**Programmer sous Fiji/ImageJ**

La programmation sous Fiji / ImageJ est une programmation en JAVA standard, il y a peu de particularité par rapport à une application Java standalone.

Le développement sous ImageJ est maintenant bien documenté pour notre projet avec des sources disponibles pour plusieurs programmes (https://github.com/salimkanoun/Scintigraphy)

Il y a essentiellement 2 particularités :

* La richesse de l’interaction des APIs d’ImageJ : Il y a un grand nombre d’API sous ImageJ qu’il convient d’utiliser autant que possible, cela permet de simplifier considérablement le développement sur toutes les opérations réalisées sur l’image (opération sur l’image, lecture des metadonnées, dessin des régions d’intérêt, des légendes, récupération de valeurs dans l’image ou d’une région d’intérêt).

Au final l’essentiel du travail du développement consiste à trouver les bonnes APIs à utiliser et gérer l’interface et actions de l’utilisateur.

La plupart des APIs utilisées sont en exemple dans les programmes déjà disponibles, en fonction de votre projet si des besoins nouveaux apparaissent renseignez-vous auprès de votre tuteur de stage pour savoir si la fonction recherchée existe déjà dans les librairies et/ou sollicitez le forum d’ImageJ.

* La mise en place d’un plugin nécessite 2 éléments : l’implémentation de l’interface plugIn d’ImageJ dans la classe principale qui force l’ouverture d’une méthode de démarrage du programme « public void run(String arg) » en remplacement de la méthode main usuelle de Java pour le demarrage et l’édition d’un fichier plugin.config signalant à ImageJ dans quel menu votre programme doit apparaitre et qu’elle est la classe principale.

1. Notions globales

Sont ici mentionnées les elements de langage des objets d’ImageJ

* 1. ImagesPlus

Une ImagePlus est un objet que vous manipulerez tres frequement, c’est un objet qui contient « l’image » dans ImageJ

Cette image est constituée :

* De metadonnées « Info » qui comporte pour nous les tags des images DICOM (nom du patient, date d’acquistion…)
* Des matrice images
  + Soit une image
  + Soit plusieurs Images (on parle aussi d’ImageStack), dans cette pile d’Image chaque Image à sa propre métadonée.
  1. ImageStack

Il s’agit d’une pile d’image numérotée de 0 à n.

Vous pouvez acceder pour chaque image du stack aux :

* Pixels de l’image via l’ImageProcessor
* Metadonées via les Info
  1. ImageProcessor

Permet de traiter avec les pixels de l’image (par exemple faire un flip de l’image)

* 1. WindowManager

C’est le gestionnaire de fenetre d’ImageJ, très utile pour savoir qu’elles images sont ouvertes (chargée en mémoire) et disponibles pour le travail.

* 1. IJ.setTool

C’est l’outil de dessin de région d’intérêt (ROI) , il existe des ROI de différentes géométrie, irrégulière (la plus utilisée appelée polygone), ou cercle, rectangle….

Quand la ROI est dessinée on peut récupérer les valeurs de cette zone (statistics), stocker la ROI dans le ROI manager…

* 1. ROI Manager

C’est un gestionnaire de region d’interet, il est à instancier à l’ouverture du programme pour y stocker les regions d’interet que l’utilisateur dessine lors de l’utilisation.

Vous pouvez stocker, mettre à jour, afficher, cacher, effacer ou même faire des opérations sur les ROI (par exemple récupérer les zones de superpositions de deux ROIs).

Très utile pour garder en mémoire le travail de l’utilisateur, réafficher une zone qui doit être ajustée sur l’image (pour éviter à tout refaire re-dessiner…)

* 1. Overlay

Ce sont des calques en surimpression sur l’image, sert à afficher des légendes, les ROIs précédemment déssinées etc…

* 1. Montage

Permet de concaténer des images dans une nouvelle image, par exemple mettre 4 images sous la forme d’un tableau de 2\*2.

Attention ne marche que pour des images de dimensions identiques.

Utile pour rassembler les différentes captures / courbes dans un rendu final

* 1. ImageStatistics

Permet de recuperer les valeurs d’une region d’interet.  
Permet de récuperer le nombre de voxel, leurs valeurs moyenne, le max, le min etc etc…

Fondamental pour faire les calculs.

* 1. Logs

Un des defauts d’ImageJ est la faiblesse des logs pour les APIs, en cas de mauvaise utilisation d’APIs le programme s’arrête sans trop d’explication.

Par contre pour vos lignes de code, les erreurs s’affichent normalement dans la console (accessible via le menu windows). La console affichera les erreurs et vos System.out.print

Il est également possible de forcer des messages à l’utilisateurs via ImageJ soit en boite de dialog (IJ.showMessage(« msg »)) soit dans une fenêtre de log (IJ.log(«log »))

1. Tips de programmation
   1. Jar à importer dans Eclipse

Au minimum vous aurez à importer le jar ij.jar qui contient l’interface plugIn pour la déclaration de votre plugin dans ImageJ.

Cette librairie contient egalement l’object ImagePlus nécéssaire à la gestion des images dans ImageJ.

Vous aurez également probablement d’autres librairies (Math, JFreeChart..).

Il n’y a pas de limites, toutes les librairies JAVA peuvent être incluses à votre code.

Cependant pensez à verifier que la librairies dont vous avez besoin n’est pas déjà distribuée avec Fiji

Si c’est le cas utilisez la version distribuée dans Fiji pour éviter des conflits de version

Sinon pensez à signaler le besoin de livrer la librairie avec votre programme.

* 1. Plugin.config

Il s’agit d’un fichier texte à placer à la racine de votre JAR qui permet de demander l’ouverture d’un entrée dans le menu d’ImageJ et qui indique quelle est votre classe principale pour le démarrage du programme (cf exemple dans gitHub).

Vous pouvez livrer plusieurs programmes dans un même jar.

* 1. Structure

Pour votre programme suivez au minium cette structure : Une classe interface graphique « Vue », une classe « Controleur » et une classe « modele » (pour vous calculs).

Evidemment vous pouvez créer d’autre classes si besoin, il s’agit d’une structure minimale.

* 1. Debogage

Lancer ImageJ dans Eclipse via un classe separée en instanciant vos programmes via leur main (voir classe test\_Debug dans les sources). Ainsi vous aurez les message d’erreur dans la console d’eclipse (autrement le lancement directement dans ImageJ est peu locace pour les bugs)

Si demarrage dans ImageJ hors Eclipse : Si besoin pour loger vos variables utilisez IJ.log pour faire apparaitre votre message dans le gestionnaire de log d’ImageJ. Autrement System.out.print est également accessible via une ouverture manuelle de la console dans ImageJ (menu Windows> Console)

* 1. GUI

Dans l’idéal les composants Swing sont les plus performants pour la gestion de l’interface graphique.

Il existe cependant une difficulté : Les composant Swing ne sont pas Thread Safe et chaque action des compostant Swing doit être placée dans l’EDT.

Ceci doit etre réalisé en utilisant la classe SwingUtilities ou SwingWorker (cf <http://gfx.developpez.com/tutoriel/java/swing/swing-threading/>)

Cependant pour ne pas complexifer le code, il est possible d’utiliser certains composants en AWT qui n’a pas cette limitation.

Le problème se pose essentiellement pour faire des mises à jour de l’affichage avant capture d’écran, si vous n’avez qu’un composant (label ou bouton) à mettre à jour, utilisez AWT.

* 1. Penser à l’internationalisation (separateur « . » ; pas d’accent)

Les programmes réalisés ont vocation à être diffusés à l’international :

Utilisez le system US pour les decimale (.) et gardez votre interface en anglais.

1. Fonctions communes disponibles

Un certain nombre de fonctions ont été créés et rendues disponible en Public Static ce qui permet d’accéder à des fonctions sans à dupliquer de code (ni à instancier d’objet)

Ces methodes permet de bénéficier de fonctions debeugées réalisant pour vous des opération assez complexe (tri des images, capture d’ecran..)

Elle sont expliquées ici et documentée dans la javadoc (cf notre git)

* 1. Tri des Images
     1. Ismultiframe
     2. IsAnterior
     3. Tri Image Ant post
     4. TrierSeries par Acquisition Time
  2. Application de la prefered LUT
  3. Capture secondaire
     1. Image seule
     2. Fenêtre entière
     3. Capture to Stack
  4. Génération du tag DICOM pour la capture secondaire
  5. ExportAll
  6. Ouverture et enregistrement DICOM

1. Documentation
   1. Javadoc ImageJ

<https://imagej.nih.gov/ij/developer/api/index.html>

* 1. Forums

http://forum.imagej.net/

* 1. Sources
     1. Projet PET/CT et Scintigraphie

Sources et javadoc : <https://github.com/salimkanoun/Scintigraphy>

* + 1. Autres

Autre exemples de programme réalisés en Java (moins utile pour votre projet)

https://github.com/salimkanoun

1. Contact

Salim Kanoun

Salim.Kanoun@gmail.com