Projet cartographie aérienne et géoréférencement

Note d'installation et d'utilisation du Matlab Computer Runtime

1) Objectifs

Lors de notre projet de cartographie aérienne, il nous a été demandé de trouver une solution d'interfacer Matlab et Java. Une fois le code testé sur Matlab nous avons commencé notre recherche pour permettre l'intégration des briques logicielles dans une interface graphiques apportant plus de fonctionnalités. Ce document résume les étapes pour la mise en place d'une telle interface.

2) Environnement et pré-requis

Nous avons développé notre interface graphique sur environnement Linux, plus particulièrement sur une version récente du système d'exploitation Ubuntu (12.04/12.10)
Les raisons expliquant ce choix seront expliquées par ce qui suit.

Pour interfacer Matlab et l'interface graphique Java tout en garantissant une portabilité maximale. Il est nécessaire d'installer une surcouche logicielle gratuite ; le Matlab Computer Runtime (MCR). Pour l'instant, il faut prendre le MCR R2012a pour Linux disponible ici :

http://www.mathworks.fr/products/compiler/mcr/

Chaque version du MCR est compatible avec un système d'exploitation Linux, Windows ou Mac, ainsi qu'avec une version particulière de Matlab. Actuellement, sur le site de Matlab
trois versions du MCR sont disponibles sur chacune des plates-formes :

MCR R2012a
MCR R2012b
MCR R2013a

Pour installer le MCR (sous Linux), décompresser l'archive téléchargée, puis lancer en root le programme d'installation (en console) :

```
cd dossier_archive_MCR
sudo ./install
```

Pour garantir une compatibilité avec le MCR, il faut une version de Matlab récente, typiquement une version R2012 convient très bien. Nous disposions de Matlab R2012a pour Linux. La version R2010 étant actuellement installée à l'école.

Nous avons développé notre interface avec Java, il suffit donc d'installer les paquets :

openjdk-7-jdk
openjdk-7-jre

Avant de parler de la manière de générer les codes, il est nécessaire de comprendre que lorsque

Matlab génère des classes à partir de vos fonctions Matlab, celles-ci sont compatibles avec la version de Java utilisée lors de la compilation et avec le MCR de la version de Matlab installée. L'ordinateur client devra donc disposé de la version de Java recommandé et du bon MCR. Dans notre cas le code fournit est donc compatible avec les système Linux, disposant de java 7 et du MCR Matlab R2012a.

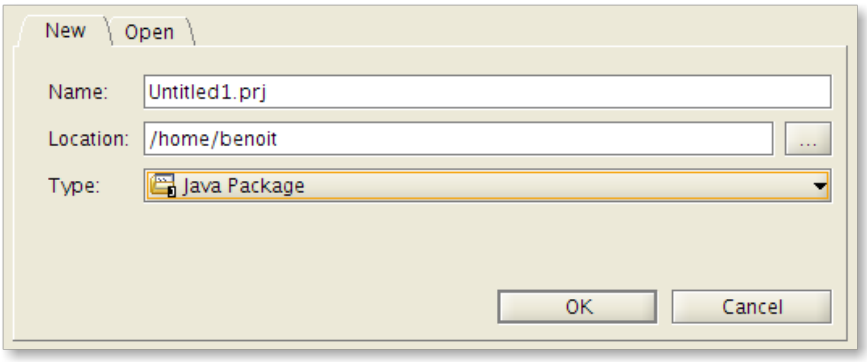
3) Comment générer du code Java depuis Matlab

Nativement, Matlab permet de générer des archives Jar et une documentation afin de pouvoir utiliser vos .m avec Java.

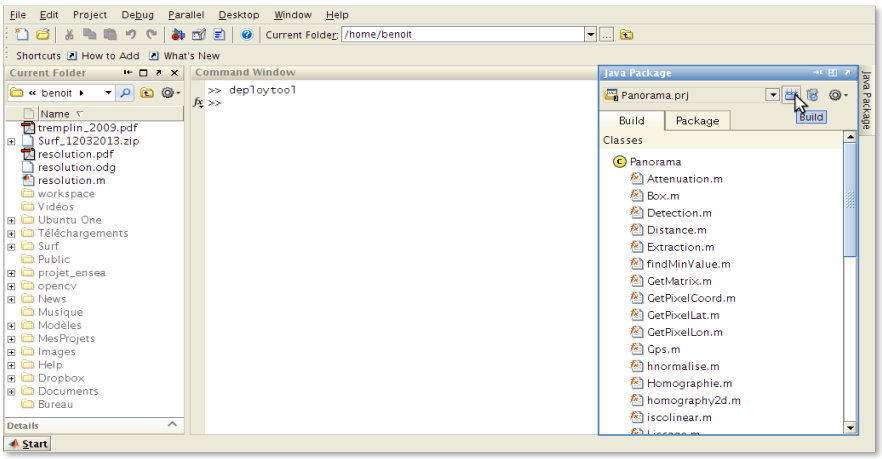
Une fois vos fonctions achevées et testées, quand vous voulez créer vos classes, il suffit de taper dans le workspace Matlab :

deploytool

Une fenêtre s'ouvre et la configuration est très simple puisqu'il suffit de choisir Java Package dans le dernier champs



Ensuite, un panneau latéral apparaît dans Matlab , c'est dans celui-ci qu'il vous sera demandé le nom de votre nouvelle classe. Une fois renseigné, il suffit de glisser vos fonction dans le panneau latéral. Ensuite, un clic sur « Build » permet de lancer la construction de vos classe.



Une fois, la construction lancée Matlab parse vos fichier .m à la recherche des fonctions que vous utilisés dans ceux-ci afin de les inclure dans votre fichier Jar. Cette étape peut-être très longue. À la fin, un fichier jar et une documentation sont générés dans le dossier du projet.

4) Comment utiliser le Jar fournit par Matlab ?

Cette étape est celle qui nous a pris le plus de temps car elle nécessite quelques subtilités. Heureusement, celles-ci sont maintenant regroupées dans un petit script de lancement. Rapidement, elles résident dans différents exports de variables systèmes et diverses modifications du classpath. Il ne faut pas oublier d'importer votre Jar dans les fichier de classes java via un import.

Voici les différents export qui seront effectuées par le script :

export LD_LIBRARY_PATH="/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/runtime/glnx86:/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/bin/glnx86:/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/sys/os/glnx86:/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/sys/java/jre/glnx86/jre/lib/i386/native_threads:/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/sys/java/jre/glnx86/jre/lib/i386/server:/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/sys/java/jre/glnx86/jre/lib/i386"

export XAPPLRESDIR="/usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/X11/app-defaults"

Voici comment le script modifie le classpath à chaque lancement du projet et à chaque compilation :

java -classpath /usr/local/MATLAB/MATLAB_Compiler_Runtime/v717/toolbox/javabuilder/jar/javabuilder.jar:$DIR/resources/Jar/Panorama.jar:$DIR/resources:$DIR:$DIR/classes Test;

($DIR = dossier d'éxécution du programme)