ENSEA
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Projet Logiciel Transversal

Anaïs COLIN – Vincent LAMBRETCHS

DuelMaster

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

Table des matières

ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications

1 Objectif

1.1 Présentation générale

Présenter ici une description générale du projet. On peut s'appuyer sur des schémas ou croquis pour illustrer cette présentation. Éventuellement, proposer des projets existants et/ou captures d'écrans permettant de rapidement comprendre la nature du projet.

L'objectif de ce projet est la réalisation d'un jeu de cartes à collectionner de type « HearthStone » sur la base d'un jeu de rôle de type « Pokémon » . Le joueur incarne un personnage qui évolue dans un monde (partie Pokémon) divisé en 4 royaumes, représentant chacun un élément, et affronte des personnages par l'intermédiaire d'un jeu de carte (partie HearthStone) pour conquérir le monde.

1.2 Règles du jeu

Présenter ici une description des principales règles du jeu. Il doit y avoir suffisamment d'éléments pour pouvoir former entièrement le jeu, sans pour autant entrer dans les détails . Notez que c'est une description en « français » qui est demandé, il n'est pas question d'informatique et de programmation dans cette section.

❖ *But du jeu*

Le joueur incarne un personnage avec un deck et un pouvoir héroïque donnés. Son but est de conquérir les 4 royaumes et de devenir le « Maitre des cartes ». Il devra pour cela battre les 4 princes au DuelMaster, un jeu de carte à collectionner. Sur son chemin, plusieurs chevaliers viendront le défier.

Il existe 3 modes de jeu :

- **Mode Observation :** un joueur artificiel évolue tout seul et montre les fonctionnalités du jeu.
- **Mode Découverte :** (mode classique) le joueur évolue sans contrainte et seul sur la carte.
- **Mode Rival :** un joueur artificiel évolue en même temps que le joueur. Le premier à devenir « Maître des cartes » a gagné.

❖ *Univers du jeu*

Le jeu se décompose en 2 sous-jeux :

➢ *Partie Exploration*

Le joueur se déplace sur une carte en 2D fixe vue du dessus contenant des montagnes, des plans d'eau, des plaines ainsi que des maisons et un château par royaume. Ces décors n'influencent en aucun cas la façon de jouer : ce ne sont que des obstacles. Le monde est divisé en 4 royaumes représentant chacun un élément (feu, etc.) et régit par un prince expert en DuelMaster. Toutefois, il existe des personnages non joueur qui pourront affronter le joueur ou simplement interagir avec lui.

➢ *Partie Duel*

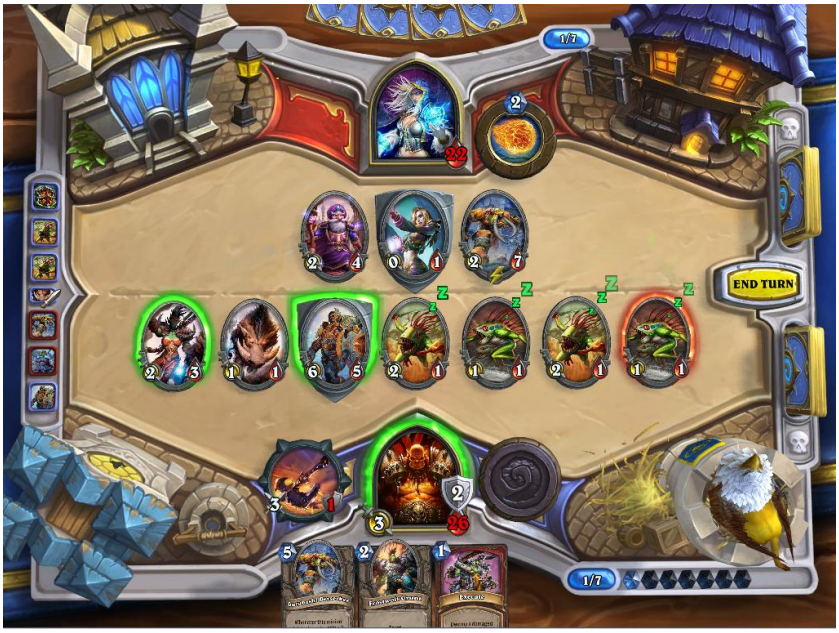
Les duels se déroulent sous forme d'un jeu de cartes tour à tour appelé DuelMaster : chacun des personnages détient un deck préparé au préalable et un nombre de points de vie (=PV). Le but du combat est de réduire les PV de l'adversaire à 0.

Pour cela, les joueurs utilisent les cartes de leur deck : chacune de ces cartes dispose de caractéristiques propres à elle-même et un coût en points de combat (=PC).

Il existe 3 zones de jeu contenant des cartes :

- **La main :** chaque joueur dispose d'un ensemble de cartes visibles et jouables lors d'un tour
- **La pioche :** il s'agit d'un tas de cartes non visibles. A chaque début de tour, le joueur pioche une carte dans **sa** pioche. Une fois la pioche épuisée, le joueur à qui elle appartient perd des PV à la place de piocher.
- **Le plateau :** pour prendre effet, une carte doit être posée sur le plateau de jeu

Il existe une banque de PC qui autorise une ou plusieurs actions. A chaque début de tour, un PC supplémentaire est attribué jusqu'à un maximum.



❖ *Evolution*

Tout au long du jeu, le joueur peut améliorer son deck. Pour cela, il existe plusieurs façons d'obtenir de nouvelles cartes :

1) Se battre contre des joueurs ou des princes
2) Gagner de l'expérience, celle-ci s'incrémente au fur et à mesure des cartes jouées
3) Remplir des quêtes

Une fois les 4 royaumes conquis, le joueur est nommé « Maître des cartes » , gagne la partie et débloque un nouveau personnage détenant un nouveau deck et un nouveau pouvoir héroïque.

1.3 Conception Logiciel

Présenter ici les packages de votre solution, ainsi que leurs dépendances.

2 Description et conception des états

L'objectif de cette section est une description très fine des états dans le projet. Plusieurs niveaux de descriptions sont attendus. Le premier doit être général, afin que le lecteur puisse comprendre les éléments et principes en jeux. Le niveau suivant est celui de la conception logiciel. Pour ce faire, on présente à la fois un diagramme des classes, ainsi qu'un commentaire détaillé de ce diagramme. Indiquer l'utilisation de patron de conception sera très apprécié. Notez bien que les règles de changement d'état ne sont pas attendues dans cette section, même s'il n'est pas interdit d'illustrer de temps à autre des états par leur possibles changements.

2.1 Description des états

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

Illustration 1: Diagramme des classes d'état

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémente pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

Illustration 2: Diagramme de classes pour le rendu

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects lié au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

Illustration 3: Diagrammes des classes pour le moteur de jeu

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est une parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

Illustration 4: Diagramme de classes pour la modularisation