SmallWorldUtbm
rapport de projet

Amani Younes - Michael Longo - Gabriel Notong - Pierre Rognon

Université de Technologies de Belfort-Montbéliard

automne 2012

Table des matières

1 Cahier des charges **4**
1.1 Étude de l'existant et adaptation 4
1.2 Conception 4
1.3 Application 4
1.4 Contraintes et choix techniques 7

2 Mise en œuvre **7**
2.1 Étude de l'existant et adaptation 7
2.2 Conception 9
2.3 Développement de l'application 10
2.4 Problèmes rencontrés 11

3 Bilan **12**
3.1 Bilan humain 12
3.2 Bilan pédagogique 12

Introduction

Dans le cadre de l'Unité de Valeur LO43, il a été demandé aux étudiants de réaliser un projet de fin de semestre. Ce projet a pour but de mettre en application les différentes notions acquises en programmation orientée objet ainsi que de pratiquer un langage de programmation : le Java. L'objectif de ce projet est de créer un jeu de rôle à partir d'un jeu de société existant.

SmallWorld est un jeu de stratégie se rapportant à l'univers fantastique. A partir d'un peuple légendaire choisi, il s'agit d'effectuer des conquêtes sur un plateau de jeu comportant des territoires. L'occupation de ces territoires permet alors de remporter de l'argent à chaque tour pour ensuite gagner la partie.

A l'occasion de ce projet, le jeu a été renommé SmallWorldUtbm et il a été demandé d'appliquer les grands principes du jeu d'origine tout en l'adaptant à la vie Utbohémienne.
Cette adaptation a donc demandé une organisation en plusieurs étapes qui seront présentées dans une première partie. Sera ensuite abordée la mise en œuvre de ce projet comprenant des détails sur la conception et le développement de l'application. Enfin, les problèmes rencontrés et un bilan succin seront dressés.

1 Cahier des charges

Le cahier des charges impliquait plusieurs parties. En effet, le sujet portant sur un jeu existant, une première étape consiste à découvrir les règles du jeu et les entités liées à celui-ci puis à les adapter au projet. Suite à cela, une seconde étape porte sur la réalisation d'une modélisation dans le langage UML. Le cahier des charges concernant l'application concerne la troisième partie du travail.

1.1 Étude de l'existant et adaptation

L'étude de l'existant doit permettre de réaliser plusieurs choses. Tout d'abord, elle doit permettre de comprendre les différentes règles du jeu. Elle doit ensuite aider à dissocier les différentes entités relatives à celui-ci. Cela doit avancer une première fois la réflexion sur la réalisation de l'application.
L'adaptation au sujet proposé est en lien direct avec l'existant. Elle doit en effet reprendre les grandes idées de l'existant tout en calquant les entités de l'UTBM. Cette adaptation doit être en réalité une sorte de première étape de conception puisqu'elle met déjà en jeu les différentes entités qui seront rassemblées lors de la conception proprement dite.

1.2 Conception

Pour la conception, le cahier des charges est en grande partie donné et la marge de manœuvre est faible. On doit donc réaliser :
- des diagrammes de cas d'utilisation ;
- des diagrammes de séquence ;
- un diagramme de classes.

Cette étape de conception s'appuie grandement sur l'adaptation faite lors de l'étude de l'existant. Les entités doivent en effet déjà avoir été isolées pour réaliser l'adaptation à notre contexte. L'étape de conception doit donc simplement transférer ces idées en respectant la syntaxe et les normes UML. Le diagramme de classes doit aussi permettre de développer plus facilement par la suite les classes Java en étant donc assez précis.

Suite à cela est réalisé le cahier des charges dit fonctionnel de l'application. Il permettra le développement de celle-ci.

1.3 Application

Le cahier des charges de l'application indique ce que devra faire celle-ci en développant son comportement. Un certain nombre de choses à faire sont donc déjà énoncées par le sujet ou par l'existant mais d'autres sont à indiquer. Ces besoins sont énumérés ci-dessous en quatre parties : tout d'abord ceux relatifs à un début de partie, ensuite d'autres en liens avec le cours de la partie, puis les besoins intervenant en fin de partie. Une dernière partie traitera des besoins liés à l'interface.

Début de partie
L'application doit répondre à un certain nombre de besoins en début de partie.

Tout d'abord, un certain nombre de règles générales ont été établies :
- l'application doit permettre d'effectuer des parties avec un nombre de joueur variables (de 2 à 4 joueurs) ;

- différents plateaux de jeu doivent être disponibles en fonction du nombre de joueurs dans une partie ;
- on doit pouvoir choisir le nom des joueurs ;
- l'application doit permettre au joueur de choisir un couple Peuple/Pouvoir ;

Pour cette application, un grand nombre de traitements se font de façon transparente par rapport au joueur. Ceux-ci sont traduits par les besoins suivants :

- l'application doit initialiser tous les peuples disponibles ainsi que les pouvoirs ;
- elle doit générer des couples aléatoires peuple/pouvoir afin que le joueur en choisisse une ;
- les territoires doivent être générés ainsi que les éléments potentiellement présents sur chacun d'eux ;
- l'application doit gérer l'argent liée à chaque couple peuple/pouvoir puisqu'un joueur peut payer pour prendre un couple.

Cours de partie

Du coté de l'utilisateur, on a les besoins suivants :

- Pour la phase de conquête de territoires :
  - il doit pouvoir tant qu'il n'a encore rien pris durant le tour cliquer sur un bouton de passage en déclin du peuple. Le joueur finit alors son tour directement ;
  - si le joueur a passé son peuple en déclin au tour précédent, l'application doit lui proposer de choisir un nouveau peuple lorsqu'il joue à nouveau ;
  - il doit pouvoir cliquer sur les territoires qu'il peut prendre (géographiquement) ;
  - l'application dans ces deux cas demander confirmation au joueur ;
  - l'application doit proposer lorsque le joueur n'a pas assez de pions pour attaquer un territoire de lancer un dé : si il accepte, on doit alors passer à la phase suivante.
  - le joueur doit pouvoir cliquer sur un bouton de fin de tour pour passer à la phase suivante.

- Pour la phase de redéploiement :
  - le joueur doit pouvoir cliquer sur tous ses territoires ;
  - l'application doit alors proposer le nombre de pions à placer sur celui-ci ;
  - le joueur doit pouvoir cliquer sur un bouton de fin de redéploiement lorsqu'il a fini ;
  - l'application ne doit pas proposer de bouton de fin tant que le joueur a des pions dans les mains.

- Pour la phase de redéploiement des autres joueurs, les besoins sont les même que pour celle du joueur qui vient d'effectuer son tour.

En arrière-plan, l'application doit :

- ne permettre au joueur que de cliquer sur les territoires auxquels il peut accéder ;
- calculer à chaque changement le nombre d'unités en main ;
- effectuer les changements d'occupant pour les territoires abandonnés ou conquis ;
- vérifier dans le cas où des joueurs ont des peuples en déclin que ceux-ci ont toujours des territoires. S'ils n'en n'ont plus, l'application doit supprimer le peuple en déclin du joueur et le remettre dans la liste des peuples libres ;
- en fin de tour, calculer l'argent gagnés par le joueur. Ceux-ci sont calculés à partir des territoires occupés par son peuple mais aussi par ceux occupés par son peuple en déclin et en fonction de ses éventuels bonus.

Fin de partie

Pour l'utilisateur, l'application doit simplement indiquer quel est le gagnant de la partie. Le joueur doit pouvoir ensuite soit rejouer une partie, soit quitter le jeu.
En arrière-plan, pour indiquer quel est le gagnant, l'application doit calculer quel joueur a le plus d'argent accumulés.

Apparence et mise en page générale

La réalisation de cette application doit intégrer une interface graphique. Un certain nombre de règles sont donc à énoncer :

- l'interface doit fournir les informations suivantes au joueur lors de son tour dans un cadre d'informations sur le joueur dans un coin de la fenêtre de jeu. Les informations suivantes seront mises en évidence dans un liseret de la couleur décernée au joueur :
  - le nom du joueur ;
  - son peuple actif ainsi que le pouvoir associé ;

  D'autres informations complétant les premières seront indiquées dans le cadre hors du liseret :
  - l'argent possédée ;
  - le nombre d'unités totales qu'il possède ;
  - le nombre de territoires occupés à l'instant où il joue ;
  - le nombre d'unités qu'il a en main à l'instant où il joue ;
  - l'éventuel peuple en déclin qu'il possède.

- elle doit aussi comprendre des informations sur les territoires mises à jour en fonction du territoire pointé par la souris. Ces informations, contenues dans un cadre d'information du territoire dans un coin de la fenêtre, sont :
  - l'occupant du territoire ;
  - le nombre d'unités présentes sur celui-ci ;
  - le coût de l'attaque pour le joueur en train de jouer ;
  - les éventuels éléments que contient le territoire.

  Le titre du cadre doit avoir pour fond la couleur décernée au joueur.
- les territoires doivent de plus indiquer sur leur espace le nombre d'unités contenues sur celui-ci. Ce chiffre doit être accompagné d'une couleur en arrière-plan correspondant à la couleur décernée au joueur. Si le peuple occupant est en déclin, la couleur sera adoucie ;

- un dernier cadre dont le titre a aussi pour fond la couleur du joueur doit contenir les boutons suivants :
  - fin du tour ;
  - passage en déclin du peuple ;
  - fin du redéploiement.

  Ces boutons doivent être disponibles en fonction de l'étape de jeu du joueur.

- un cadre temporaire doit apparaître pour expliquer au joueur les différents choix impossibles lorsque celui-ci les tente. Ce cadre doit être bien visible et aura une couleur de police rouge. Il doit servir par exemple lorsque l'on clique sur un territoire impossible à attaquer à l'indiquer au joueur.

Des informations entre les étapes et les tours doivent aussi apparaître pour situer le joueur. Ces informations sont :

– une page temporaire indiquant l'étape de conquête du joueur en cours avant que celui-ci attaque des régions ;
– une page temporaire affichant le redéploiement du joueur en cours ;
– une page temporaire affichant le redéploiement des éventuels joueurs qui ont des unités en main suite à des pertes de territoires ;
– enfin en fin de tour, une page temporaire indiquant l'argent durement acquis lors du tour du joueur.

1.4 Contraintes et choix techniques

Le cahier des charges comprend différentes contraintes qui ont été imposées par le propos du sujet. D'autres contraintes ont été rajoutées par l'équipe. Ces différentes contraintes sont de différents types.

De type temporel :
– le rapport du projet doit être rendu pour le 4 janvier et le projet doit donc être terminé pour cette même date ;
– les emplois du temps des différents membres du projet étant différents, des créneaux horaires en commun doivent être trouvés pour les réunions de projet.

De type technique :
– le langage de programmation est imposé : le Java ;
– de plus, la bibliothèque graphique présentée en cours est Swing. C'est donc cette dernière qui doit être utilisée ;
– afin d'éviter les problèmes de compatibilité, un environnement de développement est imposé : Eclipse. Il a l'avantage d'être largement utilisé et d'être disponible sur les systèmes d'exploitations les plus utilisés (Linux, Windows, MacOs) ;
– afin de pouvoir travailler chacun chez soi entre les différentes réunions, un gestionnaire de version est mis en place : Git. Ce dernier a été choisi car lié au site Github.com, il a l'avantage d'être simple à utiliser et de proposer des informations pratiques sur le développement. Le choix se justifie aussi du fait qu'un plugin liant Git à Eclipse est disponible. Ce dernier permet d'utiliser les deux en lien étroit et facilement ;
– pour regrouper les éléments et les territoires du jeu, une base de données doit être utilisée : il a été choisi SQLite pour représenter cette base de données. SQLite présente l'intérêt de ne pas nécessiter de serveur et d'avoir une interface graphique pour créer la base de données. C'est aussi un système léger puisque ne générant qu'un fichier pour une base de données, il est portable et très répandu.

2 Mise en œuvre

La mise en œuvre a vu se succéder trois étapes correspondant aux étapes du cahier des charges. Ces étapes sont l'étude de l'existant et l'adaptation au contexte, la conception et le développement de l'application. L'étude de l'existant et l'adaptation ayant été effectuées dans le même temps, ces deux parties sont intégrées en une seule. Cette partie abordera aussi les problèmes rencontrés lors de la mise en œuvre du projet.

2.1 Étude de l'existant et adaptation

Étude de l'existant

L'étude de l'existant s'est majoritairement traduite par la lecture du manuel de jeu de la ver-

sion plateau de SmallWorld. Ce manuel regroupe en effet toutes les règles du jeu, les différentes étapes ainsi que les différentes entités relatives au jeu. Cela a donc permis de faire une première analyse du jeu et de répartir les différentes entités de celui-ci.
Cette partie d'étude de l'existant a donc en grande partie consisté à la découverte du jeu et à la compréhension des différentes règles, nombreuses et diverses dans SmallWorld. Elle a aussi permis de définir les différentes entités à adapter pour la version du jeu du projet, SmallWorldUtbm.
Les règles devant rester proches du jeu de base, il a fallu que chacun comprenne chaque principe du jeu. Pour cela, un petit manuel récapitulatif a été rédigé pour que chaque membre de l'équipe puisse se resituer sur une règle un peu vague du jeu.

Adaptation

Le but du jeu restant le même, diriger la destinée de peuples dans un monde où chacun doit lutter pour sa survie, les entités à redéfinir pour notre version Utbohémienne du jeu ont été les peuples, les pouvoirs spéciaux ainsi que les éléments. De plus, les cartes utilisées pour l'interface graphique ont été inspirées des différents sites de l'UTBM (Montbéliard pour deux joueurs, Belfort pour trois joueurs, Sévenans pour quatre joueurs).

Peuples de substitution Les différents peuples qui ont été substitués sont :

- les étudiants de tronc commun ; ils sont nombreux et n'ont donc pas de pouvoir particulier ;
- les étudiants de branche ; ils ne perdent pas d'unité lorsque l'on attaque un de leurs territoires ;
- les étudiants en alternance ; ils gagnent une unité par territoire conquis en plus de deux durant un tour ;
- les chercheurs ; si sur un territoire où se trouve un laboratoire, ils ont un bonus de défense de deux ;
- les directeurs de département ; ils gagnent un dollar de plus par territoire conquis durant le tour ;
- les professeurs de connaissances scientifiques ; ils ont un bonus de deux sur le résultat des dés lorsque l'on lance ceux-ci ;
- les professeurs d'humanité ; s'ils sont sur un territoire contenant une salle de partiel, ce territoire devient imprenable ;
- les membres du C.R.I. ; chaque territoire qu'ils occupent contenant une salle informatique leur rapporte plus un en bonus d'attaque ;
- les employés de l'administration ; chaque territoire qu'ils occupent contenant une photocopieuse ou une machine à café leur rapporte plus un en bonus de défense ;
- les employés du service technique ; ils peuvent attaquer tous les territoires, y compris ceux qui sont dits imprenables ;
- les Rats ; ils gagnent un pion de peuple par tour.

Tribus oubliées Les tribus oubliées ont été changées en thésards dans l'adaptation SmallWorldUtbm. Ils sont en effet en passe de faire partie d'un peuple à part entière : les professeurs d'humanités, ceux de connaissances scientifiques ou les chercheurs !

Pouvoirs spéciaux Les différents pouvoirs spéciaux ont été inspirés des catégories que l'on retrouve dans la communauté de l'UTBM. Si elles sont plus ou moins flatteuse, on y trouve pour chacune des avantages certains ! Les pouvoirs spéciaux sont les suivants :

- les associatifs ; ils gagnent un pion de peuple à chaque tour ;

- les avares ; ils reçoivent un dollar par région conquise dans le tour ;
- les bagarreurs ; ils ont un bonus de un à l'attaque ;
- les faux-culs ; ils peuvent toujours attaquer les territoires en bord de carte ;
- les fêtards ; à chaque tour, ils reçoivent un bonus de deux unités à l'attaque ;
- les fumeurs ; s'ils occupent un territoire avec un espace de plein air, ils obtiennent deux dollars de plus à la fin de chaque tour ;
- les geeks ; s'ils occupent une salle informatisée, ils y ont un bonus de défense de un. Ils gagnent aussi un bonus d'attaque de deux lors de l'attaque d'un territoire contenant une salle informatique ;
- les gloutons ; ils posent de la nourriture à chaque territoire qu'ils prennent ;
- les intellos ; ils ont un bonus de défense de un sur les salles d'examen et un bonus d'attaque de deux lors de l'attaque de celles-ci ;
- les joueurs ; obtiennent un bonus de deux sur chaque lancer de dé ;
- les nerveux ; s'ils occupent un territoire avec une machine à café, ils obtiennent un bonus de défense de deux sur ce territoire ;
- les opportunistes ; ils peuvent lancer un dé une fois de plus dans le tour ;
- les paresseux ; ils sont gratifiés d'un bonus de défense de un sur tous leurs territoires ;
- les voyageurs ; ils peuvent attaquer partout sur la carte.

Éléments Les éléments viennent remplacer montagnes, antres et autres champs dans la version originelle du jeu. Ces éléments ont été trouvés parmi les différents objets qu'on peut trouver à l'UTBM au gré des couloirs. On a ainsi :

- l'espace de plein air ;
- le laboratoire ;
- la machine à café ; elle rapporte un bonus de un en défense pour l'occupant du territoire où elle se trouve ;
- la nourriture ; elle donne un bonus de un en défense à l'occupant du territoire en contenant sauf si l'attaquant est un rat ;
- la photocopieuse ; elle rapport un dollar par tour au possesseur du territoire ;
- la salle informatique ;
- la salle de partiel ; elle rend le territoire où elle se trouve inattaquable par un étudiant.

2.2 Conception

La conception du projet a consisté en l'élaboration de différents diagrammes UML. Des cas d'utilisation, un diagramme de classe ainsi que des diagrammes de séquence ont été crées. Ceux-ci ont permis la création du cahier des charges présenté en partie précédente. Le digramme de classe a été pour une partie largement inspiré de l'adaptation qui découpait en entités l'application.

Cas d'utilisation

Le diagramme de cas d'utilisation est assez simple pour ce projet. Il n'y a en effet qu'un seul acteur : le joueur. Le diagramme retranscrit ici les besoins du cahier des charges du côté du joueur.
Ce diagramme est présenté en annexe 1.

Diagrammes de classe

Le diagramme de classe est présenté en annexe 2.

Diagrammes de séquence

Plusieurs diagrammes de séquences ont été produits :

- un diagramme concernant l'attaque d'un territoire (annexe 3) ;
- un diagramme qui modélise la prise d'un territoire sans occupant (annexe 4) ;
- une diagramme qui modélise l'abandon d'un territoire (annexe 5) ;
- un autre qui concerne le calcul le gain d'argent du joueur à la fin du tour (annexe 6).

2.3 Développement de l'application

Le développement de l'application s'est déroulé en deux parties majeures. La première est l'implementation du "coeur" de l'application. La seconde consiste en l'élaboration de l'interface graphique du jeu.

Implémentation de la conception

La première partie du développement a consisté en l'implémentation de classes "pures" de l'application. Ces classes ont été tirées directement du diagramme UML de classes élaboré lors de l'étape de conception. Les classes Pouvoir, Element, Peuple, Partie, Plateau et Territoire ont été tout d'abord implémentées. Ce sont ensuite les sous-classes de Element, Pouvoir et Peuple qui ont été écrites. En parallèle, une classe interface a été développée : Bonusable.

Classe Bonusable Cette classe a permis de simplifier toutes les classes de l'application qui mettent en jeu les bonus d'attaque, de défense, de pions, d'argent, de dé, etc. Les classes qui implémentent cette interface sont les classes Peuple et Pouvoir.

Classes Peuple et Pouvoir et sous-classes Ces classes, implémentant Bonusable, sont respectivement les classes mères des différents peuples et des pouvoirs. Ces sous-classes ont été réunies dans des packages (com.utbm.SmallWorld.peuples et com.utbm.SmallWorld.pouvoir). Comme leur nom l'indique, elles modélisent les différents peuples et pouvoirs présents dans le jeu.

Classe Territoire Cette classe n'est pas codée "en dur" comme pour Peuple et Pouvoir. Elle n'a donc pas de sous-classes. Cependant, les différents territoires sont stockés dans une base de données SQLite. La classe qui va chercher les informations sur les territoires et crée ceux-ci a été implémentée durant la seconde étapes, avec l'interface graphique.

Classe Element La classe Element implémente l'interface Comparable<Element>. Cette interface permet de comparer la classe implémentée avec la classe entre crochets. Dans notre cas, elle permet de comparer les éléments pour pouvoir reconnaître ceux présents sur les territoires de la carte pendant une partie. Cette classe a pour sous-classe les différents éléments indiqués dans la partie d'adaptation du jeu au contexte de l'UTBM, par exemple MachineACafe.

Classe Joueur La classe Joueur permet de modéliser les différents joueurs d'une partie. Elle contient les méthodes principales du jeu qui permettent d'attaquer ou encore de calculer les gains d'argent à la fin d'un tour.

Classe Plateau Cette classe se contente de contenir une liste des territoire présents sur le plateau de jeu.

Classe Partie C'est la classe principale de l'application. Elle permet d'initialiser une partie en créant donc les joueurs, les pouvoirs, les peuples. Elle permet aussi de créer les couples pouvoir/peuple en les mélangeant. Elle gère aussi les actions à faire lors d'un clique sur un bouton ou sur un territoire. Enfin, cette classe est un singleton. Cela signifie que le constructeur est privé et qu'elle contient une méthode qui retourne l'instance de Partie si existante et qui la crée sinon. Ainsi, une seule instance de la classe peut-être crée.

Développement de l'interface graphique

L'interface graphique a été développée une fois le coeur de l'application terminé. Les classes liées à celle-ci ont été regroupées dans le package com.utbm.SmallWorld.gui.

Classe Game C'est elle qui articule toute l'interface graphique. Elle contient différentes méthodes "build" qui créent la fenêtre et les layout. Elle contient aussi toutes les méthodes qui affichent les fenêtres temporaires ainsi qu'une méthode de mise à jour de l'interface. Cette dernière rafraichis les différentes informations sur l'écran. Cette classe est tout comme Partie un singleton.

Classes JoueurAction et TerritoireCase Ces deux classes sont des MouseListener. Cela signifie qu'elles "écoutent" ce que fait la souris. JoueurAction réagit lors d'un clic sur l'un des boutons que peut cliquer le joueur. Pour éviter de créer un Listener pour chaque bouton, un switch dans la méthode MousePressed a été intégré en fonction du bouton cliqué. Cela permet de rediriger vers le bon traitement. La classe TerritoireCase permet de savoir lorsque le joueur clique sur un territoire et permet d'identifier celui-ci.

Classes WinMenu, WinWait et WinWarn Ce sont les classes qui permettent de générer des informations temporaires à l'écran. La première WinMenu affiche un menu avec des options et enregistre le choix du joueur lors d'un clic sur une option. La classe WinWait donne simplement une information au joueur, par exemple "Joueur1 attaque". Enfin la classe WinWarn s'affiche tout en haut à droite de l'écran pour indiquer au joueur un choix impossible et lui expliquer.

Classe Prompt Cette dernière implémente MouseListener et KeyListener. Elle permet de créer une fenêtre qui demande une chaîne de caractère. Elle est utilisée dans l'application pour indiquer le nom des joueurs au début d'une partie.

Classe SQLite Enfin la classe SQLite, mentionnée un peu plus haut, permet de dialoguer avec le fichier .db contenant des informations sur les territoires du jeu. Elle permet d'aller chercher les informations sur ceux-ci puis de créer les territoires à partir de ces informations.

2.4 Problèmes rencontrés

Peu de problèmes ont été rencontrés lors de la réalisation de ce projet. Cependant, l'équipe a tout de même pu déplorer quelques problèmes d'ordre temporel :

- tout d'abord, l'organisation au niveau du travail n'a pas toujours été facile, les membres du groupe ne suivant pas tous les même cours. Les emplois du temps ne correspondaient pas du tout ;
- aussi, l'articulation de l'UV n'a pas permis d'assimiler les notions avant de mettre en œuvre le projet. Il était donc difficile d'implémenter certains mécanismes sans en connaître le fonctionnement ;

– enfin, le fait de rendre le rapport durant les vacances et le peu de délai avant celles-ci n'a pas permis de se concerter de façon optimale sur celui-ci.

3 Bilan

Le bilan de ce projet peut se concentrer sur deux points principaux : un bilan au niveau humain et au niveau pédagogique.

3.1 Bilan humain

Au niveau humain, cela a permis de découvrir ou de redécouvrir l'esprit d'équipe dans un projet. Cela a permis aux uns d'apporter des connaissances aux autres et inversement. Cela a permis aussi de connaitre de nouvelles personnes, plusieurs membres de l'équipe étant à l'UTBM depuis moins d'un semestre.

3.2 Bilan pédagogique

Le bilan pédagogique est plutôt positif. Tous les membres du groupe ayant déjà eu un contact avec le langage Java, cela a permis de consolider des bases dans ce langage de programmation et de découvrir quelques nouveaux mécanismes encore jamais abordés. Cela a permis aussi à certains d'utiliser le gestionnaire de version Git très répandu aujourd'hui.

Conclusion

Le projet qui a été posé à notre équipe a été réalisé dans son ensemble et est aujourd'hui parfaitement fonctionnel. C'est donc une conclusion positive puisque toutes les étapes ont été réalisées. Un cahier des charges a été mené, la conception UML a été faite. Le développement a lui aussi été mené correctement puisqu'il s'est déroulé sans encombre majeure. SmallWorldUtbm peut donc être utilisé sans problème par les plus joueurs d'entre nous afin de se divertir durant de nombreux et bons moments !

Annexes

Annexe 1

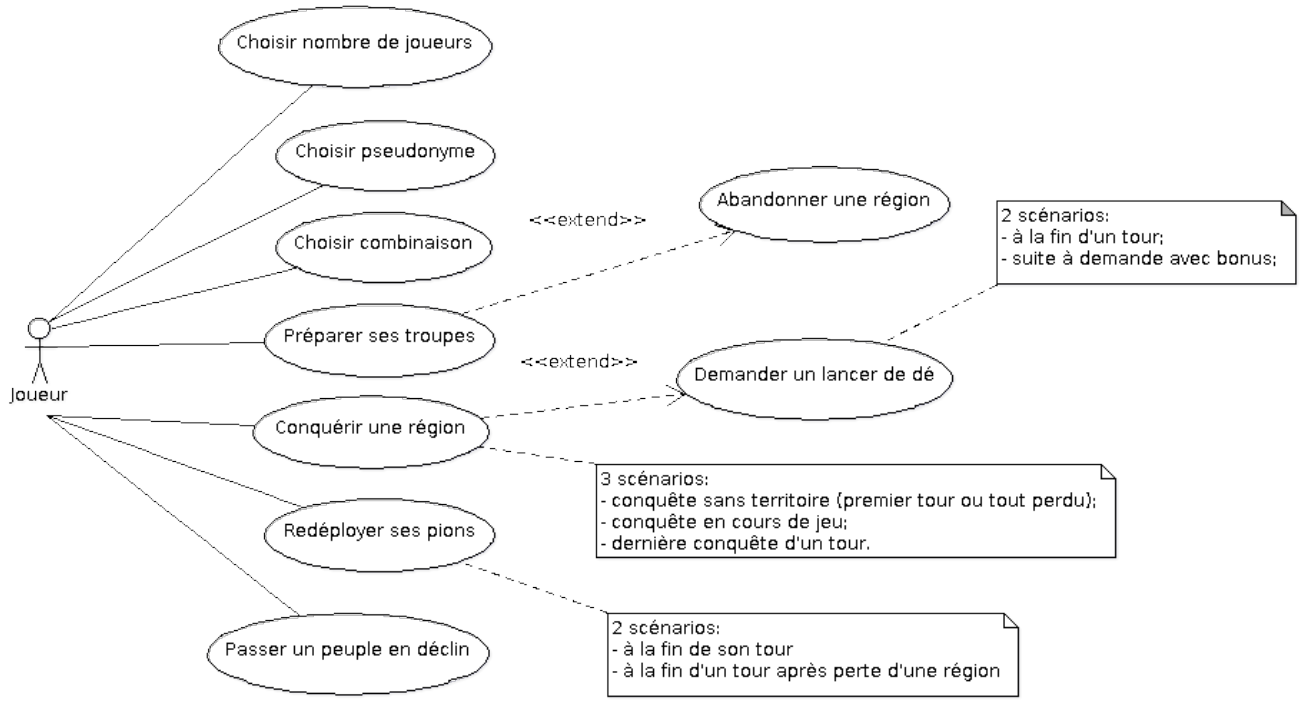

Diagramme de cas d'utilisation

Annexe 2

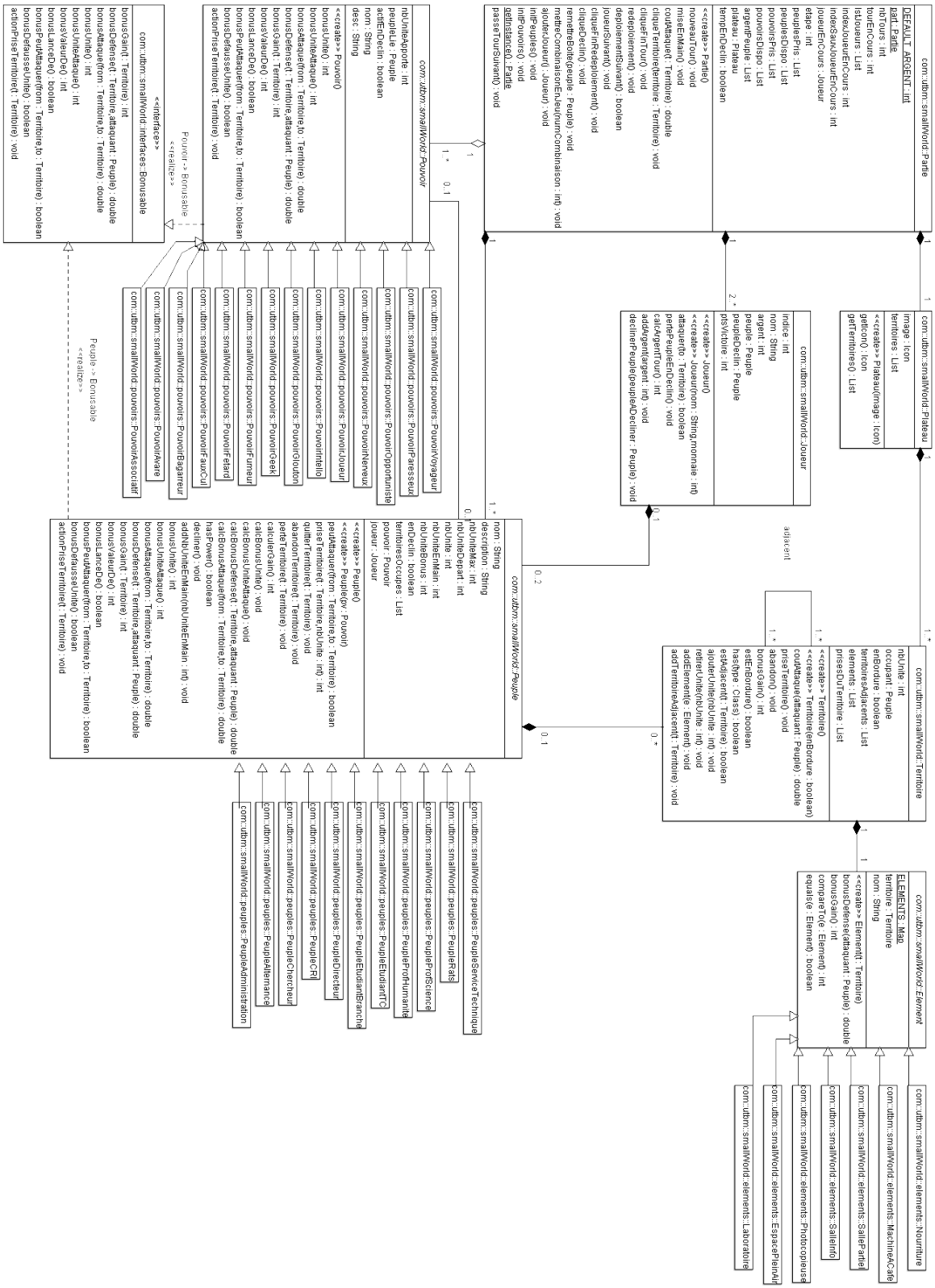

Diagramme de classe général de l'application

Annexe 3

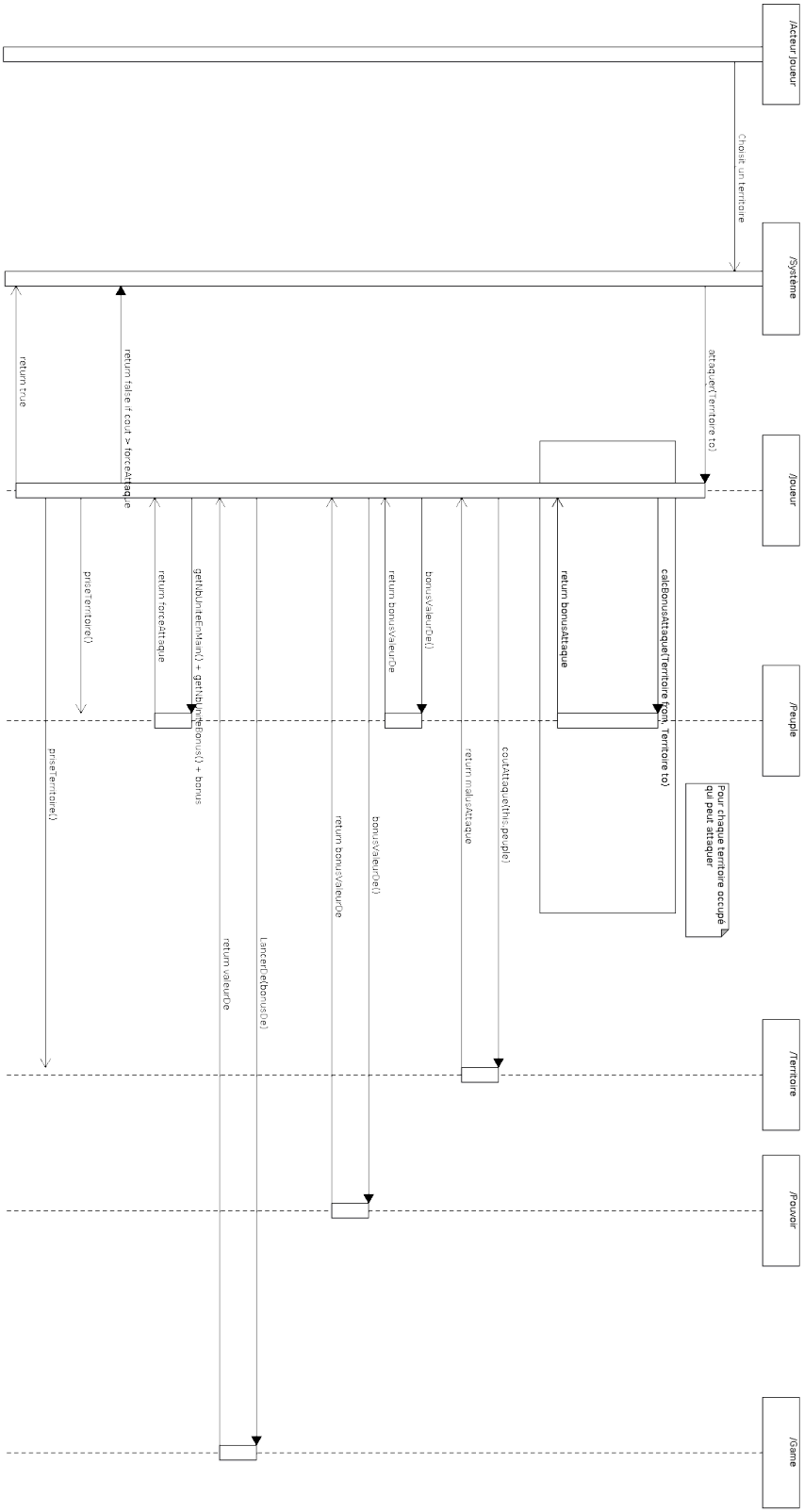

Diagramme de séquences de l'attaque d'un territoire

Annexe 4

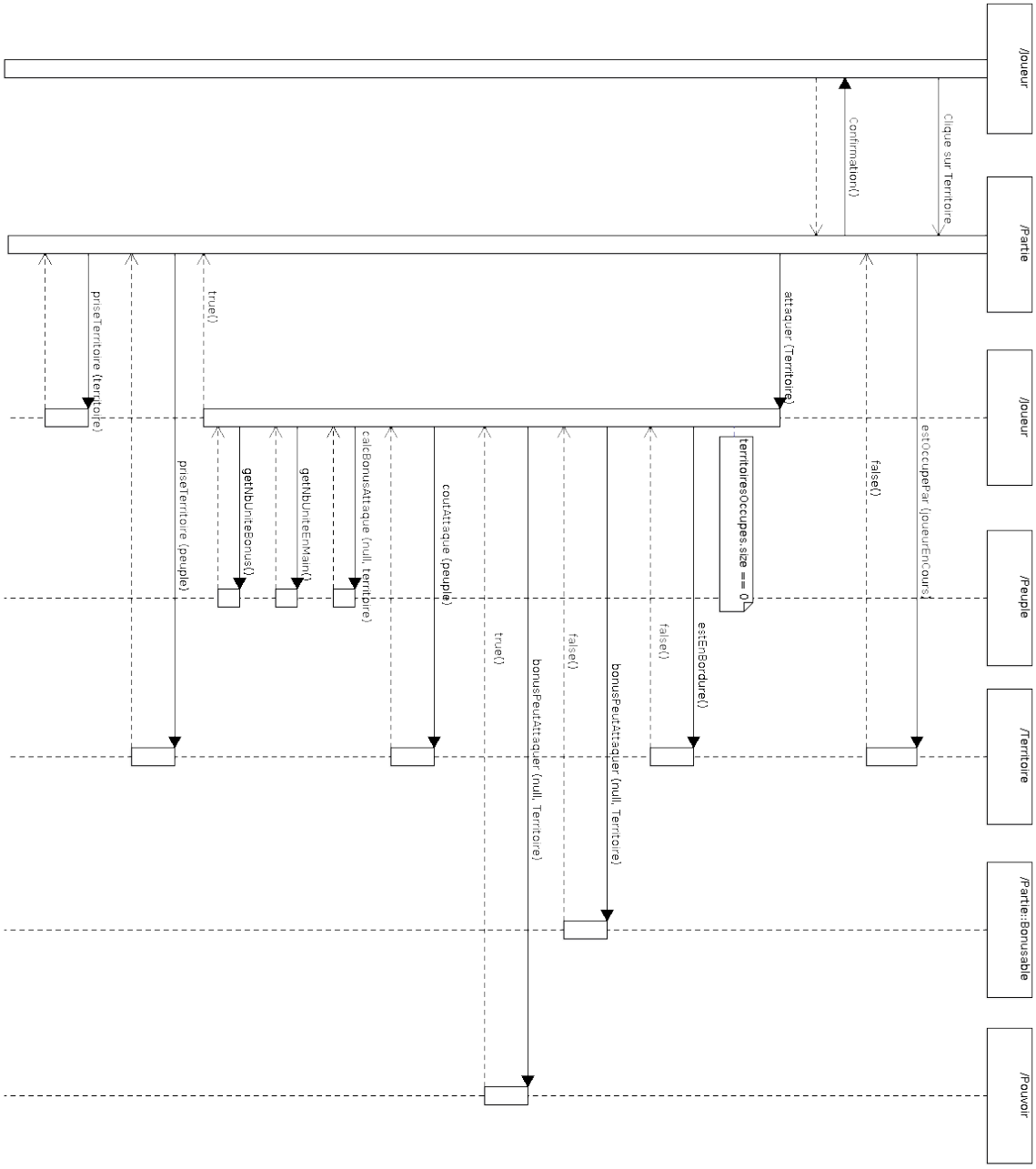

Diagramme de séquences de la prise d'un territoire sans occupant

Annexe 5

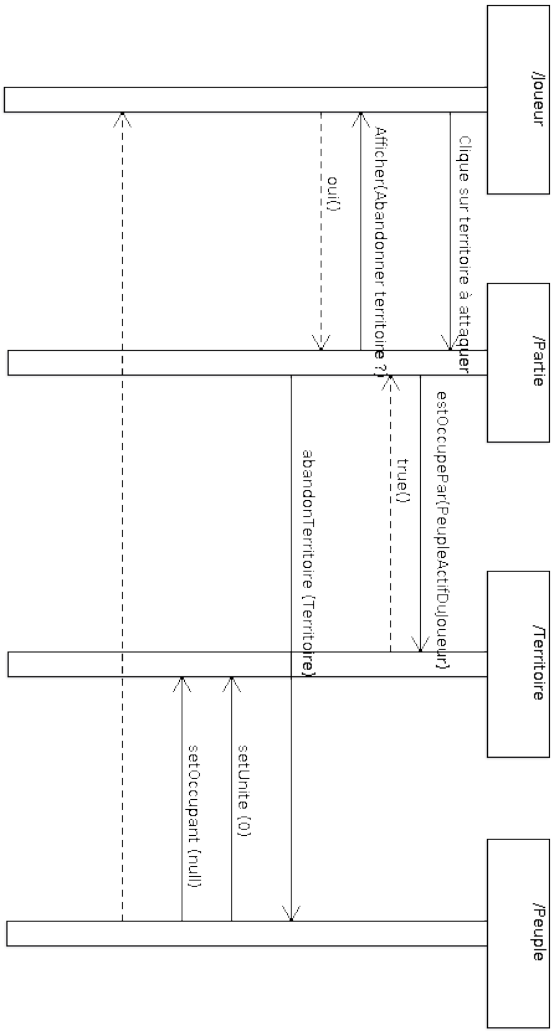

Diagramme de séquences de l'abandon d'un territoire

Annexe 6

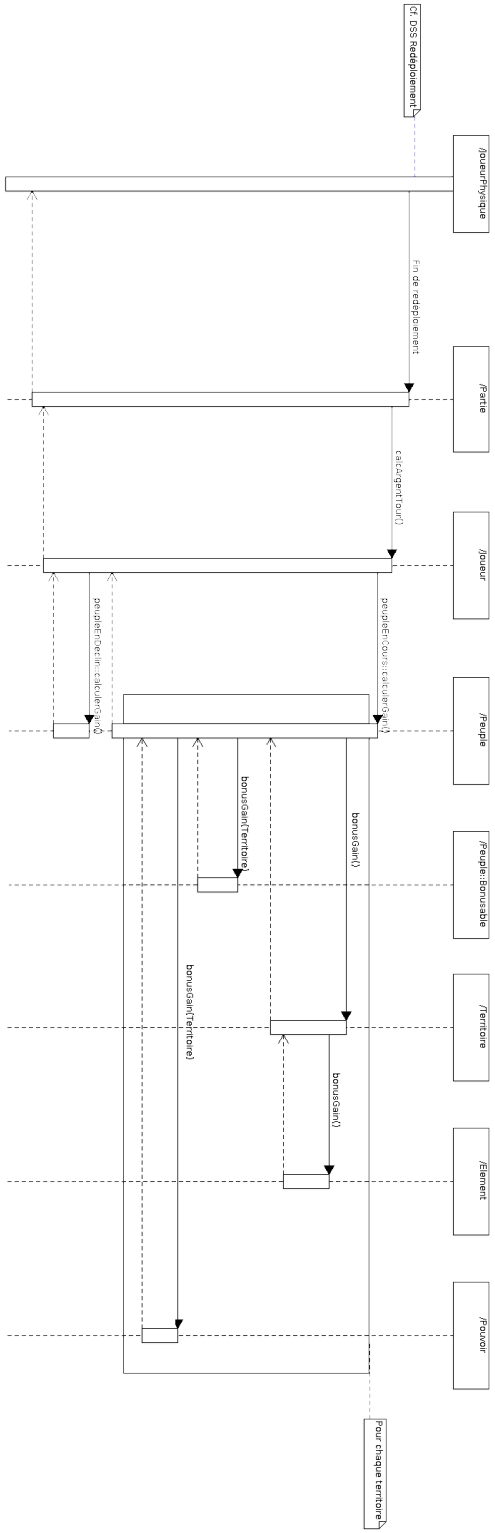

Diagramme de séquences du calcul du gain d'un joueur