FARANDJIS Matthieu
LEMOUTON Alexis

Mars - Juin 2023
INF1-C

SAÉ 2.01 et 2.02

Rapport

Exploration algorithmique d'un problème

Sommaire

1. Introduction

“Mappy RPG” est un calculateur d’itinéraire pour un jeu RPG fictif.
Comme la plateforme originale “Mappy”, notre application est capable de proposer différents itinéraires à partir des choix de l’utilisateur. Il lui suffit simplement de choisir un scénario, et de sélectionner les options qui l'intéresse.

Qu’il souhaite faire la quête finale à la toute fin ou dès que possible, avoir un itinéraire respectant certaines conditions ou bien même étudier les pires possibilités... Utiliser MappyRPG est le meilleur choix !

Ainsi, ce document présente l’élaboration de ce logiciel graphique écrit en Java. Que ce soit sur notre manière de travailler, nos idées mises en place ou abandonnées mais également nos problèmes et difficultés.
Ce compte rendu est donc une synthèse de notre travail effectué durant ces trois mois dans le cadre de notre première année de BUT Informatique.

2. Qualité de développement

a) L'organisation du travail

A ces deux SAÉ, nous pouvons joindre la SAÉ 2.05 de gestion de projet.
Dans cet SAÉ, nous avons dû découper au préalable le projet sous forme de tâches que nous avons ensuite planifiées.

Certaines tâches correspondantes à certaines méthodes, il nous a été obligatoire de prévoir d'avance les algorithmes que nous allions devoir mettre en place. Ces tâches étaient représentées dans un organigramme (WBS). Chacune d'entre elle a une lettre unique qui la représente.

Une tâche correspondant à une méthode est découpée en trois étapes : conception, développement et élaboration des tests. La première consiste à réfléchir sur ce que nous voulons faire, et construire la documentation. La deuxième correspond à la création du code et à l'essai en direct de celui-ci. Enfin, on réfléchit aux cas des tests, et on les implémente.
Ainsi nous suivons le modèle en cascade. On réfléchit, on créer, on test et on passe à autre chose. Cependant, il est tout à fait possible que plus loin dans le développement, on retouche à certaines de ces étapes. Dans le cas où on change de point de vue par rapport à l'algorithme. Pour nous, il était plus rapide et plus agréable de procédé ainsi.
En cours, nous avons vu comment réfléchir aux tests avant d'implémenter le programme. L'atout de cette méthode est qu'on est sûr du fonctionnement du programme. Nous avons donc moins besoin de le retoucher plus tard. Cependant, cela demande du temps. Vu qu'on expérimente les méthodes en même temps que leurs créations, nous avons jugé que ce n'était pas nécessaire. D'autant plus qu'en cas d'un changement concernant la conception du programme, il faudra recommencer l'entièreté des tests.

Une fois fait, nous avons ordonné ces tâches afin d'établir un planning.
Nous nous sommes ainsi attribués à chacun, des tâches auquel nous leur avons donné un temps.

Initialement, nous avons essayé de suivre cette organisation. Mais avec la recherche d'un contrat d'apprentissage, les évaluations et les autres SAÉ, nous avons essayé d'avancer au mieux. Seulement, la partie algorithmie de la SAÉ n'était pas ce que préférait faire Alexis par la difficulté de certaines tâches. Nous en avons donc discuté et nous nous sommes mis d'accord sur une nouvelle répartition des tâches. Matthieu s'occupait de la partie algorithme, et Alexis de la partie interface console et graphique du projet. En effet, l'importance est de mettre au sein du projet le meilleur de chacun.

La communication est quelque chose d’indispensable au sein d’une équipe.
Que ce soit en présentiel ou en distanciel, nous avons beaucoup travaillé et discuté ensemble du projet. Nous faisions un retour chacun de nos travaux, et chacun pouvait observer la progression de l’autre.
Si l’idéal était qu’on se partage la partie algorithmie et la partie interface, cette nouvelle organisation nous a rendu plus efficace. L’enjeu désormais était de rattraper le retard que nous avions accumulé à cause de cette désorganisation.

b) L 'utilisation des outils

Pour mener à bien notre mission, nous avons utilisés les outils suivants : IntelliJ IDEA, Git (avec BitBucket), JUnit, Discord.

Pour présenter brièvement ces outils, IntelliJ est notre IDE, c'est à dire le logiciel où nous développons notre application. BitBucket est tout comme GitHub, un service qui permet de stocker notre code et le gérer en ligne grâce à Git. Git étant un logiciel de gestion de version. Il permet d'organiser notre développement grâce à des branches et nous pouvons aussi sauvegarder notre avancé et les commentés grâce aux commit. Discord est de son côté un logiciel de discussion en ligne. Très pratique lorsque nous devions montrer notre écran à distance ou bien partager des fichiers.

Si nous avions choisi Bitbucket plutôt que le très populaire GitHub, c'était avant tout parce que nous l'avions utilisé en cours. Alexis utilisait principalement Bitbucket et GitHub lui était inconnu.

Nous avons rencontré de très grands problèmes avec Git et IntelliJ, particulièrement du côté de chez Alexis. Du côté de Matthieu, sur son ordinateur personnel, il était impossible d'ouvrir le projet sur IntelliJ sur Windows. IntelliJ fonctionnait uniquement sur Linux. Etrangement, sur les machines de l'IUT avec Windows, il n'y avait aucun problème. Concernant Alexis, il a eu de très grand problème par rapport au JDK dans IntelliJ. Lorsqu'il changeait d'ordinateur, le projet ne fonctionnait plus du tout de son côté. Parfois, il n'arrivait pas non plus à utiliser correctement Git. Nous avons cherché à résoudre ces problèmes, mais la résolution de ces problèmes pouvait prendre plus d'une heure, le temps de trouver la source du problème. On ne trouvait rien sur les forums, et cela nous a fait perdre de précieuses heures.
Nous supposons que la source de ses problèmes est dû au transfert du projet via Discord. A la place de travailler sur une clé USB, Alexis préférait se transmettre le projet en .zip via Discord. Cependant, il n'était pas le seul à le faire, peut-être que Bitbucket a contribué à ces problèmes. Nous ne savons pas exactement d'où venait les problèmes en soit.

En raison de ces problèmes, Alexis a préféré avancer simplement sur la branche master. Matthieu a préféré créer une branche pour chaque fonctionnalité ou modification importante du programme. Master étant dédié aux moments où une fonctionnalité était terminée et exploitable. Ainsi, Matthieu pouvait avancer calmement sur sa branche, annuler des modifications ou faire des ajouts sans déranger Alexis. Une fois que le résultat obtenu le satisfaisait, il faisait un merge sur la branche hôte afin de continuer sur celle-ci en faisant de nouvelles branches.

Cette méthode de travail permettait à Alexis d'être certain de pouvoir exploiter une fonctionnalité qui fonctionne. Et c'était le mieux à faire. Si une branche n'allait pas dans la bonne direction, il était très simple de recommencer sans craindre une perte de données.

Le défaut de cette méthode, est qu'à la fin nous avons plusieurs branches. Pour les comprendre, le nom des branches devait être clair et précis. C'est pour cela qu'elles contenaient la lettre associée à la tâche du WBS.

Des camarades ont préféré créer une branche chacun et avancé dessus sans créer de nouvelles branches. Ou du moins, en faisant quelques-unes de plus, mais beaucoup moins que nous.
Je (Matthieu) suppose que ce n'est pas la meilleure idée. Ils n'exploitent pas au mieux Git. En cas d'un quelconque problème sur la branche, c'est toute la progression qui est en péril. De plus, par cette manière, ils peuvent avoir des incohérences. Il faut parfois mettre à jour son travail avec son binôme afin d'être sûr d'avancer dans la même direction.
Avancer sur sa même branche, c'est aussi rendre flou son avancée. On ne sait plus exactement quand on commence.

c) L'analyse de notre méthode de développement

Nous avons suivi globalement notre WBS.
Durant le développement on est passé d'un algorithme où on ajoute des quêtes entre à un algorithme plus efficace où on cherche toutes les combinaisons possibles. C'est un algorithme récursif.
Cependant la première idée a été gardé afin de faire le scénario 10 du côté du pire exhaustif. Les tests ont été construit en même temps, et le rapport de test à été fait à la toute fin du projet. Il permettrait en cas de reprise du projet à l'avenir par exemple, d'avoir une explication de ces tests et les précédents résultats.

3. Conception générale

Diagramme de Haut Niveau

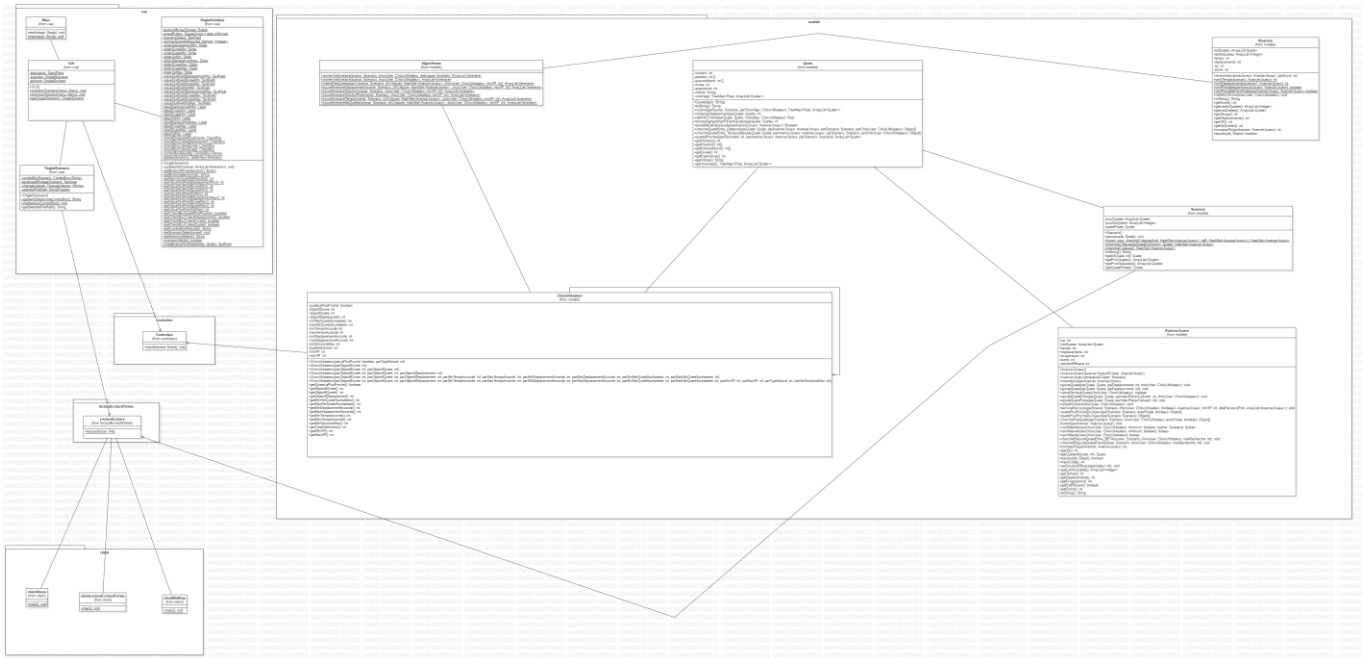
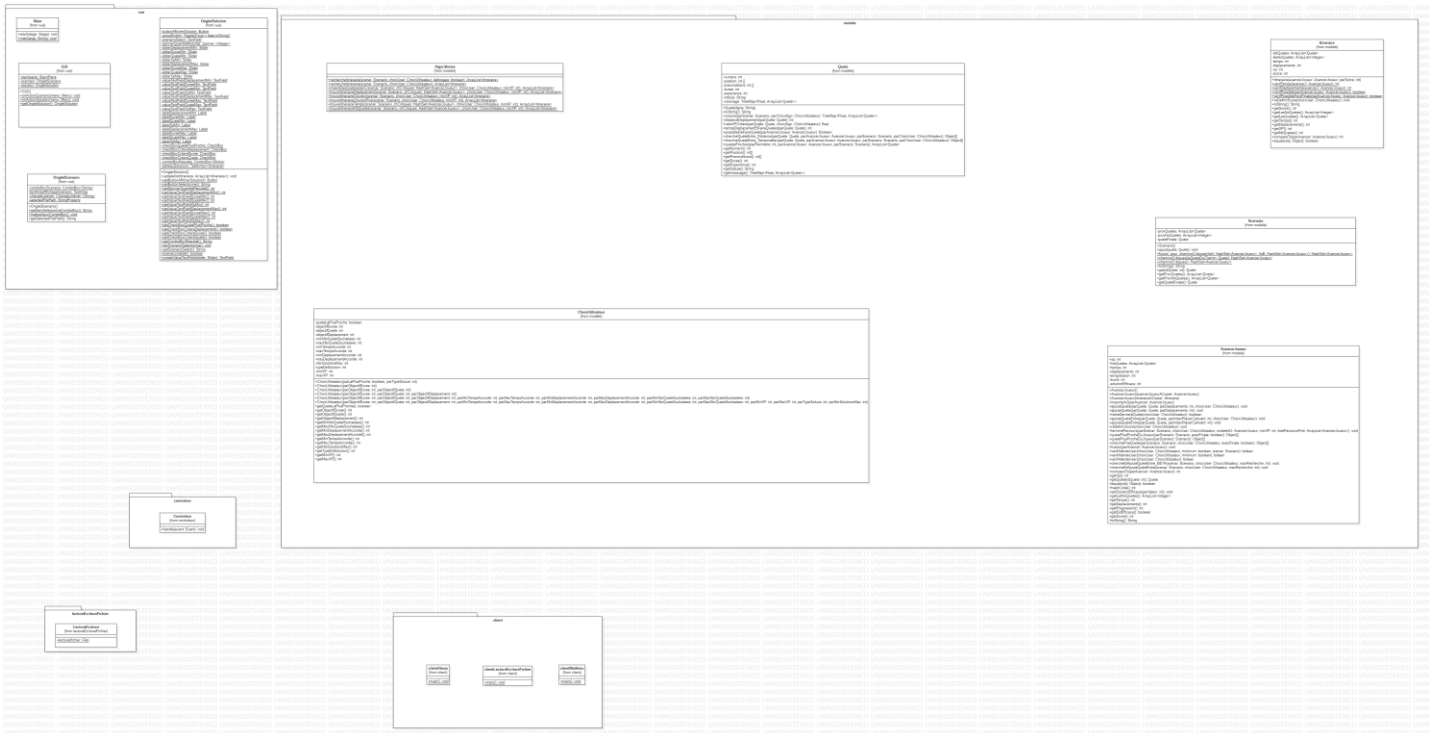


Diagramme de classe de Bas Niveau



4. Conclusion

Ce travail a été finalisé dans la précipitation mais le programme fonctionne.
Il manque des tests dans le rapport pour AvancerJoueur et le rapport pour Test_Algorithme.
Comme perspective d'évolution, on pensait créer une map graphique. Comme sous la forme du calendrier qu'on a fait en cours. Mais au lieu que ce soit un bouton par date, ça aurait été une case du jeu. On aurait colorié les cases en fonction de si c'est sur le trajet ou si c'est une quête.
Quand on cliquerait sur une quête, elle ouvrirait une boîte pour afficher ses informations.
Nous avons abandonné le mode console par manque de temps, mais les méthodes ont été testé dans clientMatthieu manuellement, sans passer par ce mode.

5. Annexe

Les rapport des tests sont dans le répertoire DossierTechnique.

Modele contient l'algorithme, vue l'affichage graphique, controleur le controleur dans le répertoire main (à côté du répertoire test).

La JavaDoc est disponible à ce chemin dans les dossiers :

SAE_RPG_202_LEMOUTON_FARANDJIS\JAVADOC\com.example.mappy_rpg

Et le fichier module-summary.html