



DGUI Scrabble

Debecker Bernard
Foncier Romain
Morel Arnaud

Jan 2013

Présentation de l'interface graphique du jeu

L'interface de l'application est divisée en deux éléments correspondant à deux vues séparées :

Game : le plateau de jeu, le rack et les boutons du jeu

Menu : les informations du joueur, de la partie (score), les paramètres d'affichages du jeu et les différents buttons de scénarios accessibles.

Vue initiale

La vue initiale présente un plateau de jeu et un chevalet vides ainsi que 3 boutons : **Play as guest**, **Log In** et **Sign Up**.



Figure 1 - Vue du jeu au lancement de l'application

Boutons initiaux

Les 3 boutons initiaux offrent au joueur différents scénarios :

- Jouer directement de façon anonyme
- Se loguer
- Créer un compte.

Remarque : Un joueur loggué aura la possibilité d'enregistrer ou de sauver sa partie à l'inverse d'un joueur anonyme.

Prenons l'exemple d'un joueur anonyme. Celui-ci clique sur le bouton **Play as guest** entraînant l'initialisation du jeu et l'affichage suivant :

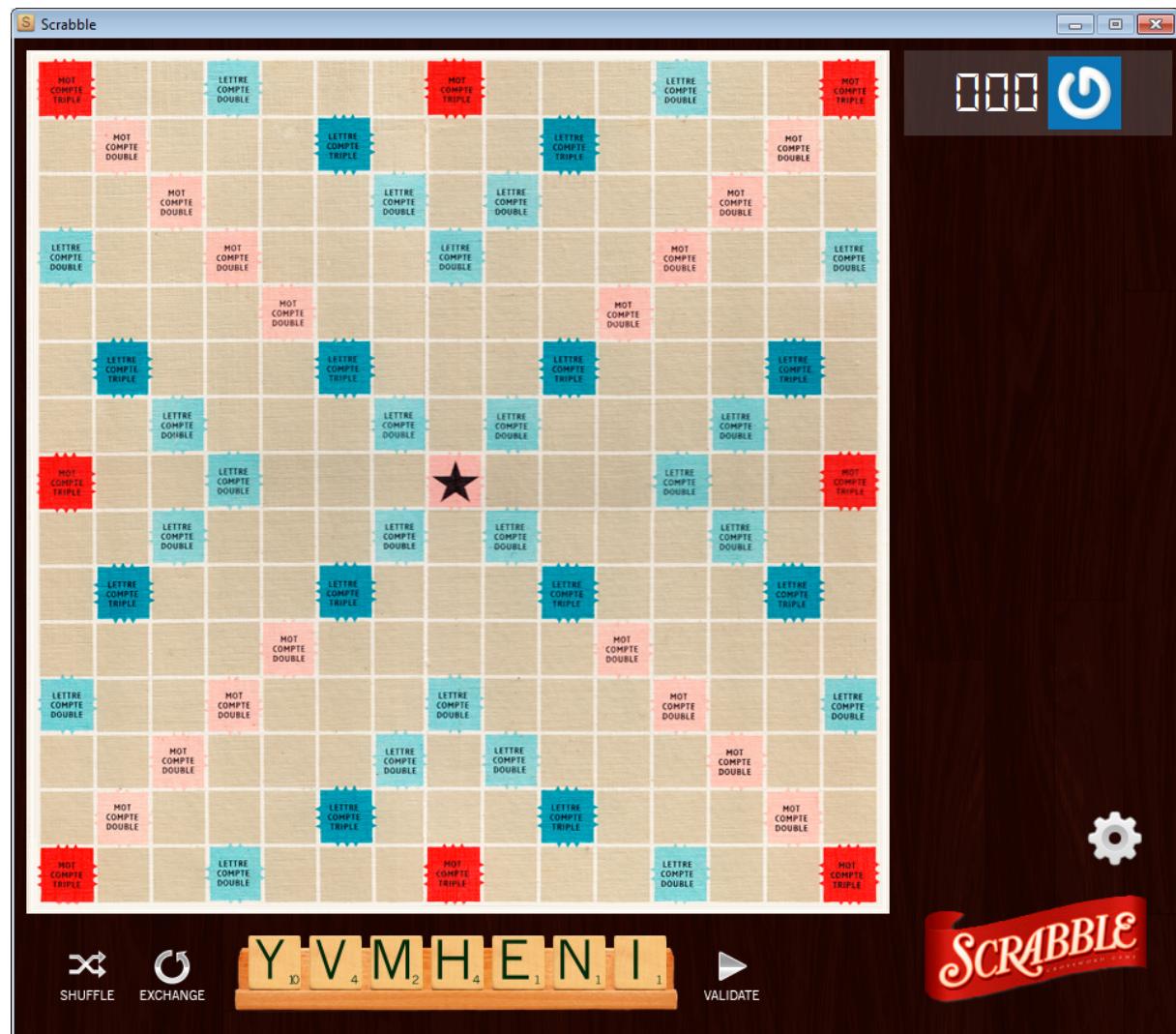


Figure 2 - Vue initiale d'une partie anonyme

S'exécutent les tâches suivantes :

Initialisations du Game :

Initialisation du chevalet

Ajout des boutons de jeu à la frame : **Mélange, Echange, Valider**

Initialisation du Menu :

Ajout des informations du joueur

Ajout des informations de la Partie (score)

Chargement des paramètres

Une fois les éléments chargés, le joueur aura le loisir d'effectuer les actions suivantes :

- Commencer sa partie
- Paramétriser les réglages de l'affichage

En cliquant sur le bouton Paramètres une pop-up s'ouvre et propose au joueur de choisir entre plusieurs fonds d'écran ou un plateau de jeu « classique » et un plus « moderne ».



Figure 3 - Fenêtre des paramètres du jeu

- Se déconnecter via le menu « drop-down » accessible en cliquant sur l'icône du joueur ou accéder à l'aide



Figure 4 - Menu drop-down

La procédure est identique si l'utilisateur se connecte. Il devra auparavant saisir son email et mot de passe via la pop-up de login :

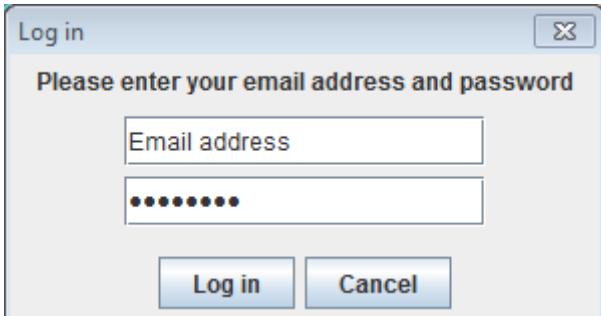


Figure 5 - Fenêtre pop-up Log In

Remarque : Les scenarios de **Log In** et de **Sign Up** ne sont actuellement pas fonctionnels.

Architecture de l'application

Le modèle MVC

Nous avons fait le choix de l'architecture MVC afin de rendre notre modèle complètement indépendant de l'affichage. Notre application est donc composée des éléments suivants :

Modèle :

Package *model* contenant une classe principale : *Play*

Contrôleur :

Package *controller* composé de deux contrôleurs : un pour la *GameView* et un pour la *MenuView*.

Vues :

Package *views* contenant une vue générale composée d'éléments java Swing, elle-même composée de deux vues *GameView* et *MenuView*.

Les fonctionnalités graphiques - Drag and drop

Nous avons mis en place dans un premier temps un système de *drag and drop* en implémentant notre propre **transferHandler**, se référant ainsi aux recommandations d'Oracle.

Pour se faire, nous allons décrire les éléments qui interviennent dans le processus :

Elements

Grid

JPanel composé de *JPanel* améliorés : *panelGrid* (Coordonnées propres dans la matrice)

Rack

JPanel composé de *JPanel* améliorés : *panelRack* (Position propre dans le rack)

DTPicture

JPanel contenant une image (celle du jeton) et implémentant deux interfaces *MouseListener* et *MouseMotionListener*. C'est cet élément qui fait office de jeton.

GlassPane

JPanel invisible sur lequel on va dessiner le déplacement de notre jeton. Cet élément est le plus haut d'un point de vue visuel par rapport à la frame. Sa coordonnée sur l'axe Z qui fait face à l'utilisateur est donc la plus élevée.

TileTransferHandler

Classe qui étend la classe *transferHandler* et redéfinit les méthodes d'import et d'export de cette dernière pour le drag and drop.

Déroulement du drag and drop

Lors du déplacement d'un jeton du chevalet vers la grille, l'image du *DTPicture* est redessinée sur le *GlassPane* qui devient alors visible (au sens logique car transparent) et suit les mouvements de souris jusqu'à sa destination finale. Le début de cette action commence lorsque l'utilisateur clique sur la souris, la déplace puis se termine lors du relâchement du bouton.

C'est à la fin de l'action que le mécanisme de drag and drop entre en jeu.

Le *DTPicture* ainsi que les *panelRack* et *panelGrid* ne sont en mesure d'effectuer le drag and drop uniquement après avoir défini le **transferHandler** à utiliser. Ceci s'effectue lors de la création de ces objets via la méthode **setTransferHandler**. Elle permet d'indiquer à la JVM, qui se chargera de l'opération, quels éléments seront impliqués.

Dans notre cas il s'agira de déplacer un *DTPicture* contenu dans un *panelRack* et de l'y ajouter dans un *panelGrid*.

Une fois que l'utilisateur lâche son bouton de souris, il suffit d'indiquer au *transferHandler* le composant déplacé à exporter. Ceci s'effectue via l'appel de la méthode **exportAsDrag(Component, event, action)**.

Entre alors en jeu les méthodes d'import et d'export que nous avions redéfinis au sein de notre *TileTransferHandler*. Les opérations effectuées sont les suivantes :

canImport : Vérifier si l'importation de l'élément est possible

importData : Importe l'information *transferable** de l'élément source au sein du composant cible.

* *L'image du DTPicture est alors transformé en élément dit transferable. Il aurait très bien pu s'agir de texte, si nous devions transférer du texte.*

exportDone : Supprimer l'élément de son conteneur source.

Remarque : Ces explications sont simplifiées pour ce rapport. Elles seront discutées plus précisément lors de la présentation.

Drag and drop et MVC

En migrant vers l'architecture MVC, nous avons du faire le choix de nous passer du mécanisme de drag and drop. En effet, lorsqu'une modification dans la vue est effectuée, celle-ci en informe son contrôleur qui se charge de mettre à jour le modèle. Ce n'est qu'une fois le modèle modifié que ce dernier informe la vue de son nouvel état afin qu'elle puisse également se mettre à jour. Toutefois, il devenait assez complexe de mettre en attente le mécanisme de drag and drop pendant le déroulement de ce processus.

Nous effectuons à présent les mêmes tâches qui ont été décrites pour le d'n'd, mais directement au sein de notre vue. C'est le *DTPicture* qui notifie de son changement de position et envoie ses nouvelles coordonnées au contrôleur, ...