LINGI1131 - Rapport de projet

Louis Navarre

March 2019

1 Introduction

Ce rapport concerne le projet pour le cours LINGI1131 - Concepts de langages de programmation (*Computer languages concepts*).

Dans un premier temps, en ce qui concerne la partie obligatoire, nous avons effectué tout ce qui était demandé. Des explications sur la structure de notre implémentation de notre contrôleur et de nos joueurs concerne la première partie de notre rapport. Dans un second temps, nous avons ajouté quelques extensions à notre travail. Tout d'abord, nous avons ajouté les deux bonus supplémentaires conseillés lorsqu'un joueur en ramasse un sur le terrain de jeu: le bouclier et la vie supplémentaire. Ensuite, nous avons modifié à notre guise le `GUI` afin de le rendre plus ludique et plus approprié à un jeu lugubre et mortel tel que le *Bomberman*. **AJOUTER ICI LES AUTRES BONUS EFFECTUES!**.

2 Contrôleur `Main`

Notre contrôleur est séparé en deux parties: la gestion des évènements et la gestion des joueurs. Faire de la sorte nous permet d'éviter de dupliquer du code pour le mode de jeu tour par tour, et le mode simultanée. Ainsi, les seules différences notables entre ces deux modes de jeu se situent au niveau de la gestion des joueuers - la gestion des évènements est donc identique peu importe le mode de jeu. Toutes les fonctions/procédures référencées dans cette section sont relatives au fichier `Main.oz`.

2.1 Gestion des joueurs

2.1.1 Mode tour par tour

Le mode tour par tour est totalement géré par la procédure `TurnByTurn`. Les arguments de cette procédure sont:

- `ThePlayersPort`: la liste de joueurs à encore traiter dans l'itération.
- `TheBombs`: les bombes posées et non encore explosées.

Cette procédure "itère" (en effectuant des appels récursifs) sur la liste des ports des joueurs présents dans la partie (même les joueurs décédés). Lorsque l'itération est terminée (i.e. le reste de la liste à traiter est `nil`), la procédure vérifie l'état des bombes en attente d'explosion, et traite les bombes devant exploser à ce tour, comme expliqué dans le paragraphe correspondant ci-dessous. Ensuite, la procédure effectue un appel récursif effectue un appel récursif **C'EST PAS LE CAS, A CHANGER DANS LE CODE** si la fin de partie n'est pas détectée; et recommençant l'itération depuis le début de la liste de ports de joueurs. Les deux paragraphes ci-dessous détaillent le fonctionnement de la procédure en fonction de l'état de l'itération.

Pour un joueur. Si le joueur est toujours en vie**??**, alors nous demandons au joueur d'effectuer sa prochaine action. S'il s'agit d'un déplacement (`move`), alors nous vérifions regardons si le joueur vient de récupérer un point/bonus. Nous traitons le point/bonus, et mettons à jour la carte (*map*)**??**. S'il s'agit d'une bombe, alors nous ajoutons la position de la bombe plantée, l'identifiant (`ID`) du joueur l'ayant posée, ainsi que le nombre de tours avant explosions (`Input.timingBomb`).

En fin de tour. Lorsque le tour est fini (i.e. tous les joueurs ont effectué leur action), la procédure vérifie l'état des bombes: nous décrémentons les *timers* de chacune des bombes posées, et nous procédons à l'explosion des bombes où ce *timer* est nul**??**.