TDLOG – séance n° 5
Notions d'architecture logicielle : TP

Xavier Clerc – xavier.clerc@enseignants.enpc.fr
Thierry Martinez – thierry.martinez@inria.fr
Olivier Cros – olivier.cros0@gmail.com

31 octobre 2016

À rendre au plus tard le 5 novembre 2016 sur educnet

On continue lors de cette séance à travailler sur le jeu commencé lors des séances précédentes. L'objectif est de remplacer l'interface textuelle par une interface graphique (à l'aide de la librairie PyQt) et de structurer le programme en modules.

1 Installation de Qt et PyQt

1.1 Avec conda

L'installation s'effectue simplement en exécutant dans un terminal (ou *shell*, ou encore *invite de commande*) :

```
conda install pyqt
```

ou, pour forcer l'utilisation de la version 4 :

```
conda install pyqt=4
```

1.2 Sous Windows sans conda

L'installation se fait simplement en téléchargeant et exécutant un installeur téléchargé à l'adresse suivante : http://www.riverbankcomputing.co.uk/software/pyqt/download
Attention : bien choisir un installeur pour *Python* 3, Qt 4, et compatible avec la version de *Python* installée (32 ou 64 bits).

1.3 Sous Linux sans conda (Debian et Ubuntu)

L'installation se fait simplement par le gestionnaire de paquets, en exécutant les commandes suivantes, pour installer respectivement Qt4 pour *Python* 3 et QtDesigner :

— `apt-get install python3-pyqt4`
— `apt-get install qt4-designer`

1.4 Sous Mac OS X sans conda

L'installation est un peu compliquée, et est assez longue (plusieurs minutes) car elle nécessite la compilation de nombreux éléments. La manière la plus simple d'installer l'ensemble des éléments est de suivre les étapes suivantes :

1. installation de Homebrew (http://brew.sh) par `ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
2. installation de *Python* 3 : `brew install python3`
3. installation de Qt : `brew install qt`
4. installation de SIP : `brew install sip --with-python3`
5. installation de PyQt : `brew install pyqt --with-python3`

Attention : bien suivre les différentes instructions données par `brew`. Typiquement, `brew` demande de modifier le fichier `.bash_profile` pour y définir une variable `PYTHONPATH`.

2 Interface pour le jeu des pièces

2.1 Première étape

La première étape consiste à remplacer l'interface textuelle par une interface graphique. Pour simplifier l'implémentation, il est raisonnable de représenter le jeu à l'aide de boutons portant des nombres représentant les valeurs des pièces, et de placer ces boutons à l'aide d'une grille en s'inspirant de l'exemple de programmation d'une calculatrice (utilisant la classe `QtGui.QGridLayout`) présenté à l'adresse suivante :

`http://zetcode.com/gui/pyqt4/layoutmanagement/`

Il s'agit de porter une attention toute particulière à l'architecture de votre code. L'implémentation doit être modulaire : le code de l'interface graphique doit appeler les fonctions qui définissent le jeu et ses règles, mais pas le contraire. Idéalement, cela doit permettre de développer une interface graphique complètement différente en ne modifiant pas la partie du code qui définit le jeu.

2.2 Seconde étape

La seconde étape consiste à permettre de jouer soit à deux joueurs, soit contre l'ordinateur, en :

— utilisant les boutons pour saisir les instructions des utilisateurs (la méthode `setEnabled` sur un `QtGui.QPushButton` permet de le rendre inactif) ;
— appelant le code min-max pour déterminer les coups de l'ordinateur.

2.3 Optionnel : menu

Pour produire un jeu complet, vous pouvez ajouter un menu (fondé par exemple sur la classe `QtGui.QDialog`) qui s'affiche au lancement du programme. Ce menu permettrait typiquement de choisir la taille de la grille, le type de joueur (humain ou ordinateur) et la force de l'IA le cas échéant.

3 Annexe : Documentation Qt4

Le site officiel de *Python* contient quelques pages à propos de PyQt :

- https://wiki.python.org/moin/PyQt
- https://wiki.python.org/moin/PyQt4
- https://wiki.python.org/moin/PyQt/SampleCode

Une documentation complète, avec notamment la liste des classes disponibles est accessible aux adresses suivantes :

- http://pyqt.sourceforge.net/Docs/PyQt4
- http://pyqt.sourceforge.net/Docs/PyQt4/classes.html

Enfin, un wikilivre (en français) et un tutoriel (en anglais) sont accessibles aux adresses suivantes :

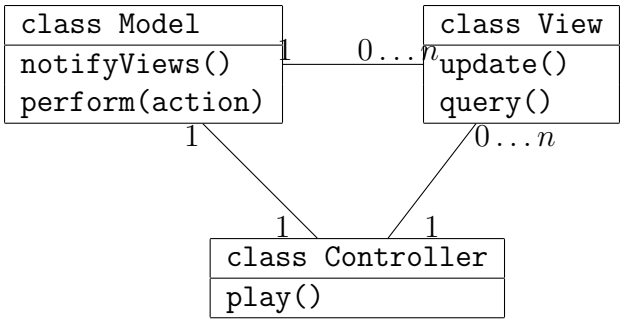
- http://fr.wikibooks.org/wiki/PyQt
- http://zetcode.com/gui/pyqt4

4 Annexe : le *design pattern* MVC

Il n'est pas *requis* d'appliquer ce design pattern dans ce TP, mais sa connaissance est néanmoins utile.

En développement logiciel, un *design pattern* est un ensemble de choix de conception documenté dans la littérature, référencé par un nom, que l'on peut appliquer pour résoudre un problème donné. C'est surtout utilisé en programmation orientée objet, car le paradigme donne beaucoup de liberté de conception au programmeur, et il est utile de pouvoir s'appuyer sur un guide de pratiques déjà éprouvées, et de pouvoir les repérer et les nommer pour se comprendre entre programmeurs.

Le design pattern *modèle–vue–contrôleur*, MVC, désigne la conception d'une application interactive, où (1) le modèle de l'application (sa "logique", son "algorithme"), (2) l'affichage des données, et (3) l'interaction avec l'utilisateur sont tous trois décrits séparément, de façon à structurer le programme en isolant chaque tâche bien définie.



On lit ce diagramme UML de la façon suivante : une instance MVC est composée d'une instance de la classe `Model`, d'une instance de la classe `Controller` et d'un nombre arbitraire d'instances de la classe `View`. Chaque instance de la classe `Model` contient l'état du jeu et ses règles. Le design pattern *Observateur* décrit le rapport entre une instance de `Model` et les instances de la classe `View` avec lesquelles celle-ci est associée : chaque instance de la classe `View` est capable de s'enregistrer auprès d'une instance de la classe `Model`, de façon à ce que la méthode `notifyViews()` de `Model` appelle les méthodes `update()` de toutes les instances de `View` enregistrées.