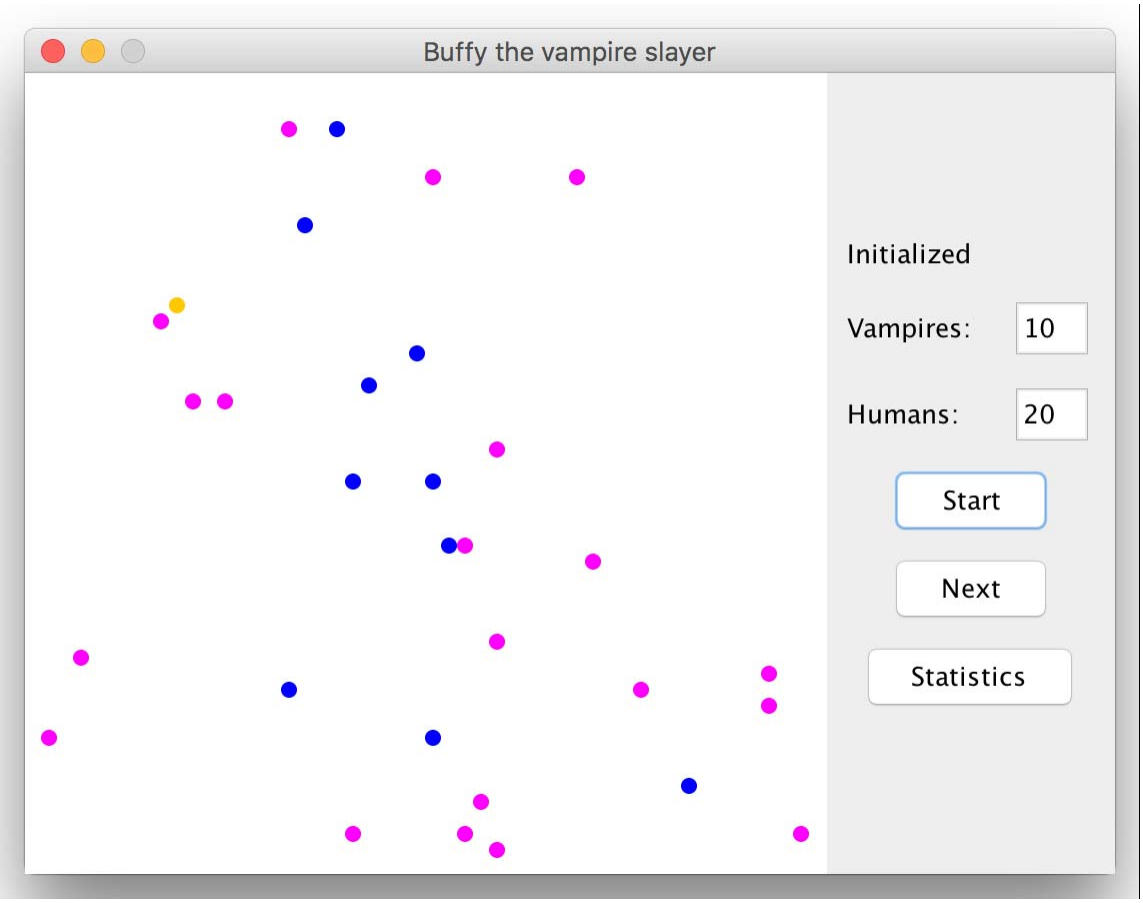
Laboratoire 11: Buffy la tueuse de vampires

1. Introduction

Buffy, la tueuse, doit sauvegarder l'espèce humaine de la menace des vampires. L'application doit permettre de simuler les actions des vampires, des humains et de Buffy (dénotés par la suite comme "humanoïdes"), en mode "tour par tour" et d'effectuer des statistiques sur un ensemble de simulations.



- Buffy (en jaune), les vampires (en bleu) et les humains (en violet) sont placés aléatoirement sur une grille 50x50 (bouton Start).
- L'interface graphique affiche le numéro du tour courant et permet de passer au tour suivant (bouton Next), d'effectuer des statistiques (bouton Statistics) ou de quitter le programme.
- A chaque tour de la simulation chaque humanoïde décide de sa prochaine action. Toutes les actions sont ensuite résolues simultanément (pour ne pas avantager les premiers humanoïdes dans la liste). L'état résultant de la simulation est ensuite affiché.

2. Actions

- Chaque humain se déplace au hasard sur la grille, d'une case à la fois.
- Buffy attaque le vampire le plus proche. S'il est à une case de distance elle peut le tuer, sinon elle le chasse en avançant de deux cases dans sa direction. Si tous les vampires sont détruits, elle se déplacera au hasard sur la grille (comme les humains).
- Chaque vampire attaque l'humain (mais pas Buffy) le plus proche. S'il est à une case de distance il peut le tuer ou le transformer en vampire (une chance sur deux), sinon, il le chasse en avançant d'une case dans sa direction. Attention: un humain ne peut donner naissance qu'à un seul vampire. S'il n'y a plus d'humains à chasser, le vampire reste sur place pour dormir.
- Les statistiques sont obtenues en effectuant 10'000 simulations pour les mêmes conditions initiales données (nombre de vampires et d'humains, mais placement aléatoire), mais sans afficher le déplacement des humanoïdes. Chaque simulation est arrêtée lorsqu'il n'y a plus de vampires. Sur l'ensemble des simulations le pourcentage de succès de Buffy (pourcentage des simulations où au moins un humain reste vivant) est calculé et affiché.

3. Mise en oeuvre

- Concevoir un diagramme de classes permettant de factoriser au maximum le code. En particulier il est important de dissocier les classes gérant l'interface graphique de celles permettant d'effectuer la simulation (celle-ci pouvant être réalisée sans interface graphique, comme dans le cas du calcul des statistiques).
- Chaque classe d'humanoïde doit référencer un objet `Action`, caractérisant la prochaine action à effectuer. Cette classe `Action` doit posséder une méthode `void execute(Field f)` permettant d'exécuter l'action courante en influant sur l'environnement, représenté par la classe `Field`.
- La classe `Field` permet de faire évoluer les humanoïdes sur une grille. Ceux-ci seront manipulés au moyen d'une liste d'humanoïdes. Cette classe doit fournir la méthode suivante permettant de gérer un tour de la simulation.

```
public int nextTurn()
{
  // Déterminer les prochaines actions
  for (Humanoid h : humanoids)
    h.nextAction(this);

  // Executer les actions
  for (Humanoid h : humanoids)
    h.getAction().execute(this);

  // Enlever les humanoides tués
  Iterator<Humanoid> it = humanoids.iterator();
  while (it.hasNext())
    if (!it.next().isAlive())
      it.remove();

  // Ajouter les humains transformés en vampires (sired)
  humanoids.addAll(sired);
  sired.clear();

  return turn++;
}
```

  La classe `Field` doit également offrir une méthode permettant de trouver l'humanoïde, instance d'une classe donnée, le plus proche d'un humanoïde donné.
- Il n'est pas interdit d'empécher que deux humanoïdes occupent la même case de la grille ou de conecevoir des algorithmes plus performants que celui proposé pour augmenter le pourcentage de succès de Buffy.