Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

Compte rendu de projet
Programmation jeu de cartes

PIQUET Florian CHARRIER Sébastien

21 septembre 2017

Sommaire

Première partie
Présentation du projet

1 Archétype du jeu

Notre projet est de réaliser une base de jeu vidéo similaire au jeu de cartes Magic : L'assemblée.

2 Règles du jeu [1]

Le jeu se joue à deux joueurs, chacun doit réduire les points de vie de l'autre à 0 en l'attaquant avec des créatures ou en lui infligeant des dégâts avec des sorts qui demandent a dépenser du mana généré par des terrains. Les créatures, sorts, terrains et invocations de chaque joueur sont représentées par des cartes que chaque joueur peut jouer depuis sa main et pioche dans sa bibliothèque.

Les cartes ont plusieurs caractéristiques :

- Un nom, différent pour chaque carte
- Un type, décrivant son comportement lorsque la carte est jouée, ainsi que les restrictions pour jouer la carte
- Un coût nécessaire a payer pour jouer la carte, sauf exceptions
- Un texte, même si il peut être vide, présentant les effets, capacités et caractéristiques d'une carte (si il y a lieu.)

Chaque joueur commence avec 20 PV, et peut en perdre ou en gagner durant la partie. Si un joueur atteint 0 PV, il perd la partie et l'autre joueur gagne, la partie s'arrête alors.

1. Les règles énumérées ici sont celles que nous allons suivre pour la réalisation de notre jeu, elles sont présentées de la manière la plus complète possible mais restent une version simplifiée des règles du jeu original, vous pouvez consulter les règles officielles au lien suivant (en anglais) : `https://media.wizards.com/2017/downloads/MagicCompRules%20170925.pdf`

Chaque carte doit être dans une zone (sauf si un effet dicte qu'elle est retirée du jeu), il en existe 5 différentes, certaines d'entre elles sont associées à un joueur, tandis que d'autres sont communes aux joueurs :

- La bibliothèque, contenant les cartes que le joueur va piocher.
- La main, contenant les cartes que le joueur peut jouer.
- La pile, commune aux deux joueurs, une fois qu'une carte est lancée ou qu'une capacité est activée ou déclenchée elle arrive dans la pile en attendant d'être résolue. Certaines cartes peuvent être jouées "en réaction" sur d'autres cartes, les cartes lancées le plus récemment se résolvent en premier (pile first in - last out), une description plus précise sera présentée plus loin.
- Le champ de bataille, commun aux deux joueurs, les cartes de types de "Permanents" arrivent sur le champ de bataille une fois résolues et y restent jusqu'à être détruites ou sacrifiées, ou qu'un autre effet les change de zone.
- Le cimetière, lorsque un permanent est détruit ou qu'un sort est résolu, celui ci va dans cette zone. Si le permanent était un jeton, il cesse d'exister, de même si le sort était une copie.

Le concept de priorité est un élément fondamental de Magic : L'assemblée qui le différencie des autres jeux de cartes similaires, c'est un principe peu intuitif mais que nous ne pouvons pas nous permettre d'ignorer ou simplifier. La priorité fonctionne ainsi :

- Les joueurs obtiennent la priorité dû à l'avancement de la partie.
- Le joueur dont c'est le tour obtient la priorité, il peut lancer des sorts ou activer des capacités.
- Si il décide de lancer un sort ou d'activer une capacité, celui ci arrive dans la pile et le joueur récupère de nouveau la priorité.
- Si le joueur ne souhaite plus lancer de sorts ni activer de capacités, il laisse la priorité.

- Le joueur suivant dans l'ordre de jeu obtient alors la priorité et peut lancer un sort ou activer une capacité.
- Si il décide de faire ainsi, le joueur dont c'est le tour récupère la priorité comme auparavant. Sinon, le joueur suivant dans l'ordre de jeu récupère la priorité.
- Lorsque tous les joueurs laissent la priorité, le dernier sort ou la dernière ajouté dans la pile se résout, et le joueur dont c'est le tour récupère la priorité.
- Si la pile est vide et que tous les joueurs laissent la priorité, la partie avance vers l'étape de tour suivante.
- Si l'étape du tour était une phase principale et que la pile est vide, le joueur dont c'est le tour peut jouer des créatures, invocations et terrains lorsqu'il a la priorité (en plus des sorts et capacités).
- Si un objet arrivant dans la pile valide les conditions d'une capacité déclenchée, celle ci arrivera dans la pile immédiatement au dessus sans que les joueurs aient la priorité entre temps. ils n'obtiendront la priorité que lorsque toutes les capacités déclenchées seront arrivées dans la pile.

Pour simplifier, on réduit le nombre de types de cartes par rapport au jeu original à 4 types de carte :

Les types suivants sont appelés "Permanents" car, une fois résolus, ils arrivent sur le champ de bataille et y restent. Ils peuvent être joués par un joueur durant une de ses phases principales et lorsque la pile est vide (aucune capacité ou carte n'attend de se résoudre), ils sont en trois types :

- Les Invocations, qui ont pour simple effet d'arriver sur le champ de bataille et d'avoir un effet passif sur le jeu, ou une capacité. (Simplification des types Enchantement et Artefact du jeu original)
- Les créatures, qui ont des caractéristiques de force et d'endurance et peuvent attaquer et bloquer lors de la phase de combat, ainsi que subir des blessures. Si la différence entre l'endurance d'une créature et le nombre de blessures qu'elle a subie pendant le tour est inférieure ou égale à 0, la créature est détruite.

– Les terrains, un joueur ne peut jouer qu'un seul terrain par tour, les terrains ne coûtent pas de mana et servent a en produire afin de payer les coûts des cartes et capacités.

On appelle le dernier type les sorts, ils ont un effet immédiat sur le jeu et peuvent être jouées à tout moment ou un joueur a la priorité.

Description des différents mots clés utilisés dans la suite de cette présentation :

Engager/Désengager : L'engagement est une propriété des permanents, qui peuvent se trouver dans deux états : engagé ou désengagé. Certaines actions, capacités ou effets demandent à engager un permanent, ce permanent passe alors dans l'état "engagé," de même pour le désengagement. Si une capacité ou action d'un permanent demande à ce qu'elle s'engage elle même, elle ne pourra pas réaliser cette action ou activer cette capacité si elle est déjà engagée.

Face cachée : Une carte est face cachée si aucun joueur ne peut voir la carte.

Piocher une carte : Le joueur retire la première carte de sa bibliothèque face cachée et la met dans sa main.

Les joueurs jouent leurs tours l'un après l'autre, un tour est structuré comme suit :

– Début du tour : le joueur dont c'est le tour "désengage" ses permanents et pioche une carte, les effets de début de tour arrivent dans la pile et les joueurs obtiennent la priorité.

– Phase principale : les joueurs ont la priorité, et le joueur dont c'est le tour peut jouer des cartes de permanents.

– Phase de combat : Elle est constituée de plusieurs étapes :

  • Étape de déclaration des attaquants : Le joueur dont c'est le tour déclare les créatures avec lesquelles il souhaite attaquer. Pour pouvoir être déclarée attaquante, une créature doit être désengagée et avoir été présente sur le champ de bataille continûment depuis le début du tour. Les joueurs obtiennent ensuite la priorité

    - Étape de déclaration des bloqueurs : Le joueur défenseur déclare les créatures avec lesquelles il souhaite bloquer, chaque créature désignée comme bloquante doit choisir la créature qu'elle bloque, plusieurs créatures bloquantes peuvent bloquer la même créature attaquante. Les joueurs obtiennent ensuite la priorité.
    - Les créatures bloquées et bloquantes s'infligent alors un nombre de blessures égale a leurs forces, les créatures non bloquées font perdre au joueur défenseur un nombre de PV égal à leur force.

– Deuxième phase principale : même chose que pour la première phase principale.

– Fin de tour : les effets de fin de tour arrivent dans la pile et les joueurs obtiennent la priorité.

– Phase de nettoyage : les effets "Jusqu'à la fin du tour" et les blessures sont retirés du jeu, le joueur actif se défausse de cartes jusqu'à sa taille de main maximale, puis le joueur suivant commence son tour.

3 Ressources

Les ressources de base sont des symboles pour les couleurs de mana ainsi que des illustrations par défaut pour les différents types de cartes que vous pouvez voir ci dessous.

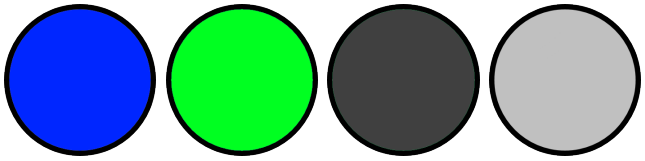

FIGURE 1 – Les différentes couleurs de mana : Bleu, Vert, Noir et Incolore.

FIGURE 2 – Images par défaut de chaque type de carte.

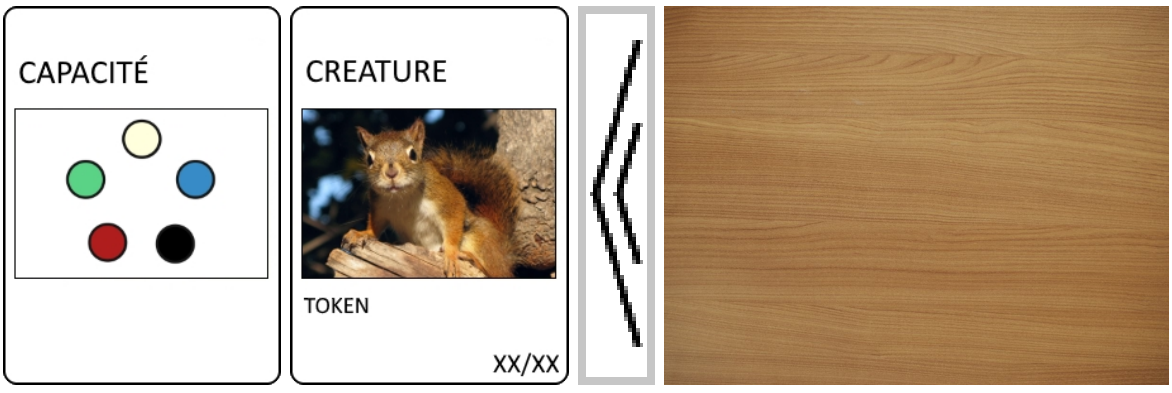

FIGURE 3 – Ressources et textures de l'interface

Deuxième partie

Description d'un état du jeu

4 L'état du jeu

On définit un état du jeu avec toutes les variables et listes définissant un état du jeu à un instant donné. Ces éléments sont séparés en deux groupes : ceux qui sont en lien avec les cartes, et ceux qui décrivent un état général de la partie

Avancement et état général : Les éléments globaux définissant l'état du jeu sont :

- Le nombre de joueurs dans la partie, entier naturel, 2 par défaut, noté N dans la suite du document.
- Le joueur actif, cad celui dont c'est le tour, entier naturel entre 1 et N.
- La phase du tour, entier entre 1 et 16.
- Le joueur qui à la priorité, entier entre 1 et N, par défaut le joueur actif est noté comme ayant la priorité.
- Le nombre de points de vie de chaque joueur, un entier naturel par joueur.
- Le fait qu'un terrain ait été joué dans ce tour, booléen.
- La quantité de mana et la couleur des manas dans la réserve de chaque joueur, N listes de 5 entiers naturels, représentant le nombre de symboles de mana de chaque couleur

Les éléments décrivant les cartes et les zones : Les cartes sont toutes réparties et ordonnées dans des zones, et peuvent subir des altérations d'états selon leur zone. Les zones et les éléments y décrivant les cartes sont :

- La bibliothèque :
  - Contient une liste des cartes présentes dans la bibliothèque du joueur (intrinsèquement leur ordre), une bibliothèque par joueur.
- Le cimetière :
  - Une liste des cartes présentes dans le cimetière du joueur (intrinsèquement leur ordre), un cimetière par joueur.
- La main :
  - Une liste des cartes présentes dans la main du joueur, une main par joueur.
- La pile :
  - Une liste des cartes et capacités dans la pile (intrinsèquement leur ordre), et des listes décrivant leurs contrôleurs et les cibles éventuelles de chaque carte.
- Le champ de bataille :
  - Une liste des cartes présentes sur le champ de bataille, et les variables associées :
    - L'attribut IsToken si la carte est un jeton, booléen.
    - Le nombre de marqueurs présents, entier relatif.
    - Les éventuels effets temporaires sur les cartes.
    - Les éventuelles blessures.

5 Notations et diagramme des classes

Tous les attributs seront privés et notés en minuscules, nous n'utiliserons pas d'arguments protégés car dia2code génère automatiquement les accesseurs et mutateurs alors que nous n'avons pas besoin, nous évitons ainsi des problèmes de lisibilité.

Toutes les méthodes seront publiques et notées en majuscules.

Organisation de l'état

Classe État : L'état est composé d'une liste d'au moins deux joueurs et s'occupe de la gestion de la priorité, des phases et des tours. Il gérera aussi les zones de la pile et du champ de bataille, et donc sera responsable des cartes en jeu et celles ayant des effets sur d'autres objets.

Classe Joueur : Un joueur comprend une réserve de mana ainsi que des ensembles d'objets, cartes et capacités, représentant les cartes que le joueur possède et les effets qu'il contrôle ainsi les zones qui leur sont affectées.

Classe ManaPool : La réserve de mana d'un joueur est une liste décrivant le nombre de manas (la monnaie du jeu) de chaque couleur que le joueur peut utiliser pour jouer des cartes ou activer des capacités.

Classe Objet : Un objet est un élément actif du jeu, pouvant changer de zone, et pouvant avoir des cibles selon son type et ses effets. Ils existent en deux types : les cartes et les capacités.

Classe Carte : Les cartes possèdent un nom, un coût, une couleur, un type, et éventuellement des capacités. Les différences entre les types de cartes étant peu nombreuses, nous les différencierons avec des booléens plutôt que de créer des classes séparées, car il n'en résulterait que des classes vides gênant la lisibilité. Seules les créatures, qui ont des caractéristiques en plus par rapport aux autres cartes, feront l'objet d'une nouvelle classe.

Classe Créature : Les créatures sont un type de carte central dans le jeu, ils possèdent des caractéristiques et suivent des règles spécifiques dû à leur propriété à pouvoir attaquer, bloquer et subir des blessures.

Classes Capacité : Les capacités sont des effets que possèdent les cartes, et qui peuvent avoir différents impacts sur le jeu. La plupart sont associées à un permanent, celles ci demandent que le joueur contrôleur de la carte paye un coût pour leur activation, elles vont ensuite dans la pile avant d'être résolues. Ces capacités peuvent être activée quand le joueur contrôleur a la priorité, et nécessitent l'engagement du permanent auquel la capacité est associée. Les autres capacités sont associées aux sorts, qui se résolvent automatiquement sans utiliser la pile. Chaque capacité sera représentée par un mot clé pour simplifier la réalisation et l'exécution du jeu.

Classe Coût : La classe coût liste les coûts qu'une carte ou capacité demandera pour être lancée, dans le jeu original ces coûts peuvent être très variés, pour simplifier nous garderons : Payer du mana depuis sa réserve, et perdre une quantité de points de vie.

FIGURE 4 – Diagramme des classes de l'état

Pour la gestion des capacités déclenchées, nous utilisons la gestion d'événements C++ [2]. Les évènements qui nous intéressent seront :

- Quand un joueur joue une carte d'un certain type.
- Quand une carte se résout.
- Quand un permanent arrive sur le champ de bataille.
- Quand un permanent quitte le champ de bataille.
- Le début du tour.
- Le début de la phase de combat.
- La fin du tour.

Classe	Effet
État	Gère la priorité et l'avancement du tour.
Joueur	Possède des cartes et des capacités
ManaPool	Liste la quantité de mana d'un joueur
Objet	Un élément qui a un effet sur le jeu
Carte	Élément principal du jeu
Créature	Type de carte qui peut attaquer
Capacité	Effet sur le jeu
Active	Capacité demandant à payer un cout
Déclenchée	Capacité attendant qu'un événement se produise.
Cout	Liste des ressources et altérations d'états nécessaires pour jouer une carte ou capacité

FIGURE 5 – Tableau des classes.

2. modèle utilisé pour la gestion des évènements : `https://msdn.microsoft.com/fr-fr/library/ee2k0a7d.aspx`

Troisième partie
Rendu graphique

6 L'interface

L'interface sera réalisée avec la bibliothèque SFML du C++, les textures et images seront réalisées à la main ou bien récupérées de banques d'images libres de réutilisation.

Le plateau de jeu sera découpé en trois parties : la zone des joueurs à gauche de l'écran, le champ de bataille au centre, et la zone d'actions, à droite de l'écran. Chaque zone est constituée d'éditeurs ou un affichage : Un éditeur est une liste déroulante contenant et affichant objets qui peuvent être du texte ou des cartes, et les affichages sont des panneaux contenant du texte et des images.

La zone des joueurs contient deux éditeurs, chacun listant les cartes dans le cimetière des joueurs, et des afficheurs avec le nombre de manas de chaque couleur dans les réserves de mana de chaque joueurs, les nombres de cartes dans les bibliothèques et mains des joueurs, la phase du tour et le nombre de PV de chaque joueur.

Le champ de bataille contient six éditeurs qui affichent respectivement : Les terrains et invocations sous le contrôle de l'adversaire, les créatures sous le contrôle de l'adversaire, les objets dans la pile, les créatures sous le contrôle du joueur, les terrains et invocations sous le contrôle du joueur, et les cartes dans la main du joueur.

La zone d'actions contient un éditeur listant les capacités de la carte sélectionnée, et des afficheurs avec, d'un coté, le texte écrit sur la carte sélectionnée, appelé texte Oracle, et ensuite, une liste de boutons permettant de choisir les actions à faire avec la créature sélectionnée.

Chaque objet sera représenté par un sprite individuel, il en existera un pour chaque carte ou jeton de créature. Les modificateurs seront affichés sur

les cartes, les marqueurs avec un nombre bleu, les bonus jusqu'à la fin du tour avec un nombre vert, et les blessures avec un nombre rouge.

En cliquant sur une carte, l'utilisateur la sélectionne, l'interface affiche alors le texte oracle de la carte et la liste de ses capacités, et la carte sélectionnée est surlignée en rouge. Le joueur peut alors choisir une des capacités de la carte, si elle en a, et la lancer.

Avant la mise en place du jeu en réseau et de l'IA, le jeu sera jouable en "HotSeat", c'est à dire que les deux joueurs joueront sur la même interface, et les cartes affichées seront actualisées selon le joueur qui a la priorité, afin qu'il puisse jouer des capacités ou lancer des sorts.

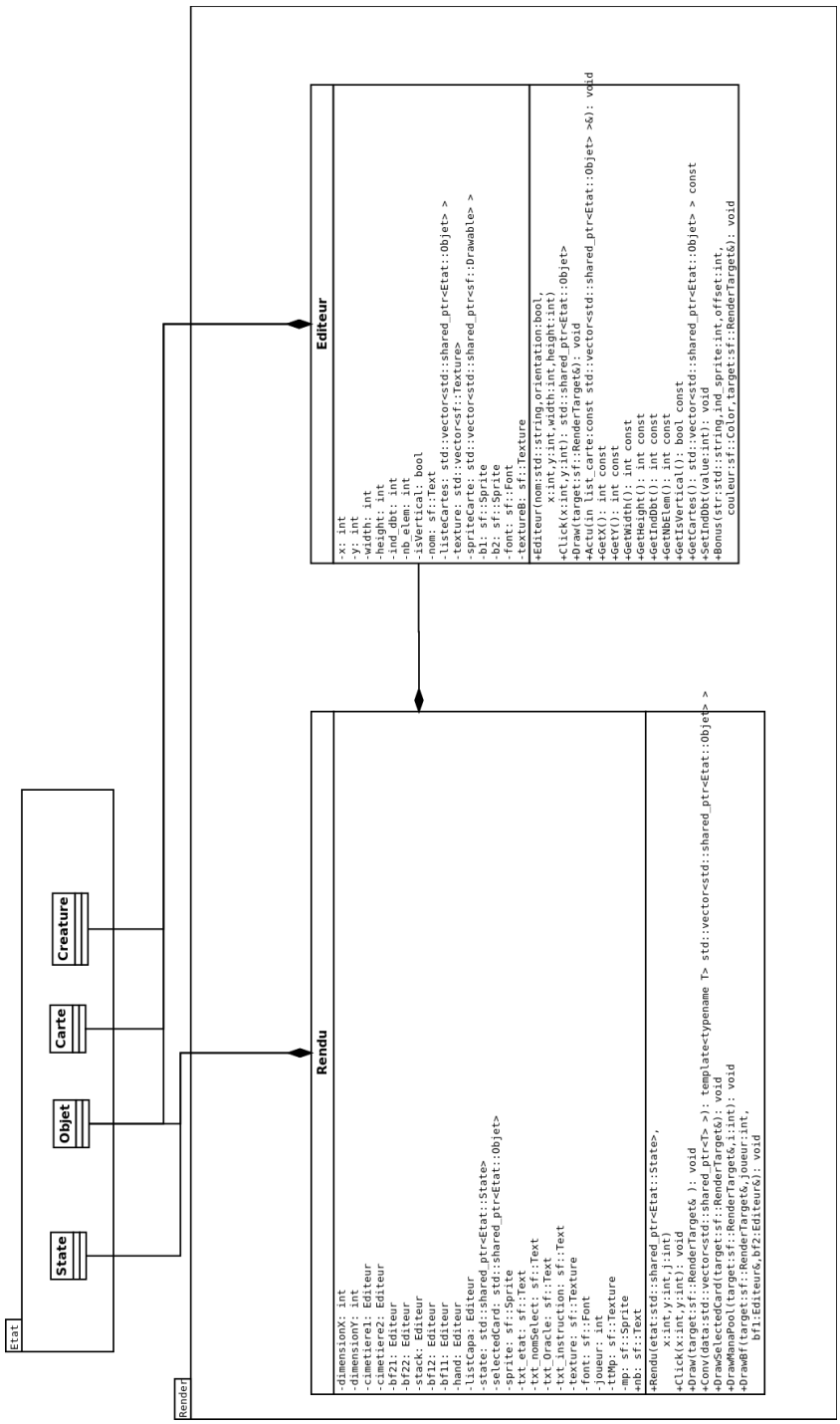

FIGURE 6 – Diagramme de classes du rendu.

Quatrième partie

Moteur du jeu et réalisation des actions

Les actions, événements et réalisations des effets se feront par l'exécution de commandes. Les actions sont gérées par des commandes que les joueurs exécutent et les réalisations automatiques sont des commandes qui ne sont pas exécutées directement par l'utilisateur.

7 Commandes du jeu

La commande "Active", exécutée par le joueur pour activer une capacité d'un permanent qu'il contrôle.

La commande "Attack", le joueur l'exécute pour déclarer une créature comme attaquante, elle n'a aucun effet si ce n'est pas la phase de combat.

La commande "Block", le joueur l'exécute pour déclarer une créature comme bloqueuse et précise la créature bloquée, elle n'a aucun effet si ce n'est pas la phase de combat.

La commande "Cast", le joueur l'exécute pour gérer le lancement d'un sort, elle appelle la commande "paye", retire le sort de la main et le met dans la pile.

La commande "Priority", le joueur l'exécute pour laisser la priorité.

8 Réalisations automatiques

La commande "Draw", retire la première carte de la bibliothèque et la met dans la main du joueur. S'exécute en début de tour ou quand une capacité le demande.

La commande "Untap", désengage les permanents du joueur, s'exécute en début de tour.

La commande "Die", s'exécute à la fin de chaque phase pour chaque créature qui doit mourir.

La commande "ResolveCapa", qui est exécutée par la commande "Priority" lorsque le dernier joueur passe la priorité et que l'objet du dessus de la pile est une capacité. Elle résout la capacité et applique ses effets.

La commande "ResolveCard", qui est exécutée par la commande "Priority" lorsque le dernier joueur passe la priorité et que l'objet du dessus de la pile est une carte. Elle résout la carte et la met sur le champ de bataille si c'est un permanent, sinon elle applique ses effets.

La commande "Phase", est exécutée par la commande "priority" à la fin d'une phase, elle passe à la phase suivante.

La commande "Tour", est exécutée par la commande "priority" à la fin du tour d'un joueur, elle passe au tour du joueur suivant.

La commande "Discard", elle fait se défausser le joueur d'une carte, elle s'exécute à la fin du tour d'un joueur quand celui ci à trop de cartes en main, ou bien quand une capacité le demande.

La commande "Paye", elle vérifie que le joueur a assez de ressources pour lancer un sort ou une capacité, et retire la quantité de mana et de PV payés ainsi.

La commande "VideMp", elle s'exécute à chaque changement de phase et retire le mana restant dans les réserves de chaque joueur.

La commande "SolveCombat", elle s'exécute après la déclaration des bloqueurs, et affecte les blessures de combat aux créatures et aux joueurs.

La commande "Clean", s'exécute à la fin du tour de chaque joueur, elle retire les blessures et les bonus EoT des créatures.

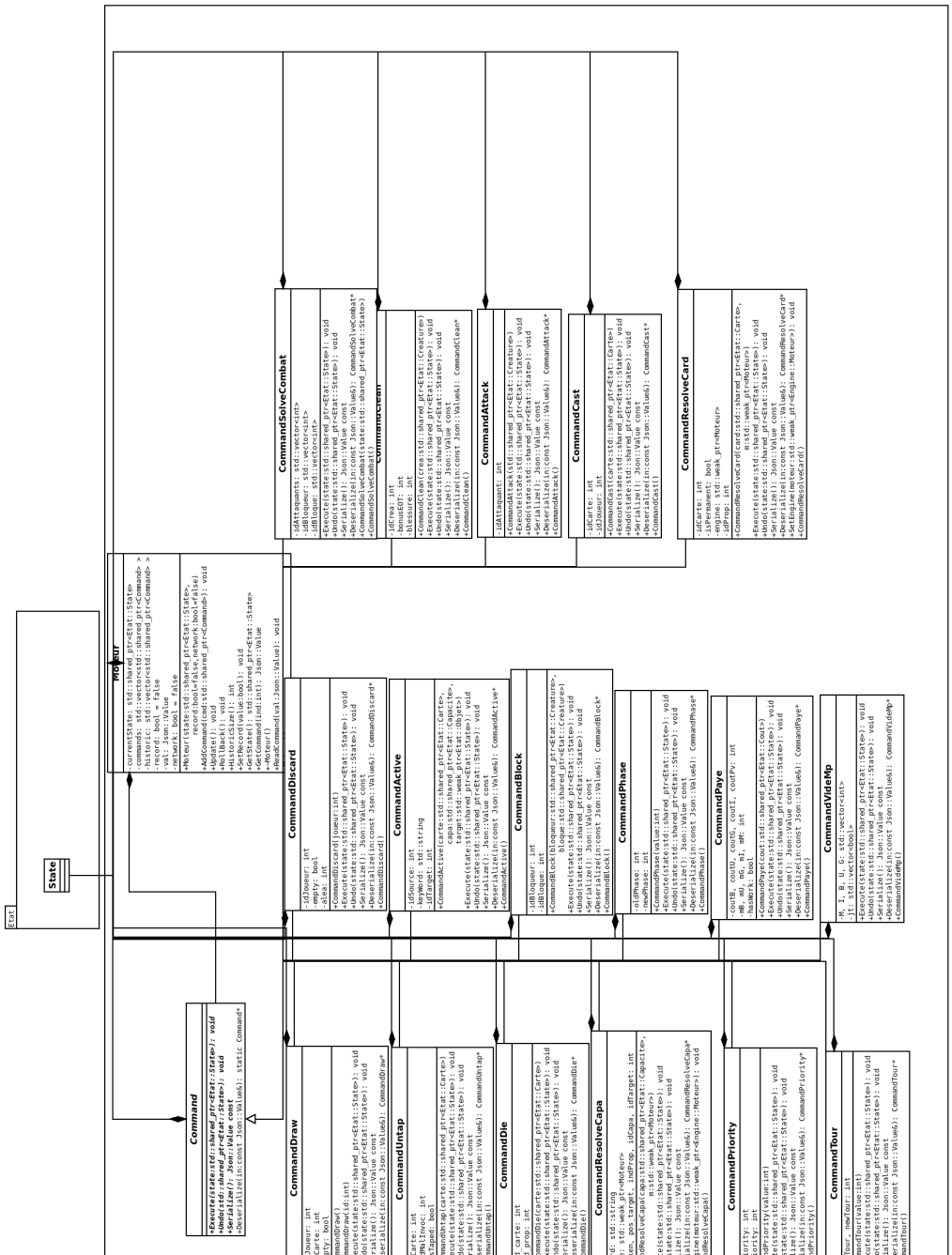

FIGURE 7 – Diagramme de classes des commandes.

Cinquième partie

Intelligence artificielle

Afin de vérifier le fonctionnement des commandes et du moteur de jeu, nous réalisons une intelligence artificielle chargée de jouer une partie en déterminant les actions optimales. Aucun joueur humain n'étant implémenté, elle prendra le rôle des deux joueurs.

9 IA Basique

Pour une première étape, l'intelligence artificielle effectuera des coups aléatoires sans prendre en compte le jeu de l'adversaire ni la liste complète des options possibles. Afin que les commandes puissent être testées sans avoir à attendre un coup de chance, le comportement de l'IA sera orienté et ainsi pas complètement aléatoire.

Au début de sa première phase principale, l'IA jouera un terrain de sa main si il y en a un, ensuite elle passera la priorité jusqu'à la phase de déclaration des attaquants, attaquera avec toutes ses créatures capable de le faire, puis passera la priorité jusqu'à la deuxième phase principale, où elle engagera tous ses terrains pour avoir du mana, et choisira une carte non terrain aléatoirement dans sa main qu'elle essaiera de lancer, enfin passera la priorité jusqu'à la fin du tour, et effectuera la même procédure pour l'autre joueur.

10 IA Heuristique

On réalise désormais une intelligence artificielle qui saura déterminer le coup le plus intéressant en fonction de la situation afin que, même si elle ne cherche pas à gagner la partie, elle essaie de rester en situation avantageuse. Pour ce faire, elle sera aussi en mesure d'annuler les actions antérieures afin de lieux jauger les conséquences d'une action.

A chaque fois que l'IA aura la priorité, elle créera un clone de l'état actuel et générera la liste des commandes possible, c'est a dire toutes les possibilités d'actions telles que lancer une carte ou activer une capacité, en filtrant les actions restreintes ou interdites au moment de la génération, en prenant en compte toutes les cibles possibles si la carte ou capacité en a besoin. Ensuite, pour chaque commande, l'IA l'exécutera, comparera l'état d'arrivée à l'état de départ, notera les changements sur différents facteurs, et attribuera une note globale à l'état. Elle reviendra ensuite sur l'état original et tentera une autre commande. Une fois toutes les commandes testées, l'IA choisira la commande ayant découlée sur la meilleure note de l'état et l'exécutera.

L'intelligence artificielle note l'état sur les paramètres suivants :

- Le nombre de créatures contrôlées,
- La capacité offensive (force totale des créatures contrôlées),
- La capacité défensive (endurance totale des créatures contrôlées),
- Le nombre de terrains contrôlés,
- Le nombre de cartes en main,
- Les points de vie actuels.

Chaque paramètre est coefficienté selon son importance dans l'avancement de la partie, par exemple le nombre de créatures est plus important que le nombre de cartes en main, sinon l'IA préférerait garder ses cartes que de les jouer et ne ferait alors aucune action.

Lors de la phase de combat, l'IA choisira ses attaquants ou bloqueurs selon un algorithme déterminé par note expérience de jeu, afin d'être transposable au maximum de situations. L'algorithme est défini comme suit :

- Phase de déclaration des attaquants :
- Si il n'y a pas de créature adverse en état de bloquer, toutes les créatures alliées sont déclarées comme attaquant.
- Si une créature alliée est sûre de tuer au moins une créature adverse qui la bloquerait et qu'elle y survivrait, elle est déclarée comme attaquant.
- Si une créature alliée est sûre de tuer au moins une créature adverse qui la bloquerait et que elle n'a pas de capacités ou alors qu'elle sont jugées inutiles, elle est déclarée comme attaquant.
- Phase de déclaration des bloqueurs :
- Si la créature survit au blocage contre une créature, elle est déclarée comme bloquant cette créature.
- Si la créature tue une créature et que sa capacité est considérée négligeable, elle est déclarée comme bloquant cette créature.

11 IA avancée

L'implémentation de l'IA avancée a été problématique, notre jeu étant à information non complète, l'IA ne peut pas correctement prévoir les coups pouvant être joués contre elle, de plus les changements de phase perturbent son fonctionnement, on ne peut donc implémenter l'algorithme de minmax que durant la phase de combat.

Durant les phases de priorité, l'IA avancée réalise un algorithme similaire à celui de l'IA heuristique, où elle évalue l'état du jeu après chaque coup possible et choisit la configuration optimale.

Durant les phases de combat, l'IA vérifie chaque combinaison d'attaquant et de bloqueur possible, et applique l'algorithme minmax dessus. Le résultat actuel est que l'IA n'attaque pas tant que elle n'a pas assez de créatures pour tuer l'adversaire quelque soit sa configuration, car elle considère que perdre une créature comme très handicapant, quelque soit le nombre de créatures que l'adversaire perds ou bien le nombre qu'il lui en reste. Les parties sont ainsi peu dynamiques, et le calcul prends beaucoup de temps dès que le nombre de créatures devient important, car les possibilités augmentent exponentiellement. (Le fait que l'un des decks de test joue sur la création de créatures 1/1 n'aide pas non plus.)

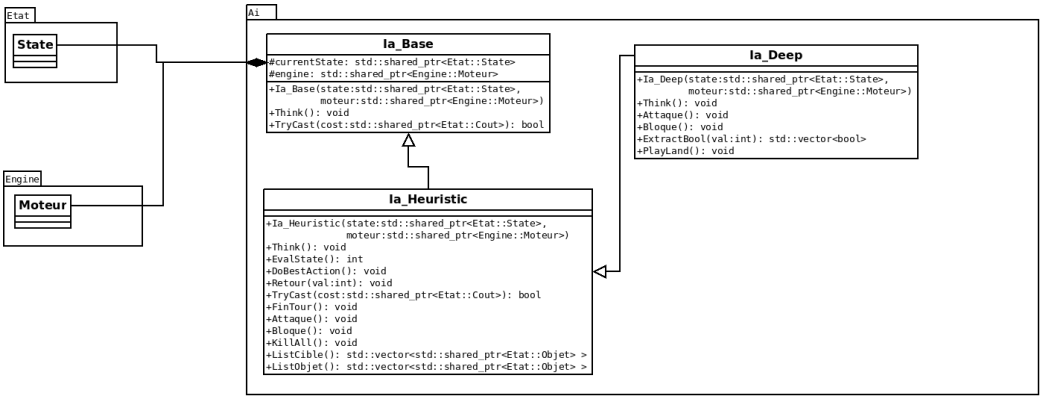

FIGURE 8 – Diagramme des classes de l'IA

Sixième partie
Threading

12 Séparation sur plusieurs threads

Dans une optique d'optimisation de l'exécution du jeu, on sépare le programme en deux threads : le thread principal gère le rendu, l'autre gère le moteur et l'intelligence artificielle. On prépare ainsi le jeu au fonctionnement en réseau, avec le client sur un ordinateur affichant le rendu, et le serveur sur une autre machine faisant tourner le moteur.

La communication entre les threads se fait par le passage de l'état en argument des deux threads et un mutex pour réguler la lecture et l'écriture sur l'état. Lors du fonctionnement en réseau, l'état ne pourra pas être passé en argument, on le modifiera par l'intermédiaire d'un fichier Json, dans lequel on stockera les attributs de chaque commande exécutée, et qui sera actualisé à chaque commande.

13 Séparation sur plusieurs machines : Lobby

Maintenant que le rendu et le moteur sont séparés sur plusieurs threads, on peut lancer chacun d'entre eux sur une machine différente. On met alors en place une API pour gérer le lobby de jeu, où les joueurs se connecteront avant de leur attribuer une partie.

Le fonctionnement du lobby est similaire à celui vu en TP de système d'exploitation, à l'exception de la requête POST, qui se comporte comme une requête PUT quand appelée avec un identifiant négatif, cette particularité avait été implanté car un appel la requête PUT renvoyait une erreur, chose qui a été corrigée depuis.

- Une requête GET avec un identifiant positif renvoie le joueur dont l'identifiant correspond à l'argument,
- Une requête GET avec un identifiant nul renvoie le joueur ayant l'identifiant 0,

- Une requête GET avec un identifiant négatif renvoie la liste des joueurs,
- Une requête PUT ajoute un joueur à la liste avec les attributs correspondants aux arguments passés en entrée,
- Une requête POST avec un identifiant positif (ou nul) modifie le joueur correspondant avec les paramètres passés en argument
- Une requête POST avec un identifiant négatif réalise la même chose que une requête PUT,
- Une requête DELETE avec un identifiant positif ou nul supprime le joueur correspondant,
- Une requête DELETE avec un identifiant négatif ne fait rien.

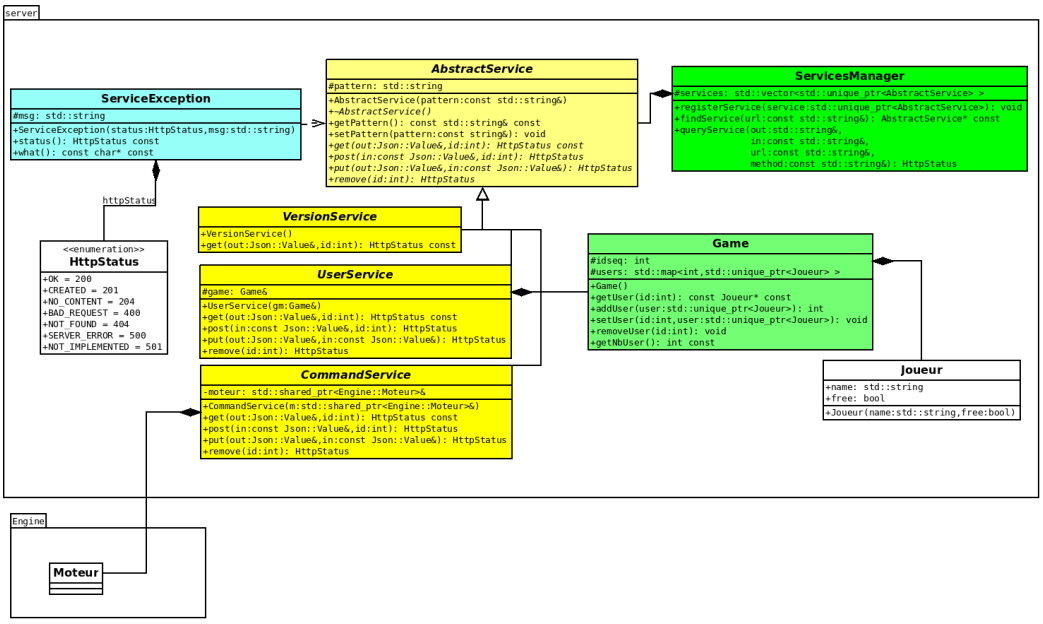

FIGURE 9 – Diagramme des classes du lobby

14 Séparation sur plusieurs machines : Jeu

Pour le fonctionnement du jeu, on ajoute des variantes aux requêtes afin de gérer les commandes.

- La requête GET Command permet d'actualiser la liste des commandes,
- Une requête GET Command avec un indice négatif renvoie le nombre de commandes dans la liste,
- Une requête GET Command avec un indice positif renvoie les commandes d'indices supérieurs à l'indice passé en argument,
- La requête PUT Command ajoute une commande dans la liste,
- La requête DELETE Command annule la dernière commande exécutée.

Lorsqu'un joueur rejoint le lobby, le client envoie une requête PUT au serveur, qui ajoute dans la liste des joueurs le nom du passé en argument en lui attribuant un identifiant.

Quand un client est dans le lobby, il envoie régulièrement des requêtes GET au serveur qui lui répond le nombre de joueurs. Tant qu'il n'y à pas de second joueur dans le lobby, le client attend et continue d'envoyer des requêtes.

Lorsque il y a deux joueurs dans la partie, le client lance le rendu et l'intelligence artificielle. A partir de ce moment, le client enverra des requêtes GET Command pour recevoir la liste des commandes depuis sa dernière actualisation, puis, si il a la priorité, enverra une requête PUT avec la commande choisie par l'IA.