Interfaces homme machine

Rapport du projet « Love letter »

Mehmet Ozkan (alias dm67x)
Jawed Bernardone
Emine Bernardone
Marija Kirandjiska
UFR de Mathématique et Informatique
Université de Strasbourg L3 S6

Avril, 2018

UFR Mathématique Informatique Strasbourg

UNIVERSITÉ DE STRASBOURG

Planification

Avant de réfléchir au code et conception du jeu, nous avons bien étudié le sujet de projet, nous avons acheté le jeu de cartes et nous l'avons testé plusieurs fois pour mieux comprendre les règles du jeu et les stratégies possibles pour l'intelligence artificielle. L'achat du jeu a permis de nous clarifier les idées avant de s'attaquer concrètement au projet.

Ensuite nous avons commencé à réfléchir à l'organisation et structuration de notre projet en prenant en compte tous les modules demandés et comment est-ce qu'on peut les séparer pour faciliter la lisibilité du code, le travail en équipe et la détection des erreurs.

Nous avons pris soin de réaliser des ébauches d'UML des classes du noyau et notre premier défi a été la façon de modéliser les effets des cartes et en particulier l'héritage de chaque carte d'une classe abstrait « carte », prenant en considération que chaque carte a des effets différents et par conséquent elle prend un nombre diffèrent des paramètres comme par exemple un joueur cible et une carte devinée.

Répartition du travail, environnement et organisation

Nous avons réfléchi ensemble à la conception et organisation du projet et nous avons décidé de concevoir ensemble le noyau, ainsi nous avons réparti les classes de la manière suivante : Jawed a fait la classe du Jeu (« Game »), Mehmet a implémenté les cartes et leurs effets (« Card », « Action »), Marija a travaillé sur la classe des joueurs (« Player ») et Emine était responsable de l'écriture des fichiers de test. Cependant au fur et à mesure on a rencontré des problèmes dans la conception de notre noyau et pour cela Mehmet a créé un prototype fonctionnel selon lequel nous avons adapté ensuite notre travail. Pour la suite, Marija a travaillé sur l'Intelligence artificielle, Jawed sur la partie réseaux et Mehmet et Emine ont travaillé sur l'interface graphique.

Concernant les différents environnement de travail, notre équipe travaillait à la fois sous Windows et Ubuntu. Bien que ça puisse sembler bizarre pour une équipe d'utiliser des technologies différentes, cela nous a néanmoins permis de voir très rapidement comment les différentes façon dont réagissait le projet chez les uns et les autres.

Côté code et IDE, nous avons décidé de réaliser le jeu en C++ (11), sous QTCreator. La raison en est très simple, nous avons eu des heures de TP de cette façon, pourquoi ne pas en tirer profit ? Par ailleurs, ces technologies nous sont assez nouvelles et l'idée de se lancer sur autre chose que ce qui est déjà connu nous a tous motivé.

Nous avons utilisé Git pour superviser le projet. Permettre à chacun de travailler sur sa branche tout en ayant la possibilité de consulter celles des autres.

Développement

Nous avons décidé d'implémenter notre projet comme un projet contenant des sous-projets (« subdirs project ») qui représentent les différents modules, ceci dans le but de les bien séparer, afin d'augmenter la lisibilité du code et faciliter le travail en équipe. De cette façon, nous distinguons clairement les différentes parties. Le Noyau est dans un dossier "core", mais est aussi un projet sous QT à part entière. Il est donc possible de le compiler en tant que librairie qu'on peut alors charger dans nos autres modules.

Interface graphique

L'interface graphique est fait en C++ (tout comme les autres parties) avec la technologie SFML (librairie multimédia). Nous avons choisi de partir sur cette librairie contrairement à QT car nous avions plus de manoeuvre et plus de liberté sur ce dont on voulait à tout pris avoir sur ce projet. Néanmoins, qui dit liberté dit responsabilité et dur à prendre en main. Ce dont on a regretté un peu plus loin au sein du projet lorsque la deadline approchait à grand pas.

L'organisation était très simple, comme nous étions que deux sur cette partie du projet nous avons décidé de coopérer sans pour autant pénaliser l'autre. Nous avons mis en place un diagramme de Pert simple pour nous faire une image de comment nous allions organiser les tâches pour justement ne pas empiéter sur le terrain de l'autre. Du coup, après moultes réflexions nous avons décidé comme suit, j'ai (Mehmet Ozkan) fait la majeure partie de ce qui concerne les éléments Card, Hand, PlayerZone, Board et quelques éléments du menu avec en prime un panel pour l'effet du garde, tandis que Emine Bernardone s'est occupé de faire les écrans de transitions, les liaisons entre ma partie et les écrans.

Nous avons également décidé de séparer les classes dans différents répertoires en fonction de leurs fonctionnalités, par exemple, le répertoire "Elements" contient tous les éléments du plateau, c'est à dire, la classe "Card", "Hand", "PlayerZone" et "Deck". Cela permettait de se retrouver plus facilement dans les fichiers.

Par contre, le gros point noir du projet se situait dans le délai mal jugé par notre part. Nous avons estimé une durée trop courte de ce que cela représente vraiment. Et nous en avons payé le prix fort, car nous n'avons pas réussi à implémenter la partie "DiscardZone" où toutes les cartes jouées par le joueur se retrouve, également délaissé

les animations que nous avions fait mais pas pu implémenter dans les éléments du jeu.

Mais néanmoins, même avec cela nous avons essayé dur comme fer de finir le projet en se battant jusqu'à la fin. Je sais que l'on a pas produit ce que l'on escompté mais nous en sommes tout de même fier.

Intelligence artificielle

L'implémentation de l'intelligence artificielle consiste de deux types d'IA, la première étant un IA bête (« IA_dummy ») adaptée pour maximum quatre joueurs et la deuxième étant l'IA qui a un peu plus de réflexion (« IA_intel ») adaptée pour 2 joueurs.

La première IA choisit une carte pour jeter, un joueur cible et une carte devinée de manière aléatoire, avec la seule contrainte de ne pas jeter la princesse et ainsi se suicider.
La deuxième IA a une stratégie de jeu sûre et prends pas trop de risque. A chaque tour elle calcule les probabilités d'apparition de toutes les cartes dans le paquet et fait une mise à jour d'une probabilité de 1 pour les cartes qu'elle a connu à travers un prêtre ou un roi qui peut être jouée de deux côtés.

A partir de là, sa stratégie fonctionne de la manière suivante : cette IA ne jette jamais la princesse si elle la possède, si elle connaît avec sûreté la carte de son adversaire et elle possède un garde, elle va jouer ce dernier, seulement si son adversaire n'est pas protégé par la servante.
Ensuite si cette carte connue en particulier est la princesse alors le prince a l'avantage d'être joué, ensuite le baron seulement si la carte connue est inférieure à la carte que l'IA possède.

De plus s'il s'agit du dernier tour il est intéressant de jouer le Roi seulement si la valeur de la carte connue est supérieure à la valeur de la carte que l'IA possède en main.

Si c'est le dernier tour et l'IA ne connaît pas la carte de son adversaire, elle jettera toujours la carte avec la valeur plus petite, histoire de maximiser sa chance pour gagner la manche.

Dans tous les autres scénarios, on jette le baron seulement si on a un roi, une comtesse ou la princesse en main, on préfère le prêtre avant la servante, la servante avant le prince et le roi et on préfère le prince entre le prince et le roi.
Si on ne se retrouve dans aucune de ses scénarios, on joue un choix aléatoire de carte et si on doit cibler une carte par le guard alors on prend celle avec la probabilité la plus grande qui n'est pas un guard.

Pour l'instant, l'implémentation de la deuxième IA ne fonctionne pas totalement. En effet, quelques bugs nous empêche de terminer une partie.

Réseaux

Cette partie nous semblait très importante à implémenter, étant donné que nous trouvions vraiment bizarre l'idée de jouer à 2-3-4 sur le même ordinateur. Jouer à distance semble beaucoup plus intuitif, étant donné qu'en Local, les joueurs sont tous sur le même écran et certains doivent se tourner pour ne pas voir les cartes de la personne qui joue.

Cependant, pour qu'on puisse développer un jeu en Réseau efficace, il faut déjà qu'on ai définis et bien réussi le jeu multijoueur sur un même ordinateur. En terme de développement, il "suffit" de calquer le jeu en Réseau sur le jeu Multijoueur Local. Nous avons donc essayé de rapidement mettre en place un prototype de jeu Local pour lancer au plus tôt le développement du jeu en Réseau.

Pour aborder le jeu en Réseau, nous avons décidé qu'on aurait :

- 1 serveur, hébergé par une personne
- Un programme "host", lancé par la personne qui héberge
- Un programme "client", lancé par les autres, qui rejoignent la partie

Pour communiquer, nous utilisons une transmission TCP car il assure une fiabilité dans l'ordre et la transmission des données.

Chaque joueur possède sa partie en local, quand c'est au tour d'un adversaire, on ne le fait pas jouer, mais on remplace ses actions par les données reçues.
Au final, l'implémentation de cette partie à causé beaucoup de problèmes, dû au fait qu'en "mergeant" l'interface avec la partie "network" on s'est rendu compte que les tests réalisés dans "network" (basés sur un prototype) ne correspondait plus à la réalité du jeu qui se déroulait sur notre branche "Master".

Notre erreur a été de partir trop tôt sur la branche "network". Un manque de mise en commun de nos éléments à fait que "network" s'est retrouvé fonctionnel pour lui même, mais n'était plus adapté aux évolutions du code ayant eu lieu sur les autres parties.

Ainsi, tout ce qui concerne le jeu en Multijoueur n'est pas encore disponible. Le rendre fonctionnel d'ici la phase de présentation serait modifier beaucoup de choses, ce qui est contraire aux consignes données.

Choix ergonomiques

Notre design est simple et simule le mieux possible un vrai jeu de cartes qui fait que l'interaction entre l'interface et l'utilisateur est plus intuitive pour ce dernier.

Nous avons également ajouté une consigne courte ainsi que l'objectif du jeu dans le menu sous l'option règles, tout dans le but de guider l'utilisateur le mieux possible.

Problèmes rencontrés et solutions apportées

Comme mentionné dans la partie planification, notre premier défi consistait à implémenter les effets des cartes tout en gardant un héritage non redondant (qui implémente une méthode pour l'effet de carte en prenant des attributs non utilisées dans la méthode) et bien propre. La solution apportée à cet défit était de créer une classe singleton « Action » qui contient les attributs de joueur cible et carte devinée, misent à jour du côté interface graphique pour chaque joueur.

Au niveaux du réseau, la grande difficulté était de debugger le client suite aux crash du système pour lequel Qt donnait aucun détail sur la source de l'erreur et le signal reçu.

Concernant l'interface graphique, plusieurs difficultés ont été rencontrés. Les cartes comme la Comtesse et le Prince étaient difficiles à implémenter, mais finalement on a réussi de les implémenter.

Un autre soucis était de bien faire la rotation du plateau du jeu en local pour qu'un joueur courant se trouve toujours devant l'écran et pas sur le côté du plateau. Ceci, nous n'avons pas réussi entièrement vu que les objets n'étaient pas bien alignés et cela faisait un décalage sur le plateau.

Le seul défi de l'intelligence artificielle était de trouver une stratégie qui prend en considération pas mal de paramètres pour augmenter ses chances pour gagner. Pour cela tous les couples possibles des cartes ont été examinés pour trouver une stratégie générique pour la carte à jeter.

Piste d'améliorations

Une des pistes d'améliorations serait un tutoriel qui montrera aux utilisateurs comment jouer une partie à 2, cependant les contraintes de temps nous ont empêché d'implémenter ceci. L'idée était de rajouter un bouton tutoriel dans le menu qui en cliquant simulera un jeu déjà prévu pour qu'on puisse pour chaque tour de joueur élaborer les choix faits dans une boîte de dialogue statique et ainsi faciliter l'apprentissage pour l'utilisateur. Ensuite, le fait qu'un utilisateur puisse glisser la carte à jeter vers un rectangle, peut être aurait été plus intuitive et plus proche d'un vrai jeu de plateau.

Concernant la localisation, la manière dont nous avons codé la permet facilement. Par exemple, les cartes sont découpés en image / valeur / titre / description, ce qui permet de s'adapter entièrement au niveau de la traduction et de l'image (une princesse n'est pas forcément représenté de la même manière dans tous les pays).

Nos erreurs

Bien que certaines aient déjà été énoncées plus haut, il est important de faire un bilan et de comprendre pourquoi nous n'avons pas réussi à atteindre les objectifs fixés. Nous avons principalement fait les frais d'une organisation faussement bien pensée. Le début était très bien, et la dynamique était bonne. Mais dans nos estimations, nous avons manqué de réalisme. En effet, estimer la réalisation d'une tâche c'est bien, prendre en considération le fait qu'on a d'autres obligations dans d'autres matières, c'est mieux.

Aussi, nous avons mal estimé le poids des modules à développer. En effet, nous avons voulu "voir les choses en grand" et réaliser le meilleur projet possible. Notre logique a donc été de paralléliser au maximum les tâches et d'avancer sur toutes les parties du projet en même temps. Cependant, il aurait été beaucoup plus judicieux dans certains cas de faire les choses les unes après les autres. De s'imposer le fait que un module doit être développé si il est prêt à être accueilli (exemple avec la branche "network").

La partie la plus fondamentale du jeu, à savoir l'interface graphique avec un jeu fonctionnel, a avancé aussi rapidement que les parties "optionnelles", car nous avons sous/mal estimé sa charge de travail. Il aurait été préférable de s'attaquer à celle là en priorité, avant de poursuivre sur la suite.

Conclusion

Ce projet reste une très bonne expérience de travail en groupe. Nous en tirons de bonnes leçons qui nous serviront pour nos travaux à venir.

Nous tenons également à vous remercier pour ce sujet de projet. En effet, c'est très rare d'avoir un projet intéressant à développer. L'idée de coder un jeu existant et d'être dans du concret nous a beaucoup motivé.

NOTE :

- Vous pourrez trouver les différentes versions de nos UML dans le dossier conception, à la racine du projet.
- L'option de "shadow build" (QT) doit être décoché pour la compilation de notre projet.
- Le sous-projet "network" donnera des erreurs à la compilation sous Windows. Il est bien sûr possible de compiler les sous-projets individuellement (clique droit -> Build).