HomePlans - Rapport
INFO-F-307

Titouan Christophe
Pierre Gérard
Florentin Hennecker
Walter Moulart
Bruno Rocha Pereira
Julian Schembri

18 novembre 2014

Table des matières

Chapitre 1

Itération 1

1.1 Introduction

Nous avons décidé de développer l'histoire 1 proposée par le client lors de cette phase. Ce choix a été fait dans le but de fournir le plus vite possible un socle pour les fonctionnalités à suivre, et se présentait logiquement comme le seul choix possible.

Vous pourrez lire dans la suite de ce chapitre la décomposition de cette histoire en top-level tasks, ainsi que la description des choix qui ont été faits quant aux librairies que l'équipe a décidé d'utiliser.

Les PV des réunions officielles qui ont eu lieu jusqu'ici peuvent être trouvés en annexe.

1.2 Librairies introduites

1.2.1 Enregistrement du projet

Utilisation de SQLite pour enregistrer les projets, à l'aide de ORMlite, et ses dépendances (ormlite-jdbc, ormlite-core, sqlite-jdbc)

- sqlite a les avantages d'une DB relationnelle, mais enregistre dans des fichiers : facilité de déplacement des projets, pas besoin de serveur
- sqlite permet d'avoir une DB en mémoire vive : pratique pour les tests
- sqlite est utilisable dans de nombreux autres langages : augmente la productivité en permettant d'éditer les fichiers de projets dans le langage préféré de chaque dev
- ORM facilite l'accès aux données (Design pattern DataMapper http ://martinfowler.com/eaaCatalog/dataMapper.html)
- ORMLite permet d'utiliser plusieurs types de bases de données différentes à travers JDBC. Possibilité donc d'utiliser autre chose qu'SQLite si nécessaire.

1.2.2 Librairie GUI

Les requirements de l'interface graphique sont les suivants

- Afficher une fenêtre esthétique
- Mettre des menus / boutons
- Incorporer une vue 2D/3D
- Compatible avec la lib 3D choisie

Les différentes possibilités de GUI sont :
- AWT (Abstract Window Toolkit)
- Swing
- SWT (Standard Widget Toolkit)
- JavaFX
- Apache Pivot
- Qt Jambi

Swing

Nous avons retenu **Swing**.
- Customisable : Oui
- Léger : Non
- Cross-Platform : Oui
- Documentation : Complet
- Faiblement couplé
- Suit le design pattern MVC
- Compatible avec jMonkeyEngine

Toutes ces raisons font qu'on utilisera Swing comme librairie pour construire notre interface graphique. Voici un aperçu des autres libraires éligibles, et les raisons pour lesquelles elles ont été refusées.

AWT

- Customisable : Non
- Léger : Oui
- Cross-Platform : Oui
- Documentation : Complet
- AWT utilise les objets du système
- Swing hérite des objets de AWT

On n'utilisera pas AWT car il n'est pas assez customisable.

SWT

- Customisable : Non
- Léger : Oui
- Cross-Platform : Oui
- Documentation : Très complet
- Développé par l'équipe d'Eclipse
- Se place un peu comme premier concurrent de Swing

On n'utilisera pas SWT car il n'est pas assez customisable.

Java FX

On n'utilisera pas Java FX car ils se sont spécialisés dans les interfaces d'application web et ce n'est pas forcément utile ici.

Apache Pivot

On n'utilisera pas Apache Pivot pour les mêmes raisons que Java FX.

Qt Jambi

- Customisable : Oui
- Léger : Non
- Cross-Platform : Oui mais limité à QT4.6
- Documentation : Mauvaise
- Intégré à Eclipse
- Assez complet
- Accès a une database sql incluse

Inconvénients :

- Pas de doc sur le site QtJambi
- Compliqué pour l'intégration de la 3D . Il existe Qt3D mais on s'est dirigé vers Jmonkey
- Bloqué a la version 4.6

1.2.3 Librairie 3D

Nous choisissons d'utiliser jMonkeyEngine comme librairie 3D pour ces raisons :

- Elle est sous license BSD
- Une des interfaces les plus high level
- Community driven
- Développement encore actif
- Documentation extensive
- Engine complet
- Refonte complête de JME2 pour le mieux
- Intégration facile dans un GUI swing : http ://hub.jmonkeyengine.org/wiki/doku.php/jme3 :advanced :swing_canvas

Justification de l'écartement d'autres librairies :

JOGL :

- Utilise SWT comme système de fenêtrage

GL4Java :

- Vieux et obsolète

Java3D :

- Abandonné

Ardor3D :

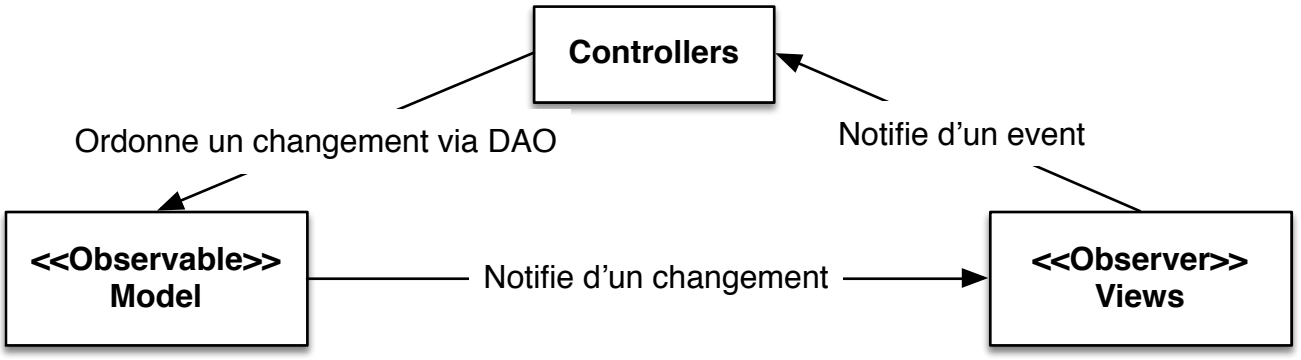
- Prise en main difficile et risque de requérir beaucoup plus de temps

1.3 Rapport de fin d'itération

Le développement de l'itération 1 s'est bien déroulé. La charge de travail a été correctement répartie le long des 3 semaines de développement, tout le monde a pris part de manière équilibrée au travail et on ne déplore aucun conflit.

1.3.1 Architecture

Architecture générale



On voit directement l'utilisation du MVC, couplé avec un design pattern Observable et un DAO. Cette architecture nous permet de :

- rajouter/supprimer une vue (+ contrôleur) très facilement
- garder toutes les vues à jour
- découpler un maximum toutes les parties de l'application
- garder l'état du projet entre deux sessions d'utilisation

Prenons un exemple : le passage d'une vue 3D à une vue 2D et inversement. Ce passage a des implications dans plusieurs vues. La première est bien évidemment l'éditeur général, qui doit afficher le monde différemment en fonction du mode ; la deuxième est la toolsbar, qui permet de changer de mode.

Quand on clique sur le bouton 2D/3D, la ToolsBarView notifie son contrôleur d'un event reçu. Le contrôleur utilise alors le DAO pour dire au modèle que le mode a été changé. Le modèle a en effet une valeur de configuration qui stocke le mode. Ce dernier notifie alors toutes les vues concernées du changement (via le pattern Observer), dont l'éditeur principal qui va alors changer son mode d'affichage.

En plus de cela, comme le modèle dispose de la sauvegarde automatique, si l'utilisateur quitte ce projet en mode 3D, il le retrouvera en mode 3D, s'il le quitte en mode 2D, il le retrouvera en mode 2D.

1.3.2 Bonnes pratiques utilisées

Plusieurs pratiques ont été mises en place pour faciliter le développement en groupe, comme par exemple le pair programming ou les sprints d'une journée.

Pair programming

À plusieurs moments, nous avons travaillé par deux sur un même ordinateur pour les problèmes plus épineux. Un développeur écrivait du code, et l'autre essayait de le corriger et de penser aux edge cases en même temps. Cela nous a permis de partager beaucoup plus facilement des idées et de compléter des features complexes de manière rapide et robuste.

Sprints d'une journée

En déployant un outil de statistiques de notre repository Git, on voit que le samedi est le jour de la semaine où l'équipe commite le plus. La raison de ce pic est simple : à deux reprises lors de cette itération, nous nous sommes retrouvés ensemble physiquement, autour d'une grande table, pour travailler sur le projet.

Au début de la matinée, nous faisions une réunion pour établir les objectifs de la journée puis nous nous lancions tous sur la production de features. C'est lors de ces sprints que l'avancement était le plus marqué et que nous pouvions prendre du recul facilement sur ce qui avait déjà été fait, et ce qu'il restait à faire.

Développent itératif/fractal

Cette pratique a été adoptée dès le début, mais nous nous sommes rendus compte vers la moitié de l'itération que nous l'utilisions mal. Le principe de cette technique est de considérer chaque feature comme une unité qu'on peut développer à plusieurs niveaux de *perfection*.

Nous nous sommes forcés de produire très vite des proofs of concept, ou Minimum Viable Products pour les unités à développer. Cette unité était alors souvent à un stade de fonctionnalité très pauvre, dont l'architecture n'était pas forcément bien pensée. Il fallait alors réécrire, rajouter ou déplacer du code pour améliorer le point de vue utilité autant que le point de vue de beauté interne du code, afin d'arriver à une unité fonctionnelle complète et qui s'intégrait bien dans l'architecture de l'application.

Toutefois, lors de la première moitié de la première itération, nous sommes souvent tombés dans le piège du "*perfectionnement d'abord*", en passant beaucoup de temps sur le perfectionnement utilitaire de l'unité. Nous aurions dû passer plus vite - une fois que l'architecture d'une unité était bien intégrée dans l'application - au développement des autres unités.

Test-Driven Development

Cette technique, qu'il ne faut plus expliquer, a été utilisée, surtout pour le développement du modèle et s'est effectivement avérée positive lorsqu'il a fallu se baser sur un modèle robuste.

Staging area

Nous avons créé une branche `stage` sur laquelle nous testions la version de développement la plus récente de l'application. De manière régulière, nous passions tous les changements de `stage` en production sur `master`. La condition de mise en production était simple : tout le code doit être documenté, et tout ce qui peut être testé doit être testé.

Couverture de la qualité du code

Des outils d'évaluation de la qualité du code ont été utilisés. Parmi eux, on peut en noter trois :

EclEMMA Cet outil nous a permis d'évaluer facilement la couverture des tests dans l'application et nous permet d'estimer visuellement très rapidement cette dernière.

Missing Javadoc (Eclipse) On peut configurer Eclipse pour afficher des warnings là où le code n'est pas documenté, ce qui est très pratique.

PMD Plusieurs d'entre nous ont installé PMD en fin d'itération pour évaluer le respect des conventions de code. Nous n'avons cependant pas pris le temps de "réparer" les quelques erreurs mises en évidence.

1.3.3 Réflexion sur les librairies choisies

Généralement, nous sommes satisfaits de nos choix.

jMonkeyEngine

Même si elle est plutôt facile à utiliser, nous nous sommes rendus compte que la documentation et les ressources disponibles pouvaient parfois être rares. Une grande partie de ce que nous avons trouvé venait directement du site officiel de jMonkeyEngine qui, en soi, est très complet, mais qui est parfois lacunaire sur certains sujets.

En prenant un peu de recul, nous nous rendons compte que nous utilisons des fonctionnalités d'assez bas niveau et que nous aurions pu nous satisfaire de moins.

Swing

Swing reste un excellent choix, nous n'avons pas à nous plaindre.

Annexe A

PV des réunions

A.1 Réunion Kick-off

22 octobre 2014

Présents : Walter, Pierre, Bruno, Titouan, Julian, Florentin

Ordre du jour

- Partage d'expérience
- Itération 1 : objectifs & planning
- Répartition des tâches
- Réunions hebdomadaires
- Conventions

Partage d'expérience

- Communication et honnêteté super importantes
- Savoir tout justifier dans son code
- Toucher à un maximum de parties du code, au moins connaître leur architecture
- Tester un maximum (TDD pour ce projet en particulier)
- Définir les objectifs de chacun clairement
- Coder en groupe un maximum, physiquement, pourquoi pas au hackerspace

Itération 1 : objectifs & planning

On va commencer par l'histoire no 1 pour avoir une "vraie" base. Les top-level tasks seront fixées plus tard mais on peut déjà discerner les suivantes :

- Enregistrement d'un projet
- GUI
- Affichage du monde 2D/3D
- Navigation dans le monde 2D/3D
- Modification de la géométrie de la pièce
- Création d'un default/demo project

Répartition des tâches

Pour l'échéance **Gestion de Projet 1**, il faudra fixer, développer et assigner les top-level tasks. Il faudra aussi se renseigner sur les librairires à utiliser.

- Librairie 3D : Bruno et Florentin
- Librairie GUI : Julian
- Enregistrement de fichiers 3D : Titou

Réunions hebdomadaires

Il a été décidé, outre le fait de se retrouver un maximum pour travailler ensemble, de se retrouver de manière hebdomadaire obligatoirement, pour tout mettre au point, les jeudis à 11h au hackerspace.

Conventions

Le groupe a accepté à l'unanimité l'utilisation de plusieurs conventions :

- Java 1.6
- camelCase, _ au début de variables/fonctions privées, majuscule au début des noms de classes
- Petits commits (quelques dizaines de lignes maximum)
- 1 feature = 1 branche
- Commentaires de commit commençant par "[fix]", "[enh]" (enhance), "[add]", "[rem]" (remove), "[ref]" (refactoring), "[test]"
- Utilisation des tags git
- Tout ce qui est pushé sur le remote doit être documenté et testé
- Utilisation de jUnit et de Javadoc
- Utilisation massive (tant que possible) des exceptions et des assertions
- Utilisation de DIA pour les diagrammes UML
- Le choix d'une lib se fait à au moins 3 personnes du groupe, et est motivé et expliqué
- Pré-échéances 1h avant l'échéance officielle. Plus rien n'est rajouté après cette pré-échéance et on teste le fonctionnement du produit

A.2 Réunion hebdomadaire 29/10

29 octobre 2014

Présents : Pierre, Julian, Walter, Titouan, Florentin

Ordre du jour

- Update client
- Mise au point sur l'avancement
- Merge du progrès
- Extensions des fichiers

Update client

On peut partir sur l'histoire 1, il faut juste réécrire les histoires dans le bon format dans leur fichier propre.

Mise au point sur l'avancement

- Julian : Menu bar qui fonctionne (testée sur mac et linux). Est en train de binder les fonctions triggerées par la GUI
- Pierre : A fait la toolbar, a déjà des fonctions bindées pour tous les boutons. Il a aussi fait un split pane fonctionnel.
- Florentin et Bruno : On a un algo qui crée les faces pour les murs à partir des points de leurs bases et d'une hauteur, et le sol aussi. On affiche cet objet dans une vue jMonkey. 1.3 n'est pas loin d'être terminée.
- Titouan : a déjà une base de modèle avec ORMLite, qui gère aussi l'enregistrement de projets. A travaillé sur la compilation avec ant.

Merge du progrès

On va merger un maximum de branches pour avoir un MVP et pour commencer à travailler sur l'architecture.

Extensions des fichiers

Le format des fichiers crées par le logiciel sera : **hpj**

A.3 Réunion hebdomadaire 05/11

5 novembre 2014

Présents : Walter, Pierre, Bruno, Titouan, Julian, Florentin

Ordre du jour

- Planning
- Avancement
- Couverture des tests et de la doc
- Architecture
- Organisation sprint
- Branch cleanup

Planning

Rappel : il reste deux semaines jour pour jour avant la remise du code de la première itération. Il reste un peu moins de deux semaines pour la remise de la partie Gestion. On parlera de ça la semaine prochaine.

Avancement

- La vue 3D est intégrée à la vue 2D, et le resize des fenêtres se fait proprement
- Le modèle n'est pas loin d'être terminé
- On arrive à afficher un objet du modèle de Titouan dans la vue 3D
- La vue 2D avance bien

Couverture des tests et de la doc

Généralement, la documentation est assez présente, mais il reste quelques modules à faire. Ce n'est pas un problème tant qu'ils ne sont pas dans la version de production. Il faut encore voir comment on peut tester la GUI.

Architecture

Pour le moment, on a surtout plusieurs proofs of concept presque indépendantes, et des proofs of concept que les proofs of concept fonctionnent ensemble. On décide, pour ce soir, de merger toutes les fonctionnalités en cours de développement dans pre-master. Demain, on procédera à un premier refactoring pour avoir une bonne architecture et passer tous les changements en production.

Organisation sprint

L'équipe prévoit de se retrouver pour travailler ensemble sur un sprint samedi, au hackerspace.

Branch cleanup

Nous avons procédé à la suppression de quelques branches mergées sur le remote. On décide d'utiliser la branche pre-master comme staging area.

A.4 Réunion hebdomadaire 13/11

13 Novembre 2014

Présents : Titouan, Bruno, Pierre, Florentin

Ordre du jour

- Avancement
- Choix de l'histoire pour l'itération 2
- Discussion sur certains aspects à implémenter

Avancement

- Rajout de boutons pour les outils de sélection, déplacement,...
- Sélection des objets dans la vue 3D et l'arbre des objets
- Lien de tout le modèle avec la vue
- Observation des vues sur le modèle

Choix de l'histoire pour l'itération 2

Nous avons décidé d'implémenter l'histoire 2 pour l'itération 2, pour les raisons suivantes : * Plusieurs des histoires 3 à 6 ont comme prérequis l'histoire 2 * C'est la suite logique de ce que nous avons fait jusqu'ici * Nous aimerions avoir un produit complet assez vite, et y rajouter des fonctionnalités plus "optionnelles" par la suite.

Discussion sur certains aspects à implémenter

Tous ces aspects devraient encore être intégrés dans le code pour la première itération et il est largement temps, pour certains d'entre eux, de s'y mettre.

Barre d'outils, key bindings Les outils de sélection et déplacement ne font encore rien, on peut sélectionner des objets hors du mode de sélection. Il faudrait que ça fonctionne et que les commandes soient unifiées. Bruno veut bien s'en occuper.

Vue isométrique 2D En vue 2D, il faut que la caméra soit isométrique, pour ne pas avoir de perspective au niveau des murs. Ca fausse notre perception des pièces. Walter et Bruno le feront.

Epaisseur des murs Quand on est pile à la verticale d'un mur, on ne le voit pas. Il faut régler ça. Florentin veut bien le faire.

Modification d'un mur On a construit un use case ensemble pour savoir comment ça devrait se faire, et Titou a déjà commencé à s'en occuper.

Création d'un mur On décide de reporter cette tâche de quelques jours. Ca serait bien de l'avoir dans la première itération.

Etages Il faut pouvoir grouper, créer, supprimer, naviguer dans les étages assez vite. Ca ne devrait pas être trop difficile avec les Groupes du modèle.