Rapport de projet

LP73 Fourmilière

El Guendouz Mohamed

Peurey Loann

18 juin 2020

Sommaire

Introduction

Ce projet s'inscrit dans le cadre du module LP73 du cycle d'ingénieur en informatique de l'UTBM.

Besoins et objectifs du projet

L'objectif est de développer un programme en C++ pour simuler la vie et l'évolution de fourmilières dans un environnement. Il doit se faire selon les principes de la programmation orientée objet et réalisé par groupe de 2 étudiants.

Approche retenue

Notre vision dans le cadre de ce projet à été de réaliser une simulation de fourmi basé sur un orchestrateur que l'on a nommé **Moteur.**

Il permet de gérer l'ensemble des interactions et déplacements dans le simulateur.

Il communique avec l'affichage en lui transmettant les nouvelles informations à chaque cycle(partie).

Ce choix à été réalisé afin de mieux gérer les objets créés.

En effet, Toutes les interactions avec les entités sont gérées **par référence** dans le projet ce qui permet de mieux contrôler la mémoire. Nous avons fait le choix de définir 3 types d'attributs pour une fourmi qui sont son état, son statut et son exploration. Nous avons défini différentes règles permettant de gérer les exceptions dans le déroulement du simulateur et ajouté des paramètres pour personnaliser son utilisation.

Spécifications

Dans cette partie nous allons essayé d'exprimer les spécifications liés au projet par rapport à l'architecture choisi.

Diagramme de classes

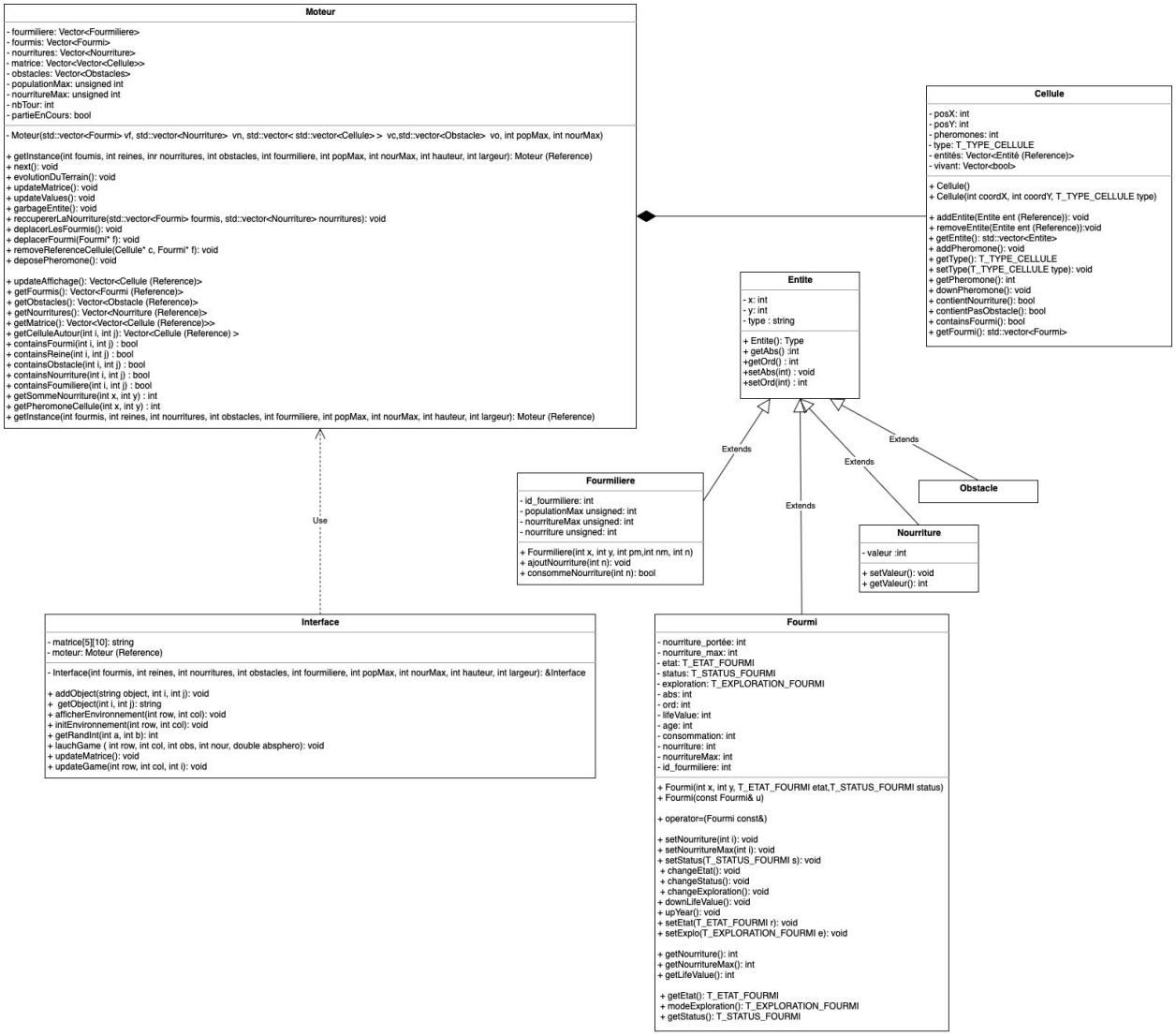
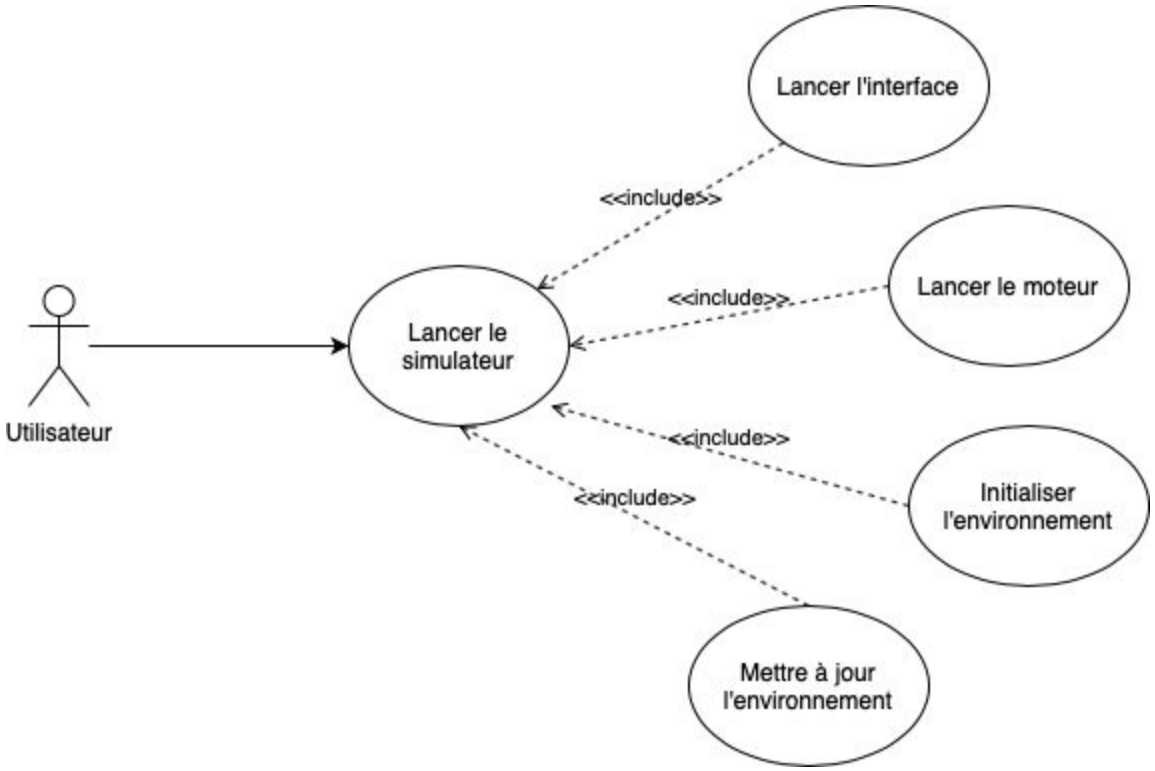


Diagramme de cas d’utilisation



Entité

toutes les classes faisant partie intégrante des éléments de jeu héritent d'entités qui est la classe des objets définissables et représentables dans le jeu

Fourmilière

La fourmilière est une entité pouvant se placer sur une case mais pouvant elle aussi contenir des Entités (toujours par référence)

Fourmi

La fourmi est une entité ayant différents types et comportement, son évolution (déplacement, ramassage) est contrôlé par le moteur.

la croissance des fourmis est géré par deux typedef enum qui permettent en se croisant de définir toutes les possibilités pour une fourmi, leur évolution est automatique et géré à chaque tour sur des contraintes d'âge et de nourriture

Nourriture

La nourriture est une entité ayant une valeur de nourriture pouvant être ramassée, lorsque cette quantité atteint 0, elle disparaît.

Obstacle

Les obstacles sont simplement des entités bloquant le passage sur une cellule.

Moteur

Organisation autour d'une classe principale de contrôle du programme (Moteur) cette classe gère toutes les actions intelligentes et de progression de l'environnement du jeu Elle contient les entités présentes dans le jeu comme Fourmis, fourmilière, cellule etc. Le moteur comporte les fonctions d'avancement de tour comprenant en autre le calcul des déplacements de fourmi, la dépose de phéromone ou encore le ramassage de nourriture.

Cellule

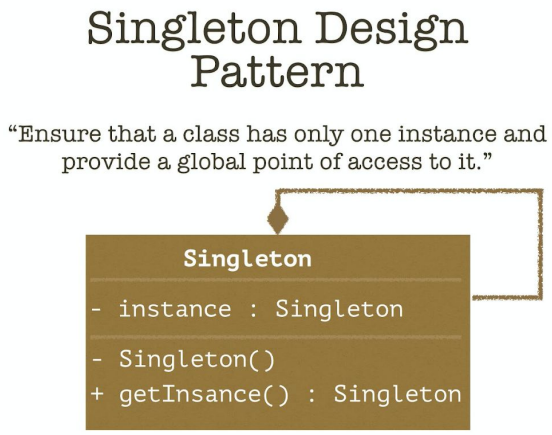
Les cellules présentes dans moteur contiennent les attributs qui les définissent et un vecteur de références des entités présentes sur la cellule, on utilise les références car les entités sont déjà présentes dans le moteur et cela permet de gérer leurs déplacements a travers les cellules plus simplement, en déplaçant les références sans recopier l'objet à chaque déplacement

Interface

une classe interface a été créée pour regrouper toutes les fonction liées au moteur permettant un affichage sur un terminal. Nous avons défini aussi défini cette interface comme étant un singleton. Dans l'énoncé, il était de demandé de réaliser un environnement 100x50 mais dû à un soucis ergonomique, nous avons divisé sa taille par 10.

Design Pattern

Un patron de conception (design pattern)est est une solution générique d'implémentation répondant à un problème spécifique. En effet, pour garder l'unicité de nos objets Moteur et Interface, nous les avons défini en suivant le patron de conception appelé **Singleton** dont l'objectif est de restreindre l'instanciation d'une classe à un seul objet.



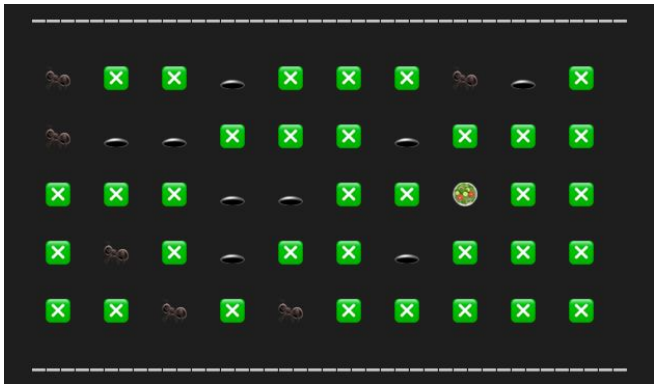
Comment utiliser le logiciel ?

Pour lancer l'interface, il faut :

1. Accéder au répertoire **src/**
2. Lancer **./interface** pour visualiser les paramètres à entrer

```
----------------------------------------
 Menu      FOURMILIERE      PROJET LP73
----------------------------------------
1er parametre : nombre de partie
2eme parametre : nombre de fourmis
3eme parametre : nombre de reine
4eme parametre : nombre de nourriture
5eme parametre : nombre de populationMax
6eme parametre : nombre de nourritureMax
7eme parametre : nombre de obstacle
8eme parametre : nombre de fourmiliere
----------------------------------------
```

3. Vous pouvez lancer l'interface avec cette ligne : **./interface 50 10 1 3 25 35 10 1**



4. Durant l'affichage, un historique à été créer permettant de visualiser différentes informations:

```
----------------------------------------
[info] Partie en cours...

[0][0] :   Nourritures : 0  Pheromones : -1
[0][1] :   Nourritures : 0  Pheromones : 0
[0][2] :   Nourritures : 0  Pheromones : 0
[0][3] :   Nourritures : 0  Pheromones : 0
[0][4] :   Nourritures : 0  Pheromones : 20
[0][5] :   Nourritures : 0  Pheromones : 0
[0][6] :   Nourritures : 0  Pheromones : 0
[0][7] :   Nourritures : 4  Pheromones : 0
[0][8] :   Nourritures : 2  Pheromones : 0
[0][9] :   Nourritures : 0  Pheromones : 0
[1][0] :   Nourritures : 0  Pheromones : -1
[1][1] :   Nourritures : 0  Pheromones : 0
[1][2] :   Nourritures : 0  Pheromones : 0
```

Gestion de projet

Durant ce projet, nous avons défini notre organisation suivant différents objectifs à atteindre. Nous avons définis ces objectifs dans un diagramme de **gantt** et défini les **deadlines**.

Nous sommes organisés en agile en définissant des sprint. regroupants plusieurs tâches et parfois de plusieurs objectifs différents (exemple: spécifications du logiciel et conception préliminaire).

Nous avons défini une **backlog** du projet suivant les objectifs à atteindres.

A la fin de chaque sprint d'une semaine, nous faisions le points et nous redéfinissons le prochain sprint avec notre backlog.

Le choix des tâches se faisait en fonction des compétences de chacun.

Conclusion

Bilan général du projet

Pour résumer le projet, nous avons eu des difficultés lors de la conception du logicielle. Nous étions partie sur une architecture complexe à utiliser. Il a donc fallu ajuster l'architecture en suivant des principes d'architecture comme l'utilisation de Design Pattern.

Perspectives d'évolution du simulateur

Il serait intéressant de mettre en place la création d'une nouvelle fourmilière à partir d'une reine ayant atteint sa maturité. Ainsi implémenter le système de conflit entre les différentes fourmilières pouvant mener à des affrontements entre fourmis de fourmilières différentes.

On pourrait souhaiter ajouter davantage de contrôle de la part de l'utilisateur comme par exemple l'ajout de nouvelles fourmis ou de nouveaux obstacles.

La crátion d'une vraie interface graphique ajouterais une meilleure compréhension et visualisation de l'ensemble.

Sources

GitHub (new_archi2) : https://github.com/MohamedElGuendouz/simulation-fourmilieres

Documentations : dans le répertoire doc/.