre plutôt que de leur conférer des coordonnées d'ancrage arbitraires, pour éviter toute distorsion des proportions. J'ai d'ailleurs implémenté une politique harmonieuse d'expansion des widgets [38] pour qu'ils se partagent équitablement le gain de largeur ou de hauteur à occuper quand on agrandit la fenêtre.

Mon travail a aussi consisté, mais dans une moindre mesure, à rédiger de la documentation [F] à l'attention des futurs stagiaires qui travailleraient sur FarmSim [10]. J'ai déterminé le rôle de chaque paquet dans l'arborescence assez vaste du GUI [14]. J'espère ainsi que cela pourra épargner un travail préliminaire de rétro-ingéniérie à chaque nouveau contributeur.

Dans un registre moins technique également, je devais maintenir l'internationalisation des libellés, c'est-à-dire renseigner leurs traductions dans des fichiers séparés, pour que le GUI [14] vienne piocher la légende adaptée selon la langue choisie par l'utilisateur.

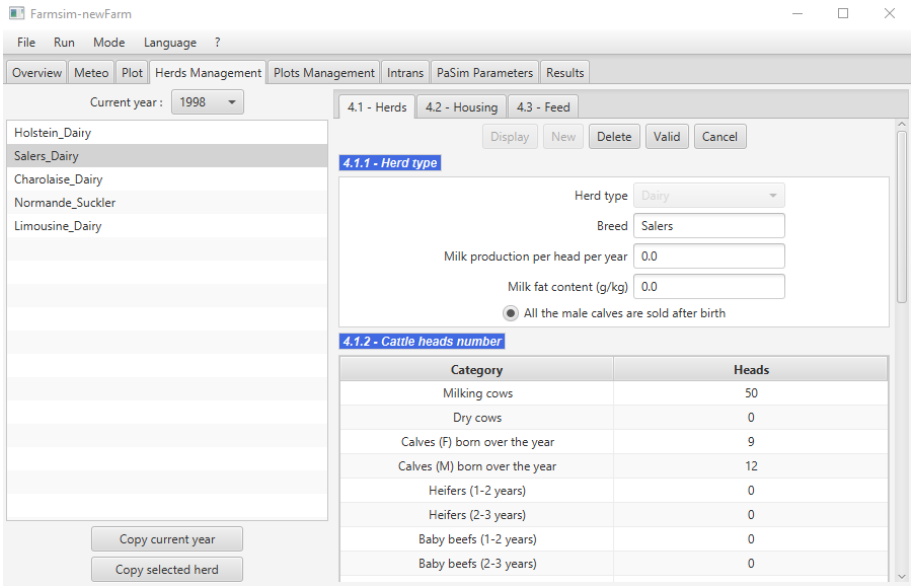



Fig. 4 Gestion anglophone des troupeaux

# II.4 - Conditions de travail

En ce qui concerne la prévention, l'INRAE [19] se positionne sérieusement vis-à-vis des obligations légales. Chaque nouveau venu est tenu de se former à la prévention des risques conformément au poste qu'il va occuper. L'Institut dispose d'un outil de pilotage de la prévention qui indique, pour chaque danger recensé, le seuil d'exposition à ne pas dépasser. Plus concrètement, l'équipe de prévention fait des recommandations pour chaque risque en terme de matériel et d'environnement de travail, de précaution individuelle et d'organisation. Elle met d'abord en place des équipements de protection collective avant d'envisager des mesures individuelles pour diminuer les risques qui restent inévitables. Cette équipe se décline au niveau régional autour des médecins de prévention et du CHSCT [7], au niveau unitaire autour de l'assistant(e) de prévention.

Les risques professionnels d'un travail de bureau comprennent les risques généraux comme le risque électrique, auxquels s'ajoutent quelques risques spécifiques qui peuvent être visuels ou posturaux. Les incidents, si tant est qu'ils surviennent, seraient classés comme des maladies professionnelles, par opposition aux accidents du travail qui se déclarent ponctuellement. C'est le cas notamment des troubles musculo-squelettiques comme les douleurs cervicales.

Mais certains agents de l'Institut sont beaucoup plus exposés que je ne l'ai jamais été, parce qu'ils manipulent des produits phytosanitaires plus régulièrement que les agriculteurs eux-mêmes, ou qu'ils travaillent encore sur les prions [31].

## II.5 - Importance des relations humaines

Comme je l'ai évoqué précédemment, j'ai trouvé qu'il y avait un véritable climat de bienveillance et de la synergie entre les corps de métiers à l'INRAE [19]. Les échanges informels font partie intégrante de la vie de l'unité et participent au tissage de connexions entre les projets comme au partage de l'expérience. Certaines compétences métrologiques pour la bonne utilisation des appareils de mesure, et botaniques pour l'identification des herbes fauchées, dépassent parfois le cadre des formations initiales, d'où l'importance d'entretenir le savoir-faire par le travail en commun. À ce sujet, la sauvegarde de jours de présence commune sur site a été débattue lors de l'assemblée générale de l'unité, afin d'éviter la raréfaction des échanges et l'isolement par le télétravail. Le comité d'entreprise organise quant à lui des évènements sportifs ou culturels pour la fédération des équipes.

De mon côté, je participais à des réunions hebdomadaires avec l'ensemble de l'équipe informatique, qui servaient d'une part à lever les points de blocage à l'avancement de chacun, d'autre part à mentionner des astuces d'intérêt collectif. C'était surtout l'occasion de contribuer de manière conviviale à la symbiose de nos projets. Ce qui nous permettait le reste de la semaine de nous solliciter les uns les autres au gré de nos besoins. De mon point de vue, c'était pratique et plutôt bienvenu de recevoir des avis sur les pistes d'amélioration de mon interface en terme d'expérience utilisateur.

# Conclusion

Le plus important était de préserver le fonctionnement correct de l'application de bout en bout de la migration. Je peux dire que c'est chose faite, étant donné qu'on pourra lancer les mêmes simulations que depuis l'ancienne interface (moyennant quelques adaptations), pour retrouver les mêmes résultats.

Un second critère de réussite était de livrer une portion de code homogène c'est-à-dire complètement débarrassée des occurences de Swing [35]. C'est ce à quoi je me suis attelé à la fin de mon stage, de manière à ce que l'évolution de l'interface soit suffisament franche et claire pour être définitivement exploitée. J'ai veillé à fonctionnaliser le plus de routines possible pour factoriser mon code et le rendre plus lisible.

En tout cas, j'ai beaucoup appris pendant ce stage. J'ai vraiment apprécié de mettre en pratique les notions abordées pendant l'année sur les bases de données. Je pourrais également réinvestir les acquis [A] de POO [30] dans l'électif consacré de deuxième année. J'aurais d'ores et déjà diversifié ma palette de langages de programmation. J'ai d'ailleurs essayé de capitaliser l'essentiel de mes découvertes dans les annexes [B] qui suivent.

À titre de suggestion, mon tuteur et moi sommes favorables à de la coopération avec l'école sur des projets transverses pour donner de la continuité à cette modernisation de FarmSim [24].

# Bibliographie


[1] **Amelung, W., Chabbi, A. et al.** (2020) *Une gestion durable des sols agricoles pour séquestrer le carbone et limiter le changement climatique.* Communiqué de presse de l'Institut.

[2] **Archimède, H., Auclair, D., Barbacci, A. et al.** (2015) *50 chercheurs, collectifs et élus tirent la sonnette d'alarme.* Manifeste de la CGT INRA [18].

[3] **Bonnin, N.** (2020) *GUI* [14] *avec JavaFX* [24]*.* Plateforme pédagogique du lycée La Martinière Monplaisir.

[4] **Bouyé, F.** (2015) *Comment définir son UI* [14] *hors de son code source en JavaFX* [24]*.* Forum de discussion Java [23].

[5] **Butler, L., Cropper, J., Johnson, R. et al.** (1997) *Livestock nutrition, husbandry and behavior.* National range and pasture handbook, Natural resources conservation service, United states department of agriculture.

[6] **Cornu, P., Valceschini, E. et Maeght-Bournay, O.** (2018) *L'histoire de l'INRA* [18]*, entre science et politique.* Éditions Quæ. ISBN : 978-2-7592-2638-2

[7] **Ghys, G. et Gille B.** (2006) *La gestion par l'INRA* [18] *de certains programmes de l'Agence nationale de la recherche.* Rapport de l'Inspection générale de l'administration de l'Éducation nationale et de la Recherche.

[8] **Gliese, R.** (2020) *LPR* [26] *et ANR* [2]*.* Presse universitaire.

[9] **Jenkov, J.** (2020) *JavaFX* [24] *tutorial.* Tutorials for software developers.

[10] **Kaeffer, C.** (2015) *Estimation du chargement en pâturage équin : le système UGB.* Techniques d'élevage.

[11] **Mauguin, P., Cherbut, C. et al.** (2020) *Plaquette de présentation de l'INRAE* [19]*.* Documentation officielle de l'Institut.

[12] **Redko, A.** (2013) *Using JavaFX* [24] *UI* [14] *Controls.* Documentation technique d'Oracle.

[13] **Yon, L., Garcia, B., Hill, D. et al.** (2018) *Support de cours sur la programmation orientée objet.* Plateforme pédagogique de l'ISIMA [22].

# Annexes

## A - Documentation avancée

Cette annexe montre les points saillants de ma recherche documentaire, et leurs domaines d'application respectifs dans le GUI [14] de FarmSim [10].

### A.1 - Héritage

A.1.1 - Principe

L'héritage consiste à spécialiser une classe mère en plusieurs classes filles dotées des champs communs de l'héritage et de champs propres supplémentaires. On peut convoquer le constructeur parental par la méthode *super()*.

*UML* [36] //représentation qui se prête à l'héritage
NomClasse
attributs
méthodes

A.1.2 - Intérêt

L'héritage fait en sorte que le code des classes supérieures soit plus éprouvé donc plus fiable, ce qui épargne des efforts de développement et de maintenance. Mais le moindre héritage doit être véritablement justifié par une discrimination fonctionnelle, sous peine sinon de multiplier inutilement les spécialisations qu'un simple attribut aurait pu départager. [13]

A.1.3 - Limites

On ne peut pas hériter des champs d'accessibilité plus restrictive que publique ou protégée.

public
protected
package-protected //niveau d'accessibilité par défaut
private

Une même classe ne peut pas non plus hériter de plusieurs parents pour ne pas soulever de conflits de méthodes. Toutefois l'interfaçage multiple permet de lever les difficultés que pose l'héritage simple.

## A.2 - Polymorphisme

### A.2.1 - Classe abstraite

Une classe abstraite n'a pas d'instances, donc on ne l'utilise qu'en tant que "superclasse d'une hiérarchie" pour définir "un cadre de travail polymorphique". [13] C'est une sorte de classe désincarnée mais qui pose des lignes de conduite pour tous ses descendants. Les méthodes abstraites qu'on y déclare, sont des méthodes dont on ne sait encore rien - hormis leur signature [33] - parce qu'elles seront redéfinies par les classes héritières. On parle plutôt de polymorphisme faible quand on ne fait que surcharger une méthode par plusieurs variantes de signatures [33]. [13]

### A.2.2 - Interface

Une interface désigne un ensemble de méthodes abstraites, autant dire une classe abstraite sans attribut. Les méthodes d'une interface fixent un contrat pour les classes candidates, dans la mesure où toute classe qui voudra l'implémenter, devra implémenter chacune des méthodes annoncées avec le bon set d'arguments passés en paramètres. On parle d'interface fonctionnelle quand il n'y a qu'une seule méthode abstraite : une classe donnée n'a plus qu'à fournir une seule opération pour l'implémenter.

## A.3 - Mise en oeuvre

### A.3.1 - Syntaxe

public class *NomClasseFille* extends *NomClasseMere* implements *PremiereInterface*, *DeuxiemeInterface* {

—Attributs
—private int *attributPropre*;

—Constructeur
—public *NomClasseFille*(int *attributPropre*) {
— —super(*signatureDuConstructeurDeLaClasseMere*);
— —this.*attributPropre* = *attributPropre*;
—}

—Méthodes
—private *methodePropre*(signature) { [...]; }
—public *methodeAbstraiteDeLaClasseMere*(signature) { [...]; }
—public *implementationDeLaPremiereInterface*(signature) { [...]; }
—public *implementationDeLaDeuxieme*(signature) { [...]; }

}

### A.3.2 - Champs d'application de l'héritage dans FarmSim [10]

En temps normal les pratiques sont relatives aux années de simulation, mais l'interface des cycles permet de dater algébriquement un enchaînement de cultures de sorte qu'il soit reproductible sur n'importe quelle parcelle à n'importe quelle époque. Les deux modes de gestion présentaient par nature les similitudes et les spécificités qui justifient complèment le recours à l'héritage.

# B - Apprentissage par l'exemple

Cette annexe identifie les points de distinction de la syntaxe propre à Java [23], et les illustre avec des exemples tirés de FarmSim [10].

## B.1 - Typologie des collections

### B.1.1 - Tableaux statiques

Listes statiques indexées numériquement d'éléments du même type.
typeElements[ ] nomTableau = new nomTableau[tailleTableau];
typeElements[ ] nomTableau =
— new nomTableau[ ]{listeDesElements};

### B.1.2 - Listes ordonnées

Listes dynamiques indexées numériquement d'éléments du même type. List⟨TypeElements⟩ nomListe =
—new ArrayList⟨TypeElements⟩();

### Méthodes de l'interface *List*

.add(x); //void
.add(i, x); //void
.set(i, x); //void
.get(i); //TypeObjet
.remove(i); //void
.size(); //int

*ArrayList* est une classe de listes qui implémente l'interface *List*. Les listes sont toujours initialisées vides, et ne contiennent pas de types primitifs mais uniquement des objets.

### B.1.3 - Ensembles non ordonnés

Combinaisons dynamiques d'éléments du même type.
Set⟨TypeElements⟩ nomEnsemble =
—new HashSet⟨TypeElements⟩();

### Méthodes de l'interface *Set*

.add(x); //void
.remove(x); //void
//On doit fournir exactement l'élément que l'on veut supprimer.
.size(); //int

*HashSet* est une classe d'ensembles qui implémente l'interface *Set*.

### B.1.4 - Dictionnaires ou tables de hachage

Ensembles de couples (clé/valeur).
Map⟨TypeCle, TypeValeur⟩ nomDictionnaire =
—new HashMap⟨TypeCle, TypeValeur⟩();

### Méthodes de l'interface *Map*

.put(clé, valeur); //void
.get(clé); //TypeValeur
.remove(clé); //void
.size(); //int

*HashMap* est une classe de dictionnaires qui implémente l'interface *Map*.

## B.2 - Autres spécificités remarquables

### B.2.1 - Types particuliers

Les types *float* et *double* représentent les décimaux avec plus ou moins de précision. On distingue aussi le type *char* du type *String* en Java.

't'; //char
"reset"; //String

### B.2.2 - Itération par compréhension de liste

C'est un parcours direct sur les éléments de l'itérable plutôt que par l'index, qui se transpose naturellement aux énumérations.

```
for (Herd herd:mainInterface.getFarm().getHerds())
—herdsListView.getItems().add(herd);
```

```
enum SelectedEventType{CROPLAND, GRASSLAND, SOWING,
GRAZING, PLOUGHING, CUTTING, IRRIGATION, FERT};
for (SelectedEventType.EventType eventType :
—SelectedEventType.values()) [...];
```

### B.2.3 - Déclinaison conditionnelle

La clause *switch* remplace avantageusement les imbrications conditionnelles pour faire de la disjonction de cas.

```
switch(eventType) {
—case CROPLAND:
— —[...];
—case GRASSLAND:
— —[...];
}
```

Mais l'exploration des cas ne s'arrête pas au premier item vérifié. Il est d'usage de moduler ce comportement par défaut, en clôturant chaque possibilité par un *break*.

### B.2.4 - Casting

Le transtypage s'écrit de manière préfixe entre parenthèses. Il permet d'exploiter correctement les résultats des opérations quand le type renvoyé est trop général ou trop restrictif.

```
DataRow dataRow =
—(DataRow) livestockUnitsTable.getItems().get(idRow);
```

# C - État de l'art des moyens disponibles

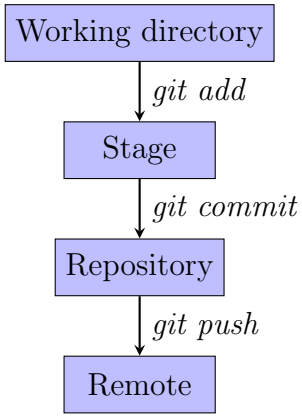
Cette annexe restitue les discussions relatives aux choix de mise en oeuvre, qui me sont venues avec l'expérience au moment de la conception de la nouvelle interface.

## C.1 - Gestion de versions

### C.1.1 - Principe

C'est une méthode de développement logiciel branché [3] qui consiste à maintenir les versions progressives du code sur un dépôt distant.

**Dépot local** Stockage sur son propre ordinateur




**Dépot distant** Stockage délocalisé sur *GitLab*

La connexion SSH [34] permet de faire dialoguer le répertoire de travail avec le dépôt distant. Une même clé peut servir de pont à n'importe quel projet.

### C.1.2 - Méthode

package ⟨ dossier
classe ⟨ fichier .java

### Console *Git* dans le package du projet

*git init* // Création du Repository local
*git add src* // Tous les fichiers .java sont ajoutés au Stage local
*git commit -m "descriptionDuCommit"* // Les fichiers du Stage migrent vers le Repository
*git remote add origin urlHttpsDuProjetSurGitLab* // On définit la zone de dépôt distant
*git push origin master* // On envoie tout le Repository sur la branche -master du dépôt distant

### C.1.3 - Développement distinct

J'ai développé la nouvelle interface dans un projet hors-sol plutôt que dans une branche [3] versionnée de FarmSim [10]. Mais c'était à mon sens l'unique moyen pour que la configuration de l'application soit propre à l'exécution sans besoin de modification intégrale des classes appelées. Je n'ai pas non plus envisagé de partir d'une classe principale codée en Swing [35] et de remplacer chaque onglet par une inclusion de JavaFX [24] (les inclusions ne sont possibles que dans ce sens), parce qu'à la fin la migration aurait demandé une telle restructuration qu'elle m'aurait privé pendant longtemps des progrès visuels.

J'ai finalement adopté une méthode hybride, à savoir tout cloner en dehors du GUI [14], et remplacer chacune des classes Swing [35] du GUI [14] sur place et une par une. Ce qui devrait garantir une certaine proximité avec le code versionné sur *GitLab*, mais j'ai quelques réserves sur l'actualité de l'ensemble parce que j'ai travaillé sur l'interface pendant que le fonctionnement central était lui-même en cours d'évolution. Je me dois de signaler à toutes fins utiles que le GUI [14] est dépendant du reste de FarmSim [10] ne serait-ce qu'au niveau du primo-chargement et de l'internationalisation. J'ai exploité le paradigme du versionnement autant que j'ai pu, je n'ai fait que le reproduire de manière plus manuelle aux endroits où le transfert demandait des efforts trop massifs.

## C.2 - IHM [17]

### C.2.1 - Mise en service de la librairie

JavaFX [24] est une bibliothèque graphique qui n'est pas contenue dans le JDK [25] de base. [9] Après le téléchargement vient donc une étape de couplage avec l'IDE [16]. Sous *Intellij IDEA*, il faut signaler qu'on ajoute une librairie pour que le projet puisse pointer dessus.

#### Ajout

File ⟩ Project Structure ⟩ Libraries ⟩ New Project Library ⟩
repertoireDeLaLibrairie

#### Raccourci du pointeur

File ⟩ Settings ⟩ Path Variables ⟩ Add ⟩
'PATH_TO_FX' ← repertoireDeLaLibrairie

#### Jonction finale

Run ⟩ Edit Configurations ⟩ Add Run Options ⟩ Add VM Options ⟩
–module-path ${PATH_TO_FX}
–add-modules javafx.controls,javafx.fxml

### C.2.2 - Architecture de l'interface

Le *Stage* permet le chargement d'une fenêtre *Windows*. La *Scene* y organise la progression de l'affichage. [9] Scène par scène, on peut construire une arborescence de noeuds graphiques par les ajouts. On distingue les contenants (généralement de suffixe *Pane*), qui répondent à des cas précis d'utilisation dans la disposition spécifique des widgets [38], des widgets [38] à proprement parler.

```
RadioButton node = new RadioButton();
StackPane root = new StackPane();
root.getChildren().add(node);
Stage primaryStage = new Stage();
primaryStage.setScene(new Scene(root, w, h));
primaryStage.show();
```

Le package *awt* parent de Swing [35] est régulièrement en concurrence avec JavaFX [24] pour le nom des widgets [38]. On conseille également de faire attention à placer les widgets [38] relativement aux dimensions du noeud parent, et de proscrire les tailles fixes.

### C.2.3 - Gestion des évènements

On qualifie la gestion d'une IHM [17] de programmation évènementielle dans la mesure où un *Event Thread* tourne en tâche de fond : c'est un processus de surveillance prêt à capter les *inputs* du réseau ou de l'utilisateur. Les gestionnaires d'évènements sont des procédures qui se lancent quand un certain type d'interaction a été détecté avec un widget [38] en particulier. [3]

Le découpage modèle/vue/contrôleur est traditionnel pour la programmation d'un GUI [14]. Le modèle désigne la couche abstraite de l'application qui s'occupe des calculs et de la gestion de données. La vue, c'est tout simplement la partie chargée du rendu de l'interface utilisateur, avec ses noeuds et ses widgets [38]. On peut enfin concevoir le contrôleur comme le centre de réaction aux évènements.

#### Gestionnaire anonyme

On n'a pas besoin de contrôleur détaché parce qu'on incorpore la réponse aux évènements dans le corps principal de l'application. C'est cette option que j'ai retenue parce qu'elle embrasse synthétiquement et localement les relations de cause à effet. C'est aussi dans la continuité de la précédente programmation du GUI [14].

```
bouton.setOnAction(new EventHandler⟨ActionEvent⟩() {
—@ Override
—public void handle(ActionEvent event) {
— —nombreDeClics += 1;
—}
}
```

Ou de manière encore plus condensée :
`bouton.setOnAction(e → nombreDeClics++);`

#### Gestion par interface

Toute classe qui implémente l'interface *EventHandler* fait office de contrôleur. La méthode obligatoire pour l'implémentation de l'interface, est complétée par nos opérations de réaction. Cela remplace la méthode anonyme mais le gestionnaire est alors une instance du contrôleur qu'on enregistre sur le widget [38] avec *.addEventHandler()*.

## C.3 - La question du fxml

### C.3.1 - Intérêt

C'est un langage certes balisé mais surtout arborescent, ce qui semble particulièrement adapté à la gestion hiérarchique des noeuds graphiques. Quand on choisit de coder la vue sur le fichier *fxml*, une seule classe de contrôle suffit pour délocaliser tous les gestionnaires puisque ce ne sont plus que des méthodes référencées. [4] Le *fxml* permet encore de se départir du moment d'ajout des widgets [38] (la construction graphique est directement consécutive au chargement), donc de se libérer des problèmes de duplication.

| Classe *Controller* | Fichier *fxml* |
|---|---|
| | fx:controller="Controller" |
| @FXML TypeObjet nomObjet; | fx:id="nomObjet" |
| @FXML void gestionnaire( ); | onAction="#gestionnaire" |

### C.3.2 - Décision

Je ne souhaitais pas déroger à la compartimentation du GUI [14] pour que l'architecture demeure en tous points fidèle à l'affichage et que la maintenance reste très abordable. La gestion anonyme des évènements avait d'ailleurs fait échouer la stricte séparation modèle/vue/contrôleur. Pour finir, le format ne se prêtait que difficilement à la génération procédurale de widgets [38], qui sont en plus en nombre variable dans certains tableaux.

# D - Quelques billets représentatifs

Cette annexe présente un aspect de mon travail au quotidien, à savoir la résolution des bugs imputables à la migration. Je les ai rangés par similitudes de manifestations.

## D.1 - Problèmes de duplication et de persistance

DESCRIPTION. Doublons dans le choix des années de simulation. ORIGINE. Un mauvais emploi de *comboBox.getItems().removeAll()* à chaque mise à jour des années susceptibles d'être simulées. SOLUTION. La méthode *removeAll()* ne retire que la liste des items qu'on lui indique. Lui préférer *comboBox.getItems().clear()* ou *comboBox.getItems().removeAll(comboBox.getItems())*.

DESCRIPTION. Réplication de colonnes dans le tableau des performances à chaque bascule du type de troupeau. Se manifeste aussi quand on modifie les dispositions de vente des yearlings [39]. ORIGINE. La méthode *afficheHerd()* est impliquée dans les deux cas de figure. La duplication venait de la mise à jour de la table *Herd* par ajout de colonnes. SOLUTION. Vider la table de ses colonnes au préalable : *table.getColumns().clear()*.

DESCRIPTION. Surimpression d'icônes dans l'arbre de rendu des pratiques sur parcelles quand on replie un item. ORIGINE. On n'examinait pas le cas hors sélection dans le préparateur de rendu. SOLUTION. Disjonction sur l'ensemble des types de sélection.

DESCRIPTION. Les tableaux ne sont pas vierges à l'ajout d'un tout nouveau troupeau. ORIGINE. Exploration partielle des stades de maturité possibles du tableau. SOLUTION. Distinction de mise à jour selon que le troupeau existe déjà ou non.

DESCRIPTION. Délai de mise à jour de l'arbre de rendu. ORIGINE. Le rafraichissement graphique n'attend pas qu'on ait interagi avec la fenêtre de configuration des pratiques. Aussi les premières modifications ne sont visibles qu'à la deuxième tentative. SOLUTION. Suspendre le code aval jusqu'à la fermeture de la fenêtre modale [27].

## D.2 - Inconséquence de l'édition

DESCRIPTION. Défaut d'enregistrement des modifications portées aux tableaux. ORIGINE. Absence de communication entre l'objet de type *TableView* et son modèle de ligne associé. SOLUTION. Transporter la valeur des cellules dans le modèle de donnée par gestion d'édition via des setters [32].

DESCRIPTION. Les tentatives d'édition de la description des pratiques ne sont pas prises en compte. ORIGINE. Ces évènements de sélection n'appartiennent pas à l'énumération des pratiques admissibles. SOLUTION. Remonter la perception de sélection au noeud parent.

DESCRIPTION. Défaut de suppression des prairies dans l'arbre de gestion des pratiques sur parcelles. ORIGINE. On essayait vainement de retirer la prairie d'une liste de cultures. SOLUTION. S'assurer avant suppression que l'occurrence existe.

DESCRIPTION. Sauvegarde limitée à la première couche du sol. ORIGINE. Défaut de restitution des données enregistrées. SOLUTION. Chargement complet de la liste dynamique d'attributs qui représente les différentes couches d'un sol.

## D.3 - Vices de chargement ou de rafraichissement

DESCRIPTION. Certains attributs de la ferme sont utilisés sans avoir été initialisés. ORIGINE. *MainInterface* ne faisait qu'incanter le GUI [14] au premier chargement. SOLUTION. Rapatriement de la procédure complète d'initialisation.

DESCRIPTION. Le chargement d'une ferme écrase le formulaire de création de parcelle au lieu de la *ListView*. ORIGINE. Forçage partiel des données chargées. SOLUTION. Rafraichissement complet du GUI [14].

DESCRIPTION. Internationalisation partielle. ORIGINE. Défaut de mise à jour de la localité. SOLUTION. Rechargement complet du GUI [14].

DESCRIPTION. Absence de cycles sur les parcelles hors chargement. ORIGINE. La bibliothèque des cycles reste vide si la ferme ne possède pas de parcelles au démarrage de FarmSim [10]. SOLUTION. Constitution de la bibliothèque postérieurement à la création d'une parcelle.

# E - Petit laïus de conversion

Cette annexe inventorie les correspondances les plus communes entre les deux bibliothèques. J'ai parfois omis le transtypage : cela dépend aussi de ce que vous voudrez en faire.

JLabel ⟩ **Label**

JButton ⟩ **Button**

JRadioButton ⟩ **RadioButton**

JTextField ⟩ **TextField**

JTextArea ⟩ **TextArea**

JListView ⟩ **ListView**

JTable ⟩ **TableView**

JTree ⟩ **TreeView**

DefaultMutableTreeNode ⟩ **TreeItem**

getContentPane().add(Button button) ⟩
— **contentPane.getChildren().add(Button button)**

button.addActionListener($e \rightarrow \{\}$); ⟩
—**button.setOnAction(**$e \rightarrow \{\}$**);**

button.setEnabled(false); ⟩
—**button.setDisable(true);**

comboBox.addItem(String item); ⟩
—**comboBox.getItems().add(String item);**

comboBox.removeAllItems(); ⟩
—**comboBox.getItems().clear();**

comboBox.getSelectedItem(); ⟩
—**comboBox.getSelectionModel().getSelectedItem();**

comboBox.setSelectedIndex(int i); ⟩
—**comboBox.getSelectionModel().select(int i);**

table.setValueAt(String val, int $row_i$, int $col_j$); ⟩
—**table.getItems().get(i).setAttribut**$_j$**(String val);**

tree.getLastSelectedPathComponent(); ⟩
—**tree.getSelectionModel().getSelectedItem();**

tree.getModel().reload(); ⟩
—**tree.refresh();**

# F - Note d'introduction à l'interface de *FarmSim*

Ce document s'adresse aux futurs contributeurs de FarmSim [10]. Il reproduit la décomposition hiérarchique du GUI [14] par souci de familiarisation avec l'arborescence de l'appliaction. Il dresse également les enjeux prégnants sur les attendus du simulateur et sur son fonctionnement. J'ai souligné les points qui méritent à mon sens d'être encore améliorés.

## F.1 - Analyse d'*Interface Overview*

C'est l'onglet de supervision générale des simulations, le premier du classeur de *Main Interface*. Il se décompose en quatre sections. Je dois mentionner que j'ai retiré l'instance de l'onglet *Meteo* des paramètres du constructeur faute d'usages ultérieurs.

### F.1.1 – Période de simulation

On peut choisir de faire une simulation exhaustive ou bien mono-annuelle. Le lancement des simulations entraine une validation des paramètres en début de procédure et l'affichage de pop-ups de contrôle en fin de simulation. J'ai essayé de rendre ces boîtes de dialogue modales par rapport à leurs fenêtres parentes, c'est-à-dire d'empêcher d'avoir la main sur la fenêtre parente tant qu'on n'a pas interagi avec la boîte de dialogue. Mais comme l'application peut être multi-fenêtrée avec le chargement de nouvelles fermes, il n'est pas évident que la modalité soit bien entretenue entre les bons couples de fenêtres.

### F.1.2 – Paramétrage annuel moyen

Le contenant *GridPane* convient parfaitement pour l'alignement du formulaire sur plusieurs rangs. La rubrique se destine à fixer les bornes de la simulation et le conditionnement général de la ferme. Le chargement d'une ferme force le remplissage des champs de texte avec les données recueillies dans le fichier *xml*.

F.1.3 – Modèles élémentaires spécifiques

Cette section propose de choisir un modèle de simulation (parmi ceux disponibles) pour chaque module autonome de FarmSim [10]. L'un d'eux se charge de la modélisation des cultures, un autre s'occupe des prairies non cultivées, un autre encore, des transferts d'azote et de carbone avec les bâtiments. Le dernier s'intéresse aux intrants de la ferme, autrement dit ses consommables pour les infrastructures et la fertilisation.

F.1.4 – Géolocalisation de la ferme

On qualifie la ferme par sa latitude. La précision est portée jusqu'à la seconde d'angle. Les libellés ne sont pas codés en dur mais délocalisés pour permettre l'internationalisation de FarmSim [10]. *FarmWord.getInstance().getString("key")* permet de pointer sur la bonne traduction dans la balise éponyme de *lang.FarmSimLanguage Properties_en_EN.xml* ou de *lang.FarmSimLanguageProperties_fr_FR.xml*.

Classes mobilisées

gui.overview.InterfaceOverview
gui.overview.*
gui.SuperPane

## F.2 - Analyse d'*Interface Meteo*

C'est l'onglet de chargement des données météorologiques de la ferme pendant ses années de simulation. Ce n'est que de la visualisation, on peut superviser librement l'affectation des fichiers météo mais pas les éditer.

F.2.1 – Sélection du fichier météo

Il y a d'abord quelques encarts pour préciser les conditions atmosphériques, puis un explorateur de fichiers pour charger les données météo. Une bascule permet de commuter entre les formats horaires et journaliers, qui modifient les variables observées. Une piste d'amélioration serait de détecter automatiquement le format du fichier météo soumis à l'affichage. On peut enfin reporter les données météo d'une année sur une ou plusieurs autres années.

F.2.2 – Données caractéristiques remarquables

Deux tableaux présentent les résultats de l'analyse statistique des données brutes. La procédure impliquée ne fait qu'extraire les extrêmes, sommer ou moyenner les valeurs enregistrées. Je dois signaler que l'implémentation des *TableView* se rapproche beaucoup du paradigme des bases de données. Chaque ligne du tableau est un enregistrement qui respecte un schéma particulier : c'est un objet dont chaque attribut coïncide avec une colonne du tableau. Cela suppose de maintenir un modèle de données associé (une abstraction de la forme attendue pour chaque ligne), qui permet d'accéder à une cellule donnée par les attributs plutôt que par les coordonnées. Je mentionne au passage que certains caractères des vieux fichiers ne sont plus supportés.

F.2.3 – Toutes les données

C'est le tableau météo le plus complet, étendu aux données hygrométriques [15] et anémométriques [1] exhaustives. Il n'est pas ouvert à l'édition mais il s'adapte au format de fichier déclaré. À titre purement ergonomique, il serait bon de forcer le saut de ligne des en-têtes de colonne trop conséquents pour être affichés entièrement.

Classes mobilisées

gui.InterfaceMeteo
gui.SuperPane

## F.3 - Analyse d'*Interface Plot* et d'*Interface Soil*

Le troisième onglet du classeur de *Main Interface* est destiné à la gestion des parcelles.

F.3.1 – Liste des parcelles

La *ListView* recense les parcelles de la ferme. J'ai verrouillé les modalités d'édition pour que les modifications soient suspendues quand il n'y a pas de parcelle selectionnée. Il semblerait que le chargement d'une ferme n'écrase pas correctement les parcelles de la *ListView*.

### F.3.2 – Edition de parcelles

Le panneau latéral abrite tantôt le menu d'édition, tantôt le formulaire de renseignement. J'ai mis fin à leur coexistence parce qu'il n'y avait jamais besoin de les voir ensemble quelque soit le cas d'utilisation. Les informations recueillies participent à la caractérisation agraire de la parcelle. Il y a superposition volontaire du bouton de fin de création avec celui de prise en compte des modifications. J'ai supprimé la persistance des champs pour qu'à chaque ajout de parcelle le formulaire soit vierge.

### F.3.3 – Configuration des sols

C'est une fenêtre de configuration avancée. Il s'agit de remplacer les standards des modèles de FarmSim [10] par des caractéristiques superficielles et souterraines spécifiques. J'ai décidé que l'inscription définitive du nouveau sol dans la bibliothèque des sols était un signal suffisant pour fermer la fenêtre (étant entendu que sa mission est alors terminée).

### Charger un sol

La bibliothèque rémanente permet de s'inspirer d'un type de sol reconnu. On peut enrichir cette collection dans la mémoire courante de l'application. On peut encore faire en sorte que les contributions à la bibliothèque des sols deviennent permanentes.

### Généralités

Ce formulaire restitue les informations indépendantes de la profondeur. Une amélioration, quoique mineure, serait d'uniformiser la précision des décimaux à l'affichage.

### Configuration personnalisée

Cinq données fondamentales constituent le quorum de renseignements nécéssaires pour prétendre à la personnalisation du sol, mais le tableau peut être étendu à une vingtaine de propriétés secondaires. La sauvegarde des modifications passe par le dialogue entre la *TableView* et son modèle associé (par des getters [11] et des setters [32]). J'ai d'ailleurs formalisé un modèle de ligne plus général (avec une liste dynamique d'attributs) pour gérer les variations du nombre de colonnes.

Classes mobilisées

gui.InterfacePlot
gui.soil.InterfaceSoil
gui.soil.*
gui.SuperPane

## F.4 - Analyse d'*Interface Herd Management*

Il reste que le changement de langue écrase anormalement les modifications du troupeau.

F.4.1 – Interface Herd

Type de troupeau

Nombre de têtes

Unités de gros bétail

Tous les animaux d'un élevage ne sont pas équivalents, ne serait-ce que par leur niveau de consommation. L'unité de gros bétail propose donc une pondération des besoins de chaque animal dans le troupeau. [10] C'est de ce point de vue que les bœufs ont plus d'importance que les veaux par exemple. Les unités de gros bétail servent aussi de référence aux agriculteurs pour caractériser la taille de leur bétail, elles remplacent parfois le nombre d'hectares pour proportionner les subventions.

Performance du troupeau

F.4.2 – Interface Housing

F.4.3 – Interface Feed

Concentrés pâturés

Fourrage pâturé

Concentrés pour l'étable

Fourrage pour l'étable

Classes mobilisées

gui.InterfaceYearHerdManagement
gui.InterfaceHerd
gui.InterfaceFeed
gui.InterfaceHousing
gui.SuperPane

## F.5 - Analyse d'*Interface Plot Management*

La gestion des évènements sur les parcelles est traitée dans l'absolu dans le cinquième onglet du classeur de *Main Interface*. Mais une classe cousine permet de mémoriser un enchaînement d'évènements pour les appliquer de manière monolithique et répétitive. L'interface des cycles introduit une datation algébrique à partir de la première année du cycle, ce qui rend tout de suite le bloc d'évènements beaucoup plus reproductible. Comme pour les sols, il existe une bibliothèque permanente de cycles.

F.5.1 – Sélection de parcelles

F.5.2 – Liste des évènements

F.5.3 – Evènements

On distingue les cultures, qui vont mobiliser Ceres [6], des prairies modélisées par PaSim [29].

F.5.4 – Pratiques

Ce sont des évènements plus intrinsèques, presque tous ponctuels comme le semis ou l'arrosage. J'ai rendu possible la modification croisée des semailles et des labours.

Classes mobilisées

gui.InterfacePlotSelection
gui.InterfaceManagement
gui.InterfaceListEvents
gui.cycle.InterfaceCycle
gui.cycle.*

gui.config.CroplandConfigurationDialog
gui.config.GrasslandConfigurationDialog
gui.config.DateEventConfigurationDialog
gui.config.GrazingConfigurationDialog
gui.config.IrrigationConfigurationDialog
gui.config.*
gui.RenduTreeCell
gui.SuperPane

## F.6 - Analyse d'*Interface Intrans*

Cet onglet s'intéresse à la conversion du niveau de consommation de la ferme en kg de CO2 eq. L'émission de gaz à effet de serres, qu'elle soit directe ou indirecte, se répartit essentiellement entre cinq postes de consommation. La colonne consacrée au volume de consommation est la seule qui puisse être éditée. L'onglet dispose aussi fort heureusement de la fonctionnalité de report des saisies d'une année sur l'autre. Je ne vois rien de nature à être changé, hormis peut-être la disposition des tableaux.

F.6.1 – Consommation de carburants

F.6.2 – Consommation de gaz

F.6.3 – Consommation d'électricité

F.6.4 – Semences

F.6.5 – Consommation de pesticides

Classes mobilisées

gui.InterfaceIntrans
gui.SuperPane

## F.7 - Analyse d'*Interface PaSim Parameters*

C'est un onglet facultatif dans la mesure où le passage en mode normal le retire du classeur de *Main Interface*.

F.7.1 – Paramètres spécifiques au modèle de prairie

Le widget de type *TextArea* permet de modifier le paramétrage avancé du modèle PaSim [29].

F.7.2 – Prise en compte des paramètres

Je me demande si cette rubrique a toujours lieu d'être, alors que d'autres onglets sont tacitement validés par changement d'onglet.

F.7.3 – Teneur en carbone du sol

Cette section porte sur le mode de détermination de la condition initiale de teneur en carbone, indispensable pour le bon déroulement des simulations du modèle de prairie. Cette question se pose notamment en cas d'alternance de Ceres [6] (le modèle de culture) et de PaSim [29] (le modèle de prairie). La mise à l'équilibre consiste à faire tourner le modèle Ceres [6] pendant X années pour obtenir une valeur de départ plausible. L'autre solution consiste à piocher cette valeur dans un fichier restart.

F.7.4 – Mise à l'équilibre du modèle de prairie

La méthode de commutation par échange des statuts de visibilité est peut-être trop sophistiquée pour la tâche demandée, à savoir basculer entre deux affichages au gré du choix de mise à l'équilibre. On peut très bien se référer à la méthode employée dans l'onglet de gestion des parcelles.

Classes mobilisées

gui.InterfacePaSimParameters
gui.SuperPane

## F.8 - Analyse d'*Interface Resultats*

Je vais illustrer l'intérêt de cet onglet à travers un cas d'utilisation. Mettons que je doive présenter les résultats d'une simulation fastidieuse ou très gourmande à des partenaires. L'onglet des résultats permettra de charger les graphiques tirés d'une simulation terminée, à condition bien sûr de l'avoir sauvegardée. Mais je dois dire qu'il n'a pas été complètement transféré.

## F.9 - *IPCC* [20]

C'est un onglet secondaire qui rassemble toutes les constantes et les variables du modèle de bâtiment. Je dois signaler que j'ai rétabli la sauvegarde des modifications aux endroits susceptibles d'être édités.

### F.9.1 – Énergie brute requise

L'énergie brute est une mesure calorique des quantités de fourrage ou de concentrés consommées par le troupeau. C'est l'énergie qu'on peut récupérer quand on les brûle pour simuler l'oxydation biologique des nutriments. [5]

### F.9.2 – Cycle du carbone

Le carbone est présent dans la ferme sous forme de dioxyde de carbone et de méthane.

### F.9.3 – Cycle de l'azote

Il est animé par les apports d'engrais nitrés et la libération des effluents [9].