POA - Buffy

Auteurs: Roland Samuel, Junod Arthur, Häffner Edwin

Introduction

Pour ce labo, nous avons dû implémenter une simulation où se trouvent des humains, des vampires et Buffy. Nous devons gérer les différentes actions que peut choisir chaque entité et ajouter une boucle de jeu qui permet de faire tourner cette simulation.

Compilation

Pour build :

```
cmake . -Bbuild && cmake --build build/ -j 8
```

Pour run :

```
./build/buffy
```

Structure

En annexe du rapport. UML

Choix d'implémentation

Utilisation des pointeurs intelligents

Nous utilisons les pointeurs intelligents pour la plupart de notre projet. Cela nous a permis de ne pas nous ennuyer avec la libération de mémoire ou de l'allocation dynamique des *Humanoid*. Ils ont étés particulièrement utiles pour gérer les *Action* et toutes les références qu'elles utilisaient.

Aléatoire

Nous avons décidé d'implémenter une classe *Random* qui s'occupe de nous procurer des numéros aléatoires entre deux bornes ainsi que des positions aléatoires. Nous avons décidé d'encapsuler cette logique dans une classe avec des fonctions statiques afin de nous permettre de facilement y faire appel, peu importe le contexte, sans devoir recréer un générateur aléatoire à chaque fois.

Recherche du plus proche humanoïde

Pour chercher un type d'humanoïde spécifique, il faut faire une vérification de type afin de savoir si l'humanoïde qui est proche de nous est bien du type recherché. Pour cela nous avons décidé d'utiliser un *enum*.

Une de nos idées de bases consistait à utiliser `dynamic_cast<>()` afin de faire de la vérification de type dynamique, mais, en se renseignant sur internet, nous avons découvert que cette fonction faisait appel au RTTI et bloquait l'arbre d'héritage des classes que nous voulions tester ce qui fait que cette solution semble plutôt boudée en général. Cependant, ces contraintes nous paraissaient proches de celles qui seraient présentes si nous utilisons un *enum* ce qui nous a poussé à l'utiliser plutôt que `dynamic_cast<>()`.

Exemple d'implémentation avec `dynamic_cast<>()` :

```
template <typename T>
std::shared_ptr<T> Field::getClosest(const std::shared_ptr<Humanoid>& from)
const {
    size_t minDist = std::numeric_limits<size_t>::max();
    std::shared_ptr<T> minHum;

    for (const auto& hum : _humanoids) {
        const auto& humCast = dynamic_cast<T>(hum);
        if (humCast != nullptr) {
            size_t dist = from->getPosition().distance(hum->getPosition());
            if(dist < minDist){
                minDist = dist;
                minHum = humCast;
            }
        }
    }
    return minHum;
}
```

Action de mouvement

Le mouvement aléatoire *MoveRandomAction* hérite de la même action *MoveAction* que *ChaseAction*. *MoveAction* permet de se déplacer en direction d'une position donnée. Dans *ChaseAction*, on donne la position de l'humanoïde que l'on essaie de pister. Dans *MoveRandomAction*, nous choisissons une position aléatoire, grâce à notre classe *Random*, puis nous la "pistons" afin de nous déplacer dans une direction aléatoire.

Tests

Test	Description	État
Affichage de la simulation	Vérification que la simulation s'affiche correctement	PASS
Touche "n" pour faire avancer la simulation	Vérifie que la simulation avance bien d'une action quand l'on appuie sur la touche "n"	PASS
Touche "q" pour quitter la simulation	Vérifie que la simulation se quitte quand on appuie sur "q"	PASS
Touche "s" lance des simulations pour faire des statistiques	En appuyant sur "s", on lance 10'000 simulations puis on affiche les statistiques du tout	PASS
Buffy court après les vampires	Buffy piste en permanence le vampire le plus proche d'elle	PASS
Buffy tue les vampires à moins d'une case d'elle	Si Buffy est assez proche d'un vampire, elle le tue	PASS
Les vampires pistent les humains	Chaque vampire piste l'humain le plus proche de lui	PASS
Les vampires mordent les humains	Si un vampire atteint un humain, il le mord et celui-ci meurt ou se transforme	PASS
Les humains se déplacent de manière aléatoire	Un humain se déplace aléatoirement	PASS
Si aucun humain n'est vivant, les vampires s'endorment	Quand tous les humains ont étés tués ou transformés, les vampires s'endorment	PASS

Win rate

Avec les paramètres indiqués dans le readMe du laboratoire, nous avons obtenu un taux de victoire de Buffy de plus ou moins **69%** sur 10'000 simulations.