Projet informatique

Samuel Givois

ENSAE

samuel.givois@ensae-paristech.fr

16 mai 2018

I. Introduction

L'objectif fixé pour ce projet informatique est la création d'un jeu, ou plus précisément la mise en place d'une structure minimale de jeu, pouvant être par la suite enrichie facilement. Le jeu choisi est « Bomberman », créé par l'éditeur Hudson Soft en 1983. Il s'agit d'un jeu en 2 dimensions, avec vue isométrique (l'univers est vu d'en haut). Le joueur est dans un labyrinthe, et peut poser des bombes. Celles-ci peuvent détruire des murs de brique ou des ennemis. En détruisant un mur, des bonus (ou malus) apparaissent. Le succès du jeu est venu de son mode multijoueur, le but étant d'être le seul survivant. On a cherché ici à développer en priorité un jeu pour un seul joueur, plus facile à réaliser. Cependant, le résultat obtenu devra permettre de passer facilement à un mode multijoueur.

II. Choix des technologies

La plupart des langages existant et l'écosystème informatique permettent facilement le développement de jeux sur ordinateur. Cela est dû à l'ancienneté de ces systèmes, avec pour conséquence une communauté informatique mature et des sources documentaires en profusion. Le développement se faisant essentiellement sur ordinateur, les tests sont par ailleurs facilités pour les applications de bureau. Cependant, il est difficile d'ignorer la prépondérance du mobile aujourd'hui, en particulier pour les jeux simples comme « Bomberman ». On a donc cherché lors de ce projet à développer un jeu qui soit compatible pour ces deux types de terminaux, les ordinateurs non tactiles *a priori* et dotés d'un clavier, et les mobiles et tablettes.

N'ayant jamais développé de jeux vidéo et afin d'avoir le temps d'assimiler les contraintes et technologies associées à ce type de projet, on a choisi de se baser sur un langage qu'on maîtrisait déjà, le langage Java. Pour permettre la réalisation d'un jeu multiplate-forme, on a étudié plusieurs solutions possibles reposant sur l'écosystème Java. Une première piste intéressante fut l'API JavaFX. Cependant, il est rapidement apparu que la portabilité sur mobile était un peu plus compliqué que prévu. On s'est alors orienté sur le « framework » libGDX, créé par Mario Zechner en 2014. Celui-ci, une fois installé et correctement configuré, permet une réalisation facilitée de livrables dédiés pour chacun des environnements cibles (bureau, Android, iOS, ...).

Par habitude, on a utilisé l'environnement de développement Eclipse pour l'écriture du code. Cependant, Google a interrompu le support des outils de développement pour Android intégrés à Eclipse en novembre 2016, et a créé son propre environnement de développement Android Studio. Les outils d'Eclipse ne fonctionnant qu'imparfaitement, on a également utilisé l'IDE de Google pour les besoins spécifiques aux applications mobiles (test et construction des livrables en particulier).

Le « framework » libGDX repose en outre sur le moteur de production Gradle, permettant notamment la gestion des dépendances et la construction de livrables (à l'instar de Maven). Eclipse et Android Studio sont capables de gérer cet outil. On a enfin choisit git pour la gestion de version, avec un dépôt central sur le site GitHub.

III. De la carte à l'écran

Un avantage supplémentaire de libGDX, est qu'il permet de manipuler des fichiers de carte au format TMX. Ceux-ci définissent des cartes carroyées (« tiled map »), souvent utilisées pour les jeux en vue isométrique. Le principal intérêt est de pouvoir découpler la partie développement de la partie création des cartes. Pour réaliser ces dernières, on a utilisé le logiciel Tiled, créé par Thorbjørn Lindeijer.

Permettre le passage de la carte au jeu à proprement parler est ce qui a constitué la majeure partie du travail de ce projet (disons 80 %). En effet, cela a nécessité un temps conséquent pour comprendre le « framework » et ses possibilités. En contrepartie, peu de développement seront nécessaires pour l'extension du jeu (création de nouveaux niveaux ou de nouveaux personnages).

LibGDX fournit un modèle complexe pour représenter l'univers du jeu (à l'instar de l'univers qu'on connaît, avec la gravité par exemple). Après de nombreux essais infructueux, on a choisi d'utiliser une représentation simplifiée, qu'on a implémenté spécifiquement. On a également adapté un autre outil, la caméra, afin de permettre la création de niveaux dépassant l'écran (dans l'optique de livrables pour mobile en particulier). On a ainsi réussi à obtenir une vision centrée sur le joueur si la carte dépasse le cadre de l'écran, dans la limite du bord de la carte.

IV. Développement du jeu

Une fois réalisé le processus de lecture et d'adaptation de la carte à l'écran, le développement du jeu à proprement parler a pu débuter. Malheureusement, étant donné le temps consacré à la réalisation d'un cadre de développement le plus propre et extensible possible, cette partie n'a pas été aussi avancée qu'escompté.

La première étape était de pouvoir faire avancer le personnage sur la carte. Pour cela, plusieurs règles on été établies. Premièrement, le personnage avance d'un quart de largeur de case. Deuxièmement, le personnage ne doit pas pouvoir atteindre une case inaccessible (comme une brique ou un mur). Enfin, une avancée ne peut se produire moins d'un dixième de seconde après une autre avancée. Les premiers et troisièmes points permettent ainsi de définir une vitesse (un quart de bloc par dixième de seconde). Ce déplacement a été associé à la manipulation du clavier pour un utilisateur sur ordinateur (flèches directionnelles) et aux déplacements de doigts pour les terminaux à écran tactile.

Dans un second temps, il a fallu permettre au personnage principal de déposer des bombes. De nouveau, il a fallu établir plusieurs règles. Deux bombes ne peuvent être posées sur la même case. Cinq secondes après avoir été déposée, une bombe explose, avec propagation dans les directions des quatre points cardinaux. Lorsque la déflagration atteint un mur, elle est stoppée ; pour une brique, elle la détruit (libérant un bonus éventuel, ou une sortie si elle la recouvrait) puis est stoppée ; pour un personnage du jeu, elle lui fait perdre un point de vie. De nouveau, la gestion du temps (ou plus précisément du temps écoulé à chaque cycle de jeu) est importante. Il a fallu prendre en considération le temps avant l'explosion, le temps de propagation et le temps avant qu'un personnage puisse perdre un nouveau point de vie.

Pour les utilisateurs sur ordinateur, le dépôt de bombe a encore été géré par l'utilisation du clavier (la touche espace). Pour les écrans tactiles, la gestion de cet évènement (en supplément de la direction) est apparue compliquée. On a donc décidé de créer un bouton dédié, en bas à droite de l'écran. Cela a nécessité de complexifier l'écran de jeu et d'utiliser des fonctionnalités plus poussées du framework libGDX. Cet investissement apporte toutefois un bénéfice : l'espace de droite pourrait être employé pour la création d'un véritable menu de jeu, utile et ergonomique, affichant des informations (le nombre de points de vie typiquement) ou offrant de nouvelles fonctionnalités (activation de pouvoirs par exemple).

V. Navigation

Afin d'offrir une interface plus évoluée qu'une simple carte et de permettre la navigation entre les différents niveaux du jeu, on a exploité un autre aspect du framework libGDX, permettant la création de menus. Cette étape, entreprise après le développement du jeu (et donc avec une capitalisation sur la connaissance de l'outil) a été relativement aisée à réaliser. Pour aller plus loin, le travail à effectuer porterait surtout sur le design et la persistane.

VI. Par la suite

On a donc réussi à mettre en place un embryon de jeu extensible facilement (que ce soit la création de nouveaux niveaux ou de nouveaux éléments du jeu). De plus, ce jeu est adapté aux contraintes des ordinateurs et des mobiles, et fonctionne (sur ordinateur et sur Android pour l'instant).

Pour permettre d'obtenir un véritable produit fini (commercialisable par exemple), de nombreuses pistes pourraient être explorées, au-delà du design. Il serait intéressant par exemple de développer le menu droit des écrans de jeu. Il serait également intéressant d'introduire de la persistance, afin qu'un joueur puisse sauvegarder ses parties. Un autre angle envisageable, plus ambitieux, consisterait à proposer une version multijoueur. Cela nécessiterait alors la création d'un serveur dédié pour la gestion des communications entre utilisateurs (architecture qui nous a semblé dépasser le cadre de ce cours).