INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

INSAGAME

Projet de programmation en C

Corto CABANTOUS, Le Bao Tin HA, Kaan KORUCAN, Lucile LE BAS--CARLEZ

Sommaire

Introduction 3

Objectifs 3

- Les objectifs pour le jeu
- Les objectifs du projet

Méthodologie 3

- Organisation du projet
- Répartition du projet

Développement 4

- Conception logicielle
- Choix techniques
- Difficultés

Résultats 6

- Présentation du résultat
  - Résultats Version 1
  - Résultats Version 2
  - Résultats Version 3
- Adéquation au cahier des charges
- Perspectives

Conclusion 7

Introduction

Tout au long du S6, nous avons réalisé un projet de langage C. Notre projet consistait en la réalisation d'un jeu où l'on doit récupérer des DS sans se faire attraper par des professeurs sur une carte représentant le plan de l'INSA. Ce projet a pour but de nous familiariser avec les projets en groupe que ce soit par rapport à l'organisation des tâches, la mise en commun du travail ou de la gestion des délais. Tous ces points nous seront essentiels pour la suite de nos études et notre vie professionnelle.

I. Objectifs

A. Les objectifs pour le jeu

Pour ce jeu nous voulions mettre en scène un élève de l'INSA volant des DS sur le campus tout en esquivant les professeurs. Pour la carte nous avons souhaité reprendre le plan de l'INSA avec ses différents bâtiments représentés en 2D par une vue du ciel. Nous voulions aussi que le personnage puisse interagir avec les professeurs, les bâtiments et le DS pour permettre au joueur de d'augmenter de niveau et augmenter la complexité à chaque fois. Pour l'aspect esthétique du jeu nous voulions mettre en place des animations et des effets sonores ce qui aurait pour but de le rendre plus abouti.

B. Les objectifs du projet

Un autre objectif de cet exercice est de nous apprendre à travailler en groupe autour d'un projet commun. Que ce soit avec la répartition des tâches entre les différents membres du groupe tout en mettant en place des délais pour chaque étape. La mise en commun du travail de chacun mais aussi nous apprendre la gestion des imprévus imputable à un projet.

II. Méthodologie

A. Organisation du projet

Pour nous aider à organiser la répartition du travail et des délais nous avons fait un diagramme de Gantt. Le projet est découpé en 3 grandes versions, chacune fonctionnelle, chaque version est détaillée en tâches représentant une fonctionnalité. Pour chacune d'elles nous avons estimé leurs durées de réalisation et les dépendances entre elles. À la fin de chaque version nous avons laissé une semaine de validation qui sert de tampon pour absorber les imprévus. Pour nous faciliter le travail, nous avons détaillé chaque fonctionnalité avec les différentes caractéristiques dans un tableau que nous avons rempli ensemble pendant les réunions.

B. Répartition du projet

Ce projet respecte le modèle Modèle-Vue-Contrôleur, un motif d'architecture logicielle destiné aux interfaces graphiques. Par conséquent, la répartition principale du projet a été partagée en trois parties : deux personnes s'occupent la partie "Modèle", deux personnes s'occupent la partie "Vue". À la fin, il y aura une personne qui mettra ces deux parties "séparées" en cascade, ce qu'on l'appelle la partie "Contrôleur" pour avoir le jeu.

En outre, les parties supplémentaires du projet concernant les tests unitaires, la détection de fuite de mémoire, la gestion de Git, ... sont divisées également entre tous les membres pour s'assurer que chacun d'entre nous développent ces compétences nécessaires lors d'un projet de programmation.

III. Développement

A. Conception logicielle

Phase d'analyse: Nous voulons réaliser un jeu top-down simple avec que la partie "Front end" qui compose le mouvement avec l'animation, le son, la gestion des cliques, du clavier et les données qui parcourent en une seule fois de lancer (les vies, le score, ...).

Phase réalisation:

- Les codes sources, les headers, les codes de test unitaire et les ressources sont divisés dans les différents dossiers ("src/", "include/", "test/", "ressource/", etc.) pour simplifier la gestion de code.
- On travaille toujours sur les modules séparés (module de mouvement, de timer, de collision, de traitement de la carte, etc.). Chaque niveau du jeu est écrit dans un fichier "niveau_X.c" (où X est le numéro de niveau). L'enchaînement entre les fenêtres sera faite dans le fichier "main.c"
- La partie "Modèle" du jeu doit être testé par les tests unitaires
- Tout au long du projet, une plateforme de gestion des versions sera utilisée.

Phase livraison:

- Avant de rendre le projet, notre jeu doit être testé et validé pour éviter tous les problèmes.
- La documentation du logiciel doit être claire pour simplifier la compréhension et l'intégration continue de ce projet.

B. Choix techniques

Pour ce type de logiciel, on a créé un jeu à partir de SDL2, une bibliothèque libre pour créer des applications multimédias contenant l'affichage graphique, l'audio numérique, les périphériques d'entrées communs, etc.

Les tests unitaires ont été faits avec GoogleTest, une bibliothèque très connue pour sa simplicité d'approche. Basé sur l'architecture xUnit, GoogleTest peut cependant être lancé sur n'importe quel système d'exploitation.

Les langages C et C++ sont souvent mentionnés comme des langages nécessitant de porter une attention particulière à la gestion de mémoire. Par conséquent, Valgrind est notre meilleur choix pour détecter les fuites et les autres problèmes de mémoire.

Le gcov a été utilisé pour vérifier quelles instructions ont été exécutées, celles qui ne l'ont pas été, et celles qui ne pourront pas l'être (code inaccessible, ou code mort). Ensuite, nous avons exécuté le lcov pour avoir notre rapport de code coverage.

Comme ce projet a été principalement travaillé sur l'IDE CLion, nous avons pu profiter de son programme CMake afin de générer notre projet (sans installer CMake à part, nous écrivons seulement les CMakeLists.txt et laissons CLion lancer son CMake). Sur Linux, Valgrind et gcov est aussi lancé sur CLion pour simplifier l'observation des détections.

C. Difficultés

Lors de la réalisation de ce projet, nous avons rencontré diverses difficultés. La première à mentionner est d'avoir travaillé à distance tout au long du projet. Cette contrainte était exigeante surtout au début, étant donné que l'on devait faire des réunions pour planifier notre trajectoire. Pour régler ce problème, nous avons fait des réunions virtuelles. Nous avons également créé un groupe sur un système réseau afin de pouvoir communiquer à tout moment.

La deuxième difficulté a été le temps limité du projet. Après avoir terminé notre deuxième version indiquée sur le diagramme de Gantt, nous nous sommes rendus compte que pour mener ce projet à bien et le compléter sans retard, nous devions faire des modifications au niveau de notre troisième version. Nous devions être efficaces et décisifs car sinon nous n'aurions pas assez de temps pour terminer notre projet. En modifiant nos cibles pour la version 3, nous avons réussi à la compléter sans retard.

Dernièrement, nous avions un souci concernant les systèmes d'exploitation. Tous les membres du groupe n'utilisaient pas le même OS, et cela nous a posé un vrai problème. Nous avons donc décidé de faire fonctionner notre jeu dans deux systèmes d'exploitation différents (Linux et Windows). Cela était un peu difficile, mais nous étions convaincus que nous ferions le fonctionner. A l'aide de notre encadrant pour ce projet, M. Sourbier, nous avons réussi ce défi. Le logiciel de débogage des fuites de mémoire Valgrind était un autre

souci concernant notre problème de système d'exploitation. Comme il était difficile de faire fonctionner Valgrind sur Windows, nous avons utilisé nos machines virtuelles pour le faire fonctionner sur Linux.

IV. Résultats

A. Présentation du résultat

Résultats Version 1

Pour la première version, nous nous sommes intéressés à la réalisation des éléments de base pour notre jeu. Nous avons tout d'abord commencé par réaliser notre carte pour le jeu. Ensuite, nous avons ajouté les fonctionnalités de déplacement du personnage avec les flèches du clavier, récupération du DS et arrêt du jeu si on touche le professeur.

Résultats Version 2

Pour la deuxième version, nous avons ajouté plusieurs fonctionnalités. Nous avons intégré la possibilité de jouer à plusieurs niveaux avec une difficulté croissante ainsi qu'une page d'accueil et de fin. De plus, un timer est ajouté pour complexifier le jeu. Nous avons aussi intégré un placement aléatoire du DS pour que le jeu ne se répète pas. Dernièrement, nous avons limité le nombre de vies de notre personnage à trois. En ajoutant toutes ces fonctionnalités, nous avons remarqué que nous avions des gros problèmes concernant les fuites de mémoire. A cause de cela, le jeu ne fonctionnait pas comme on le voulait. Donc, dans cette version et la version suivante, nous nous sommes concentrés à résoudre ce problème.

Résultats Version 3

Finalement, dans la dernière version, nous avons généré plusieurs professeurs pour les niveaux difficiles. Nous avons également intégré l'animation du personnage et des professeurs. Pour rendre le jeu plus amusant, nous avons inséré dans le jeu la musique d'ambiance et des sons. Avec l'animation des personnages, les fuites de mémoire étaient de nouveau au centre de nos préoccupations.

B. Adéquation au cahier des charges

Au cours de ce projet, nous avons utilisé le gestionnaire de version git via la plateforme GitLab et avons documenté notre code en respectant le formatage Doxygen, comme demandé dans le cahier des charges. De plus, nous avons réalisé nos tests unitaires avec GoogleTest (basé sur l'architecture xUnit) et Valgrind pour détecter les fuites mémoires. Nous avons choisi d'utiliser un CMake, ce qui nous a permis de créer un projet compatible pour Windows et Linux assez facilement.

Notre projet a été réalisé selon un modèle MVC et chacune des trois versions a ajouté de nouvelles fonctionnalités majeures au jeu. Les deux premières versions ont été fidèles au cahier des charges et on été réalisées dans le temps imparti. Afin de finir le projet à temps, nous avons cependant dû faire des concessions et modifier la version 3 initialement prévue. Nous avons choisi de nous concentrer sur la partie animation et gestion des sons. Nous avons redéfini les niveaux pour inclure certaines de nos idées initiales, notamment l'inversion des contrôles.

C. Perspectives

Pour poursuivre ce projet, nous aurions souhaité pouvoir intégrer des bâtiments avec des actions spéciales sur le personnage, comme par exemple l'infirmerie qui redonnerait une vie ou le Foy qui inverserait les contrôles. Nous aurions souhaité améliorer les graphismes du personnage et des professeurs. Sur du plus long terme, nous aurions pu imaginer un jeu avec plusieurs DS par niveaux et éventuellement jouer à deux pour avoir une stratégie d'équipe, voire même des objets pour détourner l'attention des professeurs ou se cacher. Finalement, nous aurions aussi pu faire du Q-learning pour apprendre aux professeurs à éviter les murs et donc à avoir des trajectoires plus naturelles.

V. Conclusion

Ce projet nous a permis d'apprendre à nous organiser correctement dès le début en réalisant un diagramme de Gantt. Ainsi, nous avons pu estimer le temps de travail nécessaire à la réalisation des tâches mais aussi nous séparer en sous-groupes afin de nous répartir le travail. Nous avons aussi appris à utiliser divers outils très utiles dans la vie professionnelle (GitLab, Doxygen, GoogleTest, Valgrind). Même si nous avons dû modifier notre dernière version, nous avons tout de même pu réaliser 3 versions de notre jeu. Nos attentes étaient peut-être trop élevées pour le temps imparti.

INSA Rennes
20 Avenue des Buttes de Coësmes
CS 70839
35708 Rennes Cedex 7

Tél. +33 (0) 2 23 23 82 00
Fax +33 (0) 2 23 23 83 96

www.insa-rennes.fr

INSA

MINISTÈRE
DE L'ENSEIGNEMENT
SUPÉRIEUR,
DE LA RECHERCHE
ET DE L'INNOVATION
Liberté
Égalité
Fraternité