Dufour Rémi
Cédrid Foning
Moyan Lei
Lifei Zheng

Adaptation du jeu de société Small World : rapport final

1. Objectif

L'objectif de notre travail de groupe est de mettre en pratique nos connaissances théoriques sur la programmation orientée objet et plus particulièrement sur le langage Java grâce à un projet concret : la création d'une adaptation du jeu de société Small World en un jeu vidéo, et en le modifiant pour correspondre à l'UTBM.

2. Adaptation

Afin d'adapter le jeu de société au monde de l'UTBM. Nous avons tout d'abord remplacé les concepts du jeu par des concepts lié à l'UTBM. Ainsi les peuples deviennent des départements de l'UTBM : TC, GI, UTSEUS, GMC, IMSI, EE.

De plus les pouvoirs spéciaux deviennent des particularités : geeks, alcoliques, professeurs, endormis, désordonnés, voyageurs.

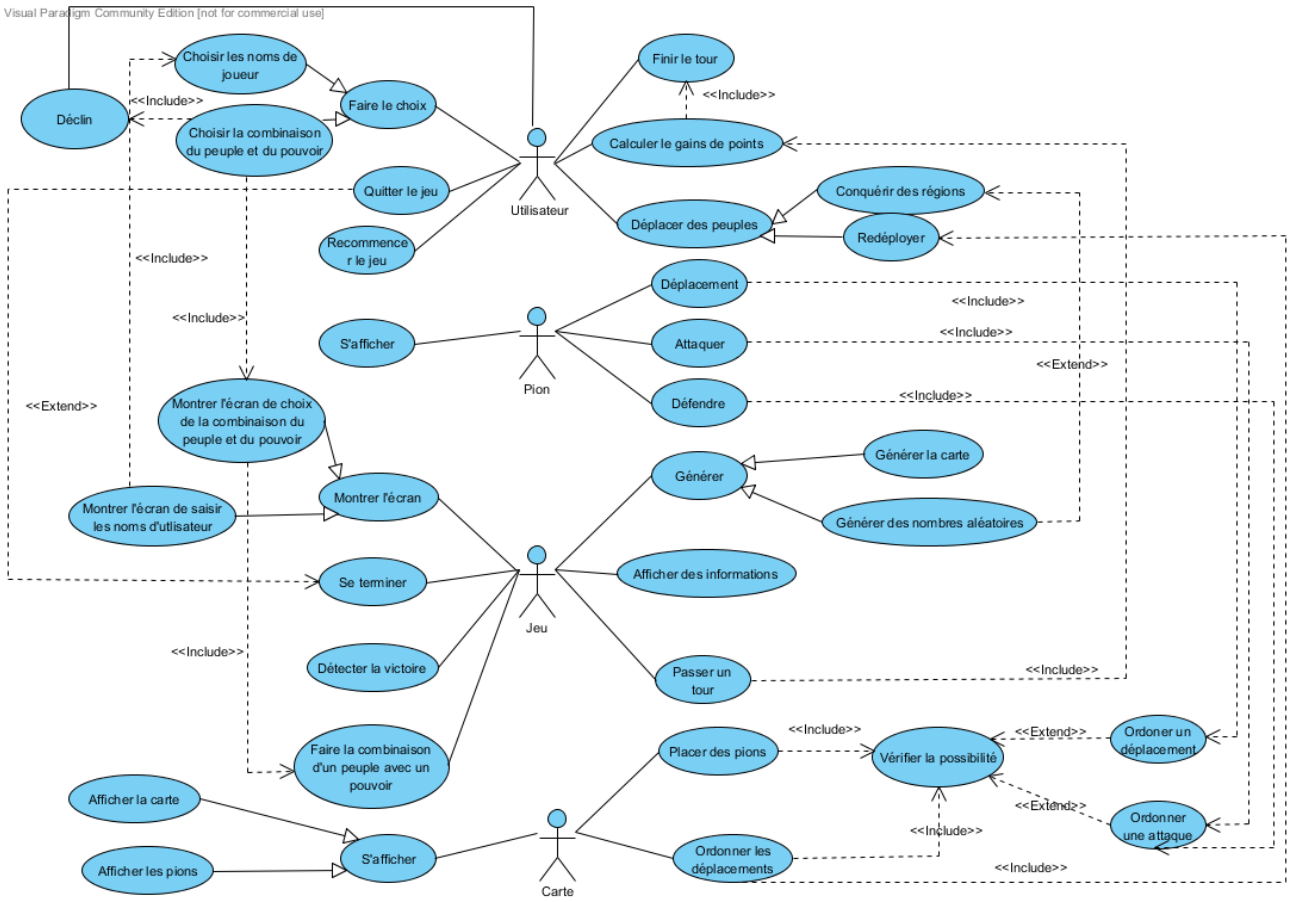
Ensuite nous avons remplacé les territoires par des salles : bar, gym, salle de td, salle de tp, salle de cours, dortoir.

Enfin, il est nécessaire d'adapter les règles du jeu de société à un jeu vidéo, et certains détails afin de les adapter à une interface souris/clavier et un écran. Ainsi la carte a été réduite à un quadrillage, la manière la plus simple de la représenter (grâce à un tableau à 2 dimensions), car l'apparence n'est pas le but principal de la création de notre application.

3. Conception

Nous avons donc commencé par la conception de la partie du design MVC n'étant pas liée à l'interface, le modèle, les compétences nécessaires pour créer l'interface n'ayant pas encore été apprises.

La première étape était d'analiser les différents cas que notre jeu doit être capable de gérer. Pour cela nous avons utilisé le diagramme de cas d'utilisation.



On a donc déduit 4 acteurs : la Carte, l'Utilisateur, le Jeu, et le Pion. La carte est un élément interactif qui doit s'afficher à l'utilisateur, qui peut par son intermédiaire conquérir des territoires à l'aide de pions, ou redéployer ces pions.

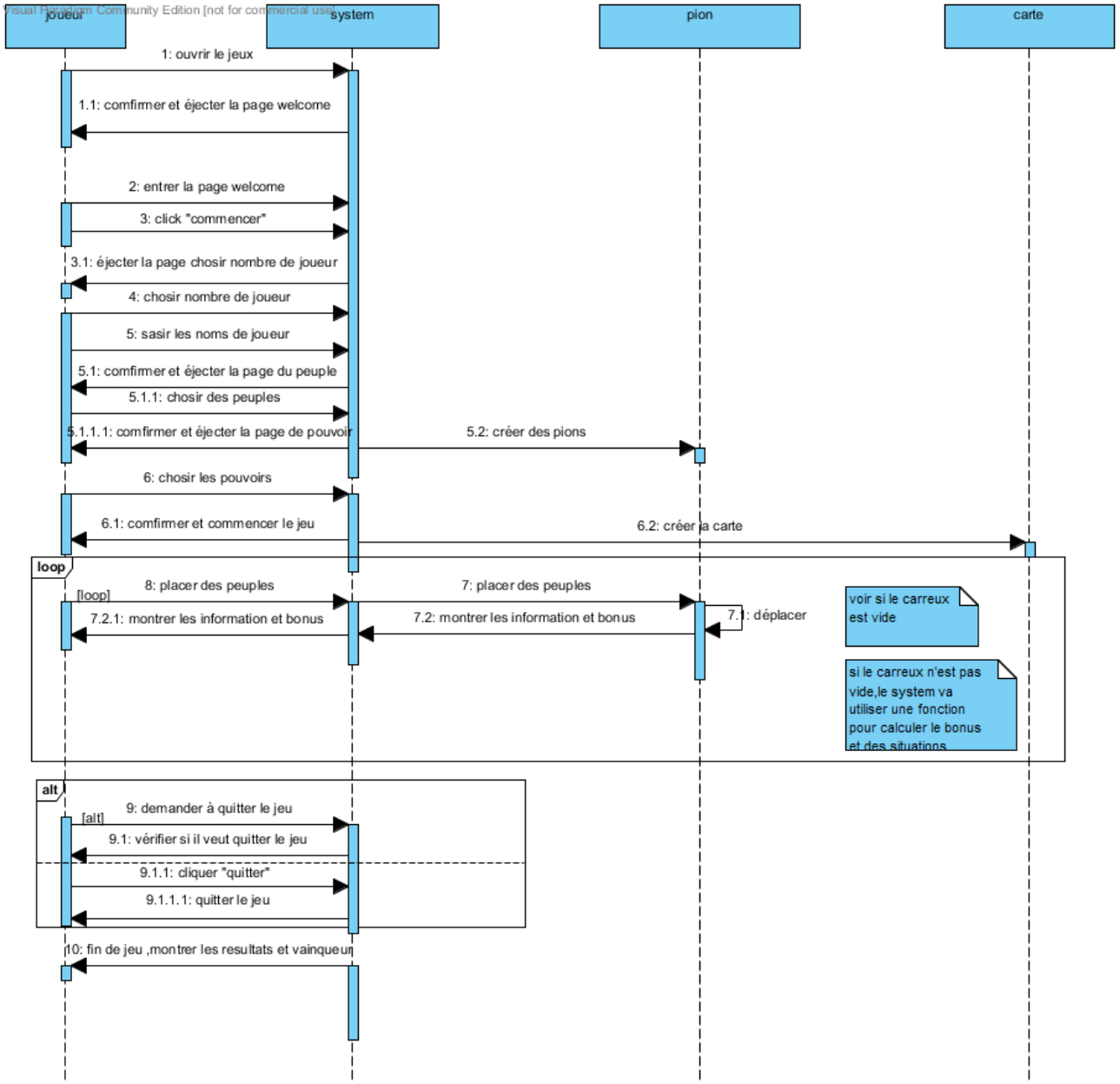
L'Utilisateur représente la ou les personnes interagissant avec le jeu, il peut choisir le ou les peuples et pouvoirs qui seront utilisés par chaque joueurs. Il peut interagir avec la carte et les pions afin de conquérir des territoires, déployer ou redéployer des pions, déclarer son peuple en déclin, finir le tour du joueur actuel, dans le but ultime de posséder le plus de points de victoires une fois le nombre de tours maximum atteint. De plus il dispose à tout moment de la possibilité de quitter le jeu, ou bien de le recommencer.

Le pion représente l'unité de base, disposant d'un peuple et d'un pouvoir, capable de se redéployer sur la carte ou de conquérir un territoire sur ordre du joueur et si le nombre de pions assaillants est suffisant. Un pion doit pouvoir s'afficher au joueur afin que celui-ci puisse le déplacer et connaître l'état actuel de ses forces.

Le jeu gère les tâches globales tels que la génération initiale de la carte, l'affichage des informations de chaque joueur à l'utilisateur, le choix initial et lors des déclins des peuples et pouvoirs, ainsi que le choix du nombre de

joueurs et de leur nom.

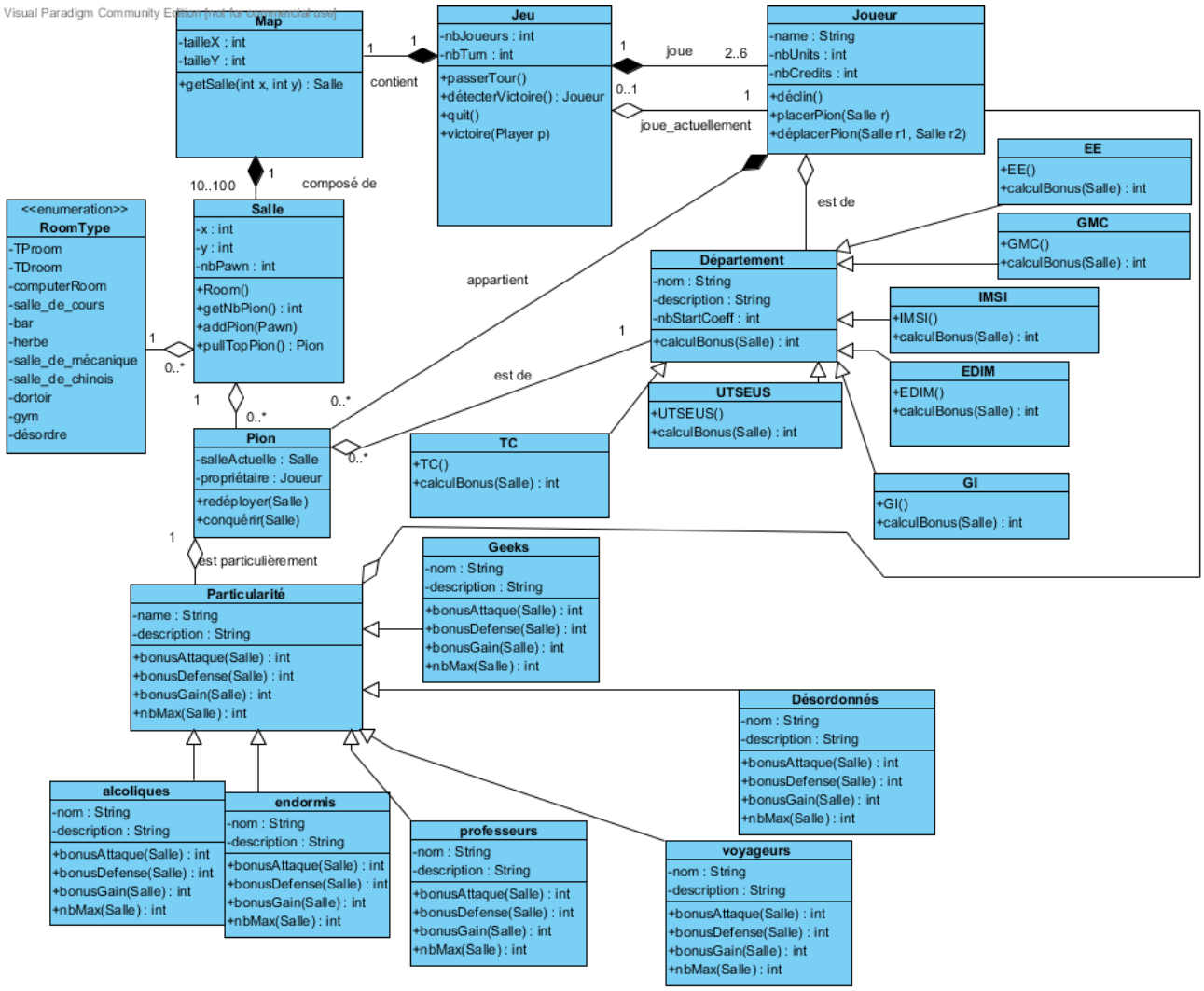
Ensuite, comme toutes les actions que peuvent effectuer les acteurs de notre jeu ne pourront pas forcément être effectuée à tous les moments et dans tous les contextes, nous avons établi un diagramme de séquence pour clarifier comment et dans quel ordre chaque étape se déroule.



Pour jouer à ce jeu, les joueurs doivent d'abord ouvrir le jeu, ensuite entrer dans la page welcome. Puis ils saisissent leurs noms. Ensuite ils vont choisir les combinaison de peuple et de pouvoir. Enfin le système créé la carte et place des pions neutres et le jeu commence. Chaque tour , les joueurs peuvent déplacer

des pions pour occuper les carreaux , pour attaquer les autres etc. Ils peuvent également quitter le jeu. Finalement ,à la fin de jeu ,le système va informer les joueurs des résultats et du vainqueur. Enfin ils peuvent recommencer une autre partie ou quitter le jeu.

Après avoir établi les besoins auxquels doit répondre notre adaptation, nous avons donc pu organiser des classes afin de répondre à ces besoins.



Ainsi le jeu contient les joueurs inscrits, et de plus 2 joueurs ne peuvent pas jouer en même temps, il est donc nécessaire de disposer également du joueur dont c'est le tour. Le joueur possède un département et un pouvoir qu'il confère aux pions qu'il place. Chaque département hérite d'une classe département et chaque particularité hérite d'une classe particularité. La carte est composée de salles qui contiennent des pions et qui sont chacune d'un certain type.

4. Organisation du travail

Afin de mener notre projet à son terme, nous nous sommes séparés les tâches à effectuer. Ainsi Rémi, Lifei et Cédric se sont occupés de l'interface : affichage et gestion des événements, tandis que Moyan a codé les packages département et pouvoirs. Plus précisément, Rémi s'est chargé de l'affichage de la carte et des fonctions permettant de déplacer les pions, Lifei s 'est chargé des panels permettant de choisir le noms de joueurs ainsi que leurs peuples et leur pouvoir, Cédric de l'affichage d'informations aux joueurs, par exemple la liste des joueurs et de leur peuple et pouvoir, ainsi que l'affichage du vainqueur une fois le jeu terminé.

De plus, nous avons choisi d'utiliser git pour gérer les changements successifs effectués, principalement pour sa simplicité des utilisateurs de gestionnaire de gestion débutants. Certaines pratiques nécessaires furent suivie, comme éviter de travailler à plusieurs en même temps sur le même fichier afin de ne pas causer de conflit.

5. Développement

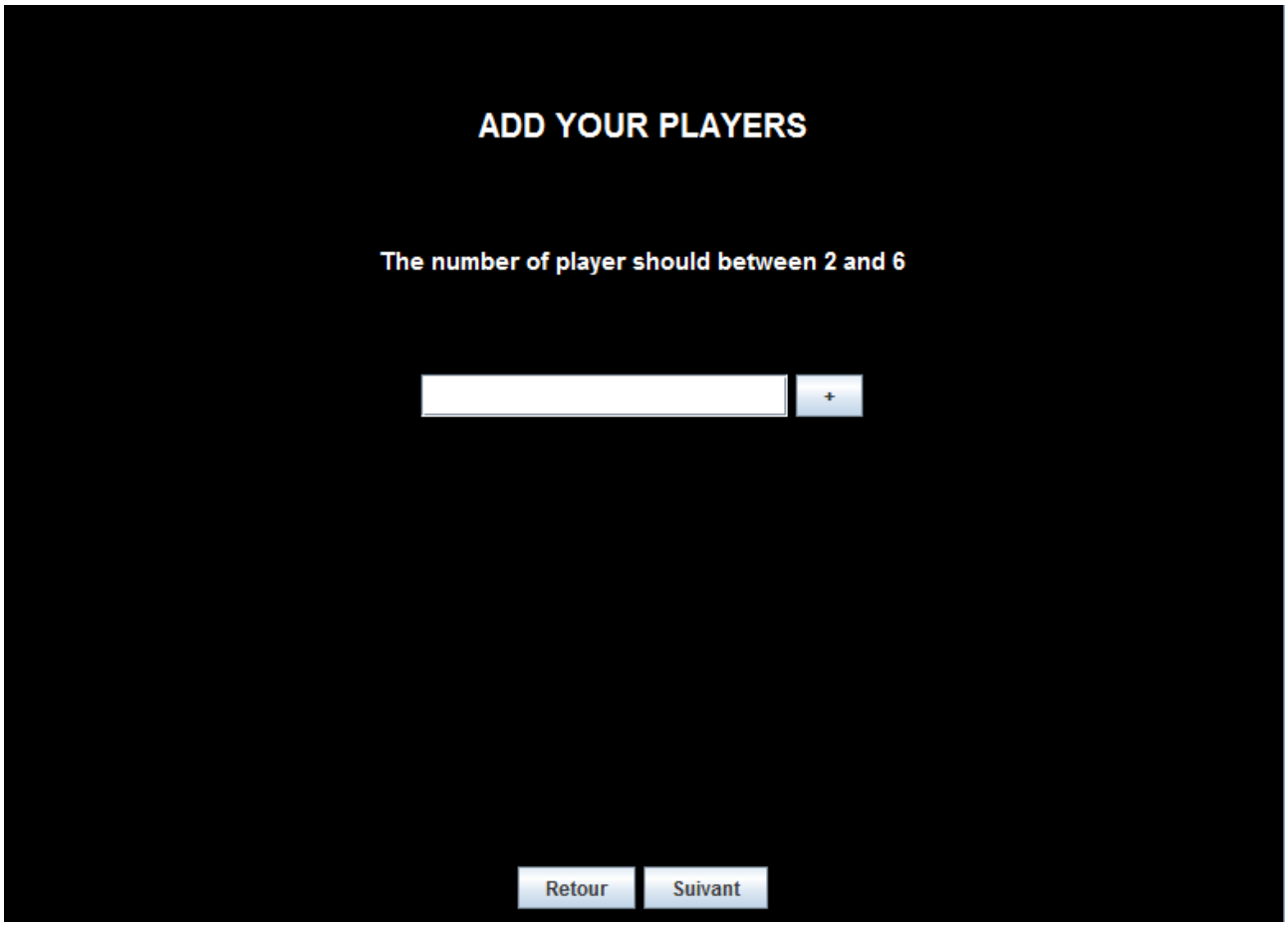
Tout d'abord nous devions décider de l'interface à implémenter. Il est nécessaire qu'avant le début du jeu, les joueurs puissent s'inscrire et choisir leur peuple. Plusieurs possibilités étaient alors possibles, par exemple : tout d'abord un écran pour choisir le nombre de joueurs, puis une fenêtre pour choisir leurs noms et enfin leur peuple, ou bien faire tous ces choix dans le même panel, avec par exemple un clique sur un peuple, puis le choix du nom du joueur, ainsi de suite pour chaque joueur.

La solution que nous avons choisi est une fenêtre permettant d'inscrire chaque joueur, puis un écran pour choisir le peuple de chaque joueur. De plus il était nécessaire de garder à l'esprit que lors d'un déclin, le joueur doit choisir un nouveau peuple, et qu'il serait donc utile de pouvoir réutiliser le Panel de choix des peuples en cas de déclin.

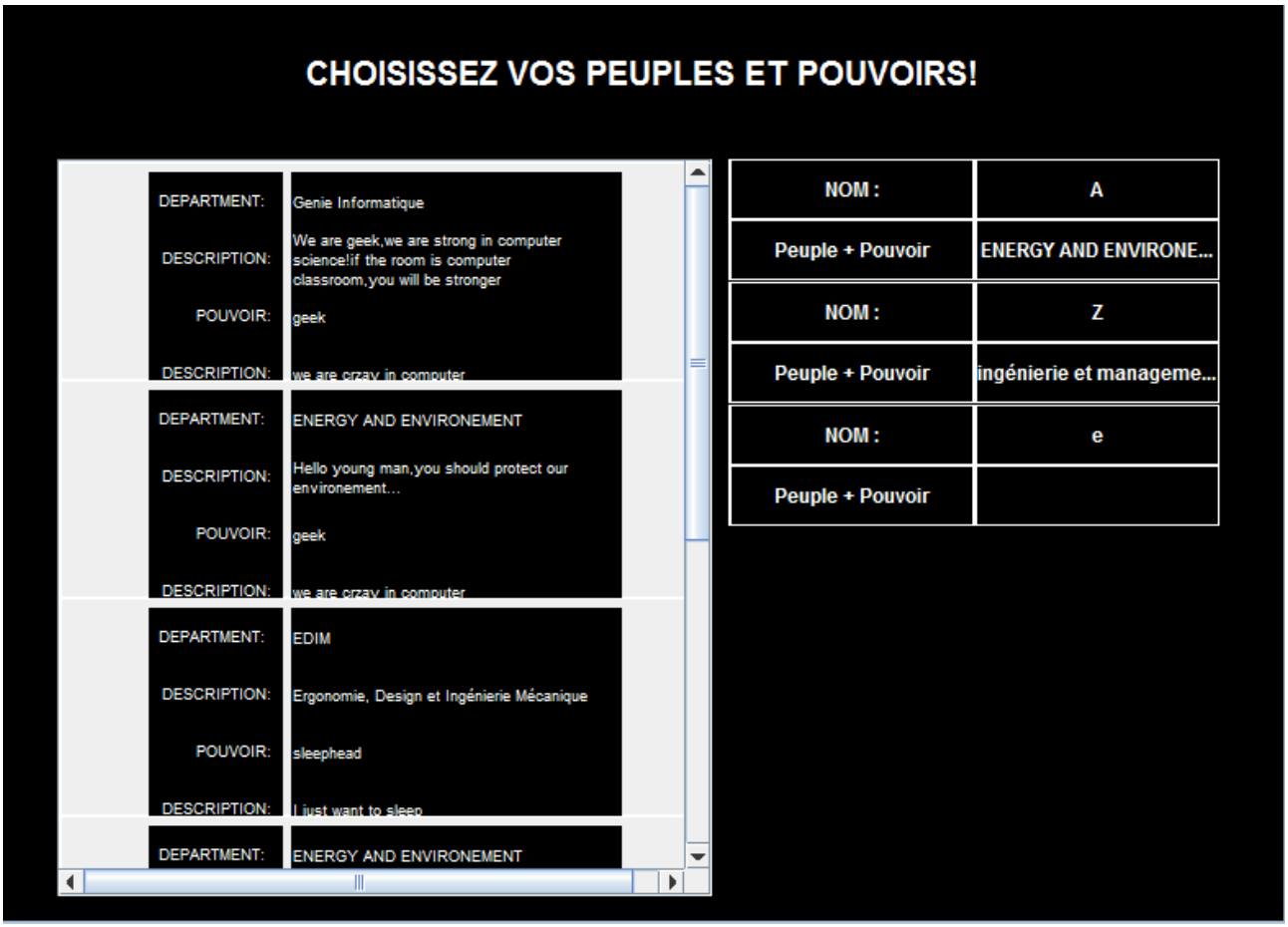
Notre interface serait donc composée de plusieurs panels qui s'afficheraient à l'écran : un panel d'accueil, un panel pour inscrire les joueurs, un panel pour choisir le peuple de chaque joueur, un panel pour le jeu lui-même , et éventuellement un panel pour afficher la victoire. Un seul de ces panels serait affiché à la fois, le panel affiché changeant à chaque étape.

La solution que nous avons choisi pour passer entre tous ces panels est de tous les placer dans un panel global, contenant tous ces panels, et disposant de méthodes pour en changer.

L'écran d'accueil affiche un message d'accueil et permet à l'utilisateur de commencer un jeu ou de lire le tutoriel, une option que nous avons décidé d'implémenter si nous en avions le temps.

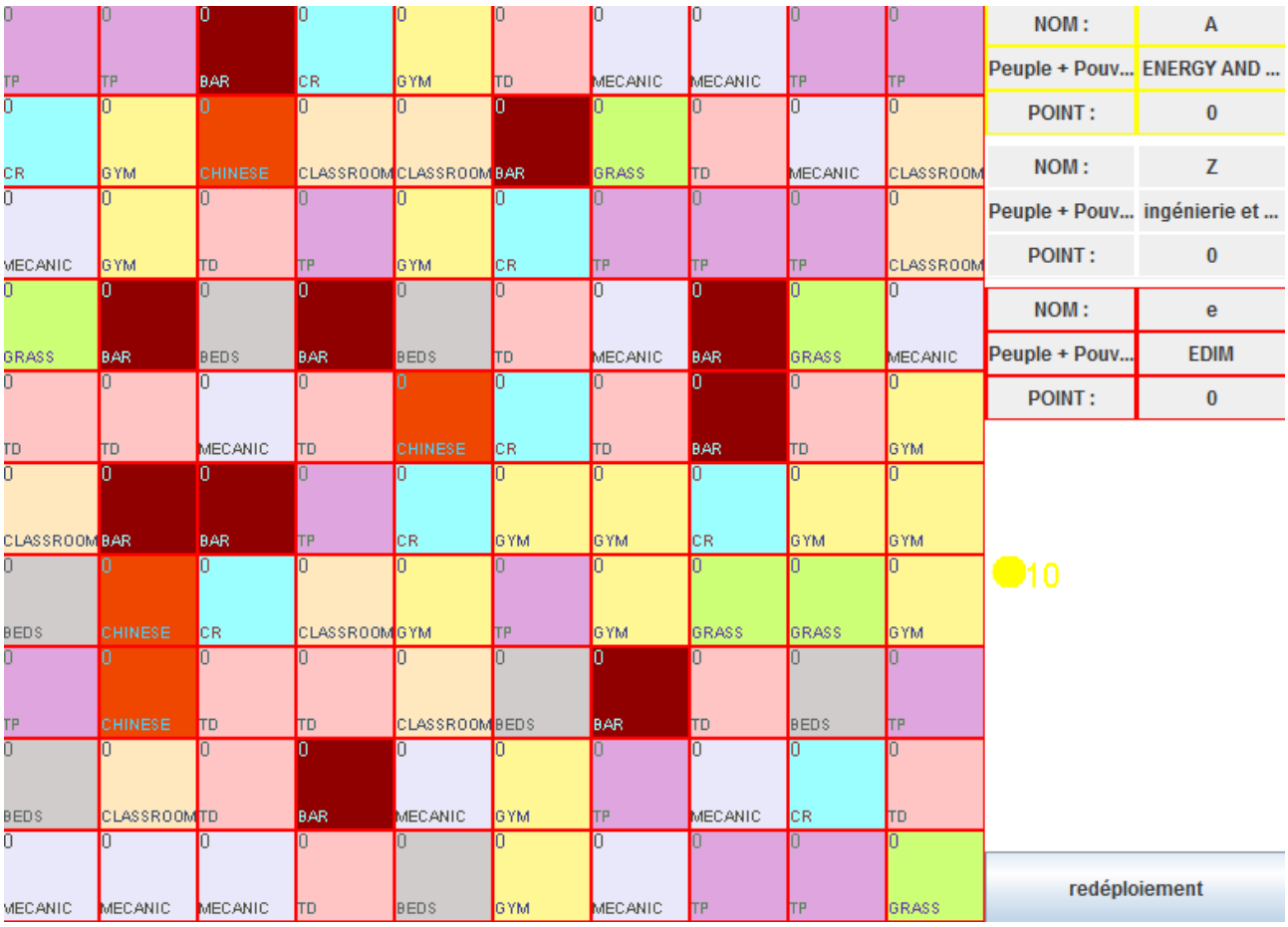


L'écran suivant permet d'inscrire les joueurs. Chaque joueur peut ici inscrire son nom ou pseudonyme dans le JtextField au milieu, puis valider son inscription en cliquant sur le bouton avec le symbole « + ». Le programme vérifie que 2 joueurs n'ont pas le même nom, et que leur nombre se situe entre 2 et 6, alors seulement l'utilisateur peut passer au panel suivant en cliquant sur le bouton « suivant ». Au cas où l'utilisateur veut revenir vers le panel d'accueil, il peut cliquer sur le bouton « retour », ce qui réinitialise au passage la liste de joueurs.

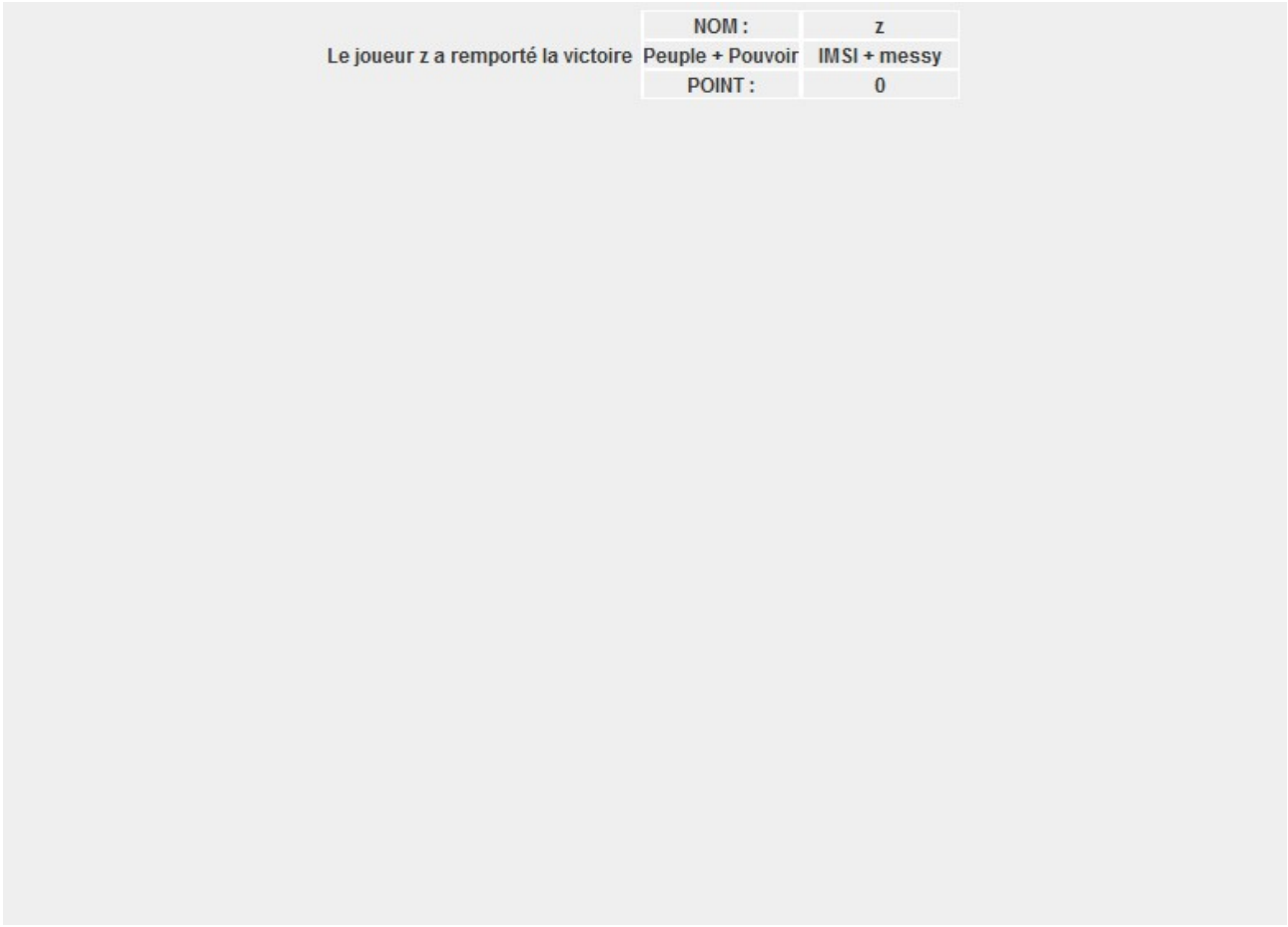


Ensuite les joueurs peuvent chacun choisir leur combinaison peuple/pouvoir. Une liste de 6 combinaisons est générée aléatoirement et affichée dans un JScrollPane, le joueur peut donc ensuite faire son choix en cliquant sur l'une d'elle. Le jeu commence automatiquement quand tous les joueurs ont fait leur choix. De plus quand une des combinaisons est choisie, elle est retirée de la liste.

Enfin, le jeu commence. Encore une fois nous avons du choisir une interface adaptée. Il était nécessaire d'afficher la carte composée de salle, et d'afficher pour chacune d'entre elles le nombre d'unité l'occupant. De plus les joueurs ont besoin d'informations telles que le nombre de points de chaque joueur, leur peuple, le nombre de pion disponible pour le joueur actuel, et l'identité de ce dernier. Nous avons donc décidé d'afficher la carte du côté gauche de l'interface et ces informations du côté droit, et d'afficher le nombre de peuple disponible pour le joueur actuel à coté du curseur. De plus, le programme assigne à chaque joueur une couleur, et un curseur circulaire de la couleur correspondante indique lequel est en train de jouer. Enfin le joueur doit disposer d'un moyen de passer à la phase suivante du jeu : le redéploiement puis le tour suivant. Nous avons donc placé un bouton remplissant ce cahier des charges en bas à droite.



Le joueur peut, dans la phase de conquête, prendre les cases du bord de la carte en utilisant le clic droit. Quand il essaye de conquérir une case alors qu'il ne peut pas, un nombre aléatoire entre 0 et 6 décide le nombre de renfort que l'on reçoit. Ces renforts sont gardés si la conquête fonctionne uniquement, puis la phase de redéploiement démarre aussitôt. Dans cette phase le joueur peut prendre dans sa main des unités dans les salles qu'il possède grâce au clic gauche, et les déposer grâce au clic droit. Dans ces 2 phases il a également la possibilité d'appuyer sur le bouton du milieu de la souris pour passer en déclin, causant l'abandon de son peuple actuel : ses salles perdent toutes leurs unités sauf une, et il en perds la possession. Il gagne néanmoins encore 1 points de victoire par case en déclin possédée. Au bout de 10 tours, le jeu se termine et le vainqueur est décidé : le joueur possédant le plus de points de victoire gagne.



5. Difficultés rencontrées

Nous nous sommes tout d'abord rendu compte de nos problèmes de conceptions. Ainsi nos diagrammes UML étant trop théoriques, des problèmes pratiques ont imposés des changements. Par exemple les diagrammes n'incluant pas la gestion des événements, des méthodes ont du être rajoutées, puis des considérations pratiques ont causé des restructurations de notre organisation.

De plus le pattern MVC n'a pas été implémenté de manière optimale. Nous avons globalement séparé les classes en 3 packages, view, controller et model, mais certaines classes comme les départements et les pouvoirs n'entraient pas vraiment dans ce pattern et sont donc dans leur propre package. De plus l'interaction entre view controller et model et globalement bien gérée, mais certains détails ne suivent pas ce pattern correctement. En particulier l'affichage d'un rectangle de sélection autour d'une salle se fait par une interaction directe entre le controller et la view, de même pour l'inscription des joueurs. De plus les interactions entre les 3 parties sont inconsistantes, ainsi parfois des classes par constructeurs, parfois par un setter. Des interfaces observable/observer auraient pu remédier à ce problème mais cela aurait nécessité de restructurer complètement le projet.

Nous avons également du apprendre à utiliser des layouts tels que GridBagLayout afin de faire notre panel de jeu, mais nous aurions souhaité disposer de plus de temps pour apprendre à les utiliser afin de pouvoir faire une fenêtre de taille variable. Malheureusement nous n'avons pu faire qu'une fenêtre de taille fixe.

Un problème lié aux MouseEvents est également apparu : les composants à l'intérieur d'un panel les intercepte, ce qui causa des complications pour l'affichage du curseur personnalisé

(affichant le nombre d'unités à placer). En effet le panel de jeu ne recevait pas le MouseEvent et ainsi le curseur ne pouvait pas être mis à jour. Nous avons corrigé ce problème en utilisant la méthode dispatchEvent(AWTEvent e) pour rediriger le MouseEvent quand il s'agit d'un mouvement de souris.

5. Conclusion

Ce projet fut une expérience intéressante pour plusieurs raison. Tout d'abord c'était une occasion de travailler sur un projet de plus grande envergure que d'habitude, impliquant ainsi l'utilité de la phase de conception. Nous avons ainsi pu utiliser ce que nous avons appris en cours concernant les diagrammes UML afin de planifier notre projet. De plus ce projet nous a permis de mettre en pratique nos connaissances afin de mettre en place une interface graphique, ce qui fut la première fois pour un certain nombre d'entre nous. Ce fut également l'occasion de travailler en équipe, avec un groupe multiculturel. Nous avons ainsi appris à s'organiser et à se coordonner de manière efficace, ce qui sera une leçon utile dans le futur. Ce fut également un travail de conception théorique intéressant, la mise en pratique de nos diagrammes théoriques ne s'étant pas passé sans problème. Ainsi ce projet nous a beaucoup rapporté en expériences qui nous seront utiles dans le futur.