Alexandre MOUTOUH
Isis ALIE-SANDEVOIR

Monsieur MAVROMATIS

PROJET
RABBITWORLD
cahier de conception

Polytech Marseille
18/12/2017

SOMMAIRE DU CAHIER DE CONCEPTION

1. Présentation du logiciel

1.1 Outils utilisés

Le projet sera réalisé sous un environnement Unix et programmé en JAVA. Nous avons choisi d'utiliser ces interfaces car l'une des contraintes du client est l'utilisation d'un langage orienté objet ainsi que l'utilisation de notions orientées objet telles que l'héritage, le polymorphismes et l'interaction entre objet.

Les schémas présentés respectent les conventions UML.

1.2 Présentation de la mission du produit logiciel

Les principaux services attendus par l'utilisateur sont de jouer un jeu de la vie, et observer l'évolution des cellules du jeu qui, dans le projet, sont des lapins.

Les principaux traitements sont la naissance des lapins, la vie des lapins, la reproduction des lapins et la mort des lapins. Nous allons aussi observer de manière plus secondaire, l'évolution du cycle de vie des carottes. En l'occurrence, dans notre cas, la naissance d'une carotte dans le champ, le vieillissement des carottes (qui deviennent pourries), l'ingestion des carottes par les lapins (qui signifie leur mort).

1.3 Position du logiciel dans le système

Le jeu devra être capable d'être exécuté sur n'importe quel ordinateur sous Unix. La machine cible sera donc la machine de l'utilisateur.

1.4 Fonctions générales du logiciel

Les principales fonctions devant être assurées par le logiciel sont donc les règles de vie du jeu. Soit la naissance, la vie et la mort des lapins dans un champ. Nous précisons à nouveau que le jeu comprendra l'évolution de vie des carottes.

2. Exigences

2.1 Exigences fonctionnelles

Les éléments constituant le système sont les lapins, les carottes (empoisonnées ou non), et le champ.
Les interactions entre les objets seront :

- Lapins et cases pour les déplacements
- Carottes et lapins pour la consommation des carottes
- Lapins et lapins pour la naissance de quatre lapins
- Champ et case pour associer les déplacements à une situation "géographique" dans le champ

2.2 Exigences non fonctionnelles

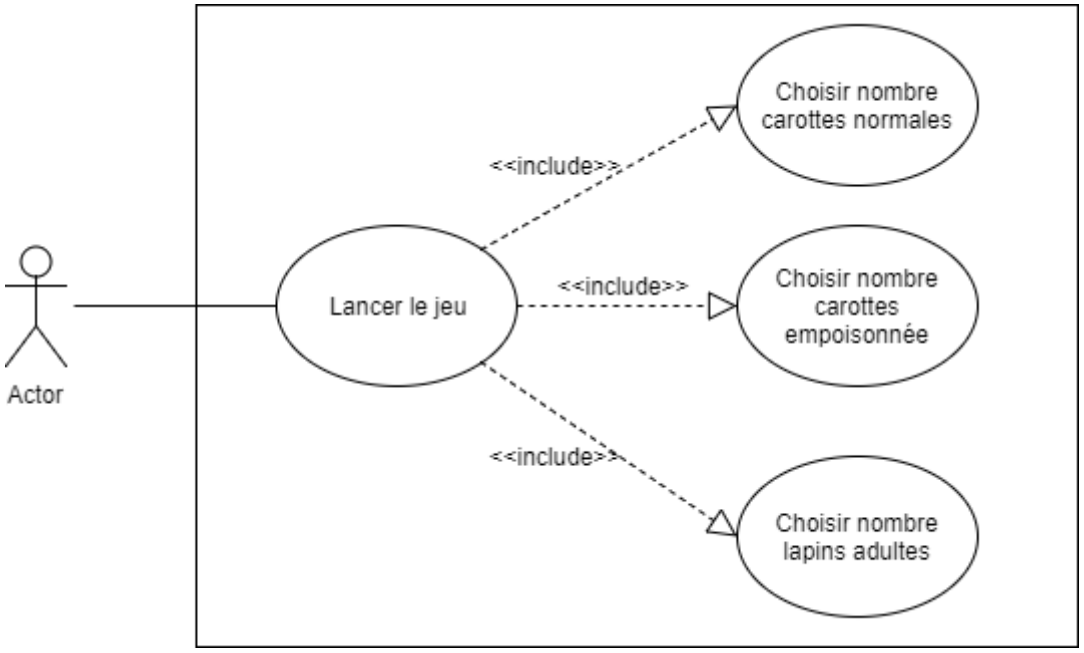
Les contraintes non-fonctionnelles sont :

- L'interface graphique qui est facultative.
- Ajout et choix d'autres règles de vie.
- Proportion de carottes différentes de sorte à laisser un temps de latence au lapin différent.
- Ajout d'ennemis (renards) et/ou de renaissance d'un lapin mort.

2.3 Rôle de l'utilisateur

Le rôle de l'utilisateur sera de paramétrer l'état initial du jeu en début de partie. Le paramétrage comprendra : le nombre de lapins, le nombre de carottes, nombres de carottes empoisonnées.

Diagramme de cas d'utilisation



2.4 Fonctionnalités

Les principales tâches de chaque acteur sont :

- ❖ Pour les lapins :
  - ➢ naître : provient de la reproduction de deux lapins
  - ➢ grandir (devenir 2-3 sortes de différents lapins)
  - ➢ manger : interaction avec les carottes
  - ➢ se reproduire entre lapins : interaction entre lapins
  - ➢ mourir
- ❖ Pour les carottes :
  - ➢ naître
  - ➢ pousser
  - ➢ être mangées par les lapins : interaction avec les lapins
  - ➢ pourrir (puis tuer les lapins si mangée) : interaction avec les lapins

2.5 Interactions

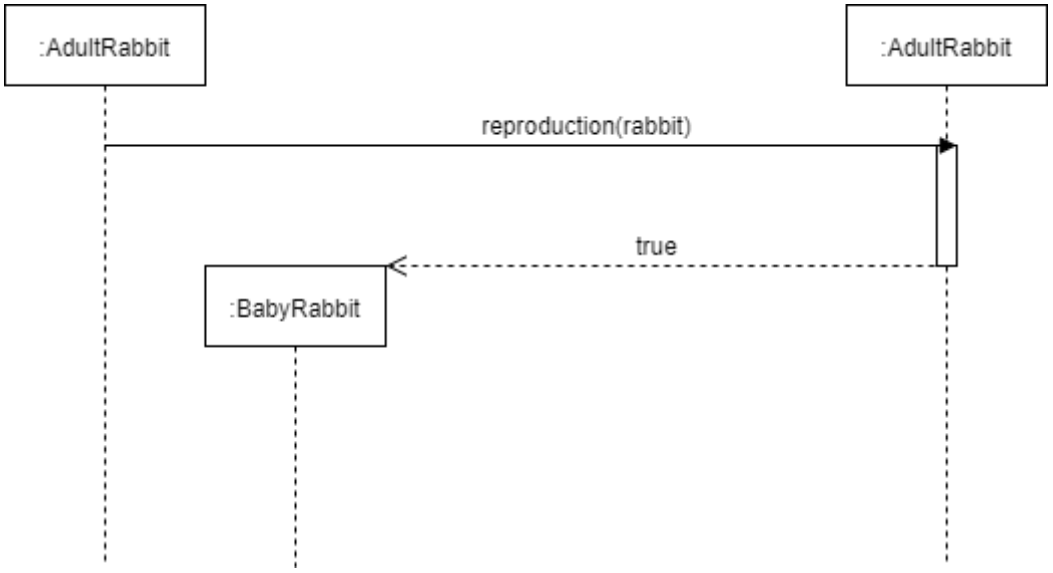
Le principe du langage objet est de réaliser des interactions entre des objets. Ainsi, voici ci-dessous les interactions entre les objets de notre jeu :

- Reproduction de deux lapins : Deux lapins sur des cases côte à côte peuvent se reproduire. Ils créent un lapin enfant qui ne peut pas se reproduire et grandit au bout d'un certains temps défini par le programmeur.
- Ingestion de carotte par les lapins : Les carottes ne se déplacent pas lorsqu'elles ont poussées. Lorsqu'un lapin arrive sur une case où se trouve une carotte alors cette dernière est mangée.
- Ingestion de carotte empoisonnées : Les carottes empoisonnées sont des carottes étant restées trop longtemps sans avoir été mangées. Lorsqu'un lapin mange une carotte empoisonnée, la carotte disparaît et le lapin aussi (ils meurent tous les deux).

Si un lapin vit trop longtemps sans manger, ce dernier meurt. La fin du jeu se définit par la mort de tous les lapins.

Diagramme de séquence

- *reproduction de deux lapins*



- *Ingestion de carotte saine*

- *Ingestion de carotte pourrie*

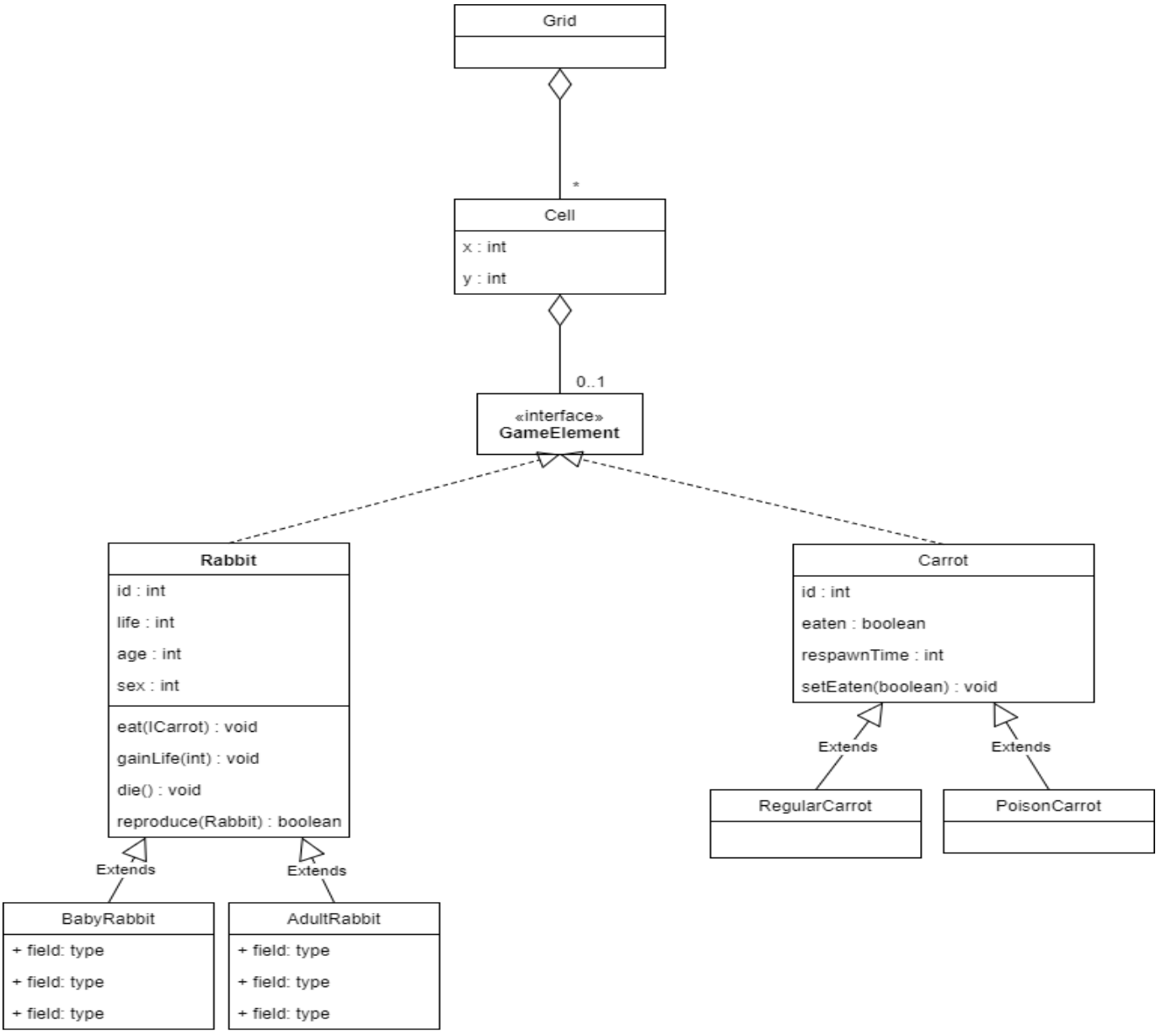
Le reste du jeu se repose simplement sur le déplacement des lapins sur le champ à chaque "unité de temps" (dans notre cas il s'agit d'un tour de jeu, c'est-à-dire à chaque actualisation de la grille). Chaque cas d'utilisation apparaîtra dans le cahier de test.

2.6 Modélisation

Le jeu est constitué de principalement 3 objets : les lapins, les carottes et le champ. Dans le programme, chaque objet est représenté sous forme de classes de la manière suivante :

Les lapins : Le lapin est une classe. Cette classe possède un héritage. Un lapin se décompose en un lapin adulte qui peut manger, se reproduire et se déplacer et un lapin enfant qui ne peut que se déplacer et devient adulte au bout d'un certains temps. Le lapin aura différents sexes qui apparaîtront en attribut.
Les carottes : La carotte est une classe. Cette classe possède un héritage. Une carotte se décompose en une carotte normale qui n'est simplement que mangée et une carotte empoisonnée qui est mangé et qui tue le lapin dès qu'elle est mangée.
Le champ : Le champ est une grille qui constitue une classe. Le champ est composé (lien de composition) de cases qui permettent le déplacement des lapins. La classe case contient alors le déplacement possible.

Diagramme de classe



3. Exigences opérationnelles

3.1 Environnement matériel

Il est requis que Java 1.8 soit installé sur la machine exécutant le jeu.

3.2 Environnement logiciel

Il n'y a pas de logiciels imposés. Le projet doit se faire en langage orienté objet (C++, Java, C#...). Notre choix s'est porté sur Java. Nous développerons sur l'environnement de développement intégré (IDE) Eclipse.

3.3 Mise en œuvre

Le logiciel se lancera et s'utilisera depuis un terminal. L'utilisateur paramètre le jeu en début de partie en choisissant le nombre de lapins, de carottes et de carottes empoisonnées.

3.4 Performances

Le temps de latence entre chaque tour de jeu devra être raisonnable (pas trop long mais pas trop rapide non plus pour laisser le temps à l'utilisateur de bien visualiser l'état pendant le tour et pouvoir ainsi bien comprendre l'évolution de l'état du jeu au tour suivant).

Le temps d'évolution des lapins (pour que le bébé lapin deviennent adulte) sera de quelques tour de jeu dans le but d'éviter de bloquer le jeu à un moment où il n'y a que des bébés lapins.

Le temps de pousse d'une carotte sera très court car il est plus intéressant que le jeu tourne et que les carottes puissent être mangées plutôt que les lapins se déplacent sans action particulière.

3.5 Sécurité

Le jeu est considéré comme une simulation visuelle. Il n'y a pas de réelles contraintes de sécurité.

3.6 Interfaces avec des fichiers ou des bases de données

Un interfaçage avec une base de données est une spécification non fonctionnelle et sera réalisée si le délais le permettent. Néanmoins, s'il venait à être réalisé, le mode d'accès serait autorisé en lecture pour l'affichage et en écriture pour l'enregistrement des données.

3.7 Interface Homme / Machine

Le jeu est ouvert à tout type d'utilisateur. Il peut s'agir de l'utilisateur lambda étant seulement au courant du principe du jeu et qui cherche à le découvrir visuellement. Ce type d'utilisateur a une connaissance normale du système d'exploitation sous lequel sa machine fonctionne. Ou il peut aussi être lancé par un type d'utilisateur

plus expérimenté, connaissant le principe du jeu ainsi que son fonctionnement et pouvant deviner quelles solutions techniques ont potentiellement pu être mises en application pour le réaliser.
Dans tous les cas, il n'est pas nécessaire de restreindre et privilégier l'accès selon les profils d'utilisateurs.
Le jeu se lancera et s'exécutera sur un terminal. De ce fait, la représentation des différents acteurs sera très limitée (un caractère différent selon l'acteur représenté). Par exemple "L" pour représenter un lapin. La légende restera affichée pendant chaque tour de jeu. En début de partie, l'utilisateur aura des données à saisir concernant le nombre de lapins et carottes initialement présents sur le terrain.

Dans le cas où le temps le permet, et que la mise en place d'une IHM est effectivement lancée, la représentation des lapins et carottes se fera avec des sprites. L'affichage sera alors fait dans une fenêtre d'une API graphique de Java à définir.

3.8 Exigences concernant la conception et la réalisation

Le jeu doit contenir au minimum des lapins se déplaçant dans un champ, mangeant des carottes et se reproduisant. Le facteur qualité principal requis est que le jeu fonctionne c'est-à-dire que l'utilisateur puisse entrer les paramètres de son choix et regarder les lapins se déplacer et manger jusqu'à la fin du jeu. Le deuxième facteur qualité qui est une exigence non-fonctionnelle mais qui est un réel point positif dans le rendu du projet est l'interface graphique. En effet, elle apporte une qualité au jeu bien supérieure.

L'objectif de l'équipe est de rendre en temps voulu les fichiers sources, les cahiers de conception et de spécifications ainsi que les différents rapport d'avancement. Une démonstration sera faites au client, d'où le fait que l'interface graphique a une certaine importance dans le projet.
Le jeu ne fonctionnera pas sous Windows mais que sous Unix.

Le jeu se joue donc sous Unix, avec une fenêtre de paramétrage en début de jeu où il suffit d'entrer le nombre de lapins, le nombre de carottes et le nombre de carottes empoisonnées. Une fois le jeu lancé, ce dernier sera totalement autonome et prendra fin à la mort de tous les lapins.

3.9 Tests

Des tests seront réalisés régulièrement au cours du projet, notamment à la fin de chaque sprint pour que tout ce qui a été fait fonctionne bien avant de commencer un nouveau sprint. De plus, en fin de jeu il sera tester les limites du jeu, principalement au niveau du paramétrage (nombre de lapins très élevé et nombre de carottes très faible par exemple). Ainsi sera tester la robustesse et la flexibilité du jeu.
Les tests seront répertoriés dans un cahier de test.