Traitement d'image

1 Objectif **1**

2 Lire ce document **2**

3 Spécifications **2**
3.1 Formats d'image supportés . . . 2
3.2 Entrées du programme . . . 2
3.3 Filtres . . . 2
3.4 Sorties du programme . . . 3
3.5 Cas d'erreur . . . 3
3.6 Affichages autorisés . . . 3
3.7 Résumé des fonctionnalités attendues . . . 3
3.8 Conseils . . . 3

4 Modalités du projet **3**
4.1 Environnement de référence . . . 3
4.2 À rendre . . . 4
4.3 Notation . . . 4

A Lire et écrire une image **5**
A.1 *JPEG* et *PNG* . . . 5
A.2 *ESI* . . . 5

B Spécification du format *ENSICA Simple Image* **6**
B.1 Entête . . . 6
B.2 Corps . . . 6
B.3 Exemple . . . 6

C Filtres **7**
C.1 Passage en niveaux de gris . . . 7
C.2 Flou . . . 7
C.3 Inversion de couleurs . . . 7

1 Objectif

Le but de ce projet est d'écrire un programme Java appliquant des filtres à des images, de manière non interactive.

2 Lire ce document

Les mots-clefs sont écrits en *italique*, les informations importantes en **gras**. Les liens, cliquables dans la version PDF, sont de deux types : interne en bleu et externe en magenta. Les extraits de code sont `écrits avec cette police`.

Les sections les plus importantes de ce document sont : la section 3 qui explicite les fonctionnalités et comportements attendus de votre programme ; la section 4 qui présente entre autres les critères de notation.

3 Spécifications

Cette section spécifie les fonctionnalités du programme que vous devez réaliser. C'est donc un cahier des charges, non négociable. **Si une fonctionnalité n'est pas écrite ici, c'est qu'elle ne vous est pas demandée**. L'équipe enseignante se réserve le droit de rajouter ou de retirer des fonctionnalités à tout moment.

3.1 Formats d'image supportés

Le logiciel doit pouvoir lire les images aux formats suivants :
— ESI (*ENSICA Simple Image*).
— JPEG (*Joint Photographic Experts Group*) ;
— PNG (*Portable Network Graphics*) ;
Le format ESI est décrit à l'annexe B. L'annexe A explique comme lire et écrire les images au format PNG et JPEG.

3.2 Entrées du programme

Le programme doit pouvoir prendre, à son lancement [1], les paramètres suivants, dans cet ordre uniquement :
— le nom du fichier contenant l'image à traiter ;
— le format du fichier d'entrée, une chaîne de caractères parmi : *ESI*, *JPEG* ou *PNG* ;
— le nom du fichier dans lequel enregistrer l'image créée après le traitement ;
— le format du fichier de sortie, une chaîne de caractères parmi : *ESI*, *JPEG* ou *PNG* ;
— une liste de filtres à appliquer, indiqués par leurs numéros, au moins un filtre étant indiqué (les numéros sont spécifiés en 3.3).

Toute autre interaction est interdite : le programme ne doit demander aucune autre information.

À titre d'exemple, pour modifier l'image `foo.png` avec les filtres numéro 3 et 1 (dans cet ordre), le résultat étant enregistré dans l'image `bar.jpeg`, les arguments seront les suivants :

```
foo.png PNG bar.jpeg JPEG 3 1
```

Si plusieurs numéros de filtre sont indiqués, ils sont alors appliqués dans leur ordre d'apparition dans les paramètres.

3.3 Filtres

3.3.1 Liste des filtres à implémenter

Voici la liste minimale de filtres à implémenter :
— passage en niveau de gris ;
— flou ;
— inversion des couleurs.

Vous trouverez à l'annexe C des pistes pour écrire de tels filtres. Vous êtes libres d'ajouter à votre guise de nouveaux filtres, à la seule condition d'avoir réalisé les 3 filtres cités ci-dessus.

1. Dans Eclipse, ces paramètres sont à spécifier dans la configuration du programme à lancer

3.3.2 Désignation d'un filtre au lancement du programme

Les numéros des filtres pour les désigner en paramètre du logiciel sont les suivants :

1. passage en niveau de gris
2. flou
3. inversion des couleurs

Vous pouvez attribuer les numéros que vous souhaitez aux filtres que vous créeriez en sus.

3.4 Sorties du programme

Le programme crée un nouveau fichier. Celui-ci contient l'image résultant de l'application des filtres sur l'image d'entrée, au format requis par l'utilisateur.

3.5 Cas d'erreur

Les scénarios suivants sont identifiés comme des cas d'erreur et doivent être gérés comme tels :
— le fichier d'entrée n'existe pas, il n'est pas possible de le lire, il n'est pas au format spécifié ;
— un ou plusieurs numéros de filtre spécifiés en paramètre sont incorrects ;
— il n'est pas possible de créer le fichier en sortie.

3.6 Affichages autorisés

Seuls les affichages des informations suivantes sont autorisés (mais non obligatoires) dans la console :
— le temps d'exécution d'un filtre.
De plus, il est autorisé d'afficher automatiquement l'image de sortie (voir 3.4) pendant l'exécution du programme.

3.7 Résumé des fonctionnalités attendues

Les fonctionnalités attendues sont donc les suivantes :

1. lire une image en entrée, contenue dans un fichier au format *ESI*, *JPEG* ou *PNG* ;
2. appliquer des filtres sur cette image ;
3. écrire l'image résultante dans un nouveau fichier au format *ESI*, *JPEG* ou *PNG*.

3.8 Conseils

Testez votre programme sur des petites images. Ne réinventez pas la roue : l'utilisation intelligente de bibliothèques externes est aussi une preuve de bonne conception !

4 Modalités du projet

4.1 Environnement de référence

L'environnement de référence est constitué des salles d'informatiques F116, F117 et A6-2 de l'ENSICA, sous Linux avec Eclipse. Autrement dit, **votre programme doit compiler et fonctionner sur les ordinateurs des salles F116, F117 et A6-2, sous Linux avec Java/Eclipse. Cette règle ne souffre aucune exception, sous peine d'une très mauvaise note**.

4.2 À rendre

Vous devez rendre les éléments suivants lors de la séance d'évaluation finale :

— Un document, **imprimé**, de 5 pages **maximum** décrivant votre architecture et vos choix techniques, comprenant les diagrammes de classes UML.

— L'archive ZIP de votre projet exportée à partir d'Eclipse [2]. Le projet **doit compiler et fonctionner dans l'environnement de référence** (voir 4.1) après importation.

De plus, vous effectuerez une démonstration de votre programme, sur les machines des salles F116, F117 ou A6-2 [3], lors de la séance finale, dans un temps maximal de 10 minutes. Aucune présentation de type PowerPoint n'est demandée.

4.3 Notation

Les critères de notation sont les suivants, par ordre d'importance décroissante, sur une note totale de 20 :

50% Modularité : couplage faible, présence claire de composants aux rôles bien identifiés et isolés

20% Qualité du code : lisibilité et commentaires

15% Tests : présence de tests unitaires utiles écrits à l'aide de JUnit

10% Qualité et pertinence des documents d'architecture rendus

5% Rapidité et qualité des filtres

De plus, pour **chaque fonctionnalité (voir 3.7) non réalisée, un malus de 2 points sera appliqué**.

Toute fonctionnalité non demandée que vous auriez écrite pourra être considérée comme apportant des points de bonification **si et seulement si toutes les fonctionnalités demandées dans ce sujet sont réalisées et testées**.

2. Menu `File > Export`. Choisir `Archive File` dans la fenêtre qui apparaît.
3. Cette démonstration ne se fait donc pas sur des ordinateurs personnels

A Lire et écrire une image

Une image n'est rien d'autre qu'une matrice de pixels, chacun indiquant la couleur d'un point dans cette matrice. Pour accéder à ces pixels, vous pouvez utiliser `BufferedImage` (voir annexe C). Toutefois, avant d'accéder à ces pixels, il faut pouvoir lire et écrire les fichiers contenant les images. Cette section vous donne donc des pistes pour lire et écrire des images dans les formats requis par le cahier des charges.

A.1 *JPEG* et *PNG*

Certains formats, tels que le *JPEG* et le *PNG* ne stockent pas directement une matrice de pixels, mais une forme compressée. Pour de tels formats, il vous faudra décompresser ces images avant d'y appliquer des filtres. Java contient par défaut tout ce qu'il faut pour cela grâce au paquetage ImageIO qui retourne un objet de type `BufferedImage`. Voici comment ouvrir et écrire une image *PNG* :

```
File src = new File("/path/to/srcimage"); // open both JPEG and PNG
File dest = new File("/path/to/dstimage");
BufferedImage img = ImageIO.read(src);
ImageIO.write(img, "png", dest);
```

Il suffit de remplacer `"png"` par `"jpg"` pour enregistrer au format *JPEG*.

A.2 *ESI*

Le format *ESI* est un format textuel (décrit à l'annexe B). Cela signifie que vous allez avoir besoin de lire et écrire des valeurs écrites en texte dans un fichier, et de les convertir en types primitifs Java (`int`, etc.).

Il existe une grande quantité de solutions pour lire un fichier textuel. L'une des plus simples est d'utiliser la classe `Scanner`. L'exemple suivant montre comment lire deux entiers, séparés par des espaces ou des retours à la ligne, dans un fichier :

```
FileInputStream fis = new FileInputStream("/path/to/file");
Scanner s = new Scanner(fis);
int value1 = s.nextInt();
int value2 = s.nextInt();
```

Dans le sens inverse, pour écrire du texte dans un fichier, vous pouvez utiliser la classe PrintWriter.

B Spécification du format *ENSICA Simple Image*

Le format *ESI* est constitué d'un en-tête et d'un corps.

B.1 Entête

L'entête, sur une ligne, donne le nombre de colonnes suivi du nombre de lignes, les deux valeurs étant séparées par un espace :

```
<colonnes> <lignes>
```

B.2 Corps

Le corps principal décrit l'image, pixel par pixel. Chaque ligne du fichier correspond à une ligne de l'image. Chaque colonne décrit une couleur d'un pixel. Trois colonnes successives représentent respectivement la composante rouge, vert et bleue, toujours séparées par un espace :

```
<red> <green> <blue> <red> <green> <blue> ...
<red> <green> <blue> <red> <green> <blue> ...
```

Les valeurs minimales et maximales d'une composante sont respectivement 0 et 255.

B.3 Exemple

Le texte correspondant à l'image de la figure 1, de 4 sur 4 pixels, est montré à la figure 2.

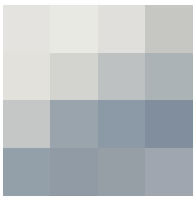

FIGURE 1 – Exemple d'image de 4 sur 4 pixels

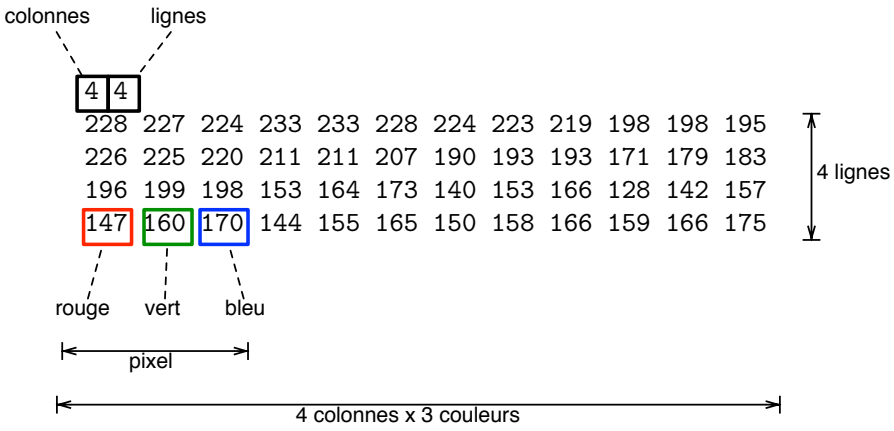

FIGURE 2 – Format ESI

C Filtres

Cette annexe vous donne quelques pistes pour écrire les filtres requis. Au préalable, il faut comprendre comment est encodée une image. Comme l'explique l'annexe A, une image n'est rien d'autre qu'une matrice de pixels, ceux-ci indiquant la couleur à afficher. Une couleur est désignée par trois valeurs : une composante rouge, une verte et une bleue (modèle RGB). Il existe bien d'autres modèles de couleurs, mais celui-ci reste le plus simple.

Vous pouvez accéder aux pixels à l'aide de `BufferedImage`, et aux composantes d'un pixel à l'aide de `Color`. À noter que si vous utilisez ce paquetage, chaque valeur de couleur est comprise entre 0 et 255. Voici un exemple de filtre qui intervertit les composantes :

```
import java.awt.Color;
import java.awt.image.BufferedImage;
...
BufferedImage src = ImageIO.read(new File("/path/to/src_file"));
BufferedImage dst = new BufferedImage( src.getWidth(), src.getHeight(),
                                       BufferedImage.TYPE_INT_RGB);
for (int y = 0; y < dst.getHeight(); ++y) {
  for (int x = 0; x < dst.getWidth(); ++x) {
    // Get color channels of the pixel located at x,y.
    Color color = new Color(src.getRGB(x, y));
    int red = color.getRed();
    int green = color.getGreen();
    int blue = color.getBlue();
    // The following line switch color channels as the normal
    // order is red, green, blue.
    color = new Color(blue, red, green);
    // Set the color of the pixel located at x,y.
    dst.setRGB(x, y, color.getRGB());
  }
}
ImageIO.write(dst, "png", new File("/path/to/dst_file"));
```

Évidemment, ce code n'est en rien modulaire.

C.1 Passage en niveaux de gris

Le principe de base est d'affecter la même valeur à chaque couleur d'un pixel. Il existe plusieurs méthodes pour calculer cette valeur :

- calculer la moyenne des trois composantes : $(R + G + B)/3$;
- faire la moyenne des maximums et minimums : $((max(R, G, B) + min(R, G, B))/2)$;
- appliquer la formule empirique suivante : $0.21R + 0.71G + 0.07B$.

C.2 Flou

Il est possible d'appliquer un flou sur une image de plein de manières différentes. Peut-être la plus simple est d'affecter à chaque pixel la moyenne des pixels d'origine dans un rayon donné $(1, 2, \ldots$ pixels). Une solution donnant des résultats plus élégants est le flou gaussien.

C.3 Inversion de couleurs

L'inversion de couleurs est très simple : il faut retrancher à la valeur maximale d'une couleur, 255, la valeur courante de la couleur. Il suffit donc d'effectuer cette opération pour chaque couleur de chaque pixel.