

Projet Logiciel Transversal

Anaïs COLIN – Vincent LAMBRECHTS

DuelMaster

Table des matières

1 Objectif.....	3
1.1 Présentation générale.....	3
1.2 Règles du jeu	3
1.3 Conception Logiciel	3
2 Description et conception des états	4
2.1 Description des états.....	4
2.2 Conception logiciel	4
2.3 Conception logiciel : extension pour le rendu.....	4
2.4 Conception logiciel : extension pour le moteur de jeu	4
2.5 Ressources	4
3 Rendu : Stratégie et Conception.....	6
3.1 Stratégie de rendu d'un état	6
3.2 Conception logiciel	6
3.3 Conception logiciel : extension pour les animations.....	6
3.4 Ressources	6
3.5 Exemple de rendu	6
4 Règles de changement d'états et moteur de jeu	8
4.1 Horloge globale	8
4.2 Changements extérieurs	8
4.3 Changements autonomes.....	8
4.4 Conception logiciel	8
4.5 Conception logiciel : extension pour l'IA.....	8
4.6 Conception logiciel : extension pour la parallélisation	8
5 Intelligence Artificielle	10
5.1 Stratégies	10
5.1.1 Intelligence minimale	10
5.1.2 Intelligence basée sur des heuristiques	10
5.1.3 Intelligence basée sur les arbres de recherche	10
5.2 Conception logiciel	10
5.3 Conception logiciel : extension pour l'IA composée.....	10
5.4 Conception logiciel : extension pour IA avancée.....	10
5.5 Conception logiciel : extension pour la parallélisation	10
6 Modularisation	11
6.1 Organisation des modules	11
6.1.1 Répartition sur différents threads	11
6.1.2 Répartition sur différentes machines	11
6.2 Conception logiciel	11
6.3 Conception logiciel : extension réseau.....	11
6.4 Conception logiciel : client Android	11

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 « Maître 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*

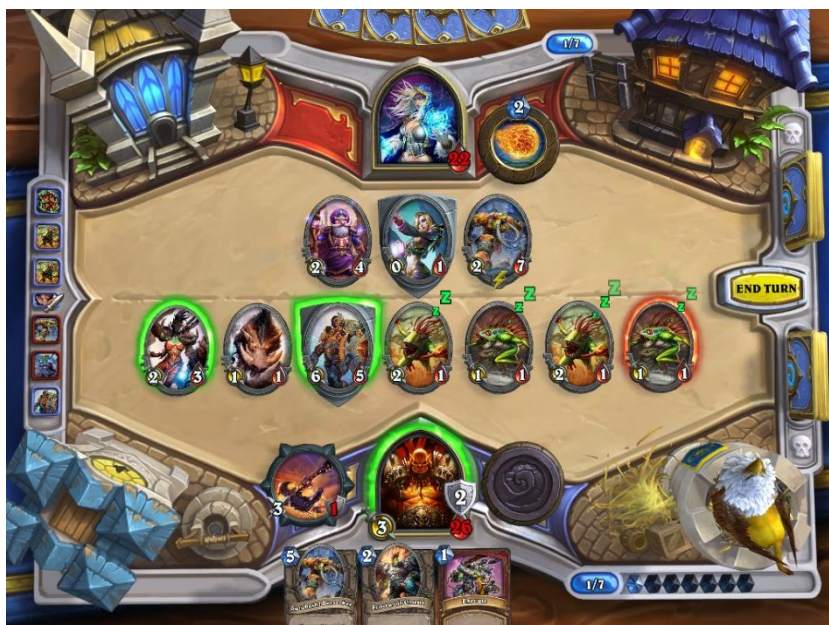
Les duels se déroulent sous forme d'un jeu de cartes tour à tour appelé DuelMaster : chacun des personnages détient un deck préparé au préalable et un nombre de points de vie (=PV). Le but du combat est de réduire les PV de l'adversaire à 0.

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 débloquent 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

Partie Exploration

Un état du jeu est formé par un ensemble d'éléments fixes et un ensemble d'éléments mobiles. Tous les éléments possèdent les propriétés suivantes :

- Coordonnées (x,y) dans la grille
- Identifiant de type d'élément : ce nombre indique la nature de l'élément (ie classe)

Partie Duel

Tous les éléments possèdent les propriétés suivantes :

- **Coût** : ce nombre indique le prix en Points de Combat à ce tour pour utiliser cet élément.
- **Attaque** : ce nombre indique le nombre de Points d'Attaque que possède cet élément.
- **Utilisation** : ce booléen est utilisé par les méthodes pour activer les effets.
- **Identifiant de type d'élément** : ce nombre indique la nature de l'élément (ie classe)

2.1.1 Etats éléments fixes

Partie Exploration

La carte est formée par une grille d'éléments nommés « cases ». La taille de cette grille est fixe.

Les types de cases sont :

- **Cases « Obstacle »** : Les cases « Obstacle » sont des éléments infranchissables pour les éléments mobiles. Le choix de la texture est purement esthétique, et n'a pas d'influence sur l'évolution du jeu.
- **Cases « Espace »** : Les cases « Espace » sont les éléments franchissables par les éléments mobiles. On considère les types de cases « Espace » suivants :

- ❖ Les espaces « vides »
- ❖ Les espaces « cadeaux », qui contiennent des cartes à récupérer

Partie Duel

Le plateau ne contient qu'un seul élément fixe nommés Case « Pouvoir ». Une Case « Pouvoir » contient le pouvoir du héros, qui est la personification du joueur.

2.1.2 Etats éléments mobiles

Partie Exploration

Les éléments mobiles possèdent une direction (aucune, gauche, droite, haut ou bas), une vitesse, et une position. Une position à zéro signifie que l'élément est exactement sur la case ; pour les autres valeurs, cela signifie qu'il est entre deux cases (l'actuelle et celle définie par la direction de l'élément). Lorsque la position est égale à la vitesse, alors l'élément passe à la position suivante. Ainsi, plus la valeur numérique de vitesse est grande, plus le personnage aura un déplacement lent. Ce système est choisi pour pouvoir être toujours synchronisé avec une horloge globale.

Les éléments sont :

- **Eléments mobiles « Personnage »** : Cet élément est dirigé par le joueur qui commande la propriété de direction qu'elle provienne d'un humain ou d'une IA.
Un personnage supplémentaire est déblocable lorsqu'une partie est gagnée
- **Eléments mobiles « Villageois »** : Cet élément est également commandé par la propriété de direction. Ils possèdent une propriété particulière appelée « status » :
 - ❖ Status « normal », cas où le villageois peut interagir avec le personnage
 - ❖ Status « combat », cas où le villageois propose au personnage un combat

Partie Duel

- **Eléments mobiles « Carte »** : Cet élément est utilisable par le joueur ou l'IA. Il possède un attribut position qui définit dans quel tableau il est contenu. Il existe quatre types de tableaux et chaque joueur possède un exemplaire de ces quatre types :
 - ❖ **Tableau « pioche »** : ce tableau contient les cartes non jouées et non jouables. Sa taille maximale est de 20.
 - ❖ **Tableau « main »** : ce tableau contient les cartes jouables et non jouées. Sa taille maximale est de 10.
 - ❖ **Tableau « terrain »** : ce tableau contient les cartes jouées. Sa taille maximale est de 7.
 - ❖ **Tableau « cimetière »** : ce tableau contient les cartes jouées qui ne sont plus présentes sur le terrain et inutilisables. Sa taille maximale est de 20.

Il existe 3 types de cartes :

- ❖ **Soldat** : un soldat détient un nombre de PV et peut avoir un effet particulier et se pose sur le terrain
- ❖ **Ordre** : un ordre détient uniquement un effet particulier et ne se pose pas sur le terrain
- ❖ **Arme** : une arme détient une durabilité et peut avoir un effet particulier et se pose sur une case particulière du terrain

2.1.3 Etat général

A l'ensemble des éléments statiques et mobiles, nous rajoutons les propriétés suivantes :

Partie Exploration

- **Epoque** : représente « l'heure » correspondant à l'état, ie c'est le nombre de « tic » de l'horloge globale depuis le début de la partie.
- **Vitesse** : le nombre d'époque par seconde, ie la vitesse à laquelle l'état du jeu est mis à jour
- **Compteur de victoire sur prince** : le nombre de victoires contre les princes (détermine l'avancée du jeu car il faut battre les 4 princes pour gagner une partie)

Partie Duel

- **Compteur Tour** : le nombre de tours joués depuis le début du duel
- **Compteur PV** : le nombre de points de vie (PV) restants ($PV=30-\text{dégâts}$)
- **Compteur Pioche** : le nombre de cartes restantes dans la pioche ($Pioche=20-\text{cartes distribuées}$)
- **Compteur PC** : le nombre de points de combat restants (disponibles) lors d'un tour ($\text{Compteur PC} = PC_{\text{dutour}} - \text{Coûts Cartes Jouées pendant le tour}$)
- **Liste des cartes** : liste complète des cartes contenues dans le jeu. Cette liste comprend les cartes sur le terrain, les cartes dans la main des joueurs, les cartes dans leur pioche respective et les cartes dans les cimetières.

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

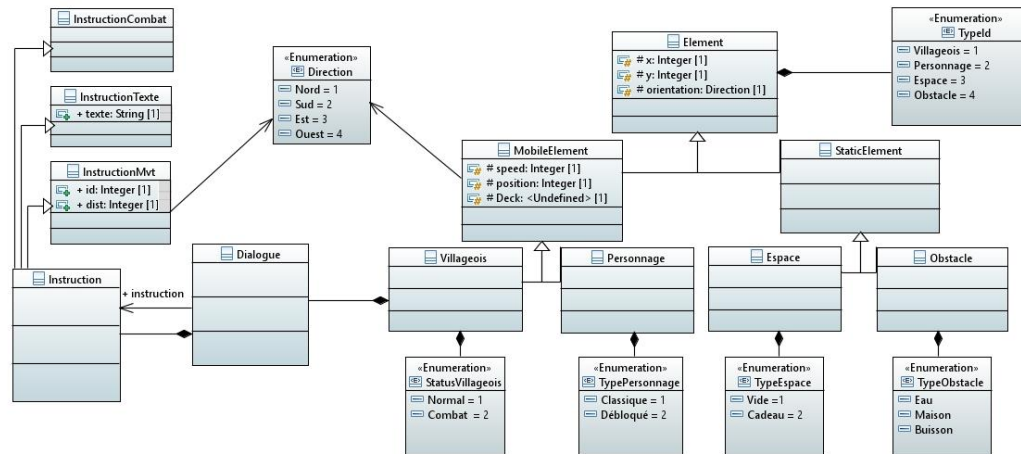


Illustration 1: Diagramme des classes d'état Exploration

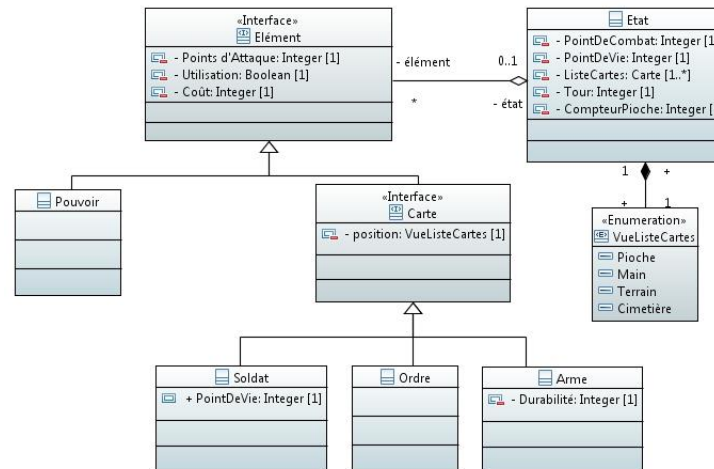


Illustration 2 : Diagramme des classes d'état Duel

4 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'implémentent 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.

4.1 Stratégie de rendu d'un état

4.2 Conception logiciel

4.3 Conception logiciel : extension pour les animations

4.4 Ressources

4.5 Exemple de rendu

Illustration 2: Diagramme de classes pour le rendu

5 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 liés 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.

5.1 Horloge globale

5.2 Changements extérieurs

5.3 Changements autonomes

5.4 Conception logiciel

5.5 Conception logiciel : extension pour l'IA

5.6 Conception logiciel : extension pour la parallélisation

Illustration 3: Diagrammes des classes pour le moteur de jeu

7 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.

7.1 Stratégies

7.1.1 Intelligence minimale

7.1.2 Intelligence basée sur des heuristiques

7.1.3 Intelligence basée sur les arbres de recherche

7.2 Conception logiciel

7.3 Conception logiciel : extension pour l'IA composée

7.4 Conception logiciel : extension pour IA avancée

7.5 Conception logiciel : extension pour la parallélisation

9 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.

9.1 Organisation des modules

9.1.1 Répartition sur différents threads

9.1.2 Répartition sur différentes machines

9.2 Conception logiciel

9.3 Conception logiciel : extension réseau

9.4 Conception logiciel : client Android

Illustration 4: Diagramme de classes pour la modularisation

