

# Annexe A

## Structure de l'application

Dans le but premier de coordonner nos actions pendant le développement de l'application, voici une liste de tout les fichiers du programme ainsi qu'un bref descriptif sur la fonction de chacun d'eux

### watershed/

#### **StandAlone.java**

Cette classe contient le programme exécutable de l'application. Elle permet de récupérer l'éventuel argument « -debug » de la ligne de commande d'exécution et d'informer au système d'exploitation de suivre chaque instruction du programme. Elle affiche ensuite l'objet MainFrame contenu dans watershed/fenetres/.

#### **MainPanel.java**

Cette classe définit le JPanel de l'application. Elle importe la plupart des éléments graphique de la bibliothèque Swing et les éléments graphique prédéfinis par nous-mêmes tels que la barre de menus,... et les dispose correctement dans le panneau. Elle contient aussi les méthodes de gestion des fenêtres filles de l'application.

### watershed/fenetres/

#### **MainFrame.java**

Cette classe étend la classe JFrame. Elle crée une fenêtre de type Swing dans laquelle sera mise un objet MainPanel définit juste au-dessus. Elle agrandit aussi cette fenêtre au maximum de l'écran.

#### **ImageFrame.java**

Cette classe étend la classe JInternalFrame. Elle permet de créer des fenêtres filles à la fenêtre principale et d'y afficher à l'intérieur une image donnée.

### watershed/langages/

#### **Const.java**

Cette classe contient l'ensemble des constantes chaînes de caractères de l'application en fonction de la langue choisit. Elle contient aussi la méthode de chargement de ces variables du fichier .wlg correspondant.

#### **Francais.wlg ; Deutsch.wlg ; English.wlg**

Ces fichiers sont des fichiers textes formatés de la façon suivante : « *nom\_de\_la\_variable\_globale=valeur* ». Ceci permet sans aucun problème de créer la gestion d'une nouvelle langue.

#### **Maker.java**

Interface graphique permettant la création d'un langage dans un format correcte.

### watershed/menus/

#### **Bar.java**

Cette classe étend la classe JMenuBar et crée les différents menus de notre application avec les diverses classes contenues dans ce même répertoire et qui sont définit ci-dessous.

#### **MEdit.java ; MFile.java ; MHelp.java ; MView.java ; MWindow.java**

Ces classes mettent en œuvre les différents menus de l'application. Seule le menu de filtre est fait de manière statique et est situé dans le répertoire des filtres.

**watershed/misc/****Chargement.java**

Cette classe étend la classe Window. Elle définit la barre de progression qui s'affiche, par exemple, pendant le chargement d'une image, l'application d'un filtre, ...

**watershed/icones/**

Ce répertoire contiendra les différentes icônes de l'application, en particulier les icônes des différents menus. Les icônes des menus sont des icônes au format GIF, de dimensions 16 x 16 pixels en 256 couleurs. Ceci permet de rendre l'interface de l'application un peu plus joviale.

**watershed/workspace/****Couleur.java**

Cette classe permet la gestion des couleurs. On y trouvera par exemple des fonctions comme retrouver le niveau de bleu dans une couleur en mode ARGB (alpha, red, green, blue) .

**Data.java**

Cette classe symbolise les images. Chaque image est copiée dans un tableau selon son type. Elle permet de récupérer une couleur sous forme d'entier sur 32 bits quelque soit le mode de l'image (N&B, niveaux de gris, couleur 32bits, couleur 24bits, indexer ...) ou la valeur brute stocker physiquement en mémoire.

**Niveaux.java**

En cours de construction... (gestion des niveaux des couleurs de l'image)

**Picture.java**

Liste de Layer.

**Layer.java**

Une couche de donnée, contient donc un Data

**watershed/formats/****FileFormat.java**

Interface définissant les méthodes .

**FileFormatChooser.java**

Cette classe permet de charger l'ensemble des formats de fichier disponible. Elle étend JFileChooser, donc on l'utilise comme boîte de dialogue pour choisir le fichier à ouvrir.

**FileFormatFilter.java**

Etend FileFilter qui permet à une JFileChooser ou plutôt dans notre cas une FileFormatChooser d'afficher uniquement les fichiers dont les extensions (\*.\*, \*.bmp, ...) sont valides par la méthode accept(File f) de FileFormatFilter

**BMP.java ; GIF.java ; JPG.java**

Ces classes définissent les types d'images ouvrables par l'application ainsi que la méthode d'ouverture de ces images. Implementent FileFormat et étendent FileFormatFilter.

**watershed/filtres/****Filter.java**

Cette classe définit les différentes méthodes qui doivent être implémentées par n'importe quel filtre. Tous filtres doivent ainsi implémenter cette classe.

**FilterAction.java**

Classe définissant l'action que va produire le clic sur le filtre dans le menu filtre.

**FilterLoader.java**

Cette classe permet de détecter l'ensemble des filtres contenus dans le répertoire, c'est à dire l'ensemble des classes implémentant Filter.java.

**Gradient.java ; GrayScale.java ; Moyenne.java ;Median.java ;Watershed.java**

Ce sont les classes des actuels filtres de l'application. Chacune définit les actions des filtres qu'elles représentent.

# Annexe B

## Implémentation de l'algorithme de Watershed

Voici l'algorithme utilisé :

```
Type Point :
| x : entier : position sur X
| y : entier : position sur Y
| lab : entier : numéro de label
Fin type

p1 : Point

hauteur : entier : hauteur en pixels de l'image
largeur : entier : largeur en pixels de l'image
DM : entier : tolérance
Max : entier : couleur d'intensité max à traiter
x, y, i, j, c, k, lab : entiers : différentes variables
x1, x2, y1, y2 : entiers : valeurs de fenêtrage
changements : booléen
t[256] : tableau de 256 piles p de <Point>
p, pileTmp : piles < Point >

Labels[][] : tableau 2 dimensions d'entiers

faire Labels.INIT()

pour
| y ← 0
| max ← 0
tantque y < hauteur répéter
| si Pixel[x,y]>max
|   alors max ← Pixel[x,y]
|   sinon rien
| fsi
| y ← y + 1
fpour

max ← max / 5;
label ← 1;

pour
| c ← 0
| labelCourant ← 1
tantque c<=max répéter

  pour
  | y ← 0
  tantque y<hauteur répéter

    pour
    | x ← 0
    tantque x < largeur répéter

      si Labels[x][y]=0 et Pixel[x,y]<=c
      alors
        k ← labelCourant
        labels[x][y] ← k
        x2 ← x
        x1 ← x2
        y2 ← y
        y1 ← y2
```

```

pour
| changements ← vrai
tantque changements répéter

    si x1 > 0
    | alors x1 ← x1 - 1
    | sinon rien
    fsi

    si y1 > 0
    | alors y1 ← y1 - 1
    | sinon rien
    fsi

    si x2 < largeur
    | alors x2 ← x2 + 1
    | sinon rien
    fsi

    si y2 < hauteur
    | alors y2 ← y2 + 1
    | sinon rien
    fsi

    changements ← faux

pour
| j ← y1
tantque j < y2 répéter

    pour
    | i ← x1
    tantque i < x2 répéter

        si Labels[i][j] = k
        | alors si i>0 et Labels[i-1][j] = 0 et Pixel[i-1,j] <= c + DM
        | | alors labels[i-1][j] ← k
        | | changements ← vrai
        | | sinon rien
        | fsi
        | si i<largeur-1 et Labels[i+1][j]=0 et Pixel[i+1,j] <= c + DM
        | | alors Labels[i+1][j] ← k
        | | changements ← vrai
        | | sinon rien
        | fsi
        | si j>0 et Labels[i][j-1]=0 et Pixel[i,j-1] <= c + DM
        | | alors Labels[i][j-1] ← k
        | | changements ← vrai
        | | sinon rien
        | fsi
        | si j<hauteur-1 et Labels[i][j+1]==0 et Pixel[i,j+1] <= c + DM
        | | alors Labels[i][j+1] ← k
        | | changements ← vrai
        | | sinon rien
        | fsi
        | sinon rien
        fsi
        i ← i + 1
    fpour

    j ← j + 1
fpour

label ← label + 1

    sinon rien
    fsi
x ← x + 1

fpour

y ← x + 1
fpour

c ← c + 1
fpour

```

```

pour
  i ← 0
tantque i < 256 répéter
  t[i].INIT()           // Initialisation des piles p du tableau t
  i ← i + 1
fpour

pour
  y ← 0
tantque y < hauteur répéter
  pour
    x ← 0
    tantque x < largeur répéter
      si Labels[x][y] <> 0
        alors i ← Pixel[x,y]
              t[i].ajouter(Point.INIT(x,y,i,Labels[x][y]))
        sinon rien
      fsi
    x ← x + 1
  fpour
  y ← y + 1
fpour

pour
  changements ← vrai
tantque changements répéter

  changement ← faux

  pour
    g ← 0
    tantque g < 256 répéter
      pour
        ListTmp ← t[g]
        tantque non changements et non ListTmp.estVide() répéter
          p1 ← ListTmp.sommet()
          faire ListTmp.dépiler()
          x ← p1.x
          y ← p1.y
          lab ← p1.lab

          si x>0 et Labels[x-1][y] = 0
            alors Labels[x-1][y] ← lab
                  faire t[Pixel[x-1,y]].empiler(Point.INIT(x-1,y,Pixel[x-1,y],lab))
                  g ← 255
                  changements ← vrai
            sinon rien
          fsi

          si x<largeur-1 et Labels[x+1][y] = 0
            alors Labels[x+1][y] ← lab
                  faire t[Pixel[x+1,y]].empiler(Point.INIT(x+1,y,Pixel[x+1,y],lab))
                  g ← 255
                  changements ← vrai
            sinon rien
          fsi

          si y>0 et Labels[x][y-1] = 0
            alors Labels[x][y-1] ← lab
                  faire t[Pixel[x,y-1]].empiler(Point.INIT(x,y-1,Pixel[x,y-1],lab))
                  g ← 255
                  changements ← vrai
            sinon rien
          fsi

          si y<hauteur-1 et Labels[x][y+1] = 0
            alors Labels[x][y+1] ← lab
                  faire t[Pixel[x,y+1]].empiler(Point(x,y+1,Pixel[x,y+1],lab))
                  g ← 255
                  changements ← vrai
            sinon rien
          fsi

        fpour
        g ← g + 1
      fpour

```

fpour

## Annexe C

### Sources d'informations

Pour nous renseigner sur les divers algorithmes utilisés et leur application, nous avons consultés les sources d'informations suivantes :

#### Bibliographie :

- Article de S. BEUCHER et F. MEYER : « The Morphological Approach to Segmentation : The Watershed Transformation »
- Documentation du professeur tuteur : « L'algorithme de lignes de partage des eaux » ainsi que « Pyramides adaptatives et stochastiques »
- Livre au format électronique PDF « Le programmeur JAVA 2 » des éditions de SUN Microsystems.

#### Netographie :

- <http://cmm.ensmp.fr/~beucher/wtshed.html> : site expliquant le principe de l'algorithme de Watershed par S. BEUCHER, docteur-chercheur à l'école des Mines de Paris.
- <http://www.cs.cmu.edu/~cil/vision.html> : site sur lequel l'on peut trouver des algorithmes pour tout ce qui concerne l'infographie de près ou de loin, on y trouve en particulier des algorithmes de transformations d'images en noir et blanc...
- <http://java.sun.com/> : site de SUN Microsystems sur JAVA, l'on peut y trouver différents tutoriaux sur les bibliothèques Swing, ainsi que la documentation complète du langage JAVA.
- <http://cauchy.fr.st/> : site d'un thésard de Strasbourg rencontré sur Internet qui nous a expliqué le fonctionnement de la détection de mouvement en utilisant l'algorithme de Watershed.
- <http://telin.rug.ac.be/~pds/papers/> : site sur lequel on peut trouver des algorithmes sur divers traitements d'images.