Dossier de validation - IHM

Brahim Berkati

June 10, 2014

1 Introduction

Ce document contient l'ensemble des informations concernant l'interface et la validation de cette dernière. Le document est divisé en quatres parties.

Dans un premier temps, nous explicitons certains concepts requis pour comprendre le fonctionnement technique de la LibGDX. Ensuite, nous justifierons les choix effectués pour améliorer l'ergonomie de l'application. La troisième partie couvre les procédures de validation utilisées en interne pour garantir la stabilité de l'interface. Enfin, nous terminons par décrire les méthodes de validations par des utilisateurs externes au projet.

2 Structure de l'interface

L'interface graphique est complètement géré par une librairie externe appelée LibGDX. Cette libraire affiche ce qu'on appelle un Screen qui contient toutes les informations de la scène. Un seul Screen est actif simultanément. C'est à travers l'utilisation de la méthode setScreen(Screen screen); que les changements sont effectués.

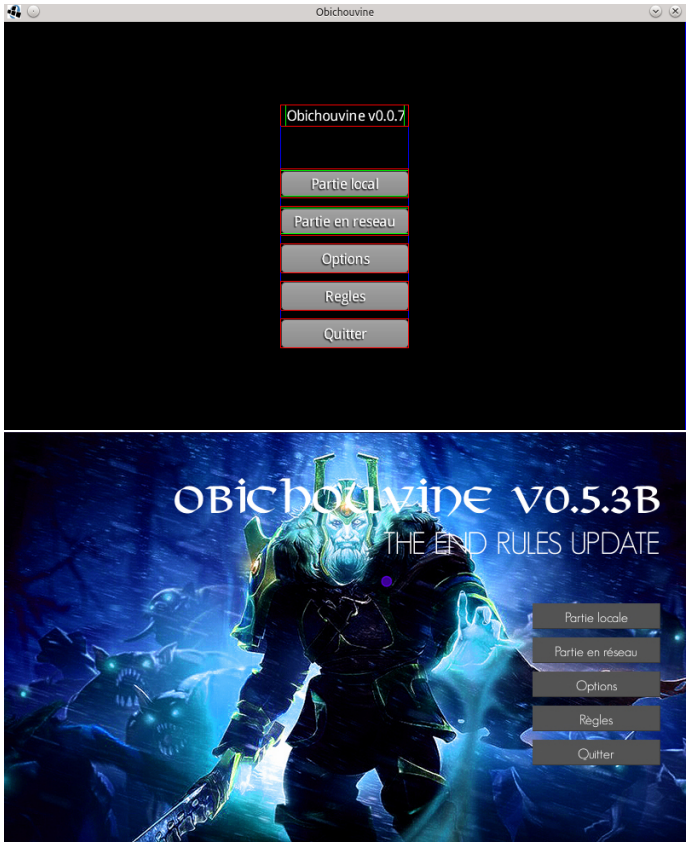
Sur une scène, il est possible d'afficher des sprites ainsi que divers widgets. Tous les widgets chargent une apparence en fonction de ce qu'on appelle un Skin. Le Skin contient toutes les informations en ce qui concerne l'ensemble des widgets. Cela permet d'uniformiser l'apparence de ces derniers.

L'ensemble des scènes comprennent donc un ensemble de widgets. Ces informations sont donc redondantes. Nous avons donc décidé de créer une abstraction d'une scène contenant toutes ces informations redondantes.

3 Prototypage

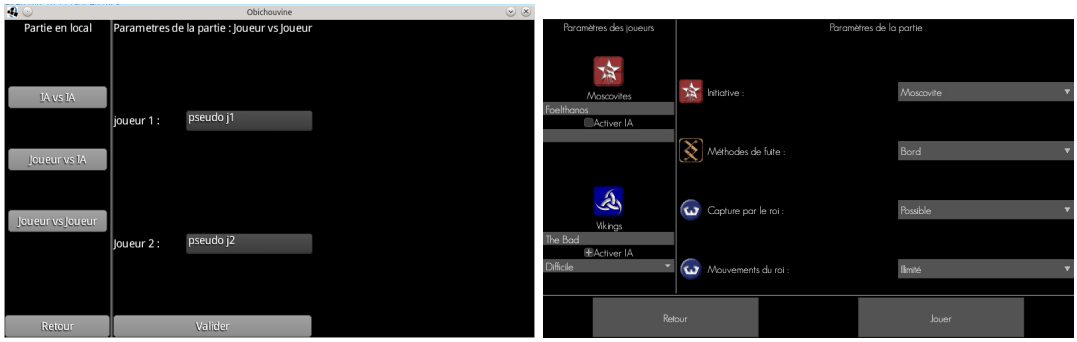
L'interface du jeu doit allié ergonomie et attractivité visuelle. Le meilleur moyen d'obtenir une ergonomie suffisante est de s'inspirer d'autres interfaces de jeux de plateau déjà existant. Ainsi, nous avons testé plusieurs jeux de Tablut en ligne, mais nous avons également recherché

l'inspiration du côté des jeux d'échecs ainsi que du jeu "42 jeux indémodables" sur Nintendo DS.



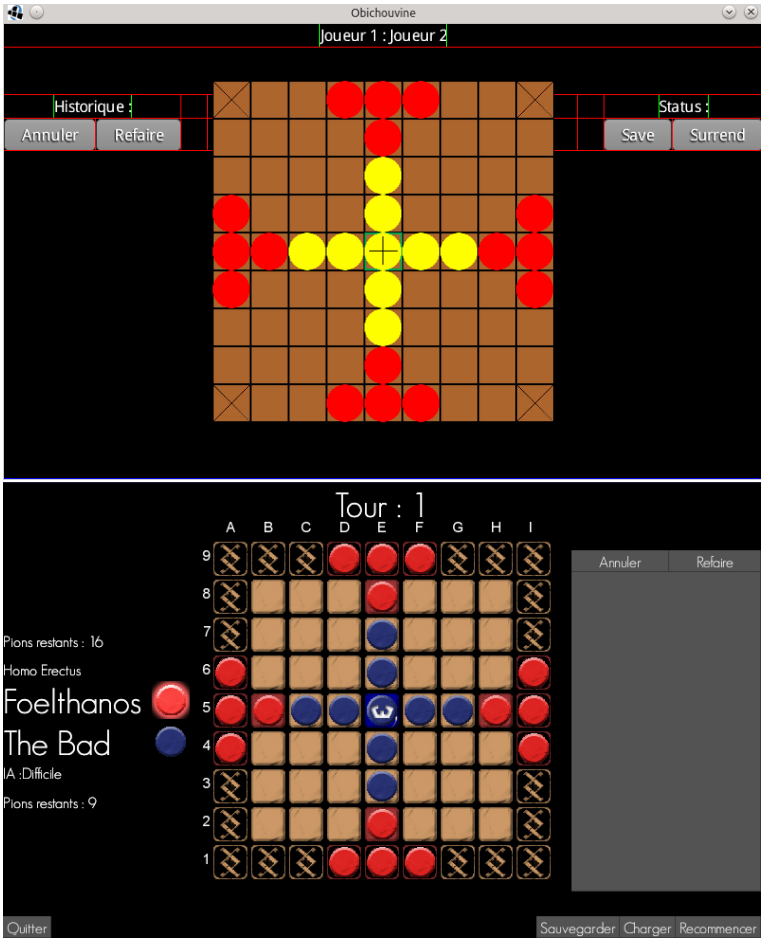
Voici une comparaison entre la première et la dernière version du menu principal. Les améliorations ne sont que visuelle. De manière analogue, le menu "Options" a subi les mêmes mutations. A noté cependant, les cadres rouges et bleu autour des widgets ne font pas partie de l'interface. Ce sont des aides visuelles en mode "développeur" pour placer plus facilement les widgets sur les écrans.

L'écran de lancement de partie locale a, par contre, été très lourdement modifié, comme en témoigne cette image :



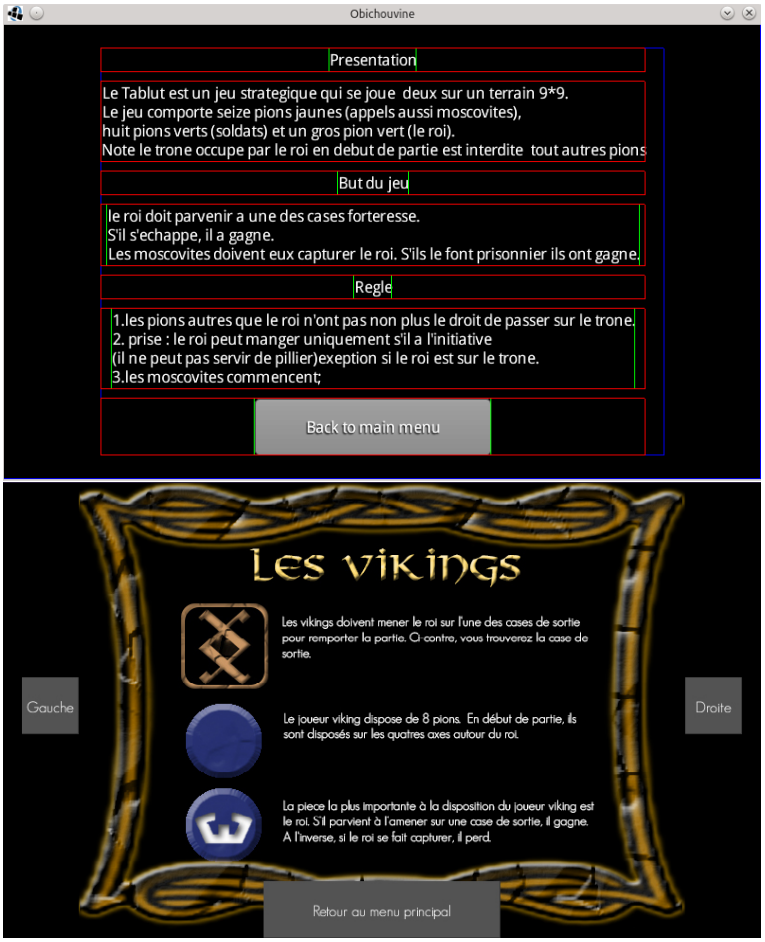
Ces modifications ont été inspirés par les différentes audits ainsi que par les rencontres avec notre tuteur.

L'écran de jeu a également fortement évolué :



Pour cet écran, ce n'est pas seulement des modifications visuelle mais surtout des modifications de confort de jeu. Surbrillance, changement de couleur, affichage des informations utiles, etc.

Un autre écran ayant subi de profonds changements est l'écran de règles :



La dernière version est directement inspiré des jeux flash en ligne et des jeux sur plateforme mobile. En effet, ce genre de jeu dispose d'une explication des règles très visuelle permettant de faire un lien avec l'écran de jeu.

4 Validation interne

La validation interne a pour objectif de vérifier le bon fonctionnement technique de l'interface.

4.1 Procédure

La procédure est très simple pour les widgets au comportement simple (boutons, label, ...). Une fois le widget intégré, on lance l'application et on teste le widget. Si le comportement effectué par le widget est celui attendu, c'est validé.

Par contre, pour les widgets complexes, il peut y avoir plusieurs paramètres différents possibles pour une plage de résultat très varié. La procédure est donc plus stricte dans ces cas. Ces widgets complexes sont bien souvent. Voici la liste des widgets complexes utilisé pour ce programme :

- OptionPane
- PlayerSelection
- GameStatusWidget
- HistoryWidget

Pour les valider, des jeux de tests sont mis en place. Toutes les possibilités doivent être couvertes.

4.2 Bilan

4.2.1 OptionPane

Le widget OptionPane est une interface permettant à l'utilisateur de générer un ensemble de paramètres. Ces paramètres modélisent les différentes variantes du jeu. Le widget récupère le Skin en entrée de manière à uniformiser l'apparence des widgets utilisés dans cette classe. L'OptionPane possède deux méthodes : printCommonContent() et generateParameter().

Le premier ne reçoit aucun paramètre et ne retourne rien, mais affiche les différents widgets que composent l'OptionPane. Pour vérifier le bon fonctionnement de cette méthode, on vérifie visuellement l'affichage de tous les widgets prévus. C'est le cas.

L'un de ces widgets possèdent des écouteurs. Il faut donc également vérifier le bon fonctionnement du comportement voulu. Le widget en question est un menu déroulant. Il doit modifier l'icône correspondant à sa ligne du tableau. La vérification est encore une fois visuelle. Cela est validé.

La méthode generateParameter() est censé récupérer les données des widgets affichés et les formater pour permettre l'utilisation en cours de jeu. Il doit donc renvoyer un Parameter (classe créer dans ce but composé d'énumérations).

La vérification se fait via l'entrée de chaque combinaison de données possibles et la vérification de l'exactitude de la sortie. Chaque combinaison a renvoyé le bon résultat.

4.2.2 PlayerSelection

Le widget PlayerSelection est une interface permettant de générer une instance de données représentant un joueur, qu'il soit humain ou non. De manière analogue à OptionPane, le widget récupère un Skin en entrée et possède deux méthodes, l'une pour afficher les widgets l'autre pour formater la réponse globale de l'instance de PlayerSelection.

La vérification suit donc exactement le même protocole. Tous les widgets prévues sont affichés. De même pour l'écouteur du widget de case à cocher. Le comportement attendu est de rafraîchir le modèle de l'instance de PlayerSelection. Une subtilité existe tout de même. Lorsque la checkbox est cochée, il faut impérativement que la liste déroulante du widget soit désactivé. La vérification confirme ce fait.

A l'instar de l'OptionPane, la méthode de récupération est vérifiée par un simple affichage dans la console. Le widget PlayerSelection est également validé.

4.2.3 GameStatusWidget

Ce widget agit légèrement différemment des autres widgets. Au lieu d'être composé de widgets disposant d'écouteur permettant la modification de son modèle, il reçoit des événements extérieurs. La validation de l'affichage est donc semblable mais pas celle des écouteurs car ils n'existent pas. Certaines méthodes sont appelées par la classe appelante, ce qui induit une modification de certains de ses attributs et rafraîchit les widgets concernés.

Heureusement pour les tests, il n'y a qu'une seule méthode à valider, switchTurn(). Elle doit inverser la valeur d'une énumération (soit moscovites, soit vikings). Le widget est validé.

4.2.4 HistoryWidget

Le HistoryWidget est composé d'écouteurs permettant la modification de son modèle mais il peut également recevoir des événements extérieurs. Lorsqu'un mouvement est effectué par le joueur ou l'IA, on appelle la méthode d'ajout. Ainsi, le widget modifie son modèle. La validation de la modification se fait de manière empirique. On joue un coup et ce coup doit apparaître dans l'historique. A chaque coup successif, l'historique ajoute le coup en haut de la liste.

Les comportements des boutons sont plus subtils. Le premier doit remettre l'état du plateau de jeu à son état précédent le dernier coup joué. Les cas possibles sont un déplacement simple ou un déplacement induisant une capture. Le premier cas est simple mais le second doit sauvegarder l'information de la capture. Il y eu un bogue causé par la non-prise en compte de ce cas.

Le second bouton est plus simple. Si nous avons annuler un coup, une autre liste sauvegarde ce coup au cas où l'annulation était mal venu. Si nous jouons un coup, la liste doit être vidé. Comportement simple, validation simple.

5 Validation externe

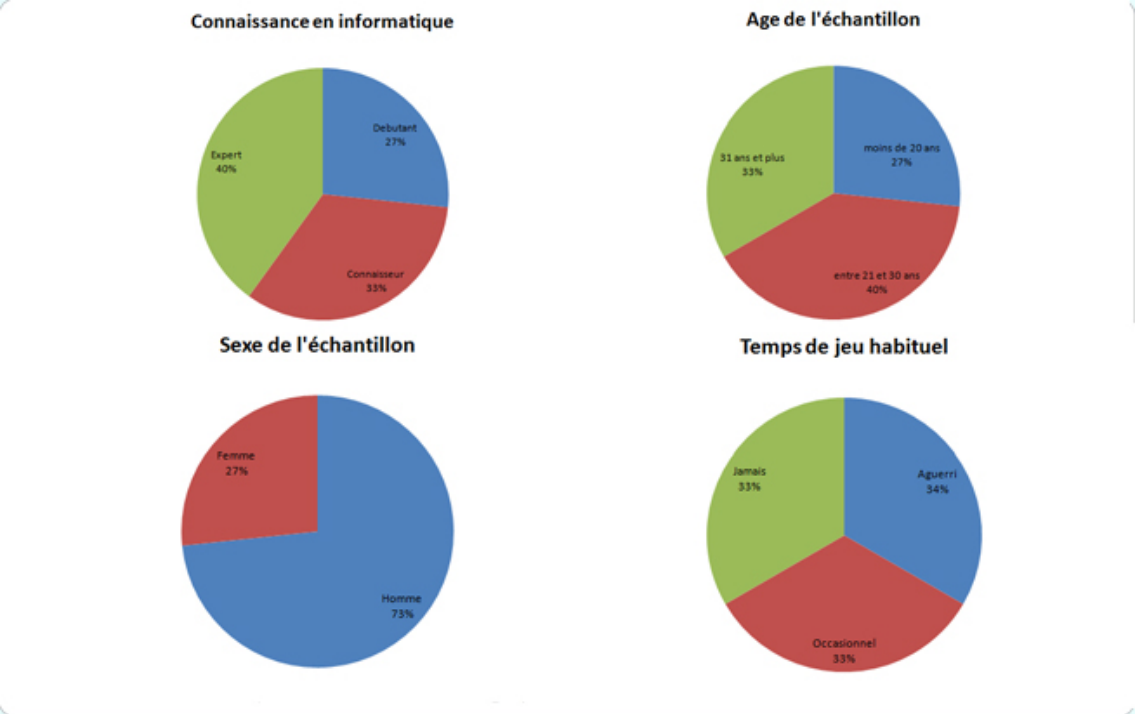
La validation externe a pour objectif d'améliorer l'expérience de l'utilisateur en recueillant les idées et critiques de personnes externes au projet.

5.1 Panel d'utilisateurs

La validation dite "externe" est divisé en deux catégories : la validation durant le développement et la validation en bêta-test.

Durant le développement, quatres personnes ont permis les améliorations successives de l'ergonomie de l'interface : le responsable de l'audit IHM, notre tuteur ainsi que deux personnes extérieures au projet. Cette validation est nécessaire pour corriger rapidement les erreurs de conception trop graves très vite durant le développement.

La bêta-test a été effectué le week-end du samedi 7 juin à 10h au dimanche 8 juin à 18h. 15 personnes ont participé à ce week-end de bêta-test. Voici les principales informations sur l'échantillon :



Parmi les "experts" en informatique, une personne est développeur professionnel dans le domaine du jeu vidéo tandis qu'une autre personne étudie actuellement dans ce sens. Cet échantillon est relativement équilibré et permet de s'assurer d'une ergonomie efficace pour le public le plus large possible.

Ce panel varié nous permettra de recueillir des informations variées. Nous pourrons ainsi utiliser ces informations afin d'optimiser l'ergonomie du jeu pour le public le plus large possible.

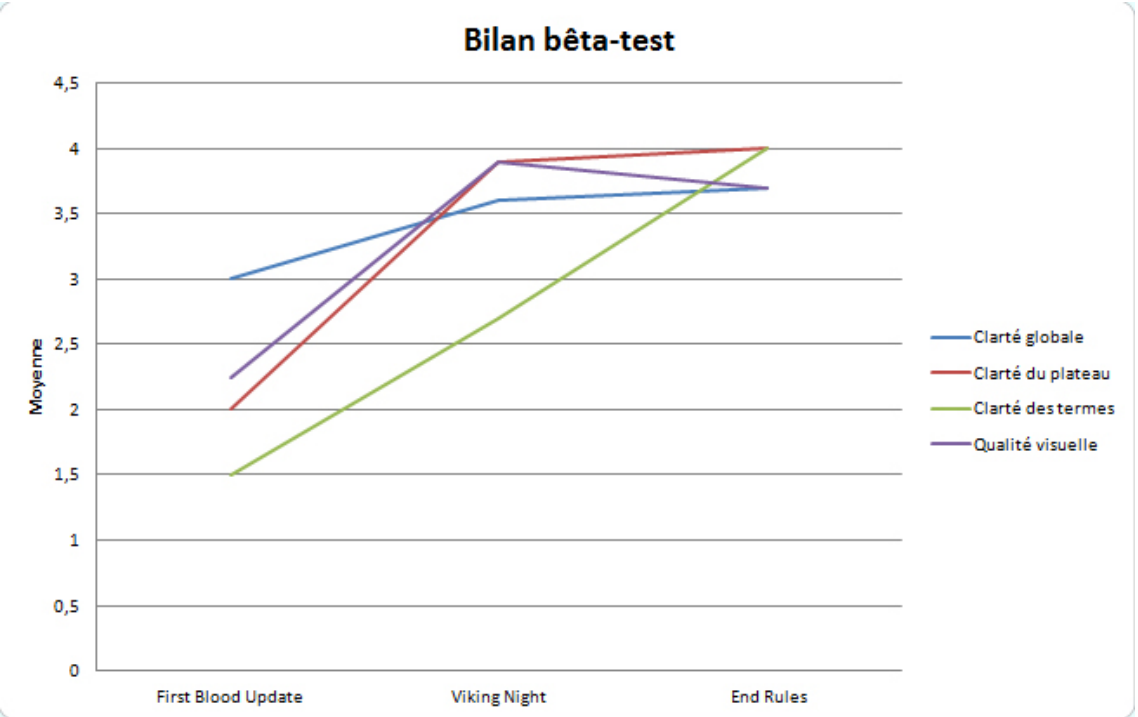
5.2 Déroulement de l'évaluation

Durant le développement, l'évaluation est surtout faite via des retours directs. Lors de discussions sur Skype ou via des audits/rencontres avec le tuteurs. Lors des audits, nous faisons tourner le jeu et le professeur émet ses interrogations et avis durant la présentation.

En ce qui concerne la bêta-test, nous envoyons la première version bêta par mail avec des directives et plusieurs documents. Les documents en question sont un sondage sur l'ergonomie du jeu et un tableau de rapport de bogues. Nous demandons à tous les participants d'être le plus honnêtes possibles lors du remplissage de ces documents.

Le sondage est principalement orienté sur la qualité de l'interface mais quelques questions ont été réservés pour l'intelligence artificielle.

5.3 Bilan



Ce graphique rassemble les moyennes des notes attribuées par les testeurs pour chacun des principaux patchs de la phase de bêta-test.

On remarque les faibles notes pour le premier patch, surtout pour la clarté des termes. L'échantillon était de seulement quatres personnes, ce n'est donc pas très représentatif, mais cela traduisait un grand manque d'explication des règles du jeu. On remarque également des notes mauvais en ce qui concerne la clarté des informations de l'écran de jeu ainsi que de la qualité visuelle du logiciel.

La mise à jour suivante, nommée "Viking Night Update", à grandement améliorer la qualité visuelle ainsi que la clarté du plateau de jeu via une mise à jour de tous les sprites du jeu. De plus, la surbrillance a été amélioré. Un meilleur retour sur les actions de l'IA ont également été ajouté à travers une image indiquant que l'IA travaille. Ces efforts ont été récompensé par une amélioration nettes des notes. Cependant, cela restait très faible en ce qui concerne la clarté des explications.

La dernière mise à jour du week-end a donc tenter d'améliorer l'explication des règles du jeu à travers un menu de règles plus précis et plus propre. L'utilisation des sprites du plateau

de jeu permet un rappel visuel. Ce nouveau menu a été récompensé par une augmentation significative des notes en ce qui concerne l'explication des règles du jeu.

En conclusion, la bêta-test a été concluante pour trouver les failles de l'interface, mais il reste encore du chemin.