IUT Sénart-Fontainebleau, UPEC

Luc Hernandez
Florent Madelaine

IUT Info 2A

2020-21

Projet IHM (FA)

Ce projet concerne le cours d'ACDA 3.1. On y retrouve les ingrédients entrevus en TP : une interface homme-machine à coder en java, et une base de donnée à mettre en oeuvre pour assurer la persistance des données.

Il faudra que votre travail s'organise selon le modèle MVC. Par ailleurs, la communication avec le modèle doit passer par une API [1] qui est fournie. Une implantation non persistante du modèle est également fournie.

Le travail se fait à 2. Les deux étudiants doivent être dans le même groupe de TP.

Date limite de rendu du rapport et de la réalisation. 7 février, minuit, heure française.

Contexte. Il s'agit d'un prototype de jeu simpliste inspiré de grands classiques comme Zork. Un joueur se déplace dans des pièces, reliées entre elles par des passages. Les pièces peuvent contenir des objets que le joueur peut transporter et déposer. Le joueur peut agir sur les passages et sur lui-même avec des objets (le mécanisme d'action reste très simpliste).

On peut imaginer qu'il s'agit d'un vieux jeu en mode textuel pour lequel on dispose d'une API.

Prototype d'IHM. *Une autre équipe* se charge de moderniser ce jeu sans changer le modèle, et de dépasser le stade de l'interface textuelle, avec une interface graphique en fausse 3D et une minimap.

Votre rôle sera de proposer une interface graphique permettant de concevoir un niveau du jeu. Il faudra ainsi pouvoir gérer pour ce niveau, les pièces et les passages, la position du joueur et les objets (les trucs). Notez que le joueur a un sac à dos qui peut contenir des trucs. Par gérer, on veut dire : création, suppression et mise à jour.

On suggère d'avoir un écran principal montrant un *dessin du niveau en cours d'élaboration*, sous forme d'une carte 2D vu de dessus.

On suggère d'ajouter des modalités pour permettre de *manipuler des listes* de certains éléments et ce le plus possible en lien avec la carte.

1. *Application Programing Interface* : ici une collection d'interfaces java et quelques objets concrets qui normalise la communication entre deux parties d'un logiciel.

Par exemple, on souhaite pouvoir accéder à une liste des trucs et lorsqu'on clique sur un truc pouvoir voir la pièce contenant l'objet (en changeant l'apparence de la pièce sur la carte 2D). Il faudrait aussi pouvoir sélectionner plusieurs trucs dans la liste et les déplacer dans une pièce ou encore les supprimer.

Un autre exemple concerne les passages qui peuvent être fermés à clé, fermés ou ouverts : il convient de pouvoir visualiser ces états facilement et de pouvoir faire des changements facilement.

Pour les diverses manipulation : on suggère de prévoir une première version « cliquodrome » utilisant des modalités de l'interface (bouton, case à cocher, etc) mais aussi une version avancée pour expert laissant une plus grande part à l'utilisation du clavier et de raccourcis.

Dans le modèle proposé, une pièce correspond implicitement à un volume cubique avec jusqu'à 4 passages pour les 4 directions cardinales. On pourra supposer pour simplifier que toutes les pièces sont de même taille.

Il n'y a toutefois pas de notion de position pour les pièces, et rien n'oblige les passages à être des couloirs rectilignes orientés nord-sud ou est-ouest, mais vous pouvez pour simplifier faire cette hypothèse.

Modèle persistant des données. Cette partie est obligatoire.

Vous devez mettre en oeuvre une version java + BdD adaptée du modèle implémentant l'API mais qui permet la persistance des données en particulier pour sauver le niveau en construction, et le charger pour recommencer ce travail d'édition.

Évolution de l'API pour stocker le dessin. Cette partie est optionnelle.

Le mécanisme des passages et pièces permet de modéliser des connexions complexes (passages avec des angles, passages qui se croisent comme dans un métro etc) en particulier le graphe dont les sommets sont des pièces et dont les arêtes sont les passages n'est pas forcément planaire. Vous pouvez travailler dans un contexte plus complexe si vous le souhaitez.

Il serait dommage de perdre le dessin réalisé par le concepteur du niveau lorsqu'on fait une sauvegarde en base puis un rechargement via l'API.

L'API propose donc une extension avec des nouvelles méthodes dans Piece et Passage pour permettre de gérer cet aspect.

Notez que le modèle non persistent donné comme exemple ne gère pour l'instant pas cet aspect et n'implémente que l'ancienne API.

Si vous faites cette partie optionnelle il faudra donc faire évoluer le modèle et votre vue pour permettre de ne pas perdre le dessin.

Recommandations. Il faudra tâcher de suivre les principes d'ergonomie vues en IHM. Il faudra fournir un *code fonctionnel*, *agréable à lire*, *documenté* (javadoc) et qui répond à toutes les recommandations et bonnes pratiques vues avec Luc Hernandez, en particulier en ce qui concerne le déploiement avec make.

Finalement, et c'est un aspect essentiel de ce projet, votre code doit s'organiser selon le **modèle MVC**. En particulier, pour la communication entre M et le reste, vous **devez utiliser l'API**.

À rendre

- Le code fonctionnel décrit aux sections précédentes **sur le git d'un étudiant du binôme** avec comme nom de projet **FAprojetIHM2020**.
- Un rapport technique de 3 pages environ explicitant la réalisation, en particulier l'API avec des diagrammes adaptés, détaillant ce qui a été fait et testé avec succès, ce qui a été fait mais n'a pas fonctionné.
- Il faut aussi décrire des cas de tests à réaliser de manière interactive sur votre interface (que ce soit des tests passés avec succès par votre réalisation ou des tests montrant un bug).

Indication sur la notation La partie IHM compte pour 14 points environ. La partie BdD compte pour 6 points environ. La notation favorisera une réalisation moins complète mais de qualité sur une tentative de résolution complète mais avec des finitions médiocres.

Description de l'API

Au tableau.
L'API vous sera communiquée via git.