Durée : 3h

Examen d’algorithmique

Jeu de cartes

1/ Combat de monstres

Nous estimons d’abord que chaque joueur ou joueuse possède un seul et unique monstre. Ce dernier possède un score d’attaque et des points de vie. Un monstre peut attaquer un autre monstre, ce qui lui permet de lui faire perdre autant de points de vie que son score d’attaque ; cela lui fait perdre aussi autant de points de vie que le score d’attaque de son adversaire. Quand un monstre attaque, il est épuisé et ne pourra plus être utilisé jusqu’à la fin du tour. Quand un monstre arrive à 0 points de vie ou moins, il est détruit. Un monstre possède aussi une méthode qui lui permet de s’afficher à l’écran (on voit alors son nom, ses points de vie, son score d’attaque, sa disponibilité c’est-à-dire s’il est épuisé ou non).

Faire le diagramme UML de la classe Monstre.
La déclarer, l’implémenter (Monstre.h et Monstre.cpp), la tester.

Durée : 3h

Examen d'algorithmique

2/ Combat de mages

Améliorez votre programme en créant une classe Mage, qui représente le joueur ou la joueuse. Chaque mage possède sa propre zone de jeu, un nom, ainsi que des points de vie.

Un mage peut jouer un nouveau monstre (pour le moment, en entrant son nom et ses caractéristiques au clavier), attaquer un monstre de son adversaire avec l'un des siens, ou attaquer son adversaire avec l'un de ses monstres. Un mage possède aussi sa propre méthode d'affichage, qui lui permet d'afficher son nom, ses points de vie, ainsi que chacune des cartes de sa zone de jeu.

Faire le diagramme UML de la classe Mage.
La déclarer, l'implémenter (Mage.h et Mage.cpp), la tester.

3/ Pioche et invocation

Les monstres ne sont pas créés librement. Ils sont piochés depuis le deck et arrivent ainsi dans la main de leur Mage propriétaire, qui pourra ensuite les poser dans la zone de jeu. Il dépensera alors autant de points de Mana que leur score d'invocation.

En début de tour, un Mage regagne tous ses points de Mana, et pioche une carte. Quand il achève son tour, c'est à son adversaire de jouer.

Faire le nouveau diagramme UML de la classe Mage.
Déclarer, implémenter et tester les modifications dans Mage.h et Mage.cpp.

4/ Sorts de dégâts et de soins

On souhaite ajouter des sorts de dégâts et des sorts de soins, qui sont piochés dans le même deck que les monstres, mais ne passent pas par la zone de jeu quand ils sont joués ; à la place, ils affectent directement les points de vie des Mages ou des Monstres.

Comment peut-on s'y prendre ?

Modéliser en UML les changements, puis les mettre en œuvre.

Selene Tonon
ETPA Rennes