[A1] Rattrapage JAVA

Juillet 2017

Contexte

Il s'agit de développer un mini jeu de combat en interface graphique dans lequel un joueur affronte une IA plus ou moins habile. Le joueur peut choisir son personnage parmi trois classes : Healer, Tank et Damager.
Chaque catégorie a ses propres statistiques et un pouvoir spécial.

Fonctionnalités demandées

→ Le jeu doit être dans une fenêtre graphique.
→ Le joueur doit pouvoir choisir la classe de son personnages (Healer/Tank/Damager).
→ Le mécanisme de choix du personnage pour l'IA est libre, les personnages possibles sont les mêmes.
→ Le joueur doit pouvoir choisir le niveau de l'IA (Alea/Normal/Expert)
→ Une partie doit pouvoir être jouée et se terminer lorsque l'un des deux joueurs au moins n'a plus de point de vie.

Contraintes

→ Projet en solo (ça veut dire tout seul).
→ L'ensemble du code et des commentaires doit être en anglais.
→ L'utilisation de GitHub doit être faite tout au long du développement.
→ **Vous devez fournir un diagramme de classes (UML) correspondant à votre réflexion en amont et un diagramme de classes (UML) revu une fois le développement terminé**.
→ **Vous devez fournir un accès à un répertoire GitHub avec l'ensemble du code source.**
→ Soutenance du projet avec démonstration le 11 juillet (5 minutes de démo + 5 minutes de questions)
→ **[BONUS]** Changer les *sprites* en fonction de l'action choisie ou du nombre de PDV.
→ **[BONUS]** Ajouter la possibilité de stocker les scores pour comparer l'efficacité des différents personnages et des différentes IA.
→ **[BONUS]** Ajouter d'autres catégories de personnages avec d'autres statistiques et pouvoirs.

Règles du jeu

Deux personnages s'affrontent en manches successives jusqu'à ce que mort s'en suive (Point de vie tombant à 0). Lors d'une manche, les deux joueurs (Humain et IA) choisissent une action, elles sont ensuite résolues simultanément et la manche se termine. Si les deux personnages sont encore vivant, le jeu continue et passe à la manche suivante.

Actions possibles :

Attaquer : Inflige un nombre de dégât égal à la force du personnage attaquant.
Bloquer : Réduit à 1 l'attaque infligée par l'adversaire quelle que soit sa force.
Pouvoir spécial : Spécifique à la classe du personnage.

Damager :

Point de vie : ♡♡♡ (3)
Force d'attaque : ♢♢ (2)
Pouvoir spécial : Inflige les dégâts reçus.
→ *En choisissant son pouvoir spécial, ce personnage répercute tous les dommages reçus sur son adversaire (mais il les subit également).*

Healer :

Point de vie : ♡♡♡♡ (4)
Force d'attaque : ♢ (1)
Pouvoir spécial : Soin +♡♡
→ *Le personnage récupère deux points de vie.*

Tank :

Point de vie : ♡♡♡♡♡ (5)
Force d'attaque : ♢ (1)
Pouvoir spécial : Attaque puissante $-\heartsuit \Rightarrow +1\diamondsuit$
→ *Effectue une attaque infligeant 1 dégât de plus donc 2 dégâts contre un personnage qui ne bloque pas, en contrepartie il perd un point de vie. Aucun dommage supplémentaire si l'adversaire bloque l'attaque.*

Graphismes réalisés par MagicFred

Pas à pas

Proposition de déroulement pas à pas, que vous pouvez suivre ou non.

1. Réflexion papier-crayon sur la conception du jeu
2. Premier diagramme de classes prévisionnel (UML)
3. Réaliser un algorithme du jeu en identifiant les différentes phases :
   a. Permettre le choix du personnage
   b. Permettre le choix du niveau de l'IA
   c. Tant que les deux personnages sont en vie :
      c.1. ...
      c.2. ...
4. En s'appuyant sur l'algorithme mis en place, écrire la fonction *main*.
5. Développer en chaîne les classes et fonctionnalités nécessaires au fonctionnement du jeu dans une première version minimaliste :
   a. Le *main* instancie le jeu et appel une méthode *play*.
   b. Création d'une classe *Game* avec entre autre un méthode *play* + Instanciation du *World* et des *Players*.
   c. Création d'une classe *World*.
   d. Création d'une classe *Player*.
   e. ...
   f. ...
6. Réflexion sur les stratégies du joueur IA.
7. Première grosse étape de validation
8. Améliorer les fonctionnalités et implémenter toutes les règles attendues.
9. Tests.
10. Diagramme de classes final (UML)