Projet POO
Simulation Orientée-Objet de systèmes multiagents

Equipe 65

Novembre 2019

1 Les balles

1.1 Choix et Implémentation

Par le biais de la classe Point déja existante, il fut plus simple pour nous de créer une classe Balls comprenant plusieurs balles représentés par ces Points. Ainsi l'utilisation du simulateur se fait directement via une classe BallsSimulateur que nous avons implanté.

1.2 Hiérarchie de classes

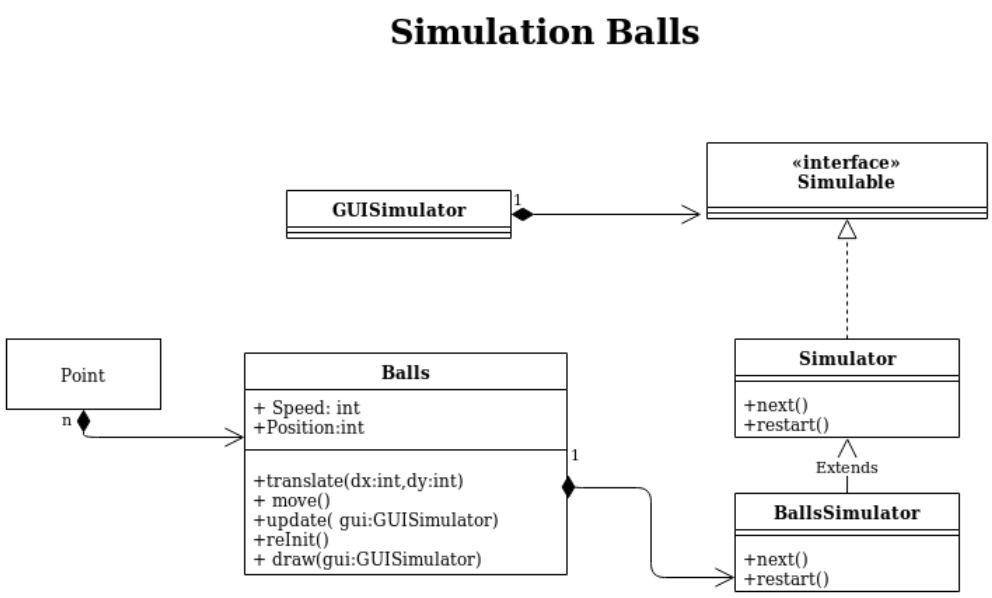

Figure 1: Diagramme ULM pour la simulation des balles

2 Jeu de la vie de Conway, Jeu de l'immigration, Modèle de Schelling

2.1 Choix et Implémentation

Pour l'implémentation de simulations d'automates cellulaires, nous avont décidé d'utiliser une classe mère Celltab représentant un tableau de cellules. Toutes les fonctions utilisées par la suite, dans les différents type d'automates, ont ainsi pu être factorisés dans cette classe. De plus, en utilisant l'héritage sur les fonctions qui diffèrent entre les automates, nous avons pu implémenter simplement chacunes des méthodes requises. Au final, avec une classe mère et 3 classes filles, nous avons pu implémenter et simuler les 3 jeux de la partie automate cellulaire.

2.2 Hiérarchie de Classes

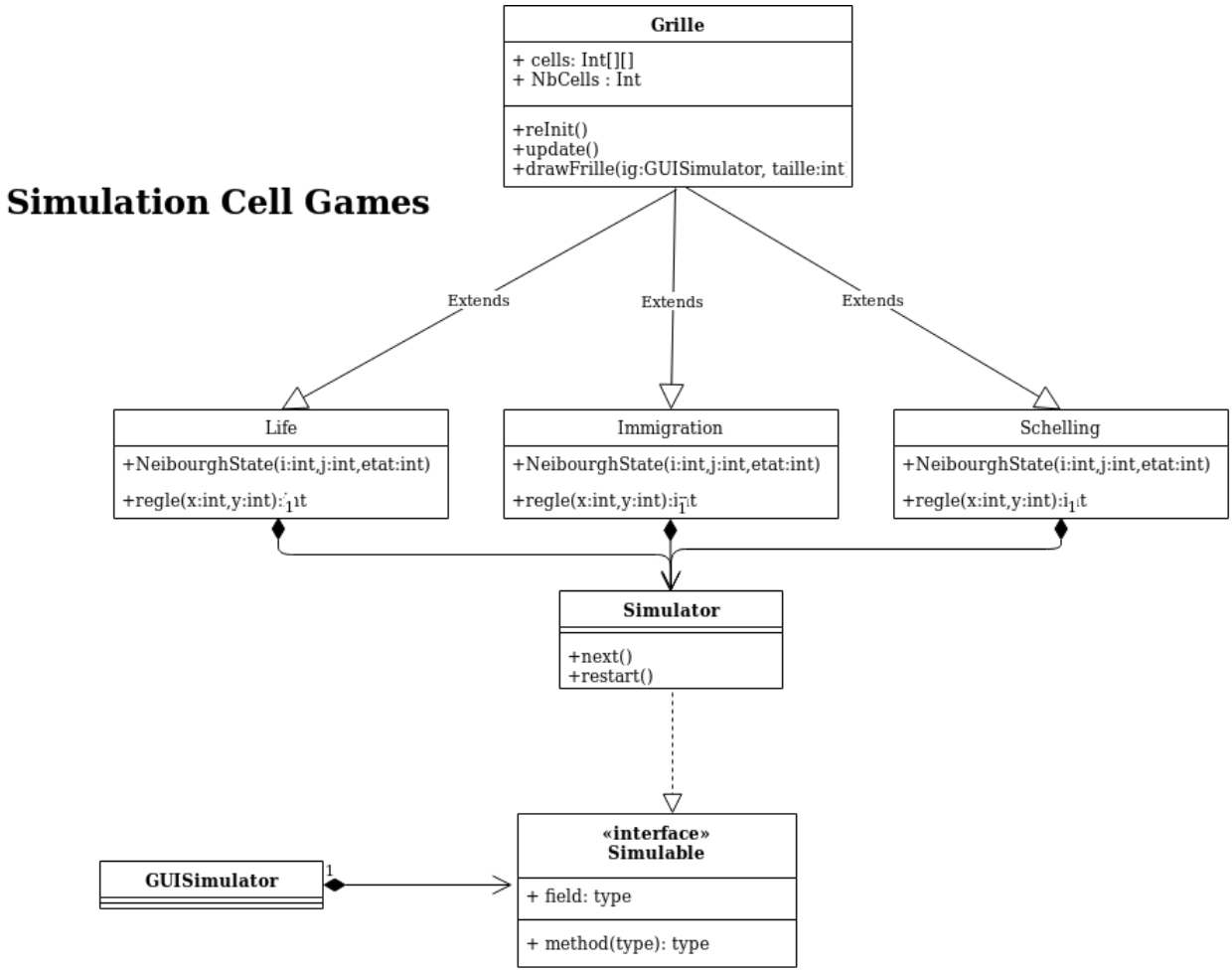

Figure 2: Diagramme ULM pour la simulation des jeux de la vie, de l'immigration et de Schelling

3 Modèle d'essaims : les boids

3.1 Choix et Implémentation

Pour l'implémentation du modèle de boids, nous avons commencé par créer une classe boid. De cette classe mère nous avons choisi de faire hériter deux types de boids différents : requins (shark) et poissons (fish) dont les interactions entre eux et avec les autres types de boids diffèrent en partie. La classe Flock représente un banc de poissons.

3.2 Hiérarchie de Classes

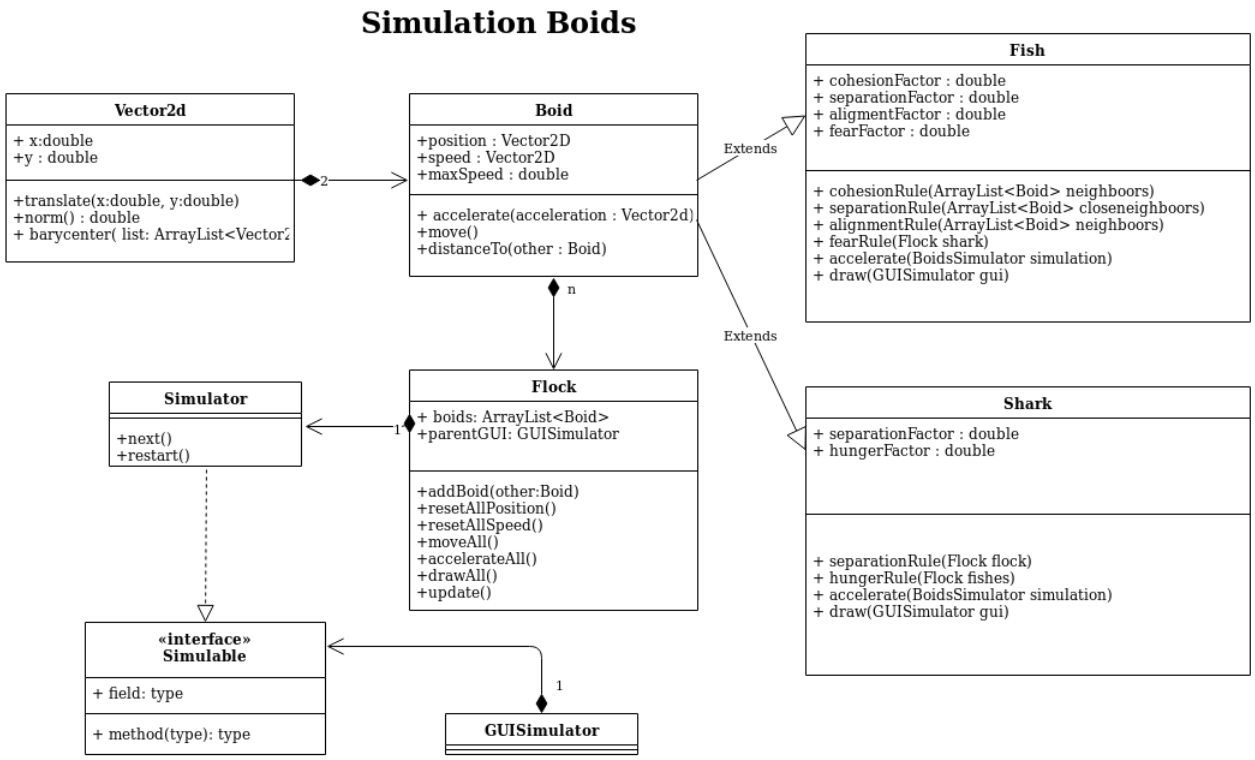

Figure 3: Diagramme ULM pour la simulation des boids

4 Les événements

4.1 Choix et Implémentation

Pour l'implémentation des événements, nous avons choisi de créer une classe mère Event, dont découlent les classes BallsMoveEvent, CellsUpdateEvent et FlockUpdateEvent. La Classe EventManager géré avec une <PriorityQueue> prends donc 0 ou plusieurs de ces Events, classé par leur ordre d'exécution temporelle grâce à la classe EventComparator. Ainsi nous avons une organisation simple et hiérarchisé sur les événements de chacune de nos simulations.

4.2 Hiérarchie de Classes

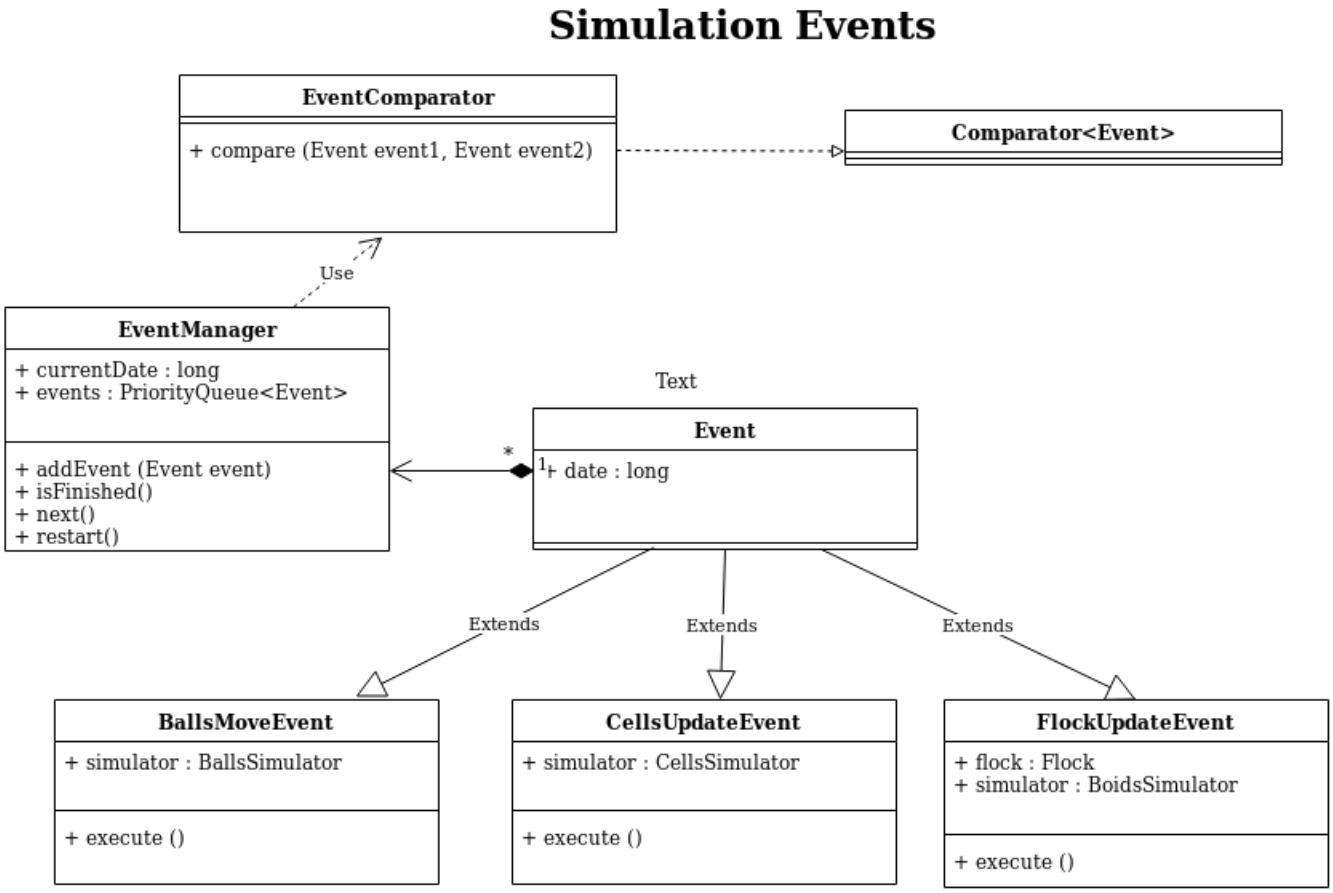

Figure 4: Diagramme ULM pour les événements