Projet Logiciel Transversal

Marine MOIROUD - Kevin LEGRAND

Table des matières

1 Objectif

1.1 Archétype

L'objectif du projet est de réaliser un jeu s'inspirant du jeu de stratégie RISK, avec des règles simplifiées.

1.2 Règles du jeu

Au début du jeu, le joueur reçoit des territoires puis gère ses armées en les plaçant sur les pays qu'il possède. A tour de rôle, les deux adversaires s'attaquent afin de gagner des territoires. Le vainqueur est celui qui conquiert les 2/3 des pays présents sur le plateau de jeu (nous pourrons modifier cet aspect du jeu pour que les parties durent moins longtemps). Pour cela, il doit lancer des dés qui décideront de l'issue du combat entre deux territoires. Le joueur peut perdre des armées et/ou en recevoir à chaque tour de jeu. Un joueur perd un territoire lorsqu'il n'a plus d'armée dessus.

Le joueur qui attaque peut décider d'attaquer avec 1, 2 ou 3 armées tant qu'une armées au moins n'est pas engagée dans le combat (par exemple, si son territoire possède 3 armées, le joueur peut attaquer avec 2 armées u maximum). Le joueur qui défend peut défendre avec 1 ou 2 armées à condition d'avoir au moins ce nombre d'armées sur le territoire, le nombre d'armées sur un territoire n'étant pas limité.

- Si l'attaquant joue avec 1 armée, la défense peut défendre avec 1 ou 2 armées. Si le plus grand lancer de dé du défenseur est supérieur ou égal à celui de l'attaquant, c'est l'attaquant qui perd une armée. Sinon, le défenseur perd une armée.
- L'attaquant peut jouer avec 2 armées. Si la défense joue avec 1 armée, la perte possible est de 1 de chaque côté. Si la défense joue avec 2 armées, la perte maximale est de 2 armées de chaque côté.
- L'attaquant joue avec 3 armées. Le cas de figure est identique au précédent bien que l'attaque ait une proportion de victoire plus importante. Le défenseur ne peut pas mettre en jeu plus de 2 armées, et donc ne peut pas perdre plus de 2 armées lors d'une attaque. Vous trouverez un exemple sur la page suivante.

Un joueur ne peut attaquer qu'un pays frontalier au sien.
A la fin de son tour, un joueur peut effectuer des déplacements stratégiques. Il peut alors déplacer des armées d'un pays qu'il possède à un autre pays qu'il possède à condition que ces deux pays soient frontaliers.

FIGURE 1 – Exemple de lancé de dés

1.3 Ressources

Plusieurs ressources sont nécessaires pour afficher l'état du jeu au cours de la partie. Le joueur joue sur un plateau représentant le monde. Chaque continent est représenté par une couleur (figure 2 et figure 3). Les armées sont représentées par des pions (figure 4). Le nombre d'armées par territoire sera indiqué par un nombre. Enfin, lorsqu'il conquiert un territoire, le joueur reçoit une carte qui pourra lui donner des avantages (figure 6). Les éléments ressources seront améliorés au fur et à mesure de la conception.

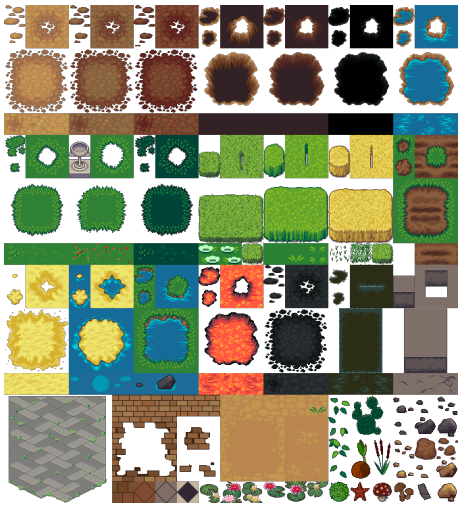

FIGURE 2 – Textures pour le plateau

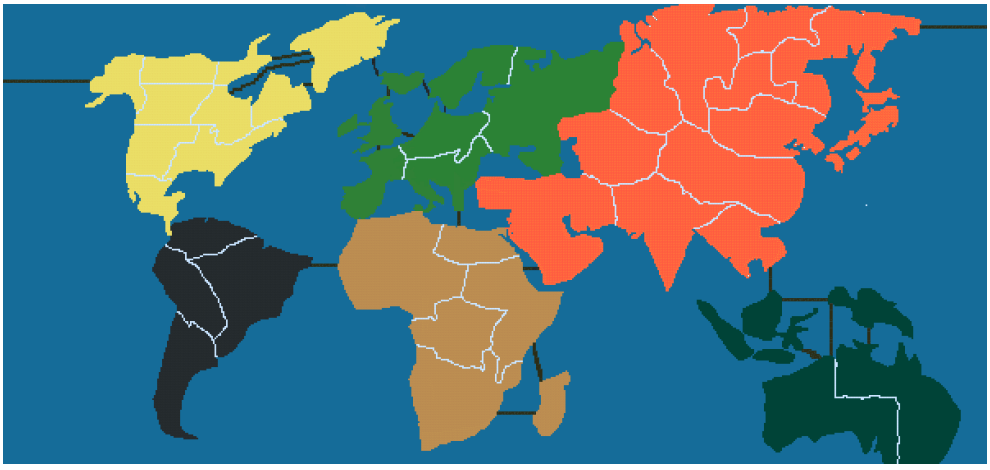

FIGURE 3 – Textures pour la plateau

Figure 4 – Textures pour les pions

Figure 5 – Textures pour les nombres

Figure 6 – Textures pour les cartes

2 Description et conception des états du jeu

2.1 Description du jeu

Pour jouer au RISK, il est nécessaire de disposer d'un plateau de jeu. Ce plateau est constitué :
— D'élements fixes. La carte du jeu est en effet composé des différents continents et pays du monde.
— D'élements mobiles. Certains éléments sont en mouvement sur la carte. Les armées sont modifiées au gré des combats. Les cartes devront également s'afficher ou non selon les joueurs et l'avancée dans la partie.

Tous ces éléments sur la carte du jeu, qu'ils soient fixes ou mobiles, seront caractérisés par :
— Une position sur la carte. Chaque élément aura donc un attribut selon x et un attribut selon l'axe y. En effet, notre plateau de jeu sera une carte en 2 dimensions.
— Un identifiant qui donnera le type de l'objet.

2.1.1 Les éléments fixes

La carte est divisée en **continents**, eux-mêmes subdivisés en **pays** ou plutôt en groupe de pays puisque tous les pays du monde ne sont pas représentés sur la carte, n'ayant pas besoin d'autant d'éléments. Sur la grille composée de tuiles, un pays occupera un ensemble de tuiles proportionnellement à sa taille. Il sera frontalier d'autres pays. Dans le jeu, l'attaque ne sera alors possible qu'entre des pays frontaliers. L'esthétique choisie pour représenter les pays et les continents n'aura aucun impact sur le déroulement du jeu. Chaque continent sera représenté par une couleur principale. Chaque pays appartenant au continent héritera alors de cette couleur.

Les continents sont donc :
— l'AFRIQUE
— l'ASIE
— l'AMERIQUE DU SUD
— l'AMERIQUE DU NORD
— l'EUROPE
— l'OCEANIE

2.1.2 Les éléments mobiles

D'autres éléments ont vocation à apparaître ou disparître à l'écran. C'est notamment le cas des **armées**.

Au début du jeu, lorsque les territoires sont attribués à chaque joueur, le joueur déploie ses armées sur les pays qui lui sont dévolus. Sur chacun des pays occupés, un pion "armée" de la couleur du joueur viendra alors s'installer, accompagné du nombre d'armées que le joueur a décidé de positionner à cet endroit. Le nombre d'armées sur chacun des territoires est donc amené à

évoluer en fonction des attaques ou défenses de chacun des joueurs. De plus, chaque joueur a la possibilité de déplacer ses armées d'un pays frontaliers à l'autre à la fin de chacun de ses tours. Enfin, les armées peuvent aussi changer de couleur si le joueur remporte un territoire.

Les **cartes** seront également assujetties à des modifications dans leur affichage. En effet, elles ne seront affichées au joueur que si elles sont en jeu. Les cartes peuvent être dans trois états différents :

- PIOCHE : Au début du jeu, les cartes sont toutes placées dans la pioche où elles ne sont pas visibles aux joueurs.
- ENJEU : Lors du gain d'un territoire, le joueur pioche la première carte de la fosse. Chaque carte dispose d'une force : TANK, CANON ou SOLDAT. Ces cartes sont alors affichées.
- DEFAUSSE : Quand il a entre 3 et 5 cartes, le joueur peut essayer de réaliser une combinaison qui peut lui donner un avantage en remportant des armées supplémentaires. Il défausse alors les cartes qui ne lui sont donc plus affichées. Il défausse également une cartes lorsqu'il en a déjà 5 et qu'il doit piocher.

2.2 Conception logiciel

Le diagramme des classes pour les états est présenté sur la page suivante (figure 7). Plusieurs groupes de classes commencent à se distinguer. Tout d'abord, un groupe de classe qui concerne uniquement **les éléments** (en orange, jaune et beige). Elles permettent de décrire le fonctionnement des éléments qui composent notre jeu.

Par ailleurs, nous pouvons distinguer un groupe **conteneurs** (en vert). Celui-ci explicite tous les éléments du jeu dans un seul objet State que l'ont pourra ensuite utiliser dans le moteur du jeu.

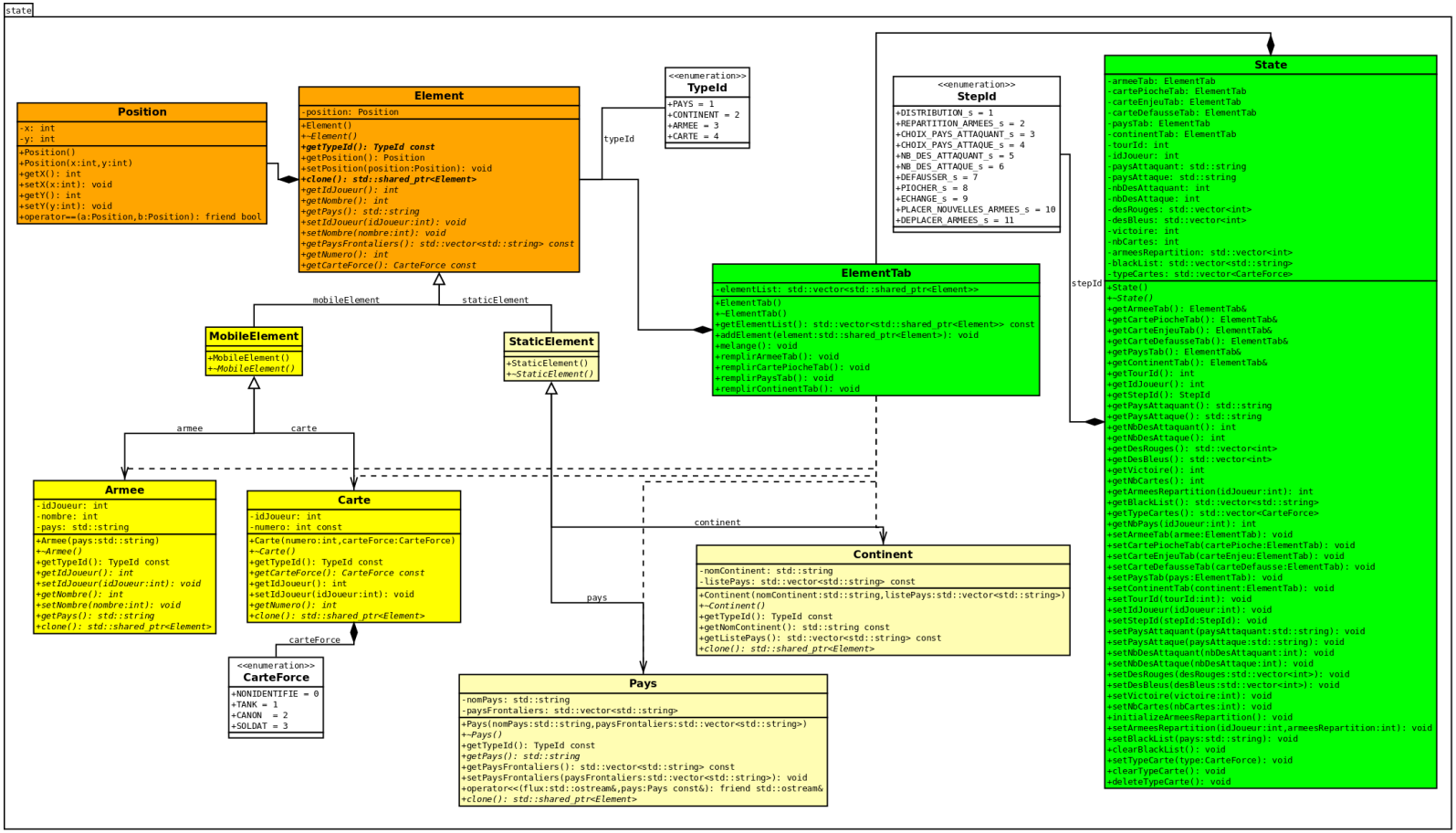

FIGURE 7 – Diagramme des états

3 Rendu : Stratégie et conception

3.1 Stratégie de rendu d'un état

Afin de pouvoir réaliser un rendu imagé de notre jeu RISK, il nous faut tout d'abord réaliser un rendu des différents éléments qui composent notre jeu. Grâce aux outils de la librairie SFML, l'affichage se fera dans une fenêtre avec laquelle le joueur pourra intéragir.

Comme pour la mise en place des différents états, nous allons différencier le rendu des éléments fixes (à savoir la map du jeu avec les pays et les continents) et le rendu des éléments mobiles amenés à se déplacer pendant le jeu (à savoir les armées et les cartes).

Pour cela, le choix s'oriente vers une stratégie assez bas niveau et relativement proche du fonctionnement des unités graphiques. Plus précisément, le découpage se fait en différents plans nommés layers. Naturellement, un plan concernera les éléments statiques et affichera donc la carte, un autre sera consacré aux armées et un dernier s'occupera des cartes.

Pour la carte, le monde et la division des différents pays a été réalisé au préalable sur Tiled. Nous pouvons donc charger le fichier image comme texture et utiliser un sprite pour l'afficher dans la fenêtre du jeu. Nous pourrons ainsi redimensionner la carte principale comme nous le souhaitons. Pour les éléments mobiles, nous utiliserons là encore des textures libres de droits trouvées sur internet pour mettre en place nos armées et les cartes. L'utilisation des sprites permettra de faire varier les couleurs par exemple.

Avant de commencer à afficher les éléments mobiles, nous avons déterminé dans quelle zone de la fenêtre chaque pays se trouvait. Puis, nous déterminons approximativement le centre de cette zone afin d'y placer les pions correspondant aux armées par la suite. Cela permettra également de savoir dans quel pays un joueur clique avec la souris.

3.2 Conception logiciel

Le diagramme des classes pour les rendus est présenté sur la figure 8.

Affichage : Le diagramme render se veut finalement assez simple. Nous l'avons beaucoup modifié afin de ne garder que les éléments et méthodes dont nous avions besoin. D'autres méthodes s'y ajouteront pour ajouter un menu où plusieurs actions et informations seront disponibles. Cela permettra de ne plus passer par le terminal ce qui n'est pas encore le cas pour le moment.

La méthode **AfficheMap** permet tout simplement d'afficher la carte du monde permettant de joueur. La fenêtre est lancée dans le main.cpp qui appelle en premier lieu cette méthode afin d'initialiser le plateau du jeu.

La méthode **AfficheArmées** initialise un pion "armée" dans chacun des territoires. Ce pion a pour le moment une couleur grise correspondante à l'état de l'initialisation. Par la suite, il prendra la couleur du joueur auquel le pays appartient.

La méthode **AfficheNombre** affichera à côté du pion "armée", le nombre d'armées présentes

sur ce territoire. Cela permettra au joueur de décider avec combien d'armée, il souhaite attaquer un territoire ennemi.

La méthode **PaysClic** permet de déterminer les clics réalisés sur la fenêtre. Ainsi, dans un second temps, le choix des pays attaquants, et du nombre d'armées se fera directement dans la fenêtre du jeu. Dans un premier temps, cela se fera dans le terminal.

Controller : Comme son nom l'indique, la classe Controller permet de contrôller l'ensemble des événements pouvant survenir sur la carte de jeu : clic de souris, touches clavier ect... Selon l'event qui est enregistré, le controller choisit les commandes qu'il faut push dans une liste de commandes. Par la suite, c'est cette liste de commande qui sera exécutée en fonction de l'état dans lequel on se trouve.

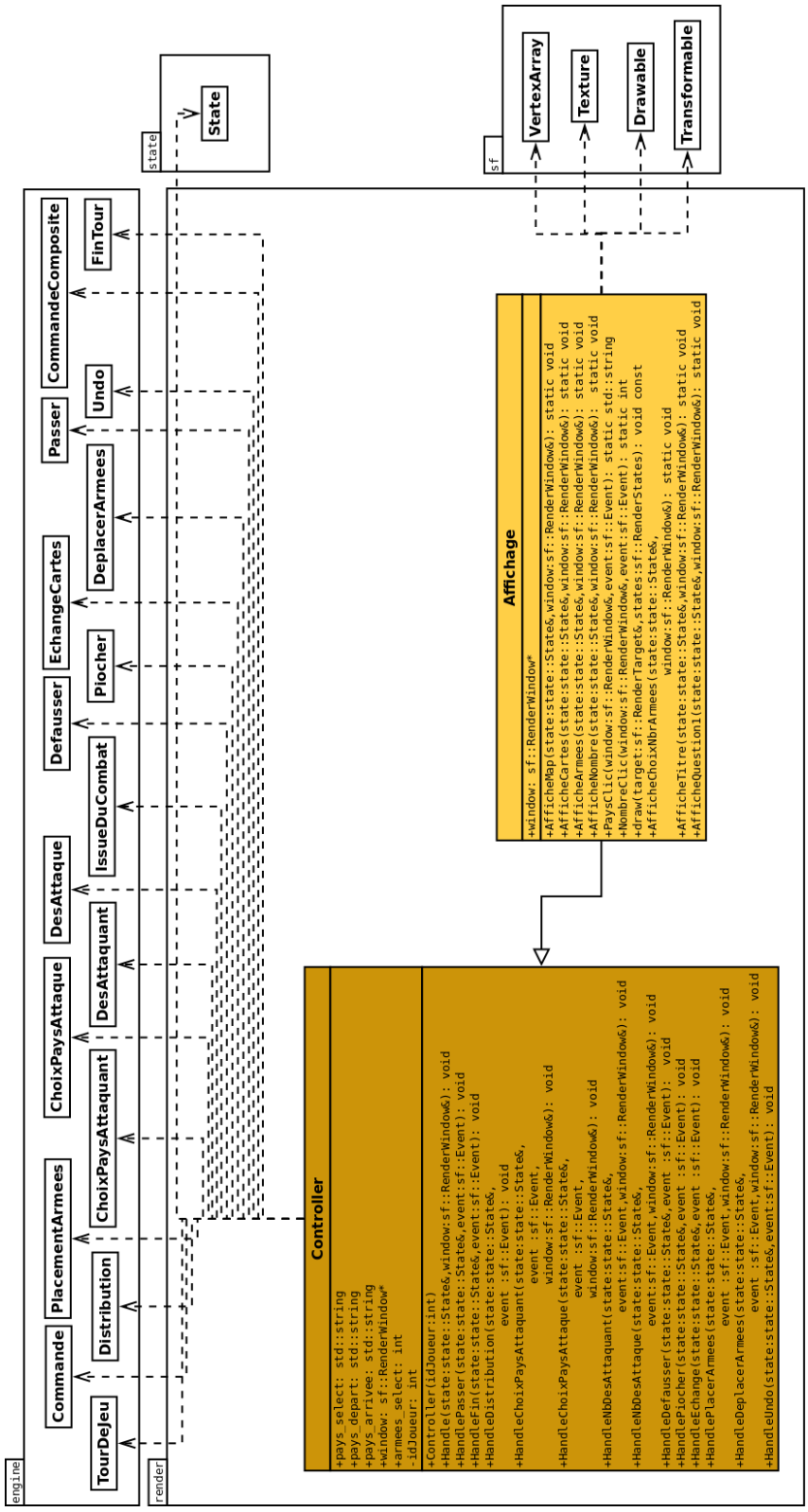

FIGURE 8 – Diagramme des rendus

4 Règles de changement d'états et moteur du jeu

4.1 Changements extérieurs

— Touche D : le joueur demande la distribution des pays parmi les joueurs à l'initialisation du jeu (distribution)
— Touche P : le joueur pioche une carte (piocher)
— Touche N : le joueur passe l'action (next)
— Touche E : le joueur indique que c'est la fin du déplacement de ses armées et donc la fin du tour de jeu (end)
— Touche U : le joueur annule la dernière action réalisée (undo)

4.2 Changements autonomes

Le jeu de déroule en deux temps : l'initialisation puis le jeu en lui-même.

L'initialisation comporte les étapes suivantes :

1. Création de tous les contients, pays, cartes et armées
2. Attribution aléatoire des territoires aux joueurs
3. Chaque joueur choisit combien d'armées il souhaite placer sur chaque territoire qui lui est attribué. Il faut qu'il y ait au moins une armée sur chaqu'un de ses territoires et que la somme des armées souhaitées sur chaque territoire soit égal au nombre total d'armées en sa possession. Tant que ce n'est pas le cas, on reste dans cet état. Si jamais le joueur demande à placer plus d'armées qu'il n'en possède, seules les armées restantes sont placées et un message en informe le joueur.
4. Gestion des cartes. Pour cela, nous créons trois listes : l'une avec les cartes dans un ordre aléatoire (pioche) et deux vides au début (enjeu et défausse). Quand quelqu'un pioche une carte, on lui donne le premier élément de la liste Pioche et on supprime l'élément de la liste. On place alors cet élément dans la liste Enjeu. Lorsque le joueur souhaite échanger des cartes contre des armées ou qu'il possède déjà 5 cartes et doit piocher, il défausse la carte. Celle-ci est alors retirée de la liste Enjeu et est placée dans la liste Defausse. Quand la pioche est vide, la liste Pioche reçoit la liste Défausse où on a mélangé les éléments aléatoirement. La liste Défausse est alors remise à vide.

Le jeu à proprement parler peut alors commencer. Les joueurs jouent chacun leur tour selon l'ordre suivant des actions :

1. Le joueur choisit avec quel pays il souhaite attaquer. On doit alors vérifier que ce pays appartient bien au joueur, qu'il a au moins un pays frontalier qui n'appartient pas au joueur et qu'il possède au moins 2 armées. Sinon, un message informe le joueur que ce pays ne peut attaquer et qu'il doit en choisir un autre (on repasse dans l'étape 1).

2. Le joueur choisit un pays qu'il souhaite attaquer. Il faut alors vérifier si ce mouvement est légal, c'est-à-dire que le pays qu'il souhaite attaquer appartienne bien à l'adversaire et soit

frontalier au territoire choisi précédemment (à l'étape 1). Si c'est bien le cas, on passe à l'étape 3. Sinon, on affiche un message informant le joueur que ce choix n'est pas possible et qu'il doit choisir un autre pays (on repasse dans l'étape 2).

3. On demande au joueur avec combien d'armées (combien de dés) il souhaite attaquer. La réponse à cette question ne peut être que 1, 2 ou 3.
   — Si la réponse est différente, on affiche un message d'erreur et on repasse dans l'étape 3.
   — Si la réponse fait partie des trois choix possibles, on vérifie que le nombre d'armées sur le territoire du joueur (choisi à l'étape 1) est au moins égal au nombre de dés choisi par le joueur + 1. (Par exemple, si le joueur veut lancer 2 dés, il lui faut au moins 3 = 2 + 1 armées sur son territoire.) Si ce n'est pas le cas, on repasse dans l'étape 3.

4. C'est le moment pour le joueur adverse de se défendre. Il choisit alors avec combien d'armées il souhaite défendre son territoire. (Le joueur choisit son nombre de défense sans que l'attaquant n'est déjà lancé ses dés.) La réponse à cette question ne peut être que 1 ou 2.
   — Si la réponse est différente, on affiche un message d'erreur et on repasse dans l'étape 4.
   — Si la réponse fait partie des deux choix possibles, on vérifie que le nombre d'armées sur le territoire du défenseur est au moins égal au nombre de dés choisi. Si ce n'est pas le cas, on repasse dans l'étape 4.

5. On crée le nombre de lancers de dés aléatoires souhaité par l'attaquant, qu'on appelera les dés rouges. Une fonction nous renverra une liste des lancers, triée dans l'ordre décroissant.

6. On crée le nombre de lancers de dés aléatoires souhaité par le défenseur, qu'on appelera les dés bleus. Une fonction nous renverra une liste des lancers, triée dans l'ordre décroissant.

7. On compare les résulats des dés rouges et bleus. On compare d'abord entre eux les deux plus grands dés (le premier élément de chaque liste de lancers de dés). Puis, s'il y a lieu, on compare les deuxièmes plus grands éléments des lancers rouges et bleus.
   — Si le dé rouge est strictement plus grand que le dé bleu, le défenseur perd une armée sur le territoire attaqué, c'est-à-dire que l'objet armée rattaché à ce pays voit son attribut nombre prendre -1.
   — Si le dé rouge est plus petit ou égal au dé bleu, le joueur perd une armée sur le territoire avec lequel il a choisi d'attaquer, de la même manière qu'évoquer précédemment.

   S'il ne reste plus d'armée sur l'un des territoires, ce territoire revient à l'ennemi qui place alors directement dessus une armée (l'armée qui a remporter le combat).

8. Le joueur pioche une carte s'il a conquis le territoire ennemi. Il doit d'abord défausser une carte de son choix s'il possède déjà 5 cartes (nombre maximum de cartes autorisé).

9. Le joueur peut maintenant décider d'échanger des cartes contre des armées. Il indique le numéro des cartes qu'il souhaite donner. On vérifie alors à quelle force chaque carte correspond (SOLDAT, CANON ou TANK). Si la combinaison des cartes est bonne, ces cartes sont placées dans la défausse et le joueur reçoit le nombre d'armées correspondant à la combinaison de cartes exécutée. Sinon, un message informe le joueur que cette combinaison de cartes n'existe pas.

10. Le joueur peut maintenant placer ses nouvelles armées et choisir de déplacer certaines armées existantes, seulement d'un pays frontalier à un autre.

    Pour ce qui est de placer les nouvelles armées, un message rappellera au joueur le nombre d'armées qu'il a remporté sur ce tour. Tant que le nombre d'armées qu'il a placé ne correpond pas au nombre d'armées remportées, le programme lui demande où il souhaite placer ses armées et combien. S'il dépasse ce nombre, seulement le nombre restant d'armées sera attribué au territoire et un message en informera le joueur.

    En ce qui concerne le déplacement des armées, nous vérifions que les deux pays appartiennent au joueur et sont bien frontaliers, et qu'il reste au moins une armée sur chaque territoire.

A la suite de toutes ces étapes (que vous pouvez retrouver figure 9), le joueur a fini son tour et c'est à son adversaire de jouer.

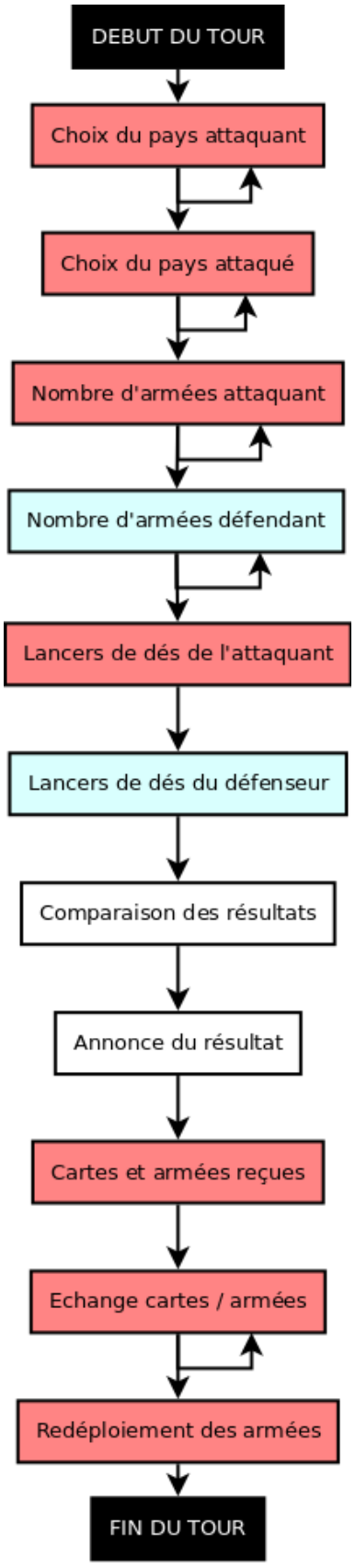

Figure 9 – Déroulement d'un tour de jeu pour un joueur

4.3 Conception logiciel

Le diagramme des classes pour le moteur du jeu est présenté en figure 10. Il comporte un grand type de classe : les classes héritières de Commande.

Tout d'abord, les classes Commande comporte un type de commande avec IdCommande afin d'identifier précisément la classe d'une instance. Les sous-classes de Commande gèrent chacune des étapes du jeu mentionnées au dessus :

- **CommandeComposite** est une commande spéciale puisqu'elle consiste en une liste de Commande*. Elle sera utile pour la DeepAI.
- **Distribution** attribut des territoires aux joueurs.
- **ChoixPays** possède deux classes filles :
  - **ChoixPaysAttaquant** permet au joueur de choisir le pays avec lequel il souhaite attaquer.
  - **ChoixPaysAttaque** permet au joueur de choisir le pays qu'il souhaite attaquer.
- **Des** possède deux classes filles :
  - **DesAttaquant** permet au joueur de choisir le nombre d'armées avec lequel il souhaite attaquer.
  - **DesAttaque** permet à l'adversaire de choisir le nombre d'armées avec lequel qu'il souhaite défendre.
- **IssueDuCombat** compare les résultats des dés et annonce s'il y a eu victoire.
- **Cartes** possède deux classes filles :
  - **Defausser** retire la carte défaussée de la liste Enjeu pour la placer dans la liste Defausse.
  - **Piocher** s'occupe de remplir la pioche avec la défausse lorsqu'elle est vide et de retourner la carte piochée.
- **EchangeCartes** vérifie que la combiniason de cartes proposée par le joueur est bonne et lui donne le nombre de nouvelles armées qu'il reçoit.
- **PlacementArmees** permet aux joueurs de placer leurs armées, que ce soit lors de l'initialisation ou à la fin de leur tour de jeu lorsqu'ils ont échangé des cartes contre des armées.
- **DeplacerArmees** permet au joueur de déplacer stratégiquement ses armées.
- **Passer** permet au joueur de passer une (EchangerCartes, DeplacerArmees) ou des étapes (ChoixPaysAttaquant, ce qui le mène directement à DeplacerArmees).
- **FinTour** met fin au tour de jeu et passe au suivant.
- **Undo** permet d'annuler la dernière action effectuée.
- **TourDeJeu** organise le jeu en permettant l'exécution d'une commande au moment où elle doit l'être.

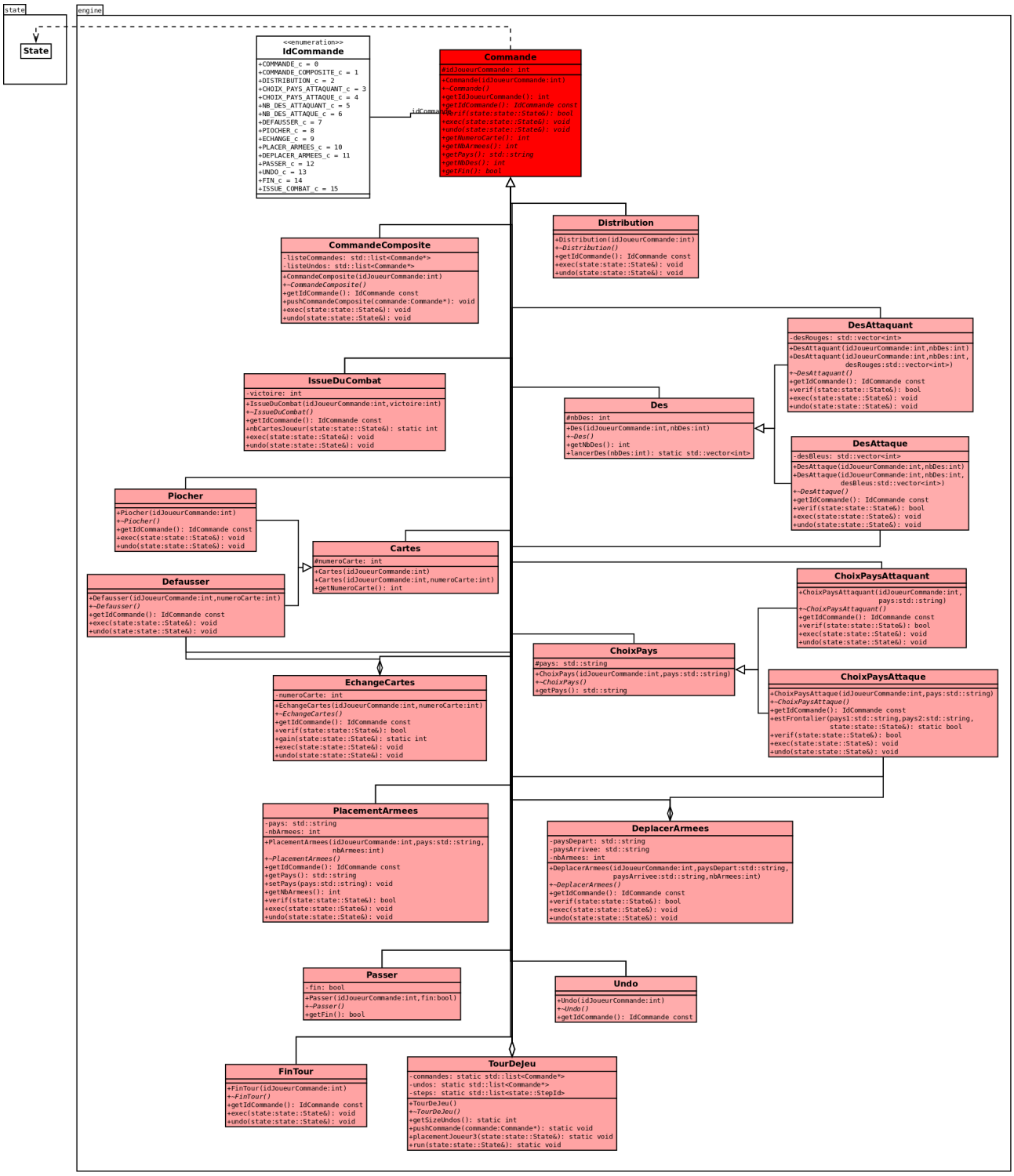

FIGURE 10 – Diagramme du moteur du jeu

5 Intelligence Artificielle

5.1 Stratégies

5.1.1 RandomAI

Cette stratégie consiste en un joueur qui, à chaque étape du jeu, aurait le choix entre plusieurs options et en choisit une de manière aléatoire.

5.1.2 HeuristicAI

Dans l'intelligence heuristique, l'IA devra avoir un meilleur comportement qu'avec les commandes générées par le hasard. Ainsi, les choix faits par l'IA seront davantage réfléchis et permettront d'avancer d'une meilleure manière vers la fin du jeu et la défaite ou la victoire de l'un des deux joueurs. Pour chaque fonction, il est clair que de nombreuses stratégies existent pour jouer au Risk, que, dans une partie réelle, plusieurs d'entre-elles s'entremêlent inévitablement pour apporter le meilleur résultat possible au joueur. Dans notre cas, nous choisissons et expliquons une stratégie envisageable, réalisable pour notre jeu.

— **aiRepartitionArmees.** Dans cette fonction, l'IA doit choisir combien d'armées elle place sur chacun des territoires qui lui sont attribués. Pour trouver la meilleure méthode et stratégie, il convient de rappeler l'objectif du jeu. Pour l'instant, conquérir une certaine partie de la carte est l'objectif fixé : disons que le joueur gagnant devra posséder 28 territoires pour gagner (donc en conquérir 14). L'idée est donc de placer le plus d'armées sur les territoires qui ne sont pas trop isolés. En effet, posséder un réseau de pays sur lesquels les déplacements sont possibles permet de mieux structurer son attaque, effectuer des déplacements, mais aussi organiser sa défense.

— **aiChoixPaysAttaquant.** Dans cette fonction, l'intelligence artificielle devra choisir le pays qu'elle possède avec le plus d'armées afin d'attaquer. Si plusieurs pays ont le même nombre d'armées, elle devra sélectionner celui qui est le plus en contact avec des états voisins appartenant au même joueur. Là encore, cette méthode permettra de renforcer les éventuelles pertes ou de ré-organiser la mise en place des armées en cas de défaite de l'attaque.

— **aiChoixPaysAttaque.** De manière très simple, nous choisissons d'attaquer le pays frontalier ennemi qui a le plus moins d'armées. Cela donne un pourcentage de chance plus important de remporter une attaque, et un éventuel territoire. Si plusieurs pays frontaliers ont le nombre d'armées minimum, on choisit le pays qui a le plus de frontaliers appartenant au joueur.

— **aiNbDesAttaquant.** On rappelle que l'attaque ne peut pas attaquer en mettant en péril son territoire. De plus, le choix du nombre d'armées attaquantes, et de défense se fait en même temps, ainsi aucun des deux joueurs ne peut connaître le choix fait par son adversaire avant de choisir à son tour.

L'attaquant peut choisir d'attaquer avec 1, 2 ou 3 armées. Lorsque c'est possible, le mieux est bien entendu d'attaquer avec trois armées puisque cela augmente la possibilité de réaliser un bon lancer de dés. Lorsque le joueur attaquant possède 4 armées ou plus, il choisit d'attaquer avec 3 armées. Sinon, il attaque avec une armée de moins que le nombre d'armées qu'il possède sur son territoire, soit le nombre maximal qui lui est autorisé.

— **aiNbDesAttaque.** Dès que possible, l'IA défend avec 2 armées. Cela augmente ses chances de battre l'attaque, bien que le risque de perdre deux pions soit aussi présent. Si le défenseur possède 2 armées ou plus sur le territoire, il défend avec 2. S'il ne possède qu'une armée, il défend avec son unique armée.

— **aiDefausser.** Le joueur compte combien de cartes de chaque "force" (TANK, CANON ou SOLDAT) il possède et décide de la meilleure carte à défausser en fonction de s'il possède déjà trois cartes de même force (qu'il pourrait ainsi échanger contre des armées) et du gain que rapporte chaque combinaison de cartes (les soldats rapportant moins que les canons ou les tanks, ils seront les premiers à être défaussés). Ainsi, selon les cartes que le joueur possède, il défaussera une carte selon ce tableau :

Nb de SOLDAT	Nb de TANK	Nb de CANON	Carte défaussée
5	0	0	SOLDAT
4	1	0	SOLDAT
4	0	1	SOLDAT
3	2	0	TANK
3	1	1	TANK
3	0	2	CANON
2	3	0	SOLDAT
2	2	1	CANON
2	1	2	TANK
2	0	3	SOLDAT
1	4	0	SOLDAT
1	3	1	SOLDAT
1	2	2	SOLDAT
1	1	3	SOLDAT
1	0	4	SOLDAT
0	5	0	TANK
0	4	1	TANK
0	3	2	CANON
0	2	3	TANK
0	1	4	TANK
0	0	5	CANON

FIGURE 11 – Carte défaussée en fonction de la main du joueur

— **aiEchange.** Dès que le joueur a trois cartes de même "force", il décide automatiquement de les échanger contre des armées.

— **aiPlacerNouvellesArmees.** Lorsque le joueur doit placer de nouvelles armées à la fin de son tour, il choisit de les placer sur ses territoires possédant le moins d'armées.

— **aiDéplacerArmees.** Plusieurs déplacements sont très intéressants à effectuer. Si le joueur possède un territoire entièrement entouré par des territoires amis, alors on ne peut laisser qu'une unique armée dessus (il ne peut être attaqué) et ainsi renforcer les défenses alentours.

5.1.3 DeepAI

Cette intelligence reprend certaines fonctions de l'intelligence heuristique. Nous donnons deux options de pays attaquant avec chacun deux options de pays attaqué (choisis selon les critères de l'heuristique). Chaque option est essayée et retire un certain nombre de points, calculés en fonction du nombre d'armées et de terrains controllés à l'issue de plusieurs tours de jeu potentiels. Ces points sont déterminés par l'algorithme du min/max basé sur le fait que le joueur veut maximiser ses gains alors que son adversaire va tenter de les minimiser.

5.2 Conception logiciel

Le diagramme des classes pour l'intelligence artificielle est présenté en figure 12.

Les Classes AI : Les classes héritières de AI implémentent plusieurs stratégies d'intelligence artificielle.

— RamdomAI : Intelligence aléatoire
— HeuristicAI : Intelligence heuristique
— DeepAI : Intelligence approfondie

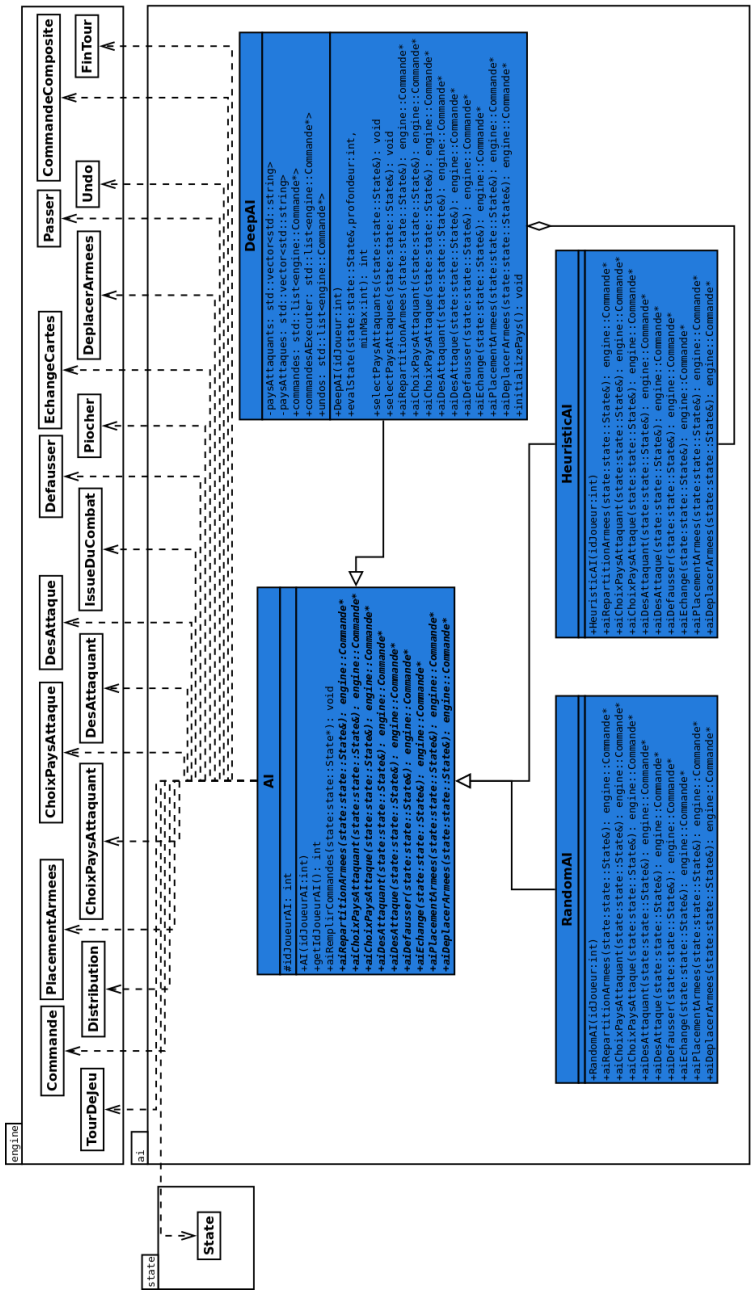

FIGURE 12 – Diagramme de l'intelligence artificielle

6 Modularisation

6.1 Serveur

Pour jouer en réseau, nous allons mettre en place un serveur dont les classes sont présentées sur le schéma suivant.

— **Classe User et UserDB.** Ces deux classes permettent de simuler une petite base de données des joueurs qui seront en jeu sur le réseau. Dans un cas plus concret, ces classes feraient le lien avec une base de données SQL ou noSQL par exemple. Les méthodes comprises dans cette classe sont assez explicites et permettent des actions de base.

— **Classe AbstractService** Cette classe abstraite gère la totalité du service REST du projet.
— **Classe Version Service et UserService**
— **Classe ServicesManager**. Il s'agit du gestionnaire de service, c'est à dire qu'il sélectionne le bon service et la bonne opération, exécute en fonction de l'URL et de la méthode HTTP.
— **Classe ServicesException** Comme son nom l'indique, il gère les cas d'exceptions qui peuvent survenir afin de pouvoir interrompre l'exécution du service.