INF3034

Projet

Aurélien TEXIER

Année scolaire 2018-2019

06/11/2018

Jeu de combat

Le but de cet exercice est de créer un petit jeu où s'affronteront deux combattants. Le combat se fera à tour de rôle : Le premier combattant fera son action ce sera ensuite le tour du joueur 2, puis retour au joueur 1, etc … Chaque combattant se battra avec une arme.

La liste des membres et méthodes indiqués dans le sujet n'est pas exhaustive. Certains ont même été consciemment omis, à vous d'en ajouter pour le bon fonctionnement du programme.

Chaque classe sera développée dans un duo de fichier .cpp/.h. Il n'y aura donc qu'une seule classe par header. Si vous avez besoin de structure, enumérateur ou union, vous pouvez les faire dans les header des classes ou créer un fichier utils.h.

Les armes

Vous allez devoir modéliser les armes qui seront utilisées par vos combattants. Pour cela, créer une classe *CWeapon* définie comme-ci :

- Membres :
  - Un nom (qui NE pourra PAS être identique à une autre arme) ;
  - Un nombre entier *m_damage* définissant les dégâts de l'arme ;
  - Un bonus entier de vie potentiel *m_bonus* pour son porteur.
  - Un nombre entier *m_criticalStrike* entre 0 et 100 indiquant la probabilité de faire un coup critique (doubler les dégâts) avec l'arme.
- Méthodes :
  - Une méthode *bool criticalStrike()* indiquant s'il y a coup critique ou non durant l'attaque avec cette arme.

Chaque arme sera soit une épée (*CSword*) un arc (*CBow*) ou un bâton (*CStaff*).

L'épée

L'épée (*CSword*) est une arme particulière. En effet, elle s'use au cours des combats et peut être détruite et donc inutilisable si le combat s'éternise. Elle sera modélisée ainsi :

- Membres :
  - Un entier *m_durability* indiquant la "vie" de notre arme.
  - Une constante *m_durabilityMax* indiquant la "vie" de départ de notre arme.
- Méthodes :
  - Une méthode *void use()* qui diminuera la durabilité de l'arme d'un nombre aléatoire entre 1 et 5 (la durabilité de l'arme ne pourra cependant pas être négative).
  - Une méthode *bool isBroken()* indiquant si l'épée est cassée (et donc inutilisable) ou non.

L'arc

L'arc (*CBow*) est l'arme des archers. Contrairement à l'épée il ne s'use pas mais il a besoin de flèches pour être utilisé (oui, c'est mieux). Il sera donc modélisé de la sorte :

- Membres :
  - Un entier *m_nbArrow* indiquant le nombre de flèches restantes dans le carquois.
- Méthodes :
  - Une méthode *void shoot()* diminuant de 1 le nombre de flèche disponible.
  - Une méthode *bool haveArrows()* indiquant s'il reste des flèches dans le carquois (et donc si l'arc est utilisable ou non).

Le bâton

Le bâton (*CStaff*) permet à un magicien de lancer des sorts. En plus des caractéristiques communes à toutes les armes, le bâton demande une consommation de mana pour être utilisée. Il possèdera donc juste un membre entier *m_manaCost* indiquant la quantité de mana requis pour être utilisé.

Le personnage

Une fois vos armes créées, il vous faudra modéliser vos différents personnages. Pour cela, il vous faudra dans un premier temps modéliser le personnage. Pour cela, créer une classe `CCharacter` définie de la sorte :

- Membres :
  - Un nom (qui NE pourra PAS être commun à un autre personnage) ;
  - Un nombre entier `m_hpMax` indiquant les points de vie maximum (au début du combat) du personnage ;
  - Un nombre entier `m_hp` désignant les points de vie courant du héros ;
  - Une arme `m_weapon` (comme vu dans la partie précédente) ;
  - Une valeur réelle `m_dodge` indiquant la probabilité pour le héros d'esquiver l'attaque de son adversaire.
- Méthodes :
  - Constructeurs et destructeurs ;
  - Une méthode `void action()` qui sera l'action que fera le personnage à chacun de ses tours ;
  - Une méthode `void unarmedAttack(CCharacter p_enemy)` permettant d'attaquer un ennemi à mains nues (la valeur de l'attaque sera définie plus tard) ;
  - Une méthode `void armedAttack(CCharacter p_enemy)` permettant d'attaquer un ennemi avec son arme (la valeur et les conditions de l'attaque seront définies plus tard) ;
  - Une méthode `void applyDamage(int p_damage)` appliquant les dégâts reçus ;
  - Une méthode `bool dodge()` indiquant si le héros a réussi ou non à esquiver l'attaque de son adversaire.
  - Une méthode `bool isAlive()` indiquant si le personnage est toujours en vie ou non.

Chaque personnage pourra être un guerrier (`CWarrior`), un archer (`CArcher`) ou un magicien (`CMage`). Bien entendu chaque type de personnage a ses facultés qui lui sont propres.

Le guerrier

Le guerrier (`CWarrior`) est un personnage ne pouvant manier que des épées mais possède un bouclier lui permettant de se protéger. En plus de ses caractéristiques héritées, il possède une valeur entière `m_attack` définissant l'attaque de notre héros ainsi qu'une seconde valeur entière `m_shield` étant la valeur défensive de son bouclier. Lors de son tour (`action()`), le guerrier pourra choisir d'attaquer (`attack()`), de se défendre (`void defend()`), ou de tenter de réparer (`void repair()`) son épée (oui il est aussi un peu forgeron).

L'attaque du guerrier (et donc les dégâts qu'il inflige à son adversaire) est calculée de la manière suivante :

Non armé : `m_attack * 0.66`
Armé : `(m_attack + m_damage) * 0.75`

Comme dis précédemment, durant son tour, le guerrier peut choisir de se défendre ou de réparer son épée.

- Concernant la défense, le guerrier pourra être protégé par son bouclier pendant un nombre de tour aléatoire compris entre 1 et 3.
  Dans le cas où le guerrier est protégé, les dégâts qu'il reçoit seront réduit de `0.25 * m_shield`.
- Pour ce qui est de la réparation de son arme, cela aura pour effet de rendre à son arme une valeur de durabilité comprise entre 3 et 15 si l'arme n'est pas cassée ou seulement 1 si elle est cassée.

L'archer

L'archer (`CArcher`) quant à lui est un héros utilisant des arcs. Sa puissance est définie par une valeur réelle `m_agility`. Lors de son tour, l'archer a la possibilité d'attaquer (avec un arc ou à main nues s'il n'a plus de flèche) mais aussi de viser (`void aim()`).

L'attaque de l'archer est calculée de la manière suivante :

Non armé : `m_agility * 0.50`
Armé : `(m_agility + m_damage) * 0.90`

La faculté de viser que possède l'archer lui permet de se concentrer et donc d'être plus précis lorsqu'il décochera sa flèche. A chaque fois qu'il utilise cette capacité, son agilité est multipliée par `1.33`. Bien entendu l'agilité de l'archer retourne à son état initial une fois sa flèche décochée et n'a pas d'influence s'il utilise son attaque à main nues.

Le magicien

Finissons avec le magicien maniant les bâtons.

Sa puissance est déterminée par son intelligence (`m_intellect`). Lorsqu'un magicien attaque à mains nues, les dégâts qu'il donne sont égaux à un tiers de son intelligence alors que lorsqu'il attaque avec son bâton, ceux-là sont égaux à :

```
(m_intellect + m_damage + 0.5*m_manaCost) * 1.10.
```

Le magicien possède donc une réserve de mana qui lui permet d'utiliser son bâton. Lors de son tour, le magicien a le choix entre deux actions : attaquer ou régénérer son mana. S'il choisit cette dernière, il pourra le régénérer à hauteur d'une valeur aléatoire comprise entre 2 et 7.

Le jeu

Une fois vos classes implémentées, il reste à implémenter le jeu. Pour cela il vous faudra créer deux personnages parmi la liste présente dans le fichier ***character.txt*** de manière aléatoire ou non. Il faudra à chacun leur affecter une arme parmi celle présente dans le fichier ***weapon.txt***. Vous pouvez (c'est fortement recommandé☺) créer une classe permettant le parsing de vos fichiers texte.

Les fichiers se présentent sous le format suivant :

character.txt

- Type de personnage
- Nom du personnage
- Nombre de point de vie max
- Valeur d'esquive du personnage
- Caractéristique(s) propre(s) au type de personnage (intelligence et mana pour le mage, agilité pour l'archer, …)
- Type de personnage suivant
- …

weapon.txt

- Type d'arme
- Nom de l'arme
- Dommages de l'arme
- Bonus
- Valeur de coup critique
- Caractéristique propre au type d'arme
- Type d'arme suivante
- …

Une fois les protagonistes choisis et prêts, le combat commence. Tour à tour, chaque personnage peut choisir l'action qu'il va réaliser (en fonction de sa classe), puis se sera au tour de son adversaire et ce jusqu'à la mort d'un des deux héros.

Une fois un vainqueur proclamé, le jeu s'arrête et propose de quitter le jeu ou de relancer une nouvelle partie.

Améliorations

Une fois le jeu fonctionnelle, à vous d'imaginer des améliorations possibles pour celui-ci, de modifier les différents calculs de dégât, réfléchir à une IA, etc…