420-5W5 – Sprint 2

Projet

Super Cartes Infinies (aka Super HeartStone Infini)

Date de remise : Semaine 11

# L’application

## **Objectifs** :

Ajouter des fonctionnalités au jeu de cartes développé lors du Sprint 1.

## Note importante :

Si vous n’avez pas réussi à compléter une partie du TP1, regardez avec votre enseignant pour savoir comment faire le TP2 sans devoir travailler en double (finir d’abord le TP1 pour pouvoir ensuite faire le TP2).

## Les nouvelles règles de jeu :

* Chaque carte prend un tour avec d’être activé (Ce que l’on appelle habituellement le « summoning sickness »).
  + Cependant, elle peut déjà se défendre dès qu’elle est en jeu.
* Jouer plusieurs cartes par tour, selon un coût en « Mana ».
  + Chaque carte à un coût en Mana différent.
  + Lorsque le Mana n’est pas utilisé, il s’accumule pour le tour d’après.
  + On doit pouvoir configurer combien de Mana les joueurs reçoivent chaque tour. Il faut ajouter un event GainManaEvent pour que les joueurs recoivent du Mana à chaque tour. Juste avant de piger leur nouvelle carte pour le tour. **Optionnel**: Vous pouvez donner un peu moins de mana au premier tour du premier joueur pour réduire l’avantage de jouer en premier.
  + Il faut ajouter un bouton « terminer » qui permet de terminer son tour lorsque l’on ne veut plus jouer de cartes. Il faut également modifier l’action PlayCard sur le serveur et ajouter une action EndTurn qui permet de terminer son tour.
* Ajouter des habilités à certaines cartes. On pourra donc associer une liste d’habiletés à nos cartes en MVC. (On parle ici de la classe Carte). La liste d’habiletés disponible n’a pas besoin d’avoir son propre UI et vous pouvez simplement utiliser un seed. Les habiletés doivent avoir :
  + Un nom
  + Une icône
  + Une valeur entière (optionnelle)
* Il faut pouvoir voir les habiletés d’une carte sur le client
* Il faut faire une animation lorsqu’une habileté est utilisé sur le client. Vous pouvez simplement montrer l’icône de l’habileté au-dessus de la carte pour un moment ou faire quelque chose de plus complexe si vous préférez.
* Mélanger les cartes dans le CardsPile pour ne pas toujours commencer avec les mêmes cartes.
* Il y a **4 habiletés (Power)** que vous devez ajouter et vous devez en créer **2 autres à votre choix**.
  + « **First Strike** » permet à une carte **d’attaquer** en « premier » et de ne pas recevoir de dégât si elle tue la carte de l’ennemie. (Fonctionne uniquement à l’attaque, pas à la défense)
  + « **Thorns X**» lorsqu’une carte **défend**, elle inflige X de dégâts AVANT de recevoir des dégâts. Si l’attaquant est tué par ces dégâts, l’attaque s’arrête et le défenseur ne reçoit pas de dégâts.
  + « **Charge** » permet à une carte d’être activé le même tour où elle est jouée (donc de faire comme c’était le cas dans le tp1).
  + « **Heal X**» soigne les cartes alliées de X incluant elle-même AVANT d’attaquer (mais les cartes ne peuvent pas avoir plus de points de vue qu’au départ.)
* Note sur les habiletés à votre choix. Pour l’instant, les pouvoirs proposés ne nécessitent pas de garder un « état ». C’est conseillé de choisir des habiletés sans état pour le TP2. Lors du TP3, nous allons exiger des pouvoirs avec un état. Un exemple d’habiletés avec état, c’est « Poison X » qui inflige des dégâts à chaque tour. Dans ce cas, il faut pouvoir ajouter un état « empoisonné » sur CardInstance. (Si vous n’arrivez pas à trouver des idées d’habiletés simples à ajouter, faites signe à votre prof.)

Le détail des sections

## Section d’administration

* L’administrateur peut ajouter et retirer des cartes en ventes au magasin, ainsi que de modifier leur prix.
* L’administrateur peut choisir le prix rachat des cartes (le prix lorsqu’un joueur vend une carte).
* L’administrateur peut modifier les contraintes des « Decks ». Il peut configurer le nombre de « Decks » maximum d’un joueur et le nombre de carte dans chaque « Deck ».
* L’administrateur peut choisir la quantité de monnaie virtuelle du jeu qu’un joueur gagne à chaque partie pour une victoire et pour une défaite.

## Deck

* Un utilisateur doit créer au moins un Deck (paquet de cartes).
* Un deck doit avoir un nom
* Un utilisateur doit choisir quel Deck il apporte dans la partie. Pour la sélection du deck, utiliser la première carte du Deck comme image en plus du nom.
* Seul les cartes du Deck sont disponibles lors d’une partie.
* Une même carte peut faire partie de plusieurs decks.

## Magasin

* Les joueurs peuvent acheter et vendre des cartes avec une monnaie virtuelle dans le jeu.
* Les cartes peuvent être achetées et vendues à l’unité.

## Partie

* À chaque partie, un joueur gagne de la monnaie virtuelle du jeu.
  + Une victoire donnera plus de monnaie qu’une défaite.
* Il sera possible d’**abandonner** une partie (Surrender).
* Il sera possible d’**abandonner** la file d’attente pour jouer un match. (Mais si la partie avait déjà commencé, c’est trop tard!)
* Lors de l’abandon, pas besoin de gérer les « race conditions » pour l’instant. Au TP3 on va utiliser des sémaphores pour protéger ce genre de situations.

Contraintes et TDD

* Il faudra modifier les informations envoyées aux clients (JSON) à propos des parties pour ne pas révéler les cartes qui sont dans la main de notre adversaire.
* Il faut ajouter des tests unitaires pour les nouveaux services pour la gestion des decks et l’achat et la vente des cartes.
* Les nouvelles règles devront être programmées avec le principe TDD (Test Driven Development).
* Les tests doivent être écrit par une personne et l’implémentation de la logique doit être fait **par une autre personne**. Quand vous allez remettre votre partie pour le TP2, vous pourrez vous faire évaluer pour avoir écrit les tests **OU** pour avoir écrit la logique de combat!
* Au Sprint 1, vous aviez déjà les tests présents pour le combat, vous devrez maintenant les réaliser par vous-même. Il faut **modifier les 3 tests de combats existants** pour supporter les changements de règles.
* Vous pouvez en ajouter PLUS, mais voici une liste des tests **obligatoires** :
  + Impossible de jouer une carte si un joueur n’a pas assez de Mana
  + Possible de jouer 2 cartes le même tour
  + Possible de jouer 0 carte pendant un tour
  + Test de l’habileté First Strike
  + Test de l’habileté Charge
  + Test de l’habileté Thorns X
  + Test de l’habileté Heal X
  + Test de l’habileté Heal X partiel (pas possible qu’un CardInstance est plus de health que sa Carte)
  + Test de votre habileté au choix 1
  + Test de votre habileté au choix 2

Tâches individuelles

Vous devez choisir une des tâches suivantes qui sera évaluer de façon individuelle. Vous devez écrire vous-même le code, mais vous pouvez collaborer avec vos collègues.

**IMPORTANT**: Vous devez choisir quelque chose de différent du TP1. Donc vous ne pouvez pas reprendre la logique de combat si vous l’aviez déjà fait pour le premier TP, par exemple.

* **Écrire les tests pour le combat TDD**. Voir la page précédente pour la liste de tous les tests requis.
* **Implémenter la logique pour le combat**. Il faut faire passer tous les tests et valider que tout fait du sens.
* **Achat et vente des cartes Angular.**
* **Toute la partie MVC.** Achat et vente des cartes et autres configurations (nombre de cartes par deck, nombre de decks, nombre de monnaie virtuelle à la création du compte, après une partie perdue et une partie gagnée.)
* **Gestion des decks Angular**. Incluant la sélection du deck avant de joindre un match.
* **Tous les services.** Ajouter les contrôleur, actions et services pour la gestion des decks et la vente et l’achat des cartes. Avec des tests unitaires.
* **Traitement des évènements Angular. ATTENTION :** Évitez de prendre cette option si votre jeu n’est pas déjà dans un très bon état après le TP1.