Rapport
Projet Programmation 2
Première Partie

Thomas Dupriez et Guillaume Hocquet

3 mars 2015

1 Bref historique

La première partie du projet s'est déroulée en trois phases, avec une réécriture du code lors de chacune d'elles :

— Écriture d'un démineur sans le séparer de l'interface graphique
— Réécriture du code précédent, en veillant à "l'objetiser" un peu plus et à le séparer en deux parties : le jeu et des fonctions d'interfaces graphiques plus générique.
— Deuxième réécriture pour sortir quasiment toutes les fonctions d'interface graphique du démineur pour les rendre réutilisable pour d'autre jeu. Cela a restreint les capacités de l'interface graphique mais celles-ci devraient rester suffisantes pour les puzzles de Simon Tatham

2 Les concepts autonomes de l'interface graphique

L'interface graphique repose sur un certains nombre de concepts génériques qui seront utilisés par les jeux en les adaptant. Les codes associés à ces concepts sont dans des fichiers séparés dont le nom commence par "GUI". Les fonctions de l'interfaces graphiques utilisent des méthodes ou des variables du jeu via leurs paramètres "game". Pour assurer que le jeu fournit bien tout ce qu'il faut, l'objet jeu doit étendre la classe Game du fichier GUI.scala qui est une signature et liste les valeurs, variables, méthodes et types que tout jeu doit fournir.

2.1 Les états de label (GUI_Label_State.scala)

Les états de labels ont vocations à être des ensembles de paramètres graphiques tels que la couleur du background ou le texte d'un label. Les états de labels seront créés par le jeu en héritant de la classe Label_State. Les états créés héritent ainsi d'une méthode qui permet de changer les paramètres graphiques d'un label pour qu'ils s'accordent à ceux de l'état.

2.2 Les labels Chronomètres (GUI_Timer_Label.scala)

La classe Timer_Label permet de récupérer simplement des labels faisant office de chronomètres et dotés de trois fonctions :

— restart : permet de changer l'origine temporelle du label, typiquement lorsqu'on démarre une nouvelle partie
— stop : permet de geler le chronomètre
— start : permet de relancer le chronomètre

2.3 Les grilles et les labels de grille (GUI_Grid.scala)

La classe Grid_Label est destinée à être étendue par le jeu pour constituer les cases de la grille. Elles contient pour ce faire des variables permettant au label de connaître ses coordonnées dans la grille (x et y) ainsi que son numéro (la numérotation se fait à partir de 0 dans le coin supérieur gauche de la grille, de gauche à droite puis de haut

en bas) et le nom de l'état dans lequel il est (state). La classe Grid crée un GridPanel d'une taille correspondant aux paramètres du jeu (nb_of_rows et nb_of_cols), puis le remplit avec des labels de la classe passée en argument. Grid fournit aussi trois fonctions pour accéder aux labels de la grille ainsi créée :

— Soit avec l'abscisse et l'ordonnée du label
— Soit avec le numéro du label
— Soit en renvoyant la liste des labels ordonnés par leurs numéros

2.4 Les formulaires à nombres (GUI_Number_Form.scala)

La classe Number_Form permet de faire s'afficher une fenêtre qui demande à l'utilisateur de renseigner certains champs numérique. On peut spécifier le nom des nombres demandés, ainsi que des bornes inférieures et supérieures pour chaque réponse demandée. La fonction appelant Number_Form peut récupérer les résultats dans la séquence indexée "result".

3 L'interface graphique proprement dite (GUI.scala)

GUI.scala prédéfinit des couleurs et des bordures de label mais elle fait bien plus que cela ! A partir d'un jeu, elle construit automatiquement la fenêtre principale ainsi que ses menus, les procédures pour lancer une partie, en recommencer une, utiliser une graine aléatoire spécifiée, lancer une partie custom, gagner et perdre. Elle construit aussi toute seule le contenu de la fenêtre pour chaque partie, avec la grille et les labels du bandeau inférieur.

3.1 La signature Game

GUI définit la signature Game, dont tout jeu doit hérité et qui spécifie ce dont elle a besoin pour créer l'interface graphique du jeu.

3.2 Les modes de difficulté

GUI fournit la classe abstraite Difficulty_Mode, dont chaque jeu doit créer uns sous-classe de type case class dont les constructeurs seront les paramètres définissant une partie du jeu (nombre de colonnes, nombres de lignes et nombre de bombes dans le cas du démineur). Elle contient une méthode devant être écrite par le jeu qui doit assigner aux variables du jeu correspondants aux paramètres d'une partie les valeurs qu'elles ont dans le mode de difficulté.

3.3 La classe Game_Frame_Content

Cette classe construit le contenu de la fenêtre du jeu pour chaque partie. Elle génère la grille et les labels du bandeau inférieur.

3.4 L'UI

La classe UI est le coeur de l'interface graphique. Elle construit la fenêtre principale et ses menus. C'est cette classe que la classe Main de chaque jeu va instancier et laisser travailler. Elle utilise les classes Generic_Game_Starter et Generic_Action_Restart et définit les méthodes action_generic_random_seed et action_generic_custom_mode pour disposer des procédures de lancement d'une partie, de relancement d'une partie, d'utilisation d'une graine aléatoire définie et de lancement d'une partie custom. Ces procèdures sont donc définies pour tout les jeux mais appellent des méthodes définies dans le jeu (ce qui permet au jeu de définir comment il souhaite s'initialiser par exemple).

4 Le Démineur

Cette partie concerne le démineur lui-même. Les fichiers correspondant au démineur commencent par "Demineur". La grande majorité du travail graphique est

effectuée par GUI et a déjà été exposée, la partie du démineur satisfaisant les besoins de l'interface graphique ne sera donc pas expliquée.

4.1 Les états de labels du démineur

Le Démineur définit trois états de label : "explored", "unexplored" et "flagged". Il définit également la classe "Demineur_Label_States_Manager" dont l'objectif est de regrouper les trois états de label et de fournir aux Demineur_Label une méthode plus agréable pour changer d'état ("change_to_state").

4.2 Fonctionnement général du démineur

Le premier label à être cliqué appelle la fonction "place_bombs" qui place les mines dans la grille de sorte à ce que le carré entourant le label ne contienne aucune mine. Ensuite, tout label non-exploré cliqué avec le clic gauche se découvre en découvrant également les labels voisins si aucun d'eux ne contient de mines. Un label cliqué avec le clic droit passe dans l'état flagged et ne pourra donc pas être exploré tant qu'il restera dans cet état.