

Projet Logiciel Transversal

Marine MOIROUD - Kevin LEGRAND



Table des matières

1	Objectif	3
1.1	Archétype	3
1.2	Règles du jeu	3
1.3	Ressources	5
2	Description et conception des états du jeu	7
2.1	Description du jeu	7
2.1.1	Les éléments fixes	7
2.1.2	Les éléments mobiles	7
2.2	Conception logiciel	8
3	Rendu : Stratégie et conception	10
3.1	Stratégie de rendu d'un état	10
3.2	Conception logiciel	10
4	Règles de changement d'états et moteur du jeu	13
4.1	Changements extérieurs	13
4.2	Changements autonomes	13
4.3	Conception logiciel	17
5	Intelligence Artificielle	19
5.1	Stratégies	19
5.1.1	Intelligence aléatoire	19
5.2	Conception logiciel	20

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 pourrions 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 à chaque phase de jeu et en reçoit de nouvelles à diverses occasions. 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 à condition d'avoir au moins ce nombre d'armées en poste sur le territoire. 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 5). Les éléments ressources seront améliorés au fur et à mesure de la conception.

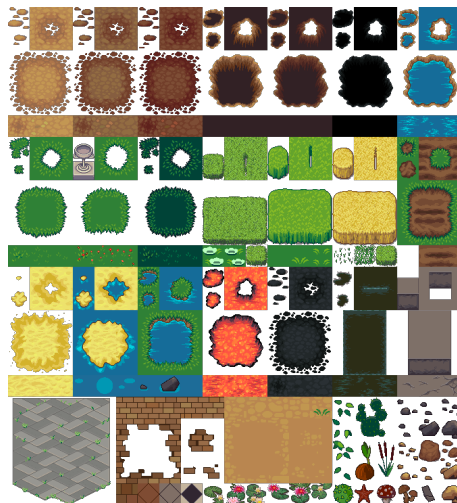


FIGURE 2 – Textures pour le plateau

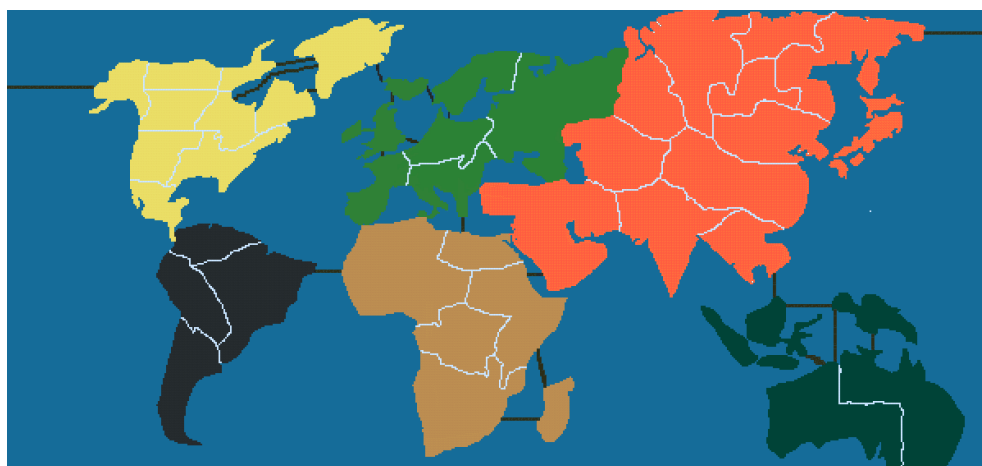


FIGURE 3 – Textures pour la plateau



FIGURE 4 – Textures pour les pions



FIGURE 5 – Textures pour les 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'éléments fixes. La carte du jeu est en effet composée des différents continents et pays du monde.
- D'éléments mobiles. Certains éléments sont en mouvement sur la carte. Les armées se déplacent sur les pays. 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 se déplacer sur la carte du jeu. 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'armée que le joueur a décidé de positionner à cet endroit. Le nombre d'armée 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 être modifiées 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** : Dans certaines conditions, notamment le 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 défause alors les cartes qui ne lui sont donc plus affichées.

2.2 Conception logiciel

Le diagramme des classes pour les états est présenté sur la page suivante (figure 6). 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 explicitera sert à avoir tous les éléments du jeu dans un seul objet State que l'ont pourra ensuite utiliser dans le moteur du jeu.

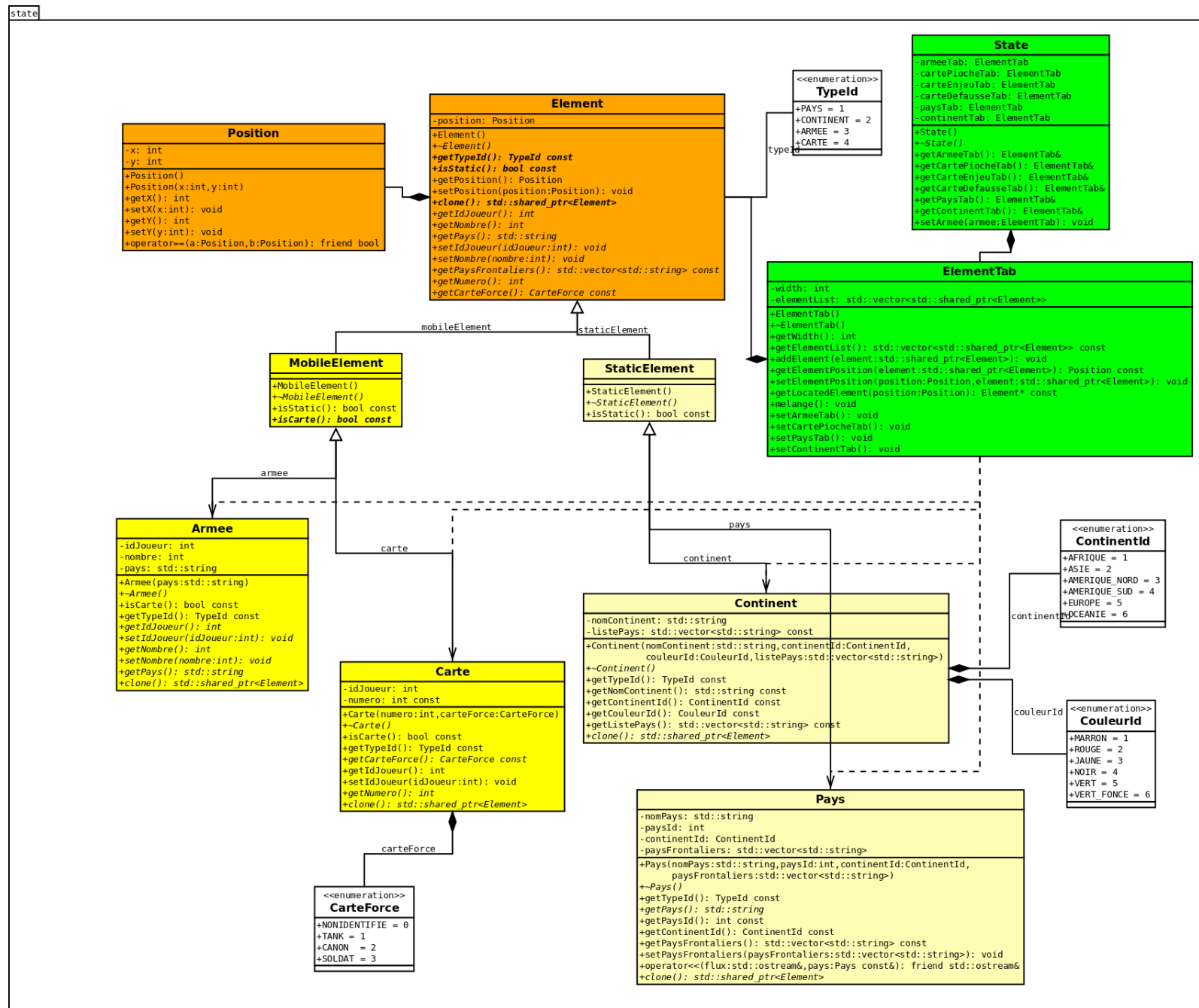


FIGURE 6 – Diagramme des états

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 interagir.

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 pourrions ainsi redimensionner la carte principale comme nous le souhaitons. Pour les éléments mobiles, nous utiliserons là encore des textures 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 7.

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 avons 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 jouer. 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.

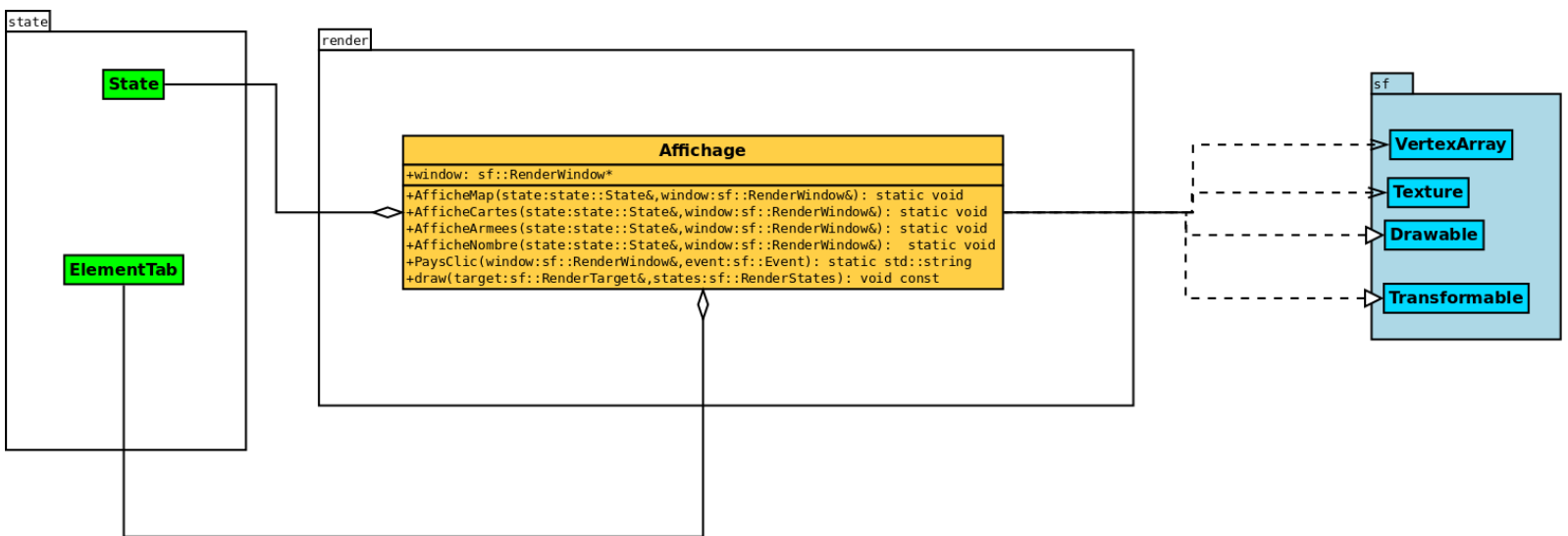


FIGURE 7 – Diagramme des rendus

4 Règles de changement d'états et moteur du jeu

4.1 Changements extérieurs

Parmi les changements extérieurs, nous allons créer une "Touche A" pour annuler et revenir au début du tour. Le joueur pourra utiliser cette touche, avant d'avoir lancé les dés, s'il veut finalement choisir d'attaquer un autre pays.

Pour lancer le jeu (étape 2 de l'initialisation), le joueur doit appuyer sur la barre d'espace.

4.2 Changements autonomes

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

L'initialisation comporte les étapes suivantes :

1. Création de tous les continents, 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 chacun de ses territoires et que la somme des armées souhaitées sur chaque territoire soit égale au nombre total d'armées en sa possession. Si ce n'est pas le cas, on informe le joueur qu'il y a une erreur et on repasse dans cette étape.
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 en formant des combinaisons, il défausse la carte. Celle-ci est alors retirée de la liste Enjeu et est placée dans la liste Défausse. Quand la pioche est vide, la liste Pioche reçoit la liste Défausse où on a mélangé les éléments aléatoirement. La liste Défausse est 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 et qu'il a au moins un pays frontalier qui n'appartient pas au joueur. 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 appellera 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 appellera les dés bleus. Une fonction nous renverra une liste des lancers, triée dans l'ordre décroissant.
7. On compare les résultats 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.
8. Le joueur pioche une carte s'il a conquis le territoire ennemi et reçoit autant d'armées qu'il a de multiples de trois territoires. (Par exemple, si le joueur possède 13 territoires, il recevra 4 nouvelles armées : $13 = 4 \times 3 + 1$.)
9. On demande au joueur s'il souhaite échanger certaines de ses cartes contre des armées supplémentaires (o/n). S'il répond oui, il indique le numéro des cartes qu'il souhaite donner. On vérifie alors à quelle catégorie 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 et on repasse dans l'étape 9.
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 correspond pas au nombre d'armées remportées, le programme lui demande ou 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 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 8), le joueur a fini son tour et c'est à son adversaire de jouer.

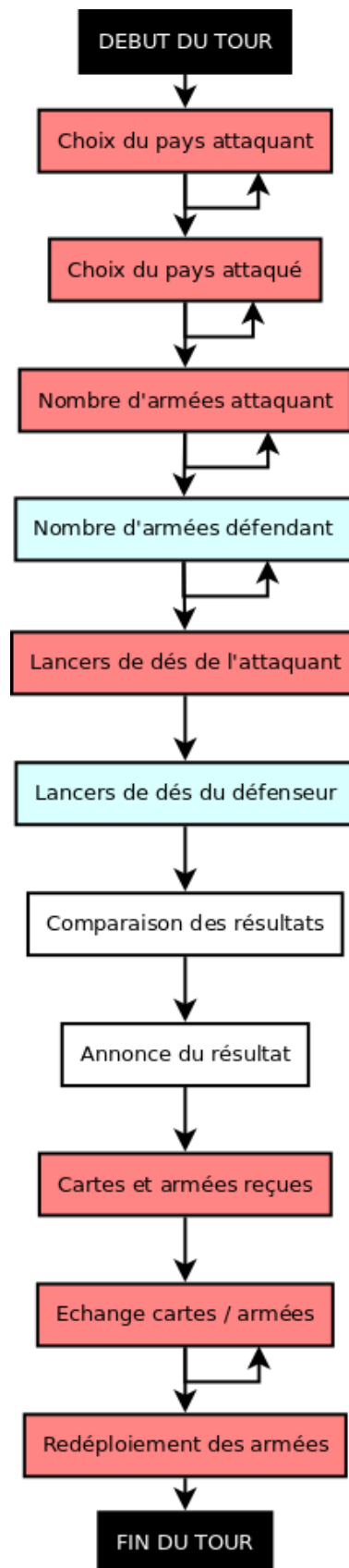


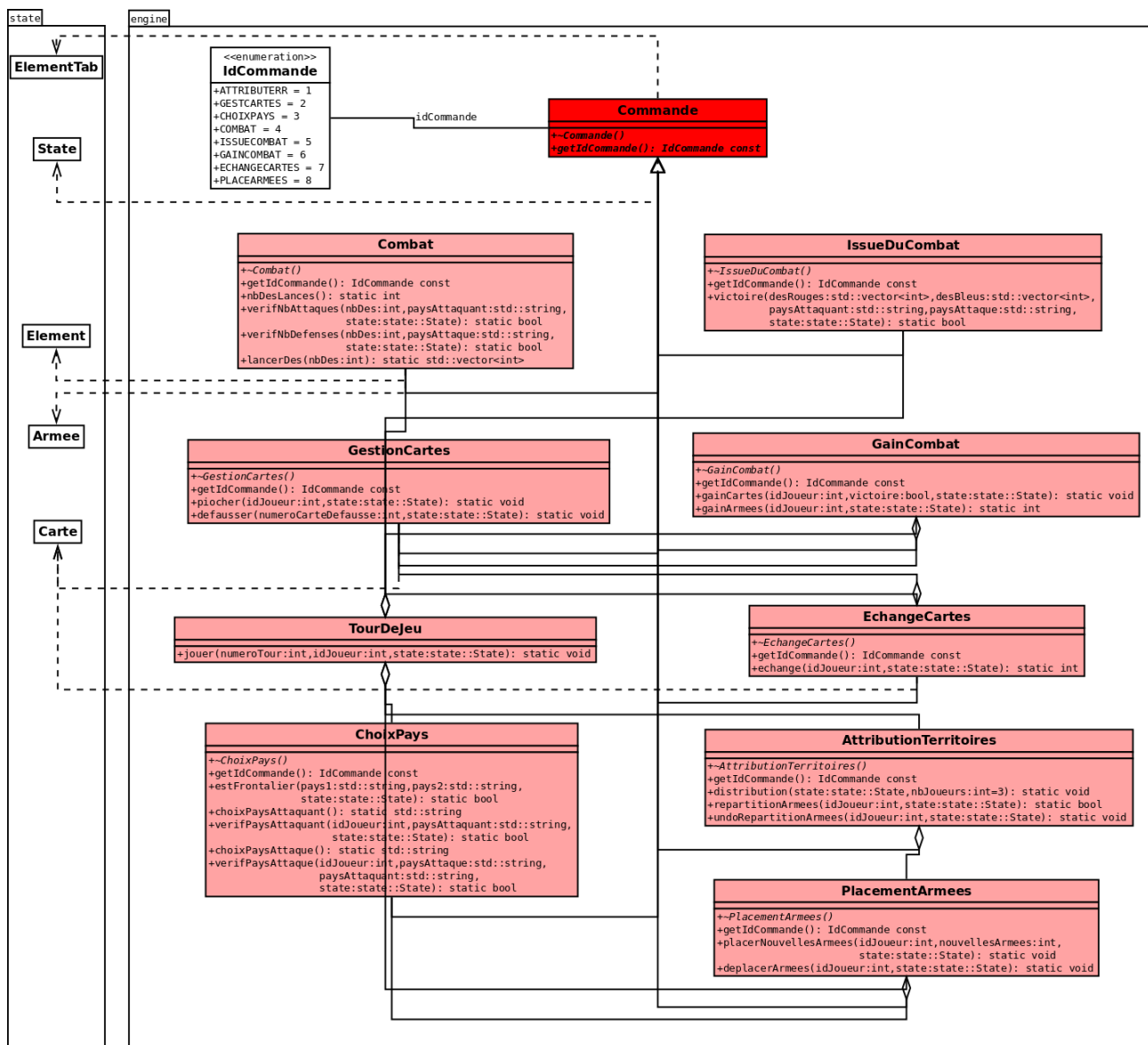
FIGURE 8 – Déroulement d'un tour de jeu pour un joueur

4.3 Conception logiciel

Le diagramme des classes pour le moteur du jeu est présenté en figure 9. 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 8 sous classes de *Commande* gèrent chacune des étapes du jeu mentionnées au dessus :

- **AttributionTerritoires** correspond aux étapes 2 et 3 de l'initialisation du jeu.
 - *distribution* attribut des territoires aux joueurs (étape 2 de l'initialisation) ;
 - *repartitionArmees* permet aux joueurs de placer leurs armées (étape 3 de l'initialisation).
- **GestionCartes** correspond à l'étape 4 de l'initialisation. Elle s'occupe de la gestion des trois listes de cartes (la pioche, les cartes en jeu et la défausse).
 - *piocher* s'occupe de remplir la pioche avec la défausse lorsqu'elle est vide et de retourner la carte piochée ;
 - *defausser* retire la carte défaussée de la liste Enjeu pour la placer dans la liste Defausse.
- **ChoixPays** correspond aux étapes 1 et 2 du jeu à proprement parler.
 - *choixPaysAttaquant* demande au joueur avec quel pays il souhaite attaquer (début de l'étape 1) ;
 - *verifPaysAttaquant* vérifie que le pays choisi pour attaquer est possible (fin de l'étape 1) ;
 - *choixPaysAttaque* demande au joueur quel pays il souhaite attaquer (début de l'étape 2) ;
 - *verifPaysAttaque* vérifie que le pays choisi à attaquer est possible (fin de l'étape 2).
- **Combat** correspond aux étapes 3 à 6 du jeu.
 - *nbDesLances* demande au joueur combien de dés il souhaite lancer (début des étapes 3 et 4) ;
 - *verifNbAttaques* vérifie que le joueur peut attaquer avec le nombre de dés qu'il a choisi (fin de l'étape 3) ;
 - *verifNbDefenses* vérifie que le joueur peut défendre avec le nombre de dés qu'il a choisi (fin de l'étape 4) ;
 - *lancerDes* effectue le lancer de dés (étapes 5 et 6).
- **IssueDuCombat** correspond à l'étape 7 du jeu.
 - *victoire* compare les résultats des dés et annonce s'il y a eu victoire.
- **GainCombat** correspond à l'étape 8 du jeu.
 - *gainCartes* donne une carte au joueur s'il a remporté une victoire durant ce tour ;
 - *gainArmees* donne de nouvelles armées au joueur.
- **EchangeCartes** correspond à l'étape 9 du jeu.
 - *echange* vérifie que la combinaison de cartes proposée par le joueur est bonne et lui retourne le nombre de nouvelles armées qu'il reçoit.
- **PlacementArmees** correspond à l'étape 10 du jeu.
 - *placerNouvellesArmees* permet au joueur de placer les nouvelles armées qu'il a reçu pendant ce tour ;
 - *deplacerArmees* permet au joueur de déplacer stratégiquement ses armées.



5 Intelligence Artificielle

5.1 Stratégies

5.1.1 Intelligence aléatoire

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.

5.2 Conception logiciel

Le diagramme des classes pour l'intelligence artificielle est présenté en figure 10.

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

- RandomAI : Intelligence aléatoire

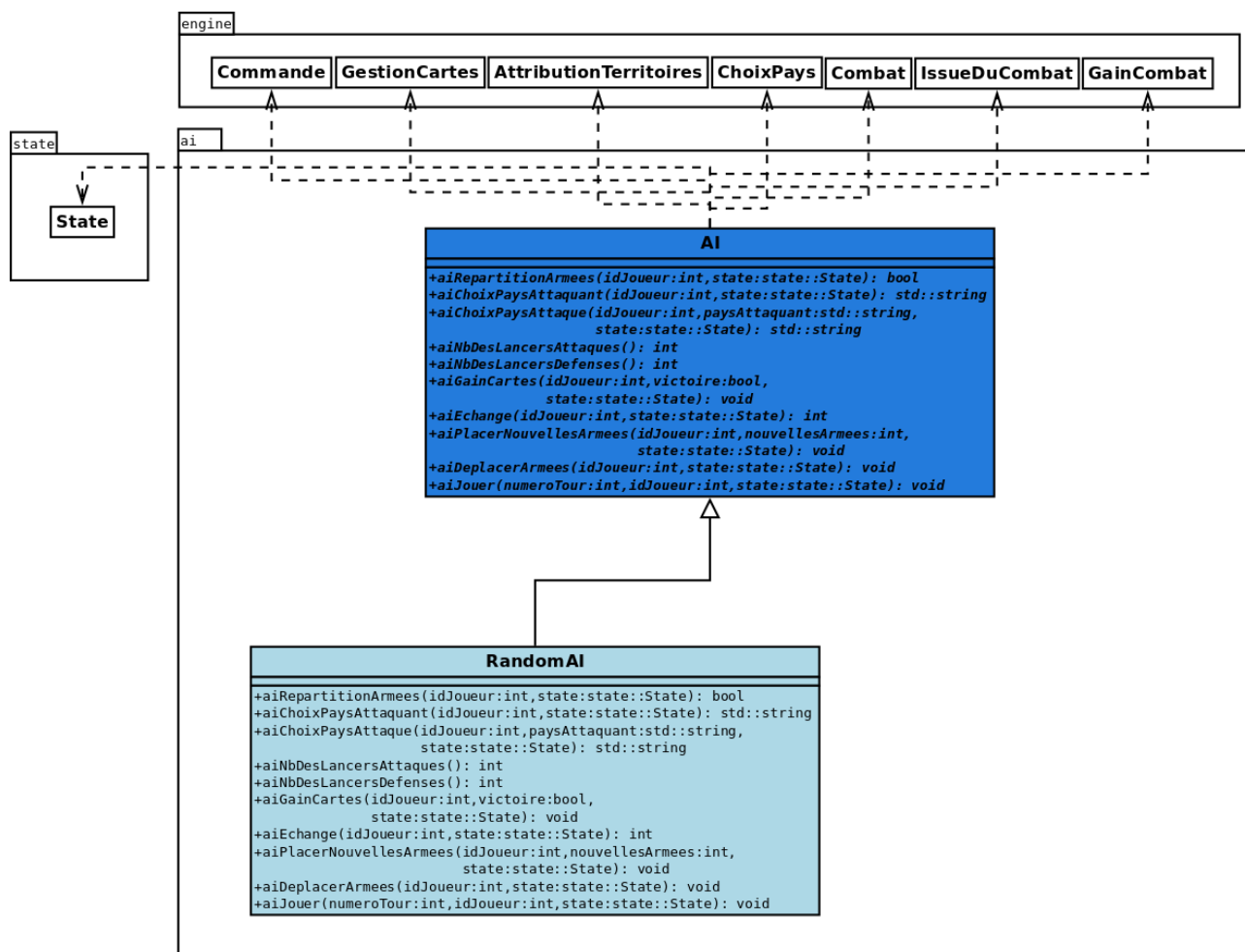


FIGURE 10 – Diagramme de l'intelligence artificielle