POO2 – Laboratoire 4

Buffy la tueuse de vampires

## UML

## Conception

#### Générale

Nous avons choisi que plusieurs humanoïdes puissent prendre la même case, pas de gestion de collisions. D’une part ceci corresponds mieux aux critères que les actions doivent être exécutés de manière a ce que personne ne soit favorisé. Si nous gérons les collisions, les premières humanoïdes qui effectuent une move peuvent perturber les attentions de bouger dans cette même case des autres humanoïdes. De plus, d’après la donnée, il est permis de modifier l’algorithme afin de donner plus de chances de réussite à Buffy. Si les collisions sont gérées, c’est Buffy qui paye le prix car son grand avantage est sa plus grande vitesse de déplacement.

D’autre part, une seule action peut être effectué en un tour par humanoïde. Si un humanoïde décide d’attaquer un autre, il va le tuer ou transformer mais ne va pas se déplacer dans cette case (comme dans le jeu d’échec par exemple).

Enfin, pour rendre le code plus simple, si un mouvement aléatoire aboutis hors des bornes du champ, l’offset qui le pousse hors des bornes sera mis à zéro. Si un humanoïde se trouve tous a gauche et veux aller dans la direction « TOP\_LEFT » il va finir par aller vers le haut. S’il veut aller à gauche, il ne bougera pas ce tour. De nouveau, le succès de Buffy n’est pas affecté car elle va se diriger en directions des vampires, si au moins un vampire existe, et donc n’aura pas une direction qui va hors des bornes.

Les classes Action

Nous avons défini trois types d’actions possible, issues d’une même classe abstraite « Action » que sont « Kill », « Move » et « Transform ».

**La classe Move** est une action qui peuvent être utilisée par tous les Humanoïdes. Tandis que les humains se déplacent n’importe où, les vampires, eux, vont se diriger vers les humains les plus proches d’eux. Il en va de même pour Buffy qui se déplacera vers le vampire le plus proche d’elle. La classe Move prends un argument une direction qui est soit donné aléatoirement par Humain ou calculé pour aller dans une certaine direction. A la charge de la classe Move est également de vérifier que le mouvement ne dépassera pas les bornes du champ. En résultat, cette classe va définir une « futurePosition » sur l’humanoïde qui deviendra sa position pendant la partie « executeAction ».

Comme la classe Move change la position d’un humanoid, elle est déclarée comme classe amie de Humanoide. Ainsi que la classe Move est capable de changer le positionnement d’un humanoide.

**La classe Kill** est une action que seuls les vampires et Buffy peuvent exécuter. Les vampires utiliseront cette action pour tuer un humain, et Buffy pour tuer des vampires. Les modalités sont vérifiées dans les méthodes « setAction » par l’humanoïde. L’action Kill ne va pas vérifier que les humanoïdes sont à distance adéquate, ceci est vérifié pendant le « setAction ». Ainsi, peut importe si un humanoïde c’est déplacé, il sera tué si une action Kill est affecté pour lui.

**La classe Transform** est une action spécifique aux vampires qui peuvent faire d’un humain, un vampire. Au moment d’exécute, cette action va tagger un humain comme étant « toTransform ». La classe Field pendant la gestion du tour va construire un nouveau Vampire à partir de cette humain, grâce au Constructeur dans la classe Vampire, est détruire cet humain.

Les classes Humanoïde

Les humanoïdes sont représentés par une classe abstraite « Humanoid » et des classes concrètes « Human », « Vampire » et « Buffy ». Chaque Humanoid possède une position et le nombre de pas qu’il peut effectuer pendant un déplacement. Elle définisse un ensemble de méthodes qui permettent d’identifier le type d’un Humanoid ainsi que son état et son positionnement.

Les classes Movement

Nous avons décomposé les déplacements d’un humanoïde de la façon suivante :

Nous avons tout d’abord la position d’un humanoïde dans la classe Position, représentée sous forme de coordonnée (x,y). x est le positionnement horizontale et y vertical

Ensuite, nous avons le Movement qui est utilisé pour gérer un déplacement. Un humanoïde est capable de se déplacer dans 8 directions possibles, horizontalement, verticalement et en diagonale. Ainsi, la classe Mouvement offre des méthodes permettant de donner une direction random mais également pour donner une direction entre 2 positions (départ et arrivé).

Elle offre également une méthode getOffset qui donne un Offset par rapport à la direction. L’offset indique comment x et y doivent changer pour que l’humanoïde va dans une direction particulière.

La classe Field

La classe principale qui encapsule une liste d’humanoïdes et s’occupe de gérer les tours comme une classe contrôleur. Elle offre un ensemble de méthodes permettant de parcourir la liste en lecture « Itérateur constant » et de rechercher un humanoïde le plus proche d’une position donnée en retournant la distance par paramètre interposé.

La recherche de l’humanoïde le plus proche se fait avec la méthode findClosestHumanoid qui prends un argument template indiquant le type de l’humanoïde qu’on recherche. Cette méthode est spécialisée explicitement pour les types Humain\* et Vampire\*. Ainsi Buffy ne peut y être recherchée.

Pour le parcours interne, la classe Field offre les méthode **privé** begin et end qui elle donnent un iterateur non constant.

Elle donne comme interface publique un ensemble de méthodes getters constant qui permettent de récupérer les informations live, au moment de l’appel, sur la simulation en cours. Ces méthodes sont utilisées pour fabriquer les statistiques sur le succès de Buffy.

La classe Field ne permet a aucune classe extérieur, appart les classe action dont elle fait exécution elle-même, de interférer avec la simulation en cours.

Les classes Simulation et Display

**Classe Display**

Cette classe gère le lancement de la simulation avec une interface utilisateur console. Elle hérite de la classe Field et change le comportement de la méthode nextTurn afin d’afficher les tours courant à l’utilisateur. Pour faire le parcours, elle se serve des méthodes begin et end qui retournent un ConstIterator qui ne permet pas la modification sur des humanoïdes.

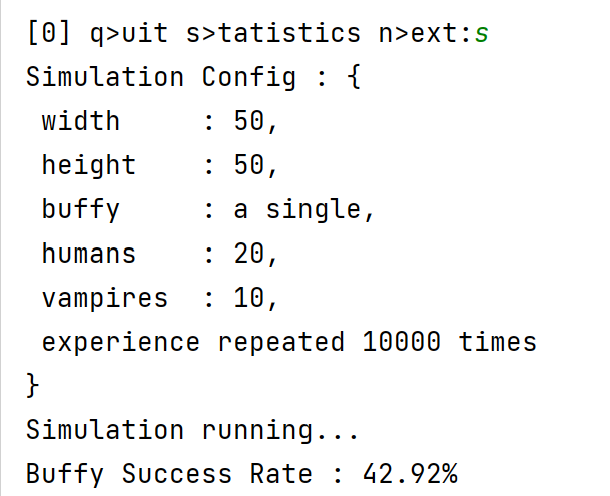
Elle encapsule un tableau à 2 dimensions de Humainoid\* qui sert de mémoire d’affichage. Comme les humanoïdes peuvent être à plusieurs sur la même case la priorité est donnée a Buffy au cas ou plusieurs se trouve sur la même.

Cette classe n’est pas capable d’interférer avec une simulation en cours, elle ne peut que lire les données et demander le prochain tour.

**Classe Simulation**

La classe Simulation prends un Field en argument ainsi que le nombre de répétitions. Elle va crée une nouvelle instance de Field à chaque répétition et effectuer les tours jusqu’à ce qu’il y ait plus de vampires sur le champ. Sur l’ensemble de répétitions, elle va tenir compte de succès de Buffy. Le succès de Buffy est caractérisé par le fait qu’il reste au moins un humain vivant au moment ou tous les vampires sont morts. Elle va calculer son percentage de succès.

## Tests

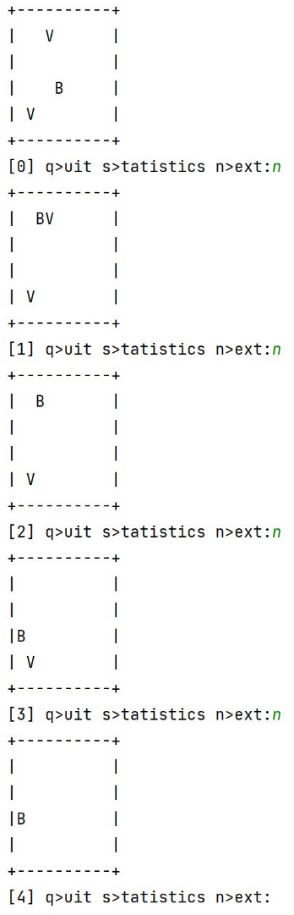
Test de simulation d’une grille 50x50 avec 20 humains et 10 vampires

Nous avons lancé une simulation du programme avec comme taille une grille de 50 cases par 50 ainsi que 20 humains et 10 vampires.

Après 10'000 expériences, nous constatons que le pourcentage de réussite de Buffy est d’environ 43%.

Tests des vampires quand il n’y a plus d’humains

Quand il n’y a plus d’humains sur le terrain et qu’ils restent des vampires, ces derniers restent sur place. Buffy continue alors à les tuer mais dans la simulation, cela comptera comme un échec.

Une image contenant table

Description générée automatiquement

Tests des humains quand il n’y a plus de vampires

Une image contenant table

Description générée automatiquementÀ l’inverse, s’il ne reste aucun vampire, les humains restant et Buffy continueront à se déplacer dans la grille :

Une image contenant table

Description générée automatiquement

Tests des vampires tuant un humain

Un vampire a une chance sur deux de tuer un humain, pour autant que ce vampire se trouve à une seule case, diagonale incluse, de l’humain.

Une image contenant table

Description générée automatiquement

Tests des vampires transformant un humain

Si le vampire a une chance sur deux de tuer un humain, il a aussi une chance sur deux de le transformer, à la même condition que précédemment, celle d’être à une case de l’humain.

Une image contenant table

Description générée automatiquement

Tests de Buffy tuant un vampire

Une image contenant table

Description générée automatiquementBuffy peut tuer un vampire que s’il se trouve à une case près d’elle. À l’itération 2, Buffy se trouve sur la même case que le vampire et le tuera au tour suivant.

Une image contenant table

Description générée automatiquement

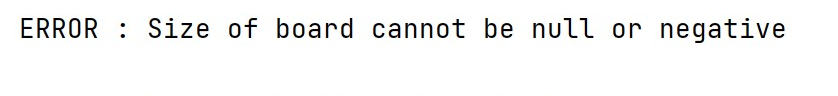
Tests de validité des valeurs de la grille

**Avec comme argument :** 12 12

Une image contenant texte

Description générée automatiquement

**Avec comme argument :** 0 0 1 1



**Avec comme argument :** 1 1 -1 0

Une image contenant texte

Description générée automatiquement

**Avec comme argument :** 1 1 2 2

Une image contenant texte

Description générée automatiquement