PROJET SOKOBAN IMDS4A
Année 2021/2022

Rapport de projet

Réalisé par :

Simon AMBLARD
Alexandre CANCEL

POLYTECH
CLERMONT-FERRAND

Résumé

Ce rapport relate la conception du jeu Sokoban. Cette conception a été réalisée dans le cadre d'un projet de programmation orienté objet durant la quatrième année d'école d'ingénieur à Polytech Clermont dans la spécialité Ingénierie Mathématique et Data Science (IMDS). La création d'un jeu tel que le Sokoban est un bon exemple de développement d'application et de jeu par le langage UML et C++. Il a été important de faire deux parties, une d'analyse et de conception du jeu (UML) et l'autre de la programmation du jeu (C++)

MOTS-CLÉS : C++, UML, Développement, Application, QT, Jeu vidéo, Programmation, Sokoban.

Abstract

This report relates the design of the game Sokoban. This design was carried out as part of an object-oriented programming project during the fourth year of engineering school at Polytech Clermont in the specialty Mathematical Engineering and Data Science (IMDS). The creation of a game like Sokoban is a good example of application and game development by the UML and C++ language. It was important to do two parts of the game analysis and design (UML) and the other part of the game programming (C++)

KEYWORDS : C++, UML, Development, Application, QT, Video game, Programming, Sokoban.

Sommaire

Table des matières

1. Introduction

Dans le cadre de notre seconde année du cycle ingénieur en Ingénierie Mathématique et Data Science à Polytech Clermont, il nous a été imposé un projet de 4 mois nous permettant de mettre en pratique nos connaissances et nos compétences professionnelles ayant pour finalité la conception et le développement d'un jeu vidéo connu qui est le Sokoban. Le projet a été séparé en deux grandes parties, une première de conception et d'analyse du jeu. Cette partie a été réalisée par groupe de 8, groupe dans lequel on retrouve notre binôme (Alexandre Cancel et Simon Amblard), et les binômes Corentin Fradier et Alexandre Jan, Rayhane Bouachrine et Younes Karrouk Doukari, Adam Chamas et Gaétan Cece. Cette partie était essentiellement axée sur les thèmes du génie logiciel et du langage UML. Elle a donc été encadrée par la professeur Marinette Bouet. Ensuite la seconde partie fut la partie programmation en langage C++ et avec QT réalisée à l'aide des schémas UML de la première. Cette partie a été faite en binôme et a été encadrée par le professeur Christophe De Vaulx et l'intervenant Sylvain Lapeyrade. À la suite de cette partie, nous avons alors le jeu Sokoban avec plusieurs niveaux.

2. Présentation du jeu

Le principe du jeu est très simple à comprendre. Le joueur dirige un magasinier au sein de son entrepôt (sorte de labyrinthe qui est une grille carrée). En fait, le magasinier doit pousser des caisses afin de les amener sur des zones de rangement prédéfinies. Le joueur peut déplacer le personnage dans les quatre directions, en général au moyen des flèches ↑, ↓, ← et → du clavier. Le personnage ne peut que pousser les caisses, à raison d'une à la fois et sous réserve qu'il n'y ait pas d'obstacle (mur, autre caisse) ; en aucun cas, il ne peut tirer les caisses. Le joueur a donc pour objectif de réussir à ranger toutes les caisses aux endroits précisés avec le moins de coups possibles ; un coup correspondant à un déplacement ou à une poussée. Une fois que c'est fait, le niveau est réussi et le joueur peut passer au niveau suivant qui est plus difficile en général. Si le joueur n'atteint pas cet objectif (caisse inatteignable, caisse bloquée), il peut recommencer le niveau en question.

3. Planification du travail

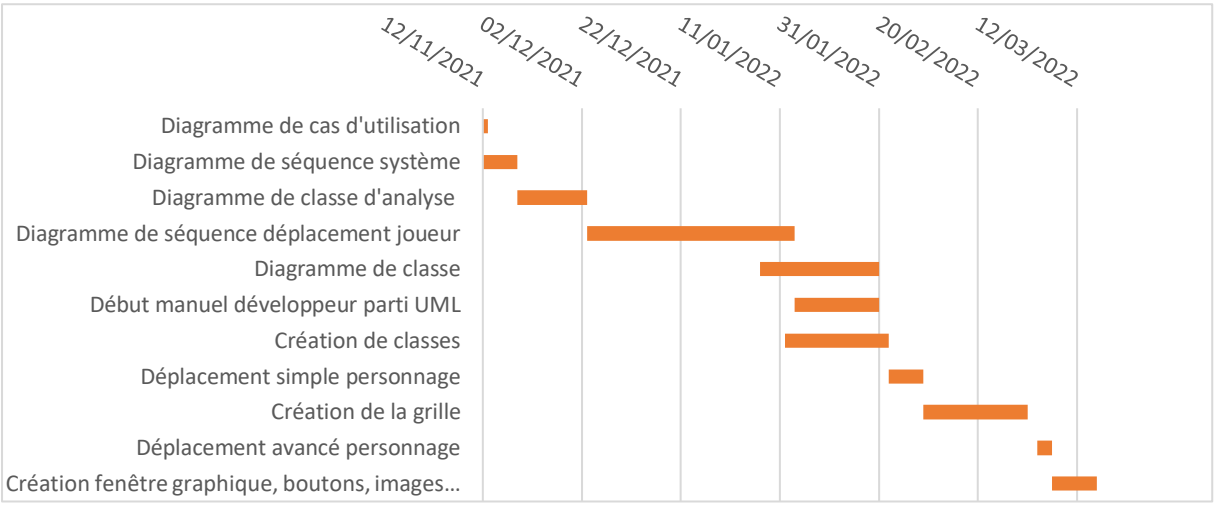

Figure 1 : Diagramme de Gantt

Afin de planifier notre projet, nous avons mis en place un diagramme de Gantt où l'on voit comment évolue le projet dans le temps.

4. Difficultés rencontrées / solutions envisagées / solution retenue

Comme ce projet était notre premier en développement d'application, nous avons rencontré plusieurs difficultés et nous avons dû chercher, à l'aide des professeurs encadrant, de nos connaissance et d'internet, les meilleures solutions à nos problèmes.

La première difficulté en UML c'est la communication. Lors de cette partie du projet nous étions 8 et nous avions beaucoup de solutions divergentes pour les différents problèmes que nous rencontrions. C'est pourquoi, il fallait que tous les membres du groupe acceptent la solution où le plus de monde était en accord avec celle-ci. Il fallait alors réussir à convaincre les personnes pas d'accord avec la majorité qu'une solution était mieux qu'une autre et parfois accepter et comprendre la solution de la majorité lorsqu'on n'en fait pas partie.

Ensuite la deuxième difficulté a été de faire le diagramme de séquence de la fonction du déplacement du joueur puisque de nombreux paramètres rentrent en compte dans ce déplacement. Nous avons alors mis du temps à nous mettre d'accord sur ce diagramme mais finalement un membre du groupe a réussi à synthétiser les différentes solutions en une seule qui convenait parfaitement à cette fonction-là.

La première difficulté en programmation a été le démarrage du projet. Après que tous les diagrammes ont été créé dans la partie UML, nous ne savions pas vraiment par quelle partie commencer. C'est pourquoi au départ nous avons commencé à implémenter les classes du diagramme de classe un peu aléatoirement. Mais il est important de pouvoir tester au fur et

à mesure nos fonctions et classes. On s'est donc rendu compte que ce n'était pas la bonne méthode et nous avons alors réfléchi à quel affichage serait le commencement. Nous avons alors décidé de commencer par essayer de faire le déplacement simple du personnage dans l'espace sans aucune contrainte.

Ensuite après ce déplacement simple, il nous fallait la grille pour avoir pouvoir mettre en place toutes les contraintes de déplacement tel que le personnage se bloque quand il rencontre un mur… Cette grille nous a pris du temps puisqu'on ne savait pas vraiment comment s'y prendre au départ. Nous hésitions entre choisir une matrice ou un vecteur pour pouvoir mettre en place cette grille. Nous avons décidé de choisir l'option du vecteur, il est composé d'entiers entre 0 et 6, les deux premiers entiers sont les dimensions de la grille. Chaque entier correspond à une case (0 : Case Néant, 1 : Mur, 2 : Case Vide, 3 : Caisse sur Case Vide, 4 : Case Vide Cible, 5 : Caisse sur Case Vide Cible et 6 : Personnage sur Case Vide) et donc selon l'entier on va afficher ce qui est demandé. Les caisses et le personnage sont à différencier de la grille puisque ce sont des objets mobiles qui bougent sur la grille.

Enfin avec cette grille nous pouvions compléter notre fonction déplacer avec toutes les contraintes. En fait, ici, nous avons rencontré une difficulté pour connaitre la case voisine du personnage. Puisque notre grille est un vecteur les cases voisines en haut et en bas sont assez difficiles à récupérer. Après plusieurs réflexions nous nous sommes rendu compte qu'il fallait récupérer la coordonnée moins la largeur de la grille pour pouvoir obtenir ces voisines-là. Enfin, pour avoir la voisine de la voisine pour ne pas oublier le cas où le personnage veut pousser une caisse il y a une autre caisse ou un mur derrière, on a simplement multiplié par deux la largeur afin récupérer cette coordonnée-là.

5. Bilan

5.1 Avancement du projet

A la fin de ce projet nous sommes plutôt satisfaits du jeu que nous avons mis en place puisque celui-ci fonctionne parfaitement pour ces 4 différents niveaux. Seulement en partie UML nous avions prévu de mettre en place une fenêtre ou l'on pouvait changer certains paramètres comme le son ou bien changer l'image du personnage selon l'envie de l'utilisateur mais nous n'avons pas eu le temps de les mettre en place dû aux difficultés rencontrées qui nous ont fait perdre du temps.

5.2 Les point à améliorer

Le point à améliorer est le manque de continuité pour le joueur. Nous aurions pu ajouter plusieurs fonctionnalités pour rendre le jeu plus attrayant. Par exemple rendre les niveaux bloqués tant que l'utilisateur n'a pas réussi le précédent. Tous les dix niveaux, l'utilisateur

aurait pu débloquer de nouveaux personnages et de nouvelles grilles. Le joueur aurait alors plus d'objectifs et donc plus de chances de jouer à notre Sokoban plutôt qu'aux autres jeux. Il aurait pu être intéressant d'ajouter une sauvegarde afin de fidéliser l'utilisateur. Enfin nous aurions pu ajouter un mode multijoueur où deux personnages peuvent simultanément se déplacer sur la grille avec deux séries de commandes différentes sur le clavier, par exemple le joueur 1 joue avec ↑, ↓, ← et → et le joueur 2 joue avec Z, S, Q et D.

Toute ces fonctionnalités auraient pu permettre à notre jeu de se démarquer des autres et ainsi attirer plus de joueur.

5.3 Ressenti personnel du projet

Tout d'abord, il est important de différencier les parties de développement UML et de programmation C++.

Dans un premier temps, la partie UML a été faite à 8, l'organisation et la communication ont donc été plus compliquées et chacun donnait ses idées. Nous n'avions jamais parlé à certains membres de l'équipe, nous avons donc dû nous adapter à leur caractère. Par exemple essayer de poser des questions aux membres plus en retrait. Ici, rien à voir avec la programmation, nous mettions en place des diagrammes UML afin de planifier et faciliter la programmation. En fait dans cette partie, ce qui prend le plus de temps est la communication entre tout les membres du groupe et non les diagrammes UML en eux même.

Dans un second temps, la partie programmation s'est faite à 2. Lors de nombreux projets, nous avons eu le même binôme depuis le début de l'année, c'est pourquoi le coté organisationnel s'est fait très simplement entre nous deux puisqu'on se connait bien et on sait quelles sont les forces et faiblesses de notre binôme. Également nous n'avons pas d'appréhension à dire les choses à notre binôme lorsque quelque chose ne va pas.

Ensuite, nous avons dû programmer en QT et en C++. Le C++ étant une compétence maitrisée puisque depuis le début de l'année de nombreuses heures ont été consacrées à ce langage de programmation, nous n'avons pas eu énormément de mal à implémenter. Néanmoins, nous avons dû effectuer quelques recherches QT pour les différents affichages du jeu. Cette partie-là a été plus longue que la précédente puisque nous avons rencontré certaines difficultés qui ont pris plus ou moins de temps à être résolues.

Pour finir, nous n'avons pas eu le même plaisir à réaliser ces différentes parties. Simon a préféré faire la partie UML puisqu'il a plus un caractère de communiquant et d'essayer de partager les différentes taches dans le groupe. En revanche, il a moins aimé la partie programmation puisque qu'il a rencontré quelques difficultés en QT notamment. Tandis qu'Alexandre a préféré la partie programmation puisque le développement est un domaine d'activité qui l'intéresse énormément dans sa vie professionnelle future et ce projet a été pour lui une mise en place concrète d'une application. Néanmoins la partie UML l'a moins intéressée puisque pour lui la création de ces diagrammes n'était pas concrète, même si lors

du début de la programmation, il a bien compris que ces diagrammes sont indispensables pour mener à bien un projet de ce type.

5.4 Compétences personnelles

Pour finir, ce projet a été l'occasion de faire le point sur nos compétences globales en conception et programmation orientées objet. Nous avons alors pu tester nos compétences dans certains langages UML, C++ et QT mais aussi en gestion de projet.

Tout d'abord pour la partie UML, le langage était plutôt bien maitrisé pour notre binôme et c'est la que la gestion de projet était la plus importante puisque nous étions nombreux. Nous avons alors utilisé une méthode qui se rapproche de celle de la méthode Agile puisque nous avions cours environ toutes les semaines et durant ses cours nous faisions des réunions où nous communiquions nos idées puis on se séparait en petit groupe pour faire les diagrammes. Comme nous avions tous vu les méthodes Agiles et leurs utilités, nous avions déjà de bonnes bases en gestion de projet, nous devions juste nous faire comprendre aux autres et comprendre les autres.

Enfin pour la partie programmation, le langage C++ était plutôt maitrisé par notre binôme, en revanche pour l'aspect QT et tout ce qui était fenêtre graphique c'était un peu plus compliqué puisque nous n'avons pas énormément travailler cela et ce travail a été fait avant le projet, nous avions alors quelques lacunes mais le support de cours de IHM nous a bien aidé à remédier à tout cela.

En somme, nous avons progressé sur tout ces compétences en faisant un projet concret comme celui-là mais le langage où nous avons le plus progresser a été le QT.

6. Conclusion

Pour conclure, nous sommes plutôt fières d'avoir pu mettre nos compétences au service d'un gros projet qui est le développement d'un jeu. Celui-ci nous a permis de voir à quel point il peut être complexe de créer une application tel qu'un jeu. La partie UML nous a montré un sens concret à ce genre de diagramme. Donc, ce projet nous permis de nous améliorer sur tous les points de la conception et programmation orientées objet et de voir à quel genre d'application ces langages servent.