TP de Traitement et d'Analyse d'images

Objectif du TP :

Ce TP se divise en deux parties indépendantes :

- Utilisation de la librairie « minilib.jar » - programmation Java de traitements simples
- Utilisation de minilab / découverte de filtres 3x3 de traitements de base.

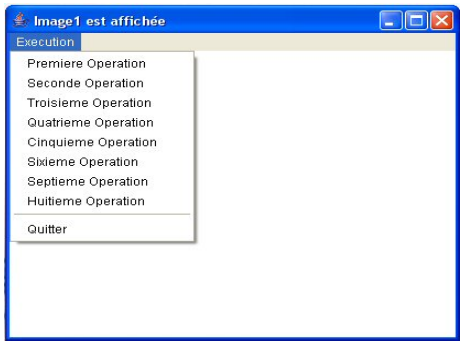
Partie 1 : Programmation et expérimentation

Présentation :

La librairie 'minilib.jar' est un ensemble de classes Java qui simplifie considérablement l'accès aux couleurs d'un fichier : cette mini librairie a été réalisé spécialement pour permettre à des débutants en programmation d'expérimenter des algorithmes simples sur des images GIF, JPG ou PNG.

Principe :

Une interface simplifiée vous est fournie : elle contient une image, un menu avec le choix de 8 actions + quitter. Vous allez programmer ces actions et visualiser le résultat sur l'image affichée.



Pour ce faire, vous allez devoir compléter un fichier (CApplication.java) qui vous permettra de créer des algorithmes simples de traitement d'image.

Ce fichier contient le code suivant :

```
public class CApplicationRobotique {
public static void ExecMenu1() { // est exécuté quand on choisit
le menu1
// à compléter
// un exemple de code est donné dans le fichier
}
public static void ExecMenu2() { // est executé quand on choisit
le menu2
// à compléter
// un exemple de code est également donné ici
}
(...)
// Vous pouvez changer le nom des menus dans l'interface,
// pour changer le nom des menus, il suffit de changer le
//contenu des chaînes qui suivent
public static String DonneNomMenu(int numMenu) {
String resultat=null;
switch(numMenu) {
case 1: resultat = /* changer ici -> */ "Premiere Operation" ;
break;
case 2: resultat = /* changer ici -> */ "Seconde Operation" ;
break;
case 3: resultat = /* changer ici -> */ "Troisieme Operation";
break;
case 4: resultat = /* changer ici -> */ "Quatrieme Operation";
break;
case 5: resultat = /* changer ici -> */ "Cinquieme Operation";
break;
case 6: resultat = /* changer ici -> */ "Sixieme Operation" ;
break;
```

```
case 7: resultat = /* changer ici -> */ "Septieme Operation" ;
break;
case 8: resultat = /* changer ici -> */ "Huitieme Operation" ;
break;
}
return resultat ;
}
}
```

Par exemple, si vous souhaitez faire exécuter du *code* quand l'utilisateur choisit l'item 'Premiere operation' du menu de l'application, il vous suffit de placer ce *code* dans la 'procédure' ExecMenu1. Vous pouvez donc réaliser 8 actions principales liées à l'interface donnée.

Vous pouvez également modifier le nom du menu en changeant la chaîne de caractères correspondante dans la 'procédure' 'DonneNomMenu' comme indiqué dans le texte.

Afin de simplifier la programmation, un certain nombre de méthodes sont mises à disposition. Vous disposez de 8 images différentes numérotées de 1 à 8. Vous pouvez décider de l'image à afficher (une seule à la fois) en utilisant 'CImage.ImageAAfficher(int num) '.

Par exemple, 'CImage.ImageAAfficher(3) ;' est une instruction qui indique que l'image à afficher dans la fenêtre est l'image numéro 3.

Par défaut, cette image est un petit carré noir. Il existe d'autres instructions pour modifier ces images. Par exemple, 'CImage.VideImage(int num, int tx, int ty) ' vous donne la possibilité de créer une image noire, (l'image numéro 'num') de taille (tx X ty).

CImage.ChargeImage(int num, String nomImage) permet de lire une image à partir du disque.

Exemple d'utilisation : vous souhaitez que le menu 'Première Opération' lise l'image 'essai.jpg' qui se trouve sur disque, puis l'affiche. Vous complèterez le code de la manière suivante :

```
public static void ExecMenu1() { // est exécuté quand on choisit
//le menu1
CImage.ChargeImage(1,"essai.jpg") ;
CImageAAffichier(1) ;
}
```

Et si vous souhaitez modifier le nom du menu, vous changerez la méthode :

```
public static String DonneNomMenu(int numMenu) {
String resultat=null;
switch(numMenu) {
case 1: resultat = /* changer ici -> */ "AFFICHAGE D'IMAGE" ;
break;
case 2: resultat = /* changer ici -> */ "Seconde Operation" ;
break;
case 3: resultat = /* changer ici -> */ "Troisieme Operation";
break;
case 4: resultat = /* changer ici -> */ "Quatrieme Operation";
break;
case 5: resultat = /* changer ici -> */ "Cinquieme Operation";
break;
case 6: resultat = /* changer ici -> */ "Sixieme Operation" ;
break;
case 7: resultat = /* changer ici -> */ "Septieme Operation" ;
break;
case 8: resultat = /* changer ici -> */ "Huitieme Operation" ;
break;
}
return resultat ;
}
```

Vous pouvez également modifier une image (après l'avoir lue sur disque) ou créer une image en utilisant les fonctions d'accès suivantes :

CImage.ImageAAfficher(int num)
indique quelle image parmi les 8 est à afficher

CImage.DonneTailleXImage(int num)
renvoie un entier qui donne la longueur de l'image 'num'

CImage.DonneTailleYImage(int num)
renvoie un entier qui donne la hauteur de l'image 'num'

CImage.ChargeImage(int num, String nomImage)
essaye de lire l'image 'nomImage' sur disque et renvoie true si ca a marché
CImage.DonneRougeImageXY(int num, int x, int y)
donne la valeur de la composante Rouge en (x,y) dans l'image num

CImage.DonneVertImageXY(int num, int x, int y)
donne la valeur de la composante Verte en (x,y) dans l'image num

CImage.DonneBleuImageXY(int num, int x, int y)
donne la valeur de la composante Bleu en (x,y) dans l'image num

CImage.ChangeRougeImageXY(int num, int x, int y, int v)

change la valeur de la composante Rouge en (x,y) dans l'image num

CImage.ChangeVertImageXY(int num, int x, int y, int v)
change la valeur de la composante Verte en (x,y) dans l'image num

CImage.ChangeBleuImageXY(int num, int x, int y, int v)
change la valeur de la composante Bleu en (x,y) dans l'image num

CImage.VideImage(int num, int tx, int ty)
permet d'effacer une image et de lui donner une autre taille (Attention, c'est à vous de vérifier que les positions (x,y) sont valides, c'est à dire dans les limites de taille de votre image)

Rappels JAVA : Vous devez utiliser un éditeur de texte + le compilateur / interpréteur Java pour pouvoir exécuter ce code. Ce travail présuppose donc un minimum de pré requis en programmation.
Vous disposerez de 'run.bat' qui exécute le programme sous windows et de 'compile.bat' qui prend en argument le fichier java que vous souhaitez compiler.

Travail à réaliser :

1. * Fabriquez une image de 100x100 entièrement orange (pas rouge, pas jaune, pas verte – orange)

2. * Fabriquez une image de 200x200 qui contient des pixels de couleurs aléatoires (Math.Random() retourne un nombre compris entre 0 et 1)

3. * Fabriquez une image avec un dégradé horizontal qui couvre toutes les couleurs saturées du spectre (rouge, jaune, vert, bleu, magenta).

4. Fabriquez un menu qui lit l'image 'test.jpg' et qui l'affiche

5. * Cette image est visiblement en négatif. Réalisez une option qui permet d'afficher l'image avec les bonnes couleurs.

6. Lisez l'image "vannes.jpg"

7. * Cette image est à l'envers. Modifiez votre option pour la retrouver à l'endroit.

8. * Vous disposez de 2 images : carte_droite et carte_gauche. Faites en sorte d'afficher les deux cartes sur la même image.

9. * 'im1.jpg' est une photographie de pièce. Chargez cette image et affichez cette pièce en noir et blanc.

10. * Colorisez cette image noir et blanc en rouge. En fait, on veut voir cette image comme à travers un filtre rouge.

Travail à rendre à la fin de la séance:

Pour chaque question marquée d'un (*), vous ferez un copié /collé de votre code dans un document texte de votre choix (en indiquant la question de référence). Vous rendrez ce document à la fin de la séance (pas après) en indiquant vos noms. Vous n'aurez peut être pas le temps de tout faire : rendez ce que vous avez fait.

Partie 2 : Expérimentations et filtres de base

Présentation :

MiniLab est un petit logiciel très simple, programmé en JAVA (donc pas extrêmement rapide), qui sert à expérimenter différents filtres de traitement d'image. Son utilisation est assez simple : commencez par charger une image de taille modeste (GIF ou JPG), puis utilisez les options pour transformer l'image et voir directement le résultat sur l'écran.

Fonctionnalités :

Le bouton 'Recharger l'image' se passe de commentaires.
Quand vous cliquez dans l'image, les couleurs RVB, apparaissent en haut dans la fenêtre prévue à cet effet. Cela vous permet d'avoir une idée de la valeur des composantes RVB pour la couleur cliquée.
Normalement, les algorithmes s'appliquent sur toute l'image. Si vous désirez appliquer l'algorithme courant (ou le filtre) seulement sur la couleur sélectionnée, alors, cochez la case 'sélection des pixels '. L'algorithme ne s'appliquera alors que sur les pixels de la couleur sélectionnée. Vous pouvez également spécifier une variation acceptée pour les valeurs RVB (Var rvb). Par exemple,
des valeurs de 10 indiquent que les pixels de la couleur sélectionnée + ou – 10 unités seront pris en compte.
La partie suivante (la matrice + le bouton 'appliquer le filtre') concerne l'application d'un filtre.
Le bouton 'Centre de gravité' calcule l'iso barycentre des points dont la couleur est différente de (0,0,0) : une croix apparaît alors à la position calculée.

Enfin, le bouton 'Appliquer opérateur' et les menus à sélection du bas vous permettent de composer un mini algorithme de traitement sur toute l'image (ou seulement sur les pixels de la couleur sélectionnée). Quand une constante apparaît dans la fonction sélectionnée, alors, vous pouvez entrer une valeur dans la case de droite prévue à cet effet.

Remarque générale sur l'utilisation de ce logiciel :
Ce logiciel a été réalisé pour l'occasion en qq heures, il est parfaitement fonctionnel mais présente les limitations suivantes :
La taille de l'image est volontairement limitée et (pour bien montrer la structure en grille de l'image) les pixels sont visibles (effet de ZOOM).
Si vous entrez des valeurs de coefficients non numériques dans les cases (histoire de rigoler), l'algorithme ne sera pas appliqué et votre nom immédiatement communiqué au poste de police le plus proche (ah, vous vouliez rigoler ?).

Travail à réaliser :

1. Commencez par ouvrir l'image d'Elvis. (elvispresley.jpg)
Utilisez les opérateurs pour effacer cette image et obtenir une image :
complètement noire
complètement jaune
complètement mauve
Essayez plusieurs niveaux de jaune (plus ou moins clairs). Vérifiez la couleur en cliquant avec la souris dans l'image.

2. Rechargez l'image d'Elvis.
Enlevez toute composante rouge de l'image. Vérifiez qu'il n'y a plus du tout de rouge en cliquant avec la souris. Faire la même chose avec la composante bleue. Vérifiez que les pixels sont tous de la forme (0,V,0) en cliquant avec la souris.

3. Rechargez l'image d'Elvis.
Cochez 'sélection des pixels' et cliquez sur le bord blanc (255,255,255) avec Var rvb = (0,0,0). Programmez R = 0, V = 0, B = 0 et cliquez sur 'appliquer opérateur'. Le fond de l'image devient noir.
Remarquez le 'halo' autours de la tête du King. Oui, c'est un Saint.
Virez moi illico cet anneau en utilisant Var rvb.

4. Rechargez l'image d'Elvis. Décochez 'selection des pixels'
Essayez le filtre :

0 0 0
0 1 0
0 0 0

Ca donne quoi ? C'était prévisible ?

5. Rechargez Elvis (!)
Essayez le filtre :

1 1 1
1 1 1
1 1 1

Ca donne quoi ? Essayez de l'appliquer plusieurs fois. Proposez un filtre pour faire marche arrière (c'est à dire revenir à l'image de départ) – est ce possible ? (Elvis est-il vraiment mort ?)

6. Rechargez Elvis.
Que font les filtres :

0 0 0	0 1 0	1 0 0
0 0 0	0 0 0	0 0 0
0 1 0	0 0 0	0 0 0

7. Elvis est un Rocker, un vrai. Faites-le danser avec le filtre :

1 0 0
0 0 0
0 0 1

8. Chargez l'image contours.gif
Le but du jeu est ici de détecter les contours des objets représentés en utilisant le filtrage.
Vous allez devoir concevoir un filtre chargé de mettre en évidence le bord des objets.
Idées :
Pensez à 3x3 pixels qui ont exactement la même couleur : la réponse du filtre en son milieu doit être de 0 (pas de contours). Cet exemple donne déjà une contrainte sur les coefficients.
Laquelle ?

Maintenant, pensez à un bord gauche d'un objet. Les pixels qui appartiennent au bord ont la configuration suivante (par exemple) :

X O O
X O O
X O O

avec X une couleur particulière et O une autre couleur. Le pixel du milieu appartient aux contours de l'objet. Commencez par réaliser un filtre chargé de détecter le bord gauche d'un objet en exploitant les deux dernières infos.
Le problème est symétrique pour les autres directions : proposez alors un filtre de détection de contours (Filtre Laplacéen).
9. Hypothèse : l'image contours.gif est une image donnée par une caméra qui filme une scène. On recherche une balle rouge dans cette scène, ou plus précisément, on cherche exactement dans cette image, la position du centre de cette balle rouge. Normalement, un système de vision enchaine automatiquement les algorithmes. Vous allez utiliser les outils de MiniLab pour mettre en évidence le centre de gravité de la balle dans cette image.
Idée : vous savez que la balle est rouge. Commencez par enlever (mettre à (0,0,0)) tout ce qui n'est pas rouge. Quand tout est noir sauf la balle, calculez le centre de gravité.

Vous n'avez pas de Documents à rendre pour cette partie, mais tous les traitements réalisés pendant ces travaux pratiques font partie des choses à savoir pour l'examen (attention)