Haute École d'Ingénierie et de Gestion du Canton de Vaud

Programmation Orientée Objet Avancée

Travail Pratique
Buffy la tueuse de vampire

Professeur: Pier Donini

Assistant: Grégoire Decorvet

Étudiant: Nelson Jeanrenaud, André Marques Nora

Table des matières

Introduction

L'objectif est de concevoir une application C++ qui permet de jouer au jeu Buffy la tueuse de vampires. …

Choix d'implémentation

Différents types d'humanoïdes

Nous avons décidé d'utiliser des rôles à la place de faire créer des spécialisations d'humanoide pour les humains, les vampires et buffy. De cette manière la transformation d'un humain en vampire nécessite seulement la création et l'affectation d'un nouveau rôle et pas la re-création complète de l'humanoïde. Cela nous évite d'avoir le problème qu'un humain peut donner naissance à plusieurs vampires s' il est mordu plusieurs fois en un tour.

Un rôle défini les actions qu'un humanoïde peut entreprendre, quand il choisit de les exécuter et des caractéristiques comme la vitesse de déplacement.

Gestion des actions

Les humanoïdes possèdent une référence sur l'action qu'il va effectuer pendant ce tour qui a été affecté au préalable dans la méthode *setAction* en dépendant de son rôle concret par liaison dynamique. La classe **Action** est donc une abstraction de l'algorithme qui est appliqué quand l'humanoïde doit exécuter une action.

Héritage Human-Buffy

Nous avons pris la décision de faire hériter **Buffy** de **Human** car elle possède également la capacité de se déplacer aléatoirement sur le plateau. En revanche, elle ajoute une référence sur une instance de la classe **Chase** pour pouvoir chasser et tuer les vampires. Les méthodes *getSpeed* et *setAction* sont donc redéfinies pour permettre à ses instances de poursuivre les vampires ou de bouger aléatoirement selon s' il reste des vampires en vie.

Cet héritage nous oblige à utiliser des *typeid* et pas des *dynamic_cast*, pour que les vampires poursuivent les humains mais pas les vampires. Certes, cette décision pourrait nous amener à devoir préciser chaque type individuellement si nous rajoutons des sous-classes dans le futur. Mais nous estimons que le bénéfice de factorisation et le fait que notre hiérarchie soit très simple justifie le choix.

Affichage

Nous avons créé une sous-classe **StreamField** à **Field** qui permet d'afficher l'état du plateau dans un flux. Pour dissocier complètement l'affichage de la logique du jeu, nous utilisons le patron visiteur pour définir un caractère à chaque rôle du jeu.

Tests

Dans le cadre de ce laboratoire, il n'est pas vraiment possible de créer une classe de test pour vérifier que tout fonctionne correctement. Il a donc fallu vérifier étape par étape d'une simulation que les entités effectuaient leurs actions correctement. Pour cela, nous avons utilisé une méthode de la classe **Simulation** nommée *graphicSimulate* qui affiche sur la console chaque tour de la simulation.

Déplacement

Test effectué	Résultat attendu	Résultat obtenu
Les humains se déplace aléatoirement	OK	OK
Buffy se déplace vers les vampires	OK	Ok
Buffy se déplace aléatoirement quand il n'y a plus de vampire	OK	OK
Les vampires chassent les humains	OK	OK
Les vampires ne bouge plus après avoir tué tous les humains	OK	OK
Aucun humanoïdes ne peut sortir du field	OK	OK

Tuer et transformer

Test effectué	Résultat attendu	Résultat obtenu
Les vampires tuent les humains à une case ou moins d'eux	OK	OK
Buffy tue les vampires à une case ou moins	OK	Ok
Les humains peuvent devenir des vampires (50%)	OK	OK
Un seul vampire est créé lors de la transformation d'un humain	Ok	OK

Affichage et simulation

Test effectué	Résultat attendu	Résultat obtenu
Affichage du jeu sur la console avec les bonnes tailles	OK	OK
Affichage du bon nombre de vampires, humains et buffy	OK	OK
Chaque humanoïde s'affiche sur le field avec le bon caractère	OK	OK
Possibilité de faire une simulation complète du jeu avec les paramètres voulu par l'utilisateur	OK	OK
Possibilité de faire 10'000 simulation et obtenir le score de réussite de Buffy	OK	OK
Utilisation de la touche 'q' permet l'arrêt du jeu	OK	OK
Utilisation de la touche 's' lance 10'000 simulation sur les paramètres actuels du jeu	OK	OK
utilisation de la 'n' affiche le prochain tour de la partie	OK	OK

Diagramme UML