Projet tutoré de C++

Jorandon Guillaume et Simon Clément
IUT d'Amiens
Tuteur : Laurent Delahoche

6 juillet 2016

Table des matières

Introduction

Dans le cadre du DUT Informatique AS de l'IUT d'Amiens, nous avons été amenés à mettre en pratique ce que nous avions étudié tout au long de l'année dans un projet tutoré de développement applicatif en C++. Le sujet était imposé et demandait la réalisation d'un *fork* de Bomberman, un *puzzle game* apparu en 1983 sur *ZX Spectrum*, puis popularisé notamment sur les consoles de la firme *Nintendo*. Ce jeu-vidéo très simple dans son principe met en scène un personnage dans un décor fermé, qui a la capacité de poser des bombes afin de progresser dans le jeu. C'est un exemple parfait pour mettre en pratique les notions de programmation et d'algorithmique apprises en cours, puisqu'il implique la création d'un moteur de jeu simple.

Nous allons dans ce rapport revenir sur les différentes étapes de la conception et de la réalisation de notre Bomberman. Nous reviendrons d'abord sur la pré-conception du projet : interprétation du sujet, choix des technologies. Puis nous expliciterons dans une deuxième partie les détails de la réalisation : analyse descendante, diagrammes de classe, boucle de jeu, etc.

Le code est téléchargeable en clonant le dépôt du projet. Pour compiler le projet, exécutez le makefile (attention à bien avoir correctement installé SFML au bon endroit). Exemple sous GNU/Linux :

```
~$ git clone https://github.com/dolfinsbizou/projet-tut-cpp.git
~$ cd projet-tut-cpp && make
~/projet-tut-cpp$ ./bomberman # -h sur bomberman pour obtenir de l'aide
```

Chapitre 1

Génèse

1.1 Interprétation du sujet

La première étape a été d'interpréter le sujet donné. Le minimum exigé par le sujet comporte plusieurs éléments de base.

Principe du jeu de base Le jeu doit correspondre à une version minimaliste de Bomberman. Cela peut sembler évident, mais il faut s'assurer de connaître un minimum le jeu afin de ne pas faire de hors-sujet. Bien heureusement, Bomberman est un jeu extrêmement connu.

Interface graphique Le jeu doit être implémenté dans une interface graphique utilisateur, et non pas en ligne de commande.

Gestion du déplacement Le personnage doit être capable de se mouvoir, contrôlé par le joueur.

Gestion des collisions avec le décor Le personnage ne doit pas traverser les obstacles, ni sortir des limites de la carte.

Pose de bombes Le personnage doit être capable de poser une bombe, qui explose au bout d'un court instant en propageant une explosion.

Nous avons cependant très vite décidé de nous fixer plusieurs exigences supplémentaires, afin d'obtenir un jeu plus achevé et jouable.

Présence d'un menu Plutôt que de lancer directement la partie au démarrage du programme, et de fermer celui-ci à la fin de la partie, un menu présente une interface moins austère à l'utilisateur.

Affichage texturé Plutôt que de représenter les éléments du jeu par des polygones colorés, l'utilisation de textures (*sprites*) rend le jeu plus élégant, en plus d'apporter des problématiques de programmation supplémentaires.

Textures animées Des animations rendent le jeu moins statiques et posent des problèmes de gestion du temps.

Deux joueurs contrôlés par deux humains Le jeu devient ainsi réellement jouable, avec un objectif clair : annihiler son adversaire.

Destruction des éléments du jeu La bombe peut détruire certains blocs et tuer les joueurs.

Ainsi, un cahier des charges se dégage du sujet initial, ce qui nous donne un certains nombre d'objectifs clairs à remplir.

1.2 Choix des technologies

Le sujet n'imposait que le langage de programmation (C++). Ainsi, l'utilisation de bibliothèques tierces, notamment pour les graphiques, était laissé à la libre appréciation de chaque binôme. Nous avons décidé de ne pas partir sur la bibliothèque graphique présentée en cours (Allegro), pour utiliser plutôt SFML (Simple and Fast Multimedia Library) [1]. Cette API, créée par Laurent Gomila, un développeur français, présente plusieurs avantages. Elle est écrite en C++ et entièrement orientée objet, et est très modulaire. De plus, elle est portable, ce qui signifie qu'un programme qui l'utilise peut être compilé et tourner sous Windows, macOS et GNU/Linux sans problème. Enfin, l'un de nous deux utilisait déjà SFML, ce qui permet de gagner du temps sur l'apprentissage de l'API.

Nous avons aussi utilisé doxygen pour commenter le code. Doxygen est un logiciel de génération de documentation qui nous a permis de créer facilement la documentation du code.

1. SFML est distribuée selon les termes de la licence zlib/png.

Chapitre 2

Développement

Table des figures

Liste des tableaux