AL ANI Mohamed
DARCISSAC Ludovic
EL JIBBAWE Nayef
WANG Pei

Projet de C++

Jeu de rôle éducatif

A l'attention de M. Jean-Philipe kotowicz

Sommaire

Introduction

Tout au long de ce projet, nous allons tenter d'implémenter un jeu RPG éducatif dans le cadre de notre cours de C++. Ce jeu qui se veut simple mais assez complet nous permettra de mettre en pratiques les connaissances que nous avons acquises pour nous familiariser avec l'implémentation en C++. Ce projet nous permettra aussi de travailler à la réalisation d'un projet concret en groupe. Il faut donc savoir synchroniser notre travail et faire en sorte de bien communiquer pour optimiser l'avancée du projet.

Dans le document suivant, nous expliciterons dans un premier temps l'étude préliminaire qui a été réalisé pour cerner les problématiques et objectifs du projet. On tachera de mettre en évidence les fonctionnalités à écarter c'est à dire les fonctions que l'on n'implémentera pas (faute de temps) et d'autres que nous essayerons de faire si le temps ne nous fait pas défaut.

Ensuite, nous présenterons le Game Design qui est le processus de création et de mise au point des règles et autres éléments constitutifs d'un jeu. Dans cette section, nous donnerons le plus d'informations possibles sur l'environnement du jeu que nous essayerons de coder.

Après cette première étape de réflexion, nous mettrons en place les spécifications fonctionnelles du projet qui consistent en la réalisation d'un diagramme de classe mais aussi de plusieurs diagrammes de séquence. Des scénarios types seront écrits pour pouvoir nous aiguiller lors de la phase d'implémentation.

Puis, nous réaliserons les spécifications techniques du projet c'est à dire que nous écrirons les différents tests unitaires et d'intégration que nous devrons réaliser qui seront nécessaires pour assurer la validité du code au fur et à mesure de notre implémentation.

Enfin, nous passerons à la phase d'implémentation qui devrait être la plus courte si notre travail préliminaire a bien été fait.

1) Etude préliminaire

Dans cette première section, nous réaliserons l'étude préliminaire du projet qui consiste à mettre en place les idées fondamentales du projet, à présenter les différentes problématiques et fonctionnalités que l'on devra implémenter. Un effort de réflexion sera fait pour cette section pour pouvoir mettre de côté certaines fonctionnalités qui ne seront pas réalisées faute de temps.

1.1) Idée générale

Le joueur incarnera un adolescent qui a des compétences (intelligence, énergie, cash, force) qui peut se trouver dans plusieurs lieux (maison, école, superette, forêt, bibliothèque et banque) et ces lieux contiennent des PNJ (Maman, Papa, bibliothécaire, professeur...).

Le jeu consistera à gagner un maximum de points en résolvant des équations mathématiques à l'école pour améliorer l'intelligence du héros et avoir accès à des missions (chercher des objets, résoudre des équations mathématiques, obtenir des objets...) données par les PNJ.

Le jeu s'arrête quand le héros atteint 50 points d'intelligence.

1.2) But du jeu

Le but du notre jeu est de réaliser plusieurs missions afin d'atteindre un certain niveau d'intelligence.

1.3) Ce que l'on va faire

Notre jeu va se présenter sous la forme d'une console. Le héros possède plusieurs caractéristiques comme l'intelligence, la force, le cash et l'énergie et en fonction de ces caractéristiques, le héros peut accéder à certaines missions ou pas.

En fait, le héros se trouve dans une ville composée de plusieurs lieux et chaque lieu est composé de plusieurs PNJ. Chaque PNJ propose plusieurs missions.

On a 3 types de missions :

- Missions maths : Elles consistent à résoudre des équations mathématiques afin de gagner des points d'intelligence.
- Missions objet : Elles consistent à récupérer un objet et à l'ajouter au sac de l'héros, ici on peut distinguer 2 types d'objets les consommables et les non-consommables.
- Missions combat : Elles consistent à comparer les statistiques de l'héros avec un PNJ.

1.4) Ce que l'on ne va pas faire

Comme énoncé dans l'introduction faute de temps, nous devons dès à présent faire des concessions et écarter certaines fonctionnalités du jeu.

Le premier aspect de la conception du jeu que l'on n'abordera pas est le modèle MVC. En fait, faute de temps, on n'a pas fait le modèle MVC mais toutes les méthodes d'affichage et toutes celles qui mériteraient d'être dans le contrôleur se trouvent dans la classe Jeu.

Le deuxième aspect que l'on n'abordera pas est la mise en place d'une version graphique.

2) Principes du Game Design

Dans cette deuxième section, nous donnerons des informations concernant le Game Design c'est-à-dire l'ensemble des données nécessaires à la création du jeu.

2.1) Lieux

- La superette

- La forêt

- La maison

- La banque

- La bibliothèque

- L'école

2.2) Personnages

2.2.1) Le Héros

Le héros : C'est le personnage principal contrôle par le joueur.

Il dispose de 6 caractéristiques (l'intelligence, la force, l'âge, le nom, le cash et l'énergie). Le héros peut parler avec les PNJ, réaliser des missions disponibles et se déplacer d'un lieu vers un autre.

2.2.2) Les PNJ

Les PNJ sont les personnages non jouables.

Ils seront principalement des personnages proposant des missions au héros comme le père, la mère, le banquier etc ou des ennemies du héros comme c'est le cas dans la forêt ou le héros peut se battre avec les PNJ.

2.3) Déroulement global du jeu

Au début du jeu, notre joueur se trouve dans la ville. La ville contient tous les lieux. Il décide alors de se déplacer afin d'aller dans un lieu en particulier comme la superette, l'école etc.

Dans chaque lieu, le héros a la possibilité de parler avec les PNJ présents dans ce lieu, de fouiller le lieu dans le but de trouver des objets consommables ou non consommables ou de quitter le lieu pour retourner dans la ville. Dans chaque lieu, ce sont les PNJ qui proposent des missions au joueur. Ce dernier peut soit les accepter soit les refuser.

Les missions varient en fonction des lieux et en fonction des PNJ. Par exemple si le héros se trouve dans la banque il peut soit discuter avec le banquier, soit retirer de l'argent, soit déposer de l'argent ou soit fouiller la banque. On a, en fait, plusieurs types de missions. Des missions de type maths permettant de résoudre des équations mathématiques, des

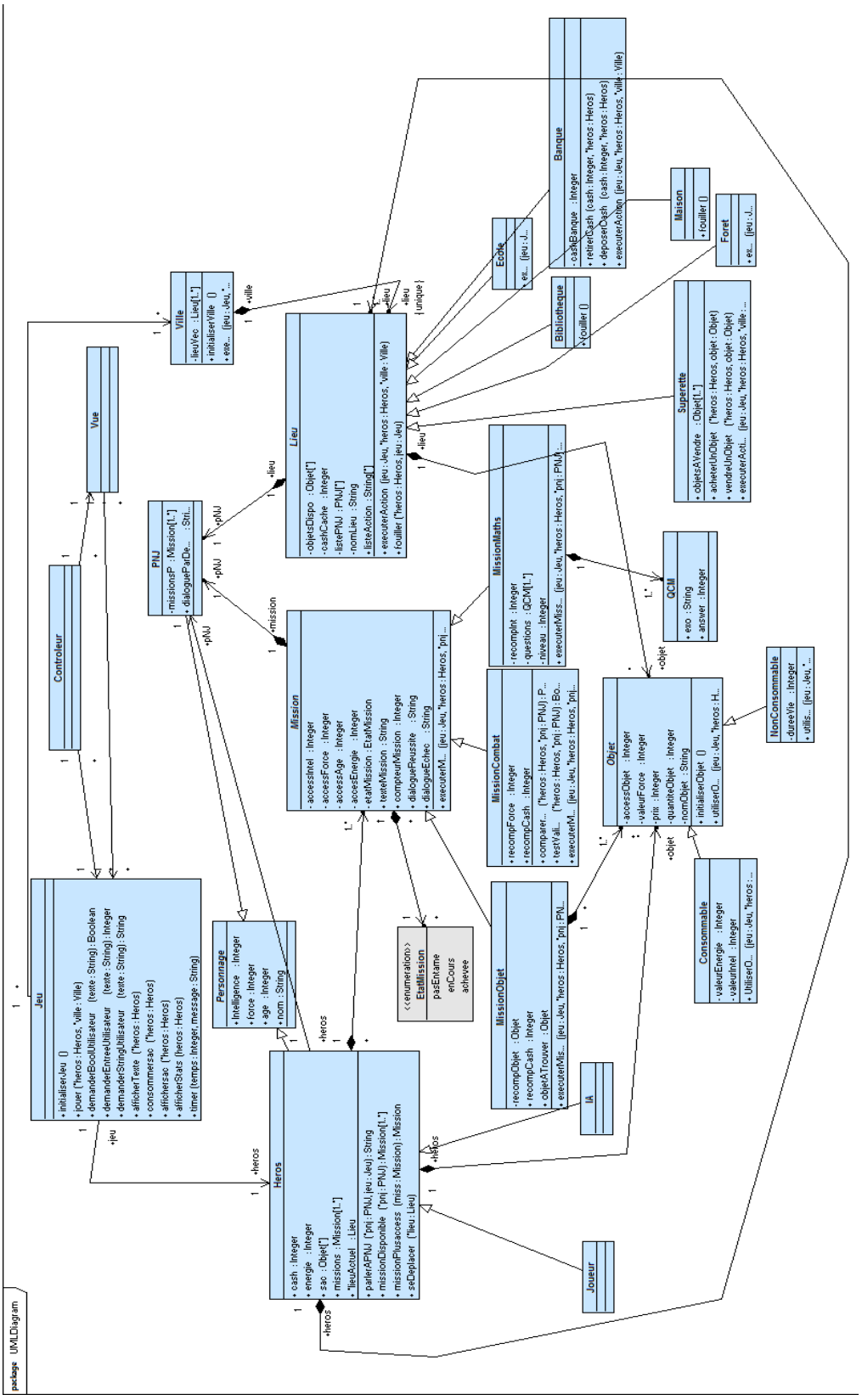
missions de type combat permettant de comparer les statistiques du héros avec celle du PNJ et des missions de type objet permettant de trouver un objet qui peut être de 2 sortes soit consommable par exemple le pain ou non consommable qui le héros peut garder dans son sac.

En fait, le héros peut à n'importe quel moment quitter un lieu pour passer dans un autre ou pour retourner dans la ville.

Le jeu s'arrête quand le héros atteint un certain nombre de points d'intelligence.

3) Spécifications fonctionnelles

3.1) Diagramme de classe



3.2) Commentaire sur le diagramme de classe

3.2.1) Jeu

C'est la classe maîtresse, qui s'occupe d'initialiser le jeu et de lancer la méthode jouer qui est formée d'un héros et d'une ville. Cette classe contient d'autres méthodes par exemple consommersac, afficherstats etc. En fait, ces méthodes ont été expliquées en rédigeant la doc.

3.2.2) Objet

C'est une classe abstraite qui permet d'introduire les classes Consommable et NonConsommable.

Cette classe a pour attribut le prix de l'objet, la quantité de l'objet, le nom de l'objet qui est une chaîne de caractères qui va être utilisée dans le map du sac du héros. On a 2 méthodes initialiserObjet qui va initialiser tous les objets à zéro et utiliserObjet qui va permettre au héros de consommer l'objet s'il consommable ou de l'utiliser s'il ne l'est pas.

Un objet peut donc être de 2 sortes :

- **Non consommable** : Ce sont des objets que le héros ne peut pas consommer et qu'il peut mettre dans son sac. Ces objets permettent de modifier les statistiques du héros comme la force. Les objets non consommables ont une durée de vie.
- **Consommable** : Ce sont les objets que le héros peut consommer comme le pain par exemple. Ces objets permettent de modifier une ou plusieurs capacités du héros comme l'énergie et l'intelligence.

3.2.3) Personnage

C'est une classe abstraite qui permet d'introduire la classe Heros et la classe PNJ.

Elle a pour attribut l'intelligence, la force, l'âge et le nom qui caractérisent chacun des personnages.

Heros : c'est une classe fondamentale, la classe Heros a comme attribut la valeur du cash, l'énergie, un vecteur d'objets correspondant au sac du héros, un vecteurs de missions et le lieu actuel du héros.

D'abord, le sac est représenté par une map <string,objet>. Chaque objet dans le sac est accessible par une chaîne de caractère portant le nom de l'objet par exemple sac[« pain »] = pain qui est l'objet correspondant. Tous les objets du sac seront initialisés dans le sac à une quantité 0.

Ensuite, Les missions sont représentées par un vecteur de missions. Toutes les missions vont être triées en fonction des PNJ avec lesquelles elles sont liées

Les méthodes de la classe Heros sont divers et variées. Les méthodes sont les suivantes : parlerAPNJ, missionDisponible, missionPlusaccess et se Deplacer.

PNJ : Cette classe représente les personnages non joueurs. Cette classe a pour attribut un vecteur de mission et un dialogue par défaut.

3.2.4) Mission

C'est une classe abstraite qui permet d'introduire les classes MissionMaths, MissionsCombat et MissionObjet.

Cette classe a comme attribut l'accessibilité en intelligence, en force, en âge, en énergie de la mission, l'état de la mission présentée par une énumération, une mission peut être pas entamée, en cours ou achevée, texte mission, compteur mission, dialogue réussite et dialogue échec.

Cette classe possède la méthode executerMission qui est une méthode virtuelle qui va être redéfinie dans les classes filles.

Les classes filles sont les suivantes :

- **MissionObjet** : Elle possède comme attribut l'objet à trouver, la récompense objet et la récompense du cash et comme méthode executerMission.
- **MissionCombat** : Elle possède comme attribut la récompense force et la récompense du cash et comme méthode comparerStats, testValidité et executerMission.
- **MissionMaths** : Elle possède comme attribut une récompense intelligence, le niveau de la mission, un vecteur de QCM correspondant à l'exercice de maths et à la réponse et comme méthode executerMission.

3.2.5) Lieu

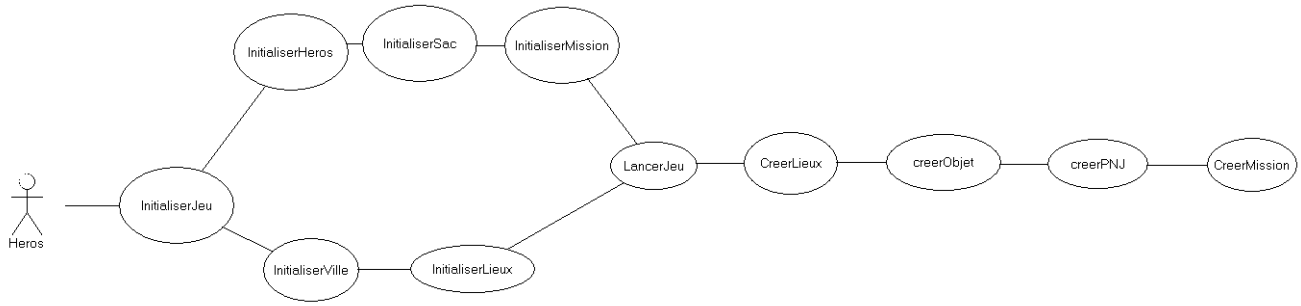
Cette classe est une classe abstraite qui permet d'introduire les classes Superette, Forêt, Maison, Banque, Ecole, Bibliothèque. La ville est composée de lieux. La classe Ville possède comme attribut un vecteur de lieu et cette classe permet d'initialiser la ville.

La classe Lieu possède comme attribut : objetsDispo un vecteur d'objets, la valeur du cash caché dans un lieu, la liste des PNJ, la liste des actions à effectuer dans un lieu donné et le nom du lieu et comme méthode executerAction qui est une méthode virtuelle de Lieu. Cette méthode va être redéfinie dans chacune des classes filles de lieu. Elle va lister toutes les actions à effectuer dans le lieu et par le biais du Jeu, proposer une action à l'utilisateur et la méthode fouiller un lieu.

Les classes filles sont les suivantes :

- Superette
- Forêt
- Maison
- Banque
- Ecole
- Bibliothèque

3.3) Diagramme de cas



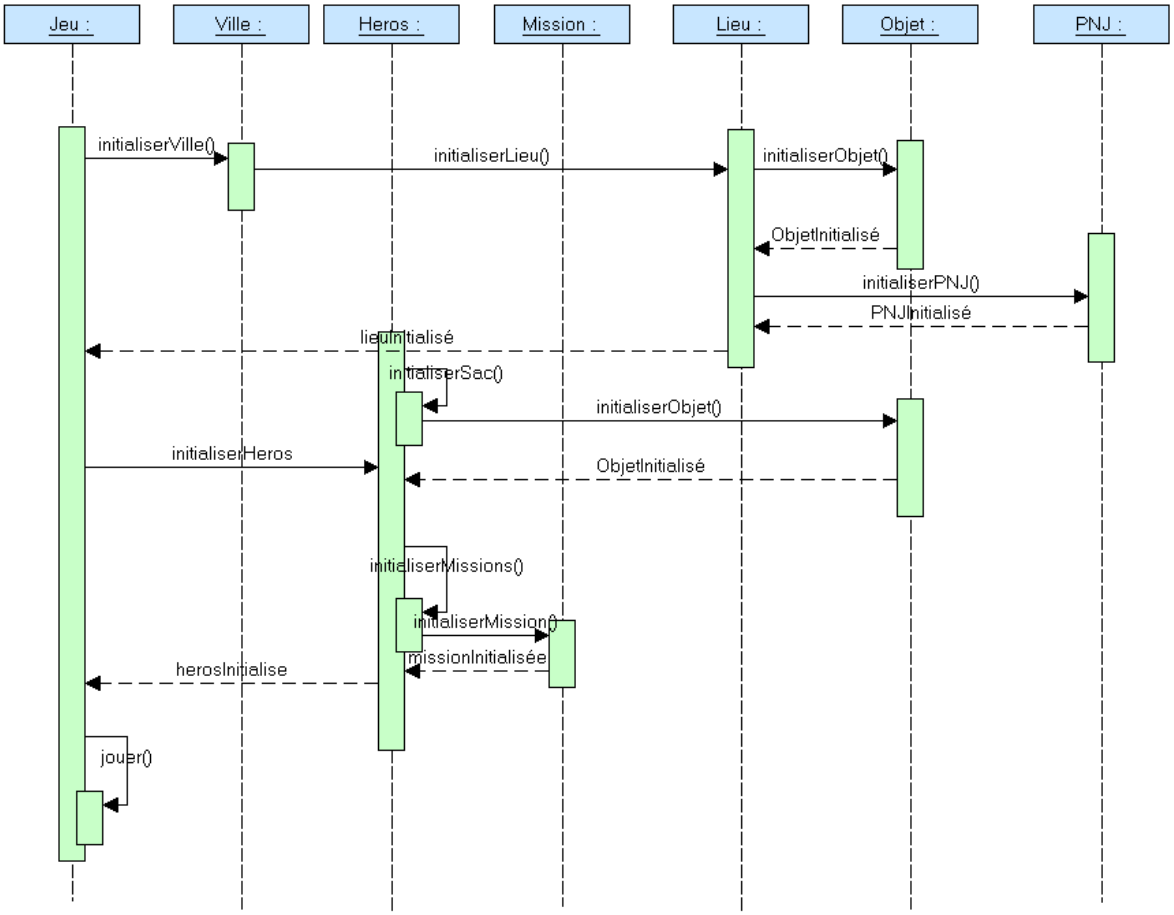
3.4) Commentaire sur le diagramme de cas

On voit bien dans le diagramme de cas que chaque partie commence par une initialisation du héros et de la ville. Ensuite, on initialise le sac du héros, les lieux et les missions. On lance après le jeu ce qui crée des lieux, des objets, des PNJ et des missions.

3.5) Diagramme de séquences et de scénarios

Scénario 1 : Initialisation du Jeu. On crée le héros:

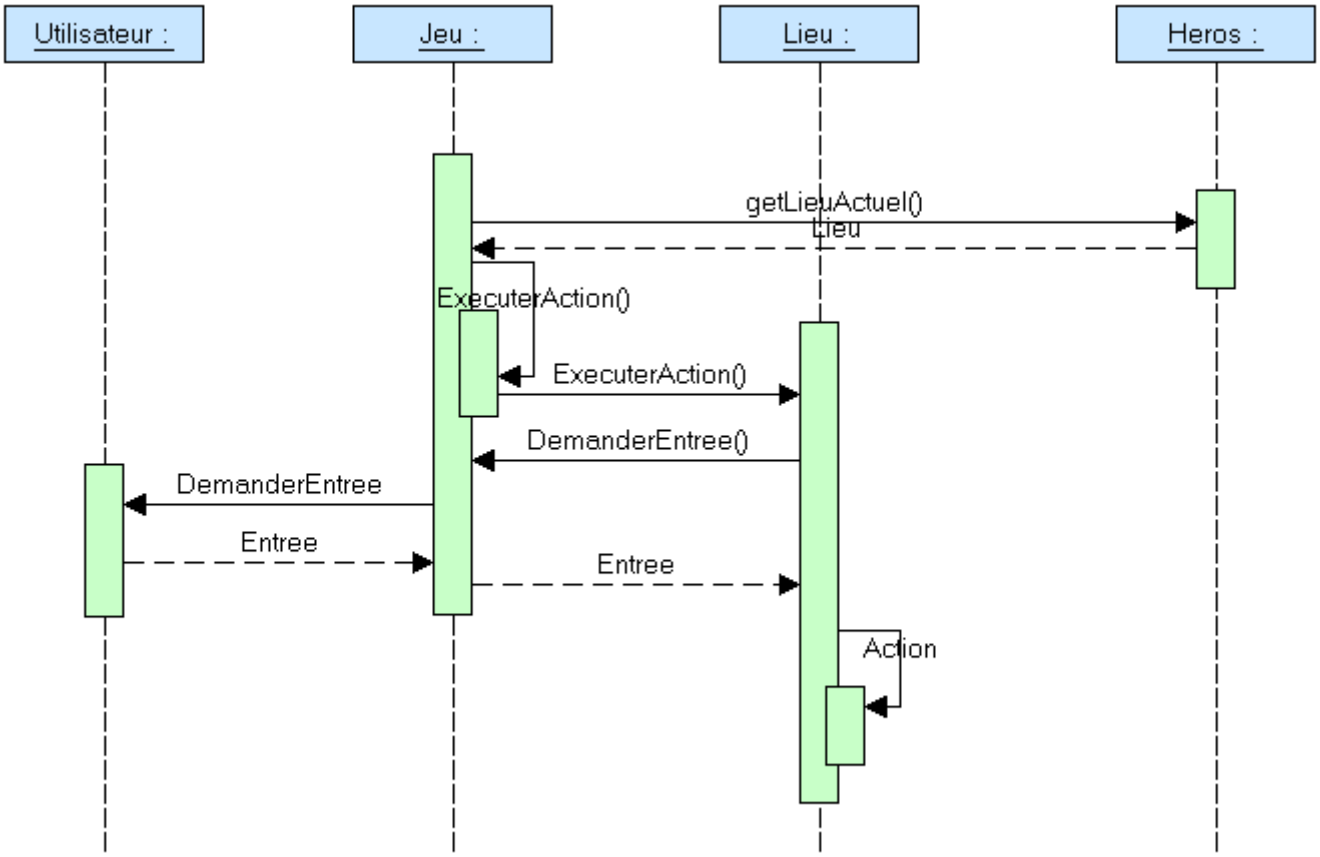
- On initialise en premier les lieux pour pouvoir ensuite initialise les Objets à l'intérieur, ainsi que les PNJ.
- On initialise le Héros : le sac avec des objets, les missions.



Scénario 2 : Executer une action dans un Lieu

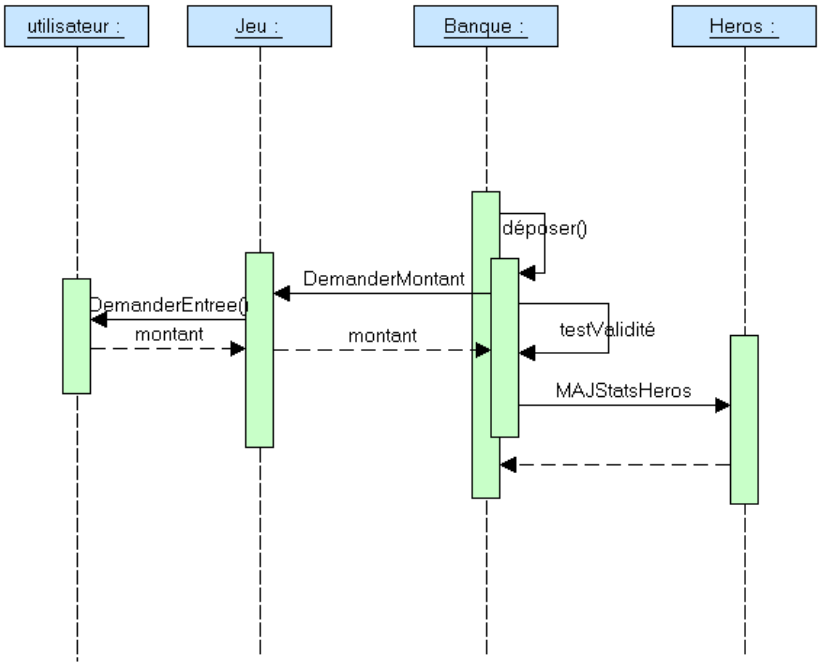
- On suppose qu'on se situe dans un Lieu.
- On demande à l'utilisateur l'action qu'il veut effectuer dans le Lieu.

L'action peut être un déplacement, l'appel d'une méthode spécifique au lieu, un affichage du sac...



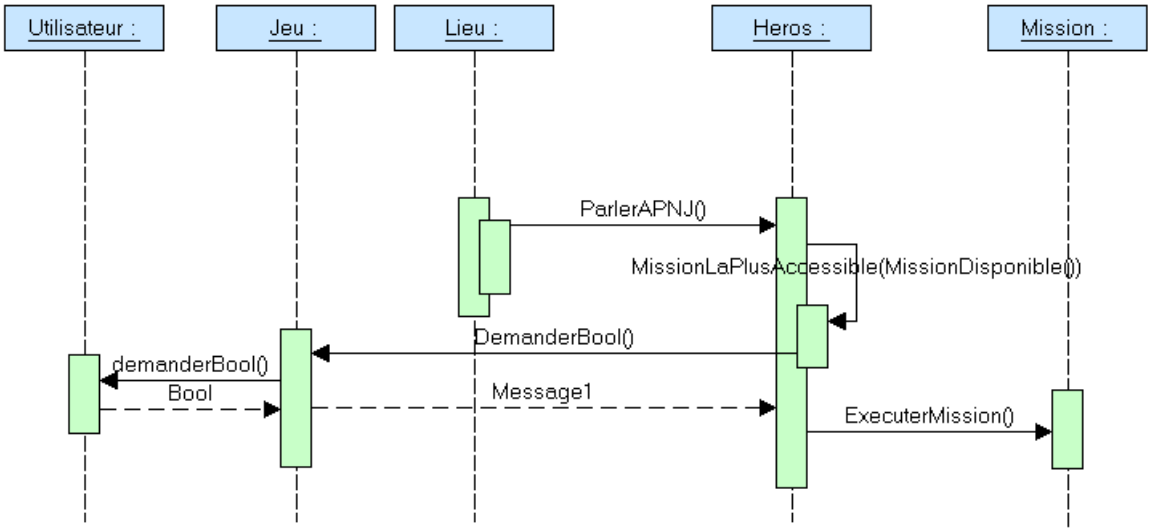
Scénario 3 : Déposer Cash dans la banque

- On suppose qu'on se trouve dans la Banque et qu'on ait proposé une action à l'utilisateur et que celui-ci ait choisi de déposer du cash.



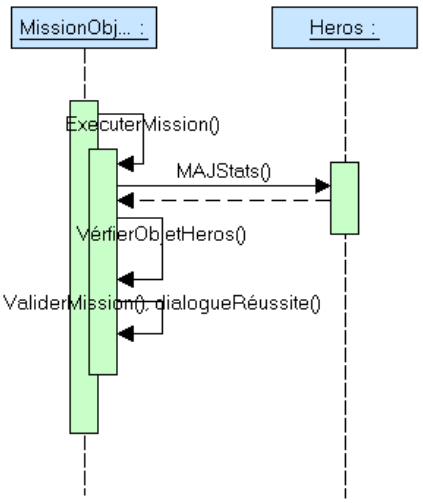
Scénario 4 : Exécuter une mission

- On suppose que l'on est dans un lieu et que l'utilisateur ait choisi de parler à un PNJ (par le biais d'executerAction).
- On trie les missions disponibles liées au PNJ.



Scénario 5 : Mission Objet

- On suppose que le héros a parlé à un PNJ qui avait des missions à proposer. Les missions ont été triées et on a fait appel à la méthode ExecuterMission().
- Les stats du héros sont mises à jour : il perd de l'énergie.
- On vérifie que le héros a l'objet demandé dans le sac.
- c'est le cas, donc on valide la mission, on met à jour les stats du héros, on lui met un objet dans le sac (on augmente la quantité de l'objet dans le map) puis on renvoie le dialogue de réussite.



4) Répartition des tâches

	Mohamed ALANI	Ludovic DARCISSAC	Nayef EL JIBBAWE	Pei WANG
Conception	Oui	Oui	Oui	Oui
Code	Oui	Oui	Oui (un peu au début)	Oui
Rédaction du rapport	Non	Non	Oui	Non
Génération de la Doc	Non	Non	Oui	Non

Conception :

Avant de commencer notre projet, on s'était réuni plusieurs fois afin de fixer le but de notre jeu et de faire la conception du jeu. Tous les membres du groupe ont participé à cette conception. La première était de faire le diagramme de classe ensuite les diagrammes de séquence et scénario finalement Nayef s'est occupé de synchroniser le code avec le modèle UML et de mettre au propre les diagrammes.

Code :

- Mohamed ALANI : Missions et les classes filles de Missions.
  Objets et les classes filles de Objets.
  Timer dans le jeu.
- Ludovic Darcissac : Lieux et les classes filles de Lieu.
- Nayef EL JIBBAWE : Jeu (modification de quelque méthodes).
- Pei Wang : La méthode consommerObjet dans la classe Jeu.
  Personnage et les classes filles de personnage.

Chacun faisait ses tests unitaires.

Rédaction du rapport :

Nayef s'est occupé de rédiger le rapport.

Génération de la doc :

Nayef s'est occupé de commenter les attributs et les méthodes de chaque classe et ensuite de générer la doc en utilisant Doxygen.

5) Synthèse critique de notre travail

Ce projet a été très enrichissant pour nous car on a appris à développer un jeu de rôle éducatif en C++. Lors de ce projet, on a eu quelques difficultés comme les inclusions circulaires.

En fait, on a affaire à des inclusions circulaires quand une classe A appelle B qui elle aussi appelle A. Pour régler ce problème, on a utilisé la forward déclaration ou la déclaration avancée qui est une déclaration d'un identificateur représentant une entité (par exemple un type, une variable, une fonction) pour laquelle la définition n'est fournie qu'ultérieurement dans le code.

En fait, on s'est rendu compte que lorsqu'on devait appeler la classe mère ou lorsqu'une classe était en attribut, on devait faire les include <….> dans les .hpp et les .cpp

Le passage par pointeur était obligatoire dans notre projet. Pour le faire, on a fait comme on faisait en Java. En fait, on avait besoin du passage par référence pour les méthodes virtuelles et de pointeurs pour les vecteurs d'objets sinon on ne peut pas les modifier.

Par manque de temps, on n'a pas pu faire le modèle MVC mais toutes les méthodes d'affichage et toutes celles qui méritaient d'être dans le contrôleur se trouvent dans Jeu. Par manque de temps aussi, on n'a pas pu réaliser une version graphique de notre Jeu. En fait, notre jeu tourne en mode console.

En fait, les initialisations méritaient d'être dans chacune des classes auxquelles elles appartiennent mais par manque de temps nous avons tout mis dans le main.

Conclusion

La première difficulté était bien entendu un diagramme de classe complet et de lister tous les scénarios possibles de notre jeu.

Cependant, ce projet a été l'occasion de mettre en pratique les cours d'UML ainsi que d'apprendre un nouveau langage informatique qui est le C++.

Par contre, nous n'avons pas eu le temps de faire le modèle MVC mais on a compris comment faut-il le faire. Faute de temps aussi, nous n'avons pas pu faire une version graphique.

Ce projet nous a permis de travailler à plusieurs sur un même sujet. En effet, nous sommes habitués à travailler en binôme, mais n'avons que très rarement l'occasion d'être dans des groupes plus importants.

La version console de notre jeu est opérationnelle, et ce premier aboutissement nous encourage pour nos futurs projets de programmation en C++.