Rapport projet BOMBEIRB - PG110 - GbT2

Julien MATHET & Théo LEPINE

16 mai 2018

1 Introduction

Ce rapport présente de manière succincte les fonctionnalités de notre jeu. Pour une meilleur compréhension, veuillez noter la différence suivante : le mot **player** représente le bombeirbman et le mot **joueur**, l'humain derrière son clavier.

2 Lancement d'une partie

Depuis le menu, le jeu peut se lancer de deux manières différentes :

- par une difficulté : les propriétés sont récupérées depuis un fichier *game_*.txt* grâce à la structure **game_infos** qui permet de génerer la structure **game**
- par une sauvegarde : la structure **game** est reconstruite depuis le fichier *player_saved.txt*

3 La structure game

La structure **game** permet de stocker toutes les informations du jeu. Elle peut-être créée ou chargée. Elle est composée notamment de pointeurs vers les autres structures du jeu tel que :

- **player** qui contient position, vie, inventaire et propriétés du player
- **bombs** qui désigne la liste chaînée de toutes les bombes du jeu
- **maps** peut être considéré comme un tableau des cartes (propriétés + grille)
- **monsters** qui désigne la liste chaînée des monstres du jeu

Nous avons également ajouté des champs comme :

- **exit_reason** indique la raison pour laquelle le jeu s'arrête (game over, victoire, pause)
- **game status** indique si le jeu est en pause ou non
- **break_time** qui est le temps de pause
- **break_time_begin** qui est le temps depuis le début de la pause (permet de calculer simplement *break_time*)

4 Gestion des bombes

Nous utilisons une **liste chaînée** pour gérer les bombes. La structure d'une bombe a les paramètres suivants : ses coordonnées dans l'espace (x et y), sa portée, son niveau dans laquelle elle se trouve, son temps d'initialisation, son état (état de la mèche ou explosion).

Au lancement du jeu, l'**initialisation** du pointeur des bombes est un pointeur NULL. Pour poser une bombe, le joueur doit appuyer sur la touche ESPACE, cela appelle la fonction *bomb_add* sur la position du player à condition d'avoir suffisamment de bombes dans l'inventaire. Ceci entraîne une décrémentation du nombre de bombes disponibles pour la player qui sera à nouveau incrémenté une fois l'explosion terminée.

L'**affichage** des bombes se fait avec l'appel de *bomb_display* qui met également à jour l'état de toutes les bombes (fonction *bomb_update*).

— Pour un **mèche** (état entre 3 et 0), la fonction affiche l'image correspondante.
— Pour une **"explosion"** (état −1), la fonction *bomb_explosion* va regarder dans les 4 directions, en tenant compte de la portée.
— Pour une bombe **"explosion terminée"** (état −2), la fonction *bomb_destruction* permet de supprimer la bombe en question et d'incrémenter le nombre de bombes disponibles pour le player.

5 Gestion des bonus

Les bonus sont contenus dans les caisses. Ils apparaissent sur la carte après la destruction de la caisse. Lors d'une explosion, la fonction *bomb_explosion_box_type* permet de de laisser un bonus à l'ancien emplacement de la caisse.

6 Gestion des monstres

La gestion des montres se fait grâce à une **liste chainée**. La structure d'un monstre est composé des attributs suivants : ses positions, son niveau de carte (numéro de la carte sur laquelle il se trouve), son temps d'initialisation et sa vitesse de déplacement.

L'**initialisation** consiste à parcourir les cartes pour ajouter les monstres qui ne se cachent pas dans des caisses.

Le **déplacement** des monstres dans toutes les directions se fait aléatoirement grâce à la fonction *rand()*. Avant de déplacer un monstre, la possibilité du déplacement est vérifié. La vitesse de déplacement des monstres est fonction du niveau sur lequel le player se trouve.

La **mort** d'un monstre est causée par une explosion sur celui-ci. La fonction *monsters_delete_monster* est appelée pour supprimer de la liste le monstre tué.

7 Les menus et la pause

Ces écrans permettent au joueur d'interagir avec **sa souris**. Les boutons sont placés par rapport à la taille de la fenêtre. Une version *hover* est affichée au passage du curseur. Le clic est détecté pour le lancement du mode de jeu.
Le menu principal sert également en cas de **sortie du jeu** (game over ou victoire) avec un affichage temporisé d'image avant l'affichage des boutons.
Le menu de pause s'ouvre en pressant **P** en jeu et permet de sauvegarder la partie.

8 Lecture des cartes

Les cartes sont stockées dans des fichiers *map_<difficulte>_<number>.txt*. Le format des fichier est **amélioré** grâce à : une structure **compréhensible** et un champ de **couleur de fond**.

Chaque fichier contient ainsi les propriétés et la grille complète de la carte. La lecture de ce fichier **détecte automatiquement** chaque champ et est **robuste aux surplus** d'espaces dans la grille.

L'ensemble des cartes est chargé dès le lancement de la partie, soit depuis les fichiers de cartes, soit depuis un fichier de sauvegarde.

9 La sauvegarde

Une partie peut-être **sauvegardée** depuis le menu pause. Les informations sont stockées dans le fichier *data/player_saved.txt*.
Ce fichier est **compréhensible** avec la succession de tous les champs des structures *game*, *player*, *maps*. Ceci permet de conserver toutes les avancées du joueur.

Pour la **restauration** de la sauvegarde, le fichier est parcouru pour régénérer les différentes structures. Certaines propriétés comme les timers sont néanmoins remis à zéro.
Il n'est possible de charger une partie que si une sauvegarde existe. La sauvegarde se détruit à chaque nouvelle sauvegarde, en cas de défaite ou de victoire.