Bastien Marie

Leroy Guillaume

14 mai 2018

Projet Informatique

Simulation d'une fourmilière

Document de synthèse

TABLE DES MATIERES

BILAN DU PROJET

ORGANISATION DU TRAVAIL

Au cours de la phase d'analyse du 3 Avril au 9 Avril, nous avons travaillé ensemble afin de poser les hypothèses de base du projet et d'être directement d'accord sur les hypothèses prises. La réalisation des diagrammes UML s'est également déroulée en commun.

Les hypothèses prises ou modifiées après cette phase d'analyse ont été adoptées lorsque les deux membres du binôme étaient d'accord.

Lors du développement, nous avons choisi de travailler en alternance, en nous transmettant le code commenté en zip par mail. Nous avons choisi de ne pas utiliser GitHub car nous ne savions pas comment nous en servir. Lors de cette phase, nous avons fait des réunions fréquentes pour faire des mises au point sur nos avancements respectifs et faire les choix des priorités de développement. Nous avons donc choisi de renoncer à l'enregistrement des jeux dans une base de données au profit de la réalisation d'une interface graphique plus poussée. Nous avons commencé l'interface graphique ensemble et Guillaume l'a finie.

Marie a rédigé le rapport d'analyse alors que Guillaume continuait de programmer les méthodes des classes. Guillaume a relu le rapport. Le document de synthèse a été réalisé par Marie et a été relu par Guillaume.

BILAN DES NOTIONS ACQUISES ET APPROFONDIES

Le projet nous a permis de gagner de l'aisance en Java. Lorsque nous avons essayé de supprimer des animaux, nous avons compris la distinction entre un objet et les références de celui-ci. Nous maîtrisons mieux la gestion des objets en Java. Le projet nous a conduit à créer une vraie architecture de classes interdépendantes et donc à gérer les liens entre les classes (héritage, appel des méthodes d'une autre classe). Ensuite, notre projet a nécessité des classes abstraites, des énumérations, des interfaces, des héritages et des implémentations, cela nous a donc permis de mieux distinguer les

différentes natures de ces outils et de mieux maîtriser la syntaxe associée à chacun. Nous avons aussi découvert de nouvelles fonctionnalités de Java : les affichages avec Swing.

En matière d'analyse, nous avons dû créer des diagrammes UML, cela nous a donc permis de nous remémorer le cours d'analyse informatique vu plus tôt dans l'année et de l'utiliser dans un cas plus concret pour nous. Lors de la réalisation des diagrammes d'activité et de séquence, nous avions représenté les diagrammes de séquence sous la forme de diagramme de séquence, nous comprenons maintenant mieux la différence entre ces deux types de diagrammes. Nous avons appris à nous servir de StarUML pour réaliser des diagrammes UML plus efficacement qu'à la main (gestion des suppressions et des ajouts plus faciles sur StarUML).

Enfin, nous avons acquis des compétences en termes de gestion de projet. Ce projet fut notre premier projet informatique long sur la durée (1 mois et demi) et assez conséquent en termes de code (plus de 15 classes à gérer). Nous nous sommes familiarisés avec le travail de groupe : expliquer son point de vue, comprendre mieux les choix de l'autre, revenir sur les choix pris précédemment. Les échanges de code par mail nous a contraint à gérer les versions des codes que nous possédions. En termes de gestion du temps, le GANTT nous a été utile pour juger de l'état d'avancement de la programmation et pour nous organiser dans notre travail, surtout au niveau de la répartition des tâches.

FONCTIONALITES REALISEES

Pour les animaux, nous avons réalisé les fonctionnalités suivantes :

- Déplacement des animaux suivant les hypothèses
- Pose des phéromones par les fourmis
- Détection des phéromones par les fourmis
- Gestion des combats entre les ennemis et les combattantes
- Détection des fourmis par les ennemis
- Gestion de la mort des animaux
- Alimentation des animaux (graines ou fourmis)
- Transport de la nourriture

Pour le terrain, nous avons réalisé les fonctionnalités suivantes :

- Mise en place de la grille qui porte toutes les informations des animaux et des cases
- Génération des obstacles et des cases nourriture aléatoirement
- Gestion des tours
- Possibilité de choix de paramètres par l'utilisateur (via la console)

Pour la visualisation, nous avons réalisé les fonctionnalités suivantes :

- Affichage de la grille à l'aide d'icône (vous trouverez une description des icônes dans le *Readme*).
- Mise à jour de l'affichage à chaque déplacement.

FONCTIONNALITES NON REALISEES

Nous n'avons pas eu le temps de gérer tout ce que nous aurions aimé faire. A savoir :

- Création des larves, couveuses et reine.
- Possibilité pour une fourmi non combattante de fuir si un ennemi se rapproche d'elle.
- Enregistrement dans une base de données (nous n'avons donc pas appris à utiliser SQLite).
- Possibilité pour l'utilisateur de placer lui-même la nourriture, les obstacles et les ennemis.

ORGANISATION DES DEVELOPPEMENTS

ARCHITECTURE GENERALE DU CODE

Les trois classes centrales pour l'exécution du code sont Grille, Jeu et InterfaceGraphique. La première porte l'ensemble des informations concernant la simulation à un instant donné. En effet, la grille a en attributs l'ensemble des objets de type animal ainsi que les cases, qu'elle peut modifier par l'intermédiaire de ses méthodes. La classe Jeu permet la succession des tours et la mise à jour de la grille en appelant les méthodes de cette dernière. Enfin, la classe InterfaceGraphique permet la visualisation de la simulation.

HYPOTHESES MODIFIEES PAR RAPPORT A L'ANALYSE INITALE

Les fourmis ne se déplacent plus aléatoirement si elles n'ont pas d'objectifs, elles s'éloignent le plus possible de la fourmilière (afin d'explorer au mieux le terrain). De même les ennemis détectent les fourmis non combattantes qui se situent autours d'eux et les prennent en chasse.

Par manque de temps, nous n'avons pas pu faire arrêter la simulation si toutes les fourmis étaient mortes. La simulation s'arrête uniquement si le compteur de tours atteint le maximum.

MODIFICATIONS DU DIAGRAMME DE CLASSE PAR RAPPORT A L'ANALYSE INITALE

Nous avons réalisé des modifications du code (qui se sont donc répercutées sur le diagramme de classes), nous présentons ci-dessous les différents changements majeurs. Les simples renommages de méthodes ou les ajouts de méthodes simples ne figurent pas dans la liste ci-dessous. Vous trouverez le diagramme de classes mis à jour en annexe, qui comporte uniquement les classes que nous avons eu le temps d'instancier (pas de classes BaseDeDonnees, Larve, Couveuse, Reine ni d'interface iEntreeSortie)

- Suppression de la classe fourmilière : nous avons privilégié une création des fourmis et ennemis dans la classe Grille. La classe fourmilière avait un nom assez mal choisi : c'était une case et aussi un lieu de création de fourmis. De plus les ennemis auraient quand même dû être générées dans une autre classe que Fourmilière. Donc nous avons préféré mettre toutes les méthodes de création d'animaux dans la classe Grille.
- N'ayant pas eu le temps de réaliser l'interface utilisateur (i.e. des menus contextuels), nous avons supprimé la classe InterfaceUtilisateur.
- L'ajout de la classe JPanelGrille a été nécessaire afin de créer le contenu utilisé par l'interfaceGraphique. Cette classe s'occupe de la création des icônes.
- Nous avons retiré l'attribut identifiantAnimal de la classe animal car il ne nous était utile que si nous enregistrions le jeu dans une base de données (ce que nous n'avons pas eu le temps de faire).
- Nous avons simplifié les attaques des animaux : tous les animaux ont une distance d'attaque d'une case.
- Tous les animaux n'ayant pas les mêmes priorités d'action, chaque type d'animal dispose d'une méthode jouerTour() qui lui est spécifique.
- Les fourmis ne se comportent également pas de la même façon sur le terrain et sur la fourmilière, nous avons donc muni les fourmis des méthodes actionSurFourmiliere() et actionHorsFourmiliere(). Par exemple : les transporteuses posent de la nourriture dans la fourmilière et peuvent se nourrir sur cette case, alors que les autres fourmis se nourrissent juste dans la fourmilière.
- Lorsqu'elles meurent, les fourmis mortes deviennent de la nourriture pour l'ennemi uniquement pendant un tour, après elle disparaissent. Les ennemis deviennent de la nourriture qui reste de manière pérenne (jusqu'à ce que des fourmis viennent prélever la nourriture de la case).
- Il n'y a plus d'attribut compteur dans la classe Jeu, le nombre de tours apparaît dans la boucle de l'affichage du jeu.

CONCLUSION

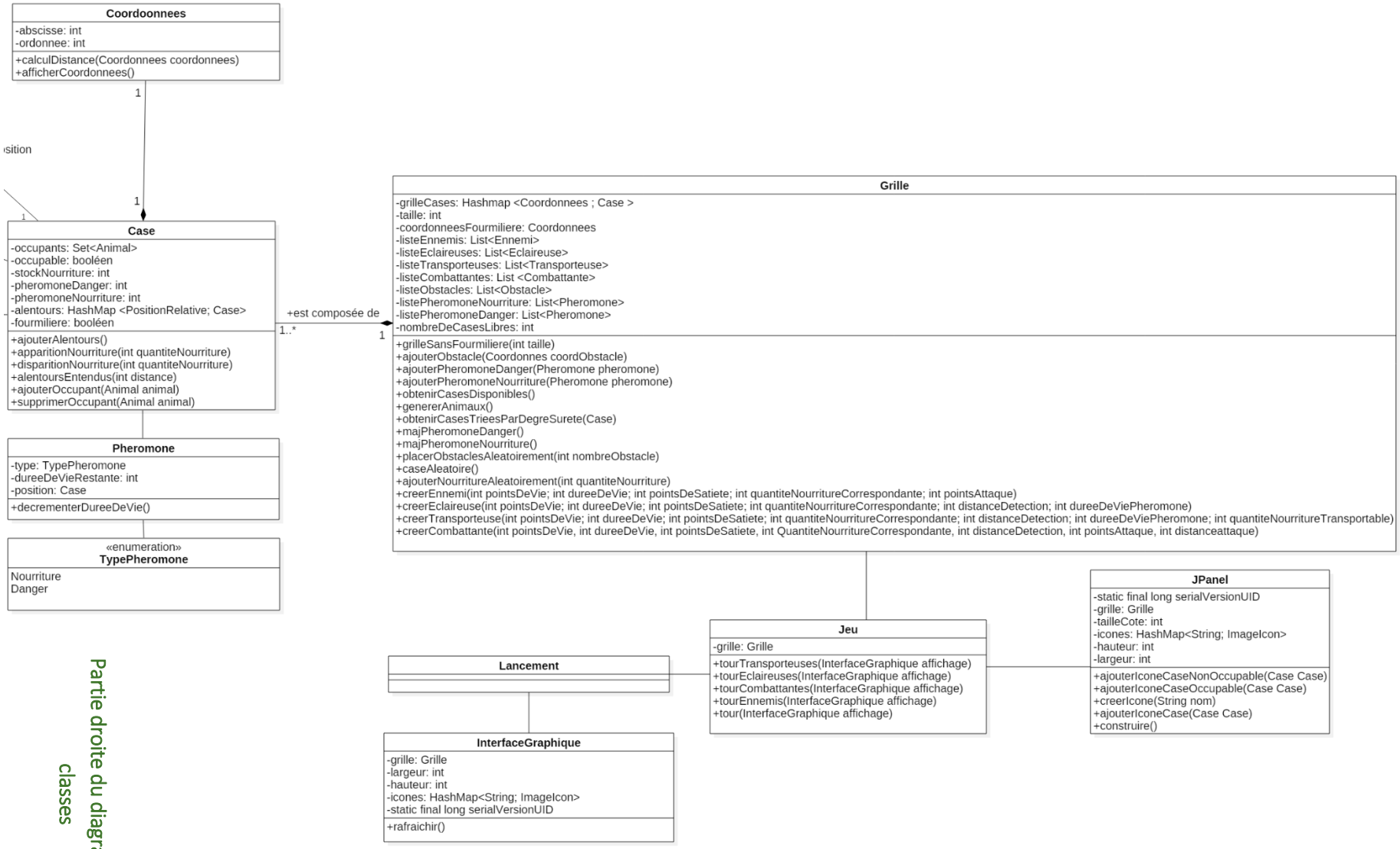
Outre le fait de nous avoir permis de revoir l'ensemble des notions informatiques étudiées cette année, ce projet nous a offert la possibilité de gagner de l'aisance en développement et en gestion de projet. La réalisation de la simulation de la fourmilière nous a peu à peu amené à envisager le développement de fonctionnalités plus poussées que celles exigées par le sujet. Cela nous permettra par la suite de modifier le code du projet en faisant appel à d'autres outils, que nous n'avons pas eu le temps d'utiliser (boutons, base de données avec SQLite).

SOURCES DES IMAGES UTILISEES

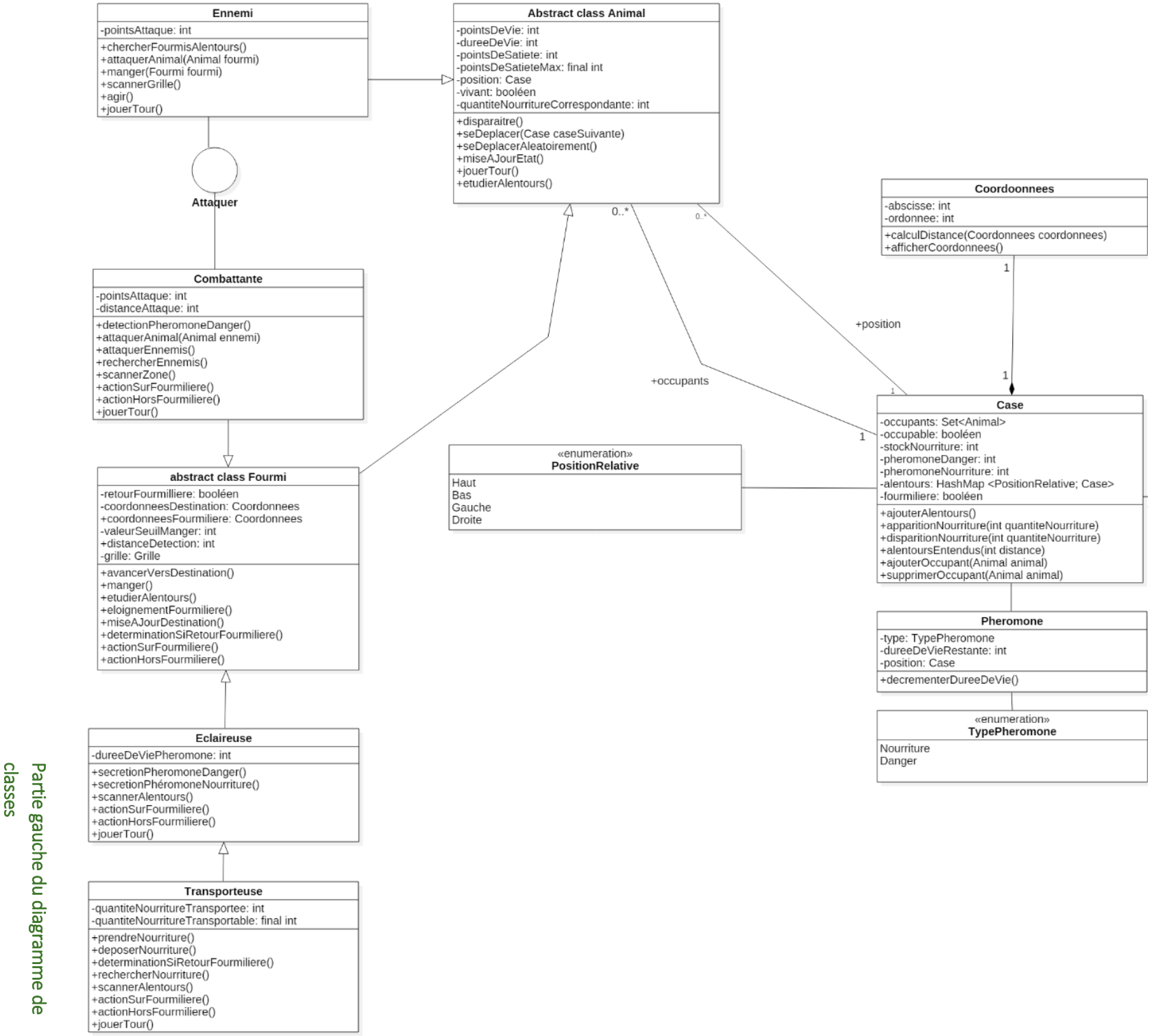
Graine : https://pixabay.com
La pierre : http://www.clker.com
Herbe : https://aspen-landscaping.com
Sac à dos : http://pluspng.com
L'araignée : http://www.clker.com
La fourmi : http://www.clker.com

ANNEXE

Vous trouverez ci-dessous le diagramme de classes final de notre projet. Par souci de lisibilité, nous avons coupé le diagramme en deux parties.



Partie droite du diagramme de classes



Partie gauche du diagramme de classes