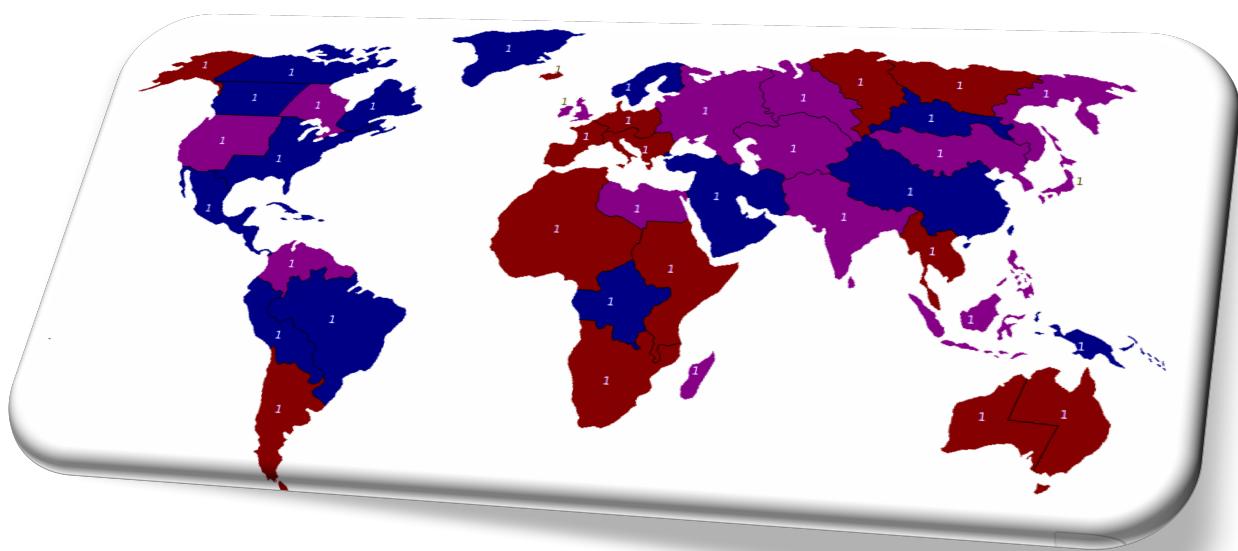


Document technique Risk'ISEP



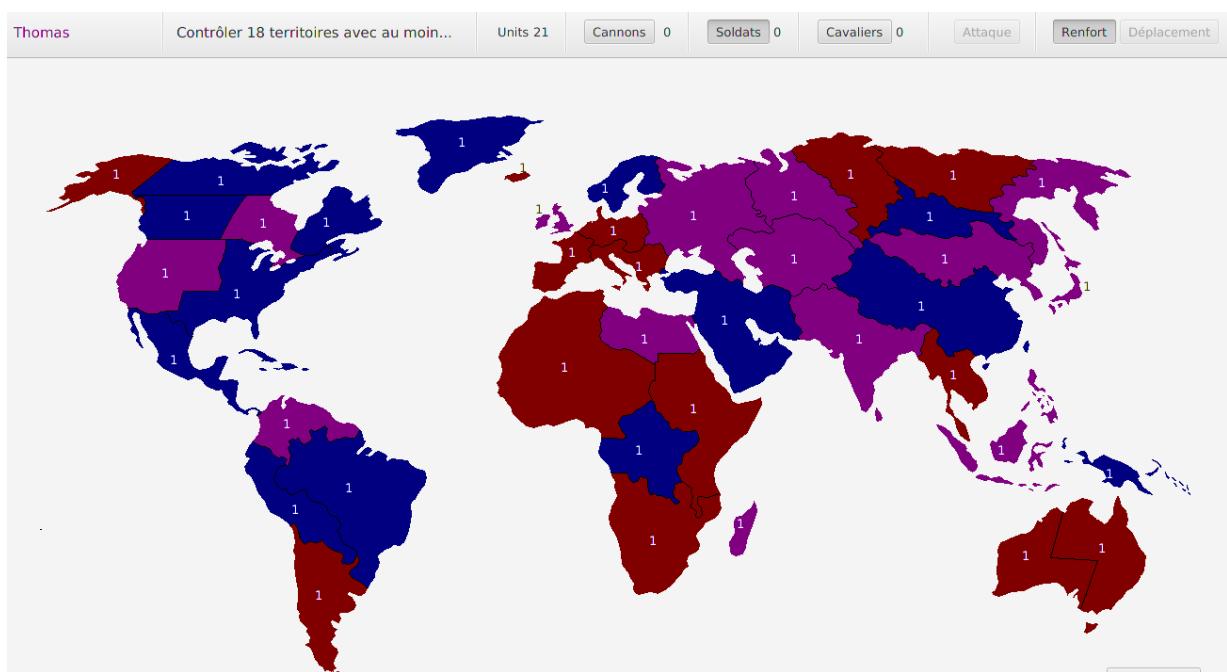
Groupe composé de : Thomas Buatois, Maxime Golfier, Lancelot Kinkelin

FONCTIONNALITES

Voici la liste des fonctionnalités que nous avons réussi à coder ainsi que quelques précisions :

Une interface graphique rendant le jeu jouable de deux à six joueurs. Pour cela, nous avons fait le choix de l'API Java FX. Pour manipuler le front, nous avons utilisé l'outil SceneBuilder qui permet de générer automatiquement du code FXML en drag'n'drop. Nous avons fait ces choix techniques car nous trouvions que le jeu Risk se prêtait bien "au jeu" de la programmation évènementielle. Nous avons d'ailleurs hésité avec la librairie LibGDX 2D qui était plus axée jeux vidéo.

L'implémentation de la carte de base.



L'implémentation des fonctionnalités de base du jeu : implémentation des différents types d'unités, des tours ainsi que des phases de jeu. Le jeu se déroule par un système de tour, tour lui-même divisé en deux phases : renfort et déplacement/attaque.

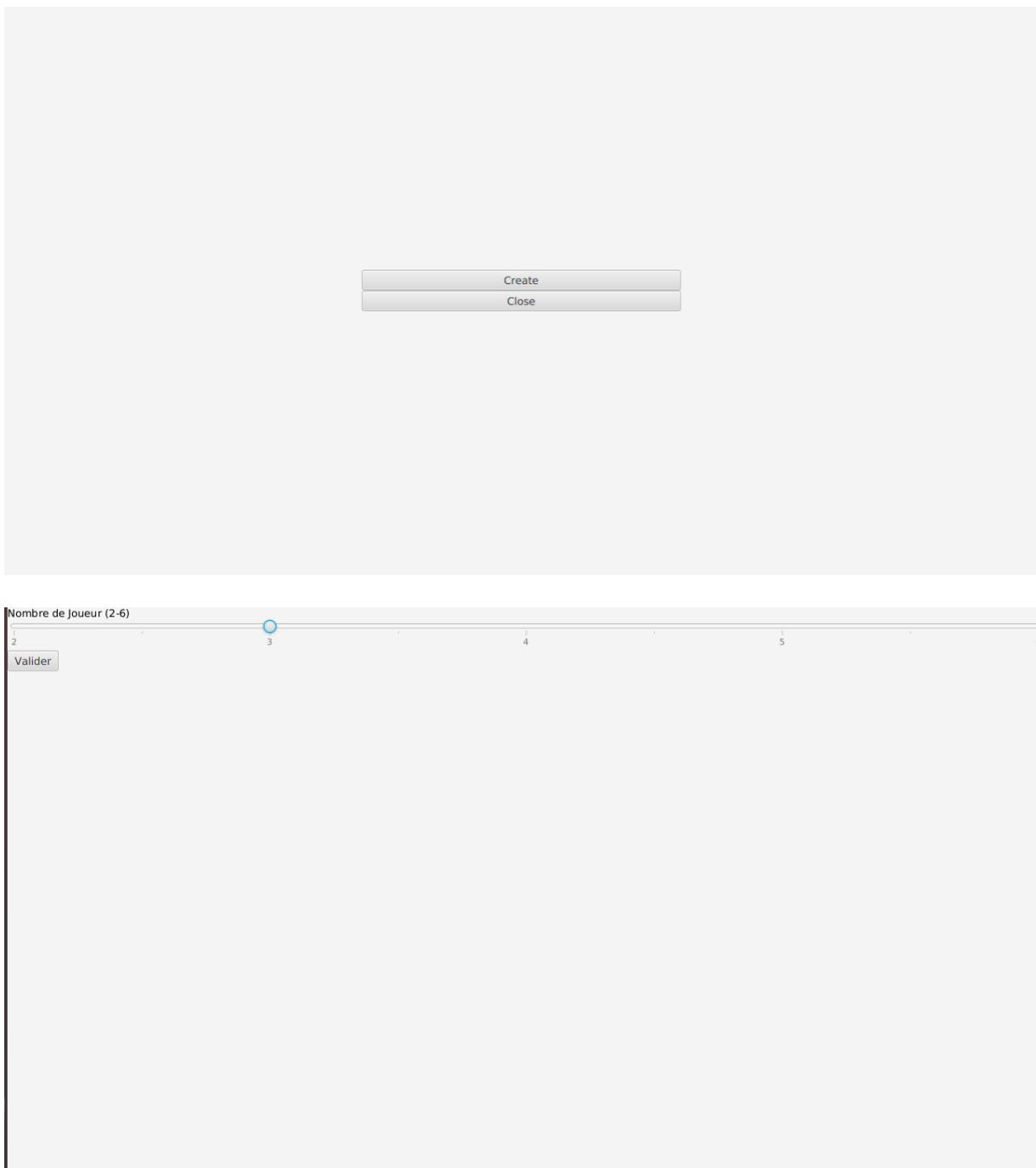
L'implémentation des missions et de la victoire par destruction de tous les ennemis. Nous avons réalisé cette fonctionnalité grâce à une classe Mission. Tout d'abord, elle attribue automatiquement les missions au début de la partie. Elle permet également de vérifier à chaque tour si un joueur donné a atteint son objectif et par conséquent s'il gagne la partie. Pour la victoire par destruction, on vérifie si les joueurs ont toujours des territoires.

L'implémentation d'une IA capable de jouer pour détruire tout le monde. Nous avons réalisé une IA qui permet de simuler le comportement d'un joueur. C'est une classe qui hérite de la classe joueur et donc de ses fonctionnalités. Elle contient une méthode qui permet à chaque tour d'attaquer les territoires aux alentours. Elle n'apparaît cependant pas dans le diagramme UML car elle n'est pas encore codée à l'heure où l'on rédige ce document.

MANUEL D'UTILISATION

Pour déployer notre jeu, il suffit de le compiler en .jar. Il se lance automatiquement quand on l'exécute et il n'y a pas de librairies à installer, Java FX étant inclus dans le JRE.

Pour créer une partie, on navigue à travers le menu et on sélectionne le nombre de joueurs souhaité.



On donne ensuite des noms si on le souhaite aux joueurs. Si on ne le fait pas, des noms aléatoires leur seront attribués.

Nombre de joueur (2-6)

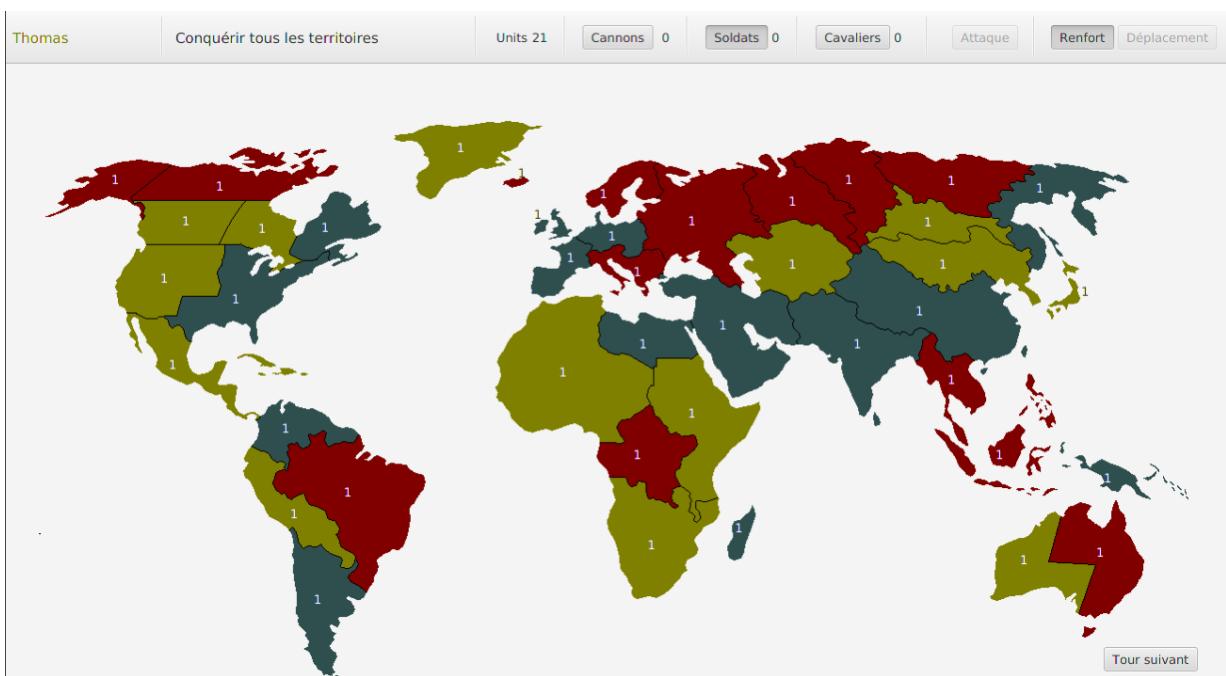
2 3 4 5 6

Nom du joueur 1
Thomas

Nom du joueur 2
Maxime

Nom du joueur 3
Lancelot

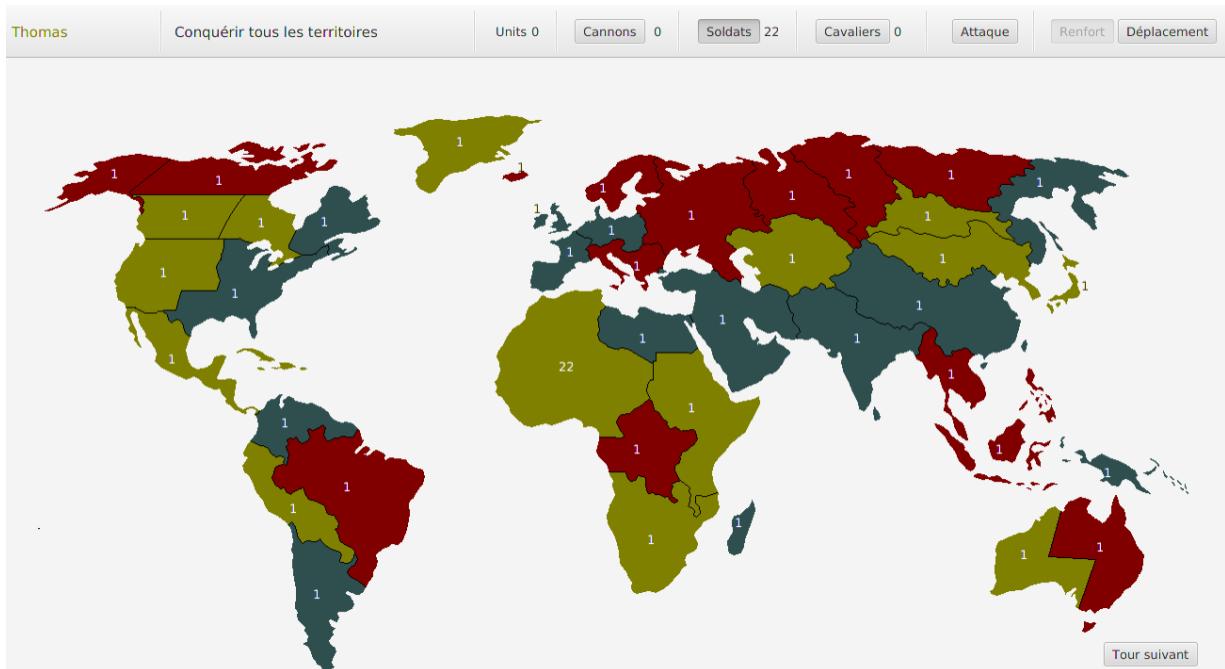
On clique sur valider et notre carte apparaît avec des couleurs générées aléatoirement (dans un panel de couleur sur lesquelles le blanc sera facilement visible.)



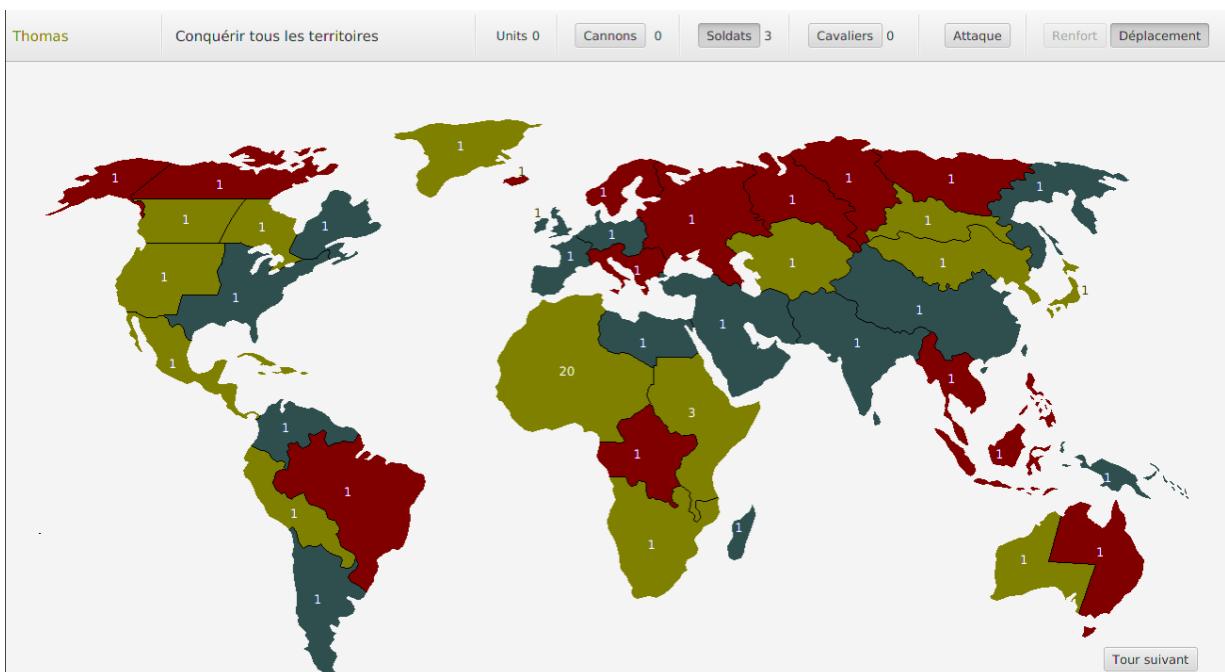
Les territoires sont distribués aléatoirement entre les trois joueurs avec une unité soldat sur chaque territoire.

Nous sommes au premier tour, dans la phase renforts du joueur 1, Thomas en l'occurrence. Celui-ci peut placer ses unités en sélectionnant le type d'unités qu'il souhaite, puis en cliquant sur le territoire souhaité. Il a un crédit d'unités à utiliser entièrement avant de passer au mode suivant. Ce crédit est calculé selon les modalités indiquées dans les consignes. On précise que chaque type

d'unité ne coûte pas la même chose : 1 pour un soldat, 3 pour un cavalier et 7 pour un canon. Une fois ses crédits d'unités utilisés, les boutons Déplacement et Attaque se dégrisent et il peut effectuer les deux actions.



Pour cela, il lui suffit de cliquer sur l'une des deux actions, "Déplacement" par exemple et il passera en mode déplacement. Pour attaquer, il lui suffira de sélectionner le mode attaque, puis de cliquer sur le territoire attaquant et enfin le territoire attaqué. S'il gagne, il possèdera le nouveau territoire.



Une fois qu'il a fini son tour, il peut passer au tour suivant en cliquant sur le bouton et c'est à Maxime de jouer. Et ainsi de suite.

Voici le diagramme UML de classes de notre code.

