Lecture et Ecriture d'Images numériques (Java, C++)

Alban Gaignard, alban.gaignard@cnrs.fr

11 février 2014

Solutions

A Partie 1

Fichier de construction Maven du projet Java : `pom.xml`. Les codes source Java doivent se situer dans un répertoire `src/main/java` au niveau du fichier `pom.xml`.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM
    /4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>fr.cnrs.i3s</groupId>
    <artifactId>TDimageIO</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>TDimageIO</name>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.10</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                    <archive>
                        <manifest>
```

```
                            <mainClass>fr.cnrs.i3s.tdimageio.solution.
                                ImageReaderWriter</mainClass>
                        </manifest>
                    </archive>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>attached</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Code source de la classe à réaliser.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package fr.cnrs.i3s.tdimageio.solution;

import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

/**
 *
 * @author gaignard
 */
public class ImageReaderWriter {

    public BufferedImage load(String filePath) {
        File fileSelected = new File(filePath);
        BufferedImage bi = null;
        try {
            bi = ImageIO.read(fileSelected);

        } catch (IOException e) {
            System.out.println(e.getStackTrace());
        }
        return bi;
    }

    public void histogram(BufferedImage bi) {
        int[] rHist = new int[256];

```

```
34          //Populates the histogram
35          int w = bi.getWidth();
36          int h = bi.getHeight();
37          for (int i = 0; i < w; i++) {
38              for (int j = 0; j < h; j++) {
39                  Color c = new Color(bi.getRGB(i, j));
40                  rHist[c.getRed()]++;
41              }
42          }
43
44          //Computes the max of the histogram
45          double rMax = 0.0;
46          for (int i = 0; i < rHist.length; i++) {
47              if (rHist[i] > rMax) {
48                  rMax = rHist[i];
49              }
50          }
51
52          //Normalizes the histogram
53          int maxNorm = 20;
54          long[] rHistNorm = new long[256];
55          for (int i = 0; i < rHist.length; i++) {
56              double n = rHist[i] * (maxNorm / rMax);
57              rHistNorm[i] = Math.round(n);
58          }
59
60          //Prints the histogram
61          for (int i = maxNorm; i >= 0; i--) {
62              String s = "";
63              for (int j = 0; j < rHistNorm.length; j++) {
64                  if (rHistNorm[j] == i) {
65                      s += "+";
66                  } else {
67                      s += " ";
68                  }
69              }
70              System.out.println(s);
71          }
72          System.out.println("");
73      }
74
75      public void dump(BufferedImage bi) {
76          int w = bi.getWidth();
77          int h = bi.getHeight();
78          int cpt = 0;
79          for (int i = 0; i < w; i++) {
80              for (int j = 0; j < h; j++) {
81                  if ((cpt % 10) == 0) {
82                      Color c = new Color(bi.getRGB(i, j));
83                      System.out.println("[" + i + "," + j + "] = [" + c.getRed() + "
                            ," + c.getGreen() + "," + c.getBlue() + "]");
84                  }
85                  cpt++;
```

```
            }
        }
    }

    public void save(BufferedImage bi, String format, String filePath) {
        try {
            File outFile = new File(filePath);
            ImageIO.write(bi, format, outFile);
        } catch (IOException e) {
            System.out.println(e.getStackTrace());
        }
    }

    public static void main(String[] args) {
        ImageReaderWriter rw = new ImageReaderWriter();
        BufferedImage img = rw.load("../../Images/lena.jpg");
        rw.dump(img);
        rw.histogram(img);
        rw.save(img, "bmp", "../../Images/lena.bmp");
    }
}
```

B Partie 2

Programme C++ à compiler pour linux (et mac) avec :

```
g++ -o hello_word.exe hello_world.cpp -O2 -L/usr/X11R6/lib -lm -lpthread -lX11
```

```
// Include CImg library file and use its main namespace
#include "CImg.h"
#include <iostream>
using namespace cimg_library;

int main(int argc,char **argv) {
  // Display program usage, when invoked from the command line with option '-h'.
  cimg_usage("Read an image, dump rgb values for each pixels, and save the image in bmp ");
  // Program options
  const char* file_i = cimg_option("-i","../../Images/lena.jpg","Input image");
  const char* file_o = cimg_option("-o","../../Images/lena.bmp","Output image");

  // Load an image
  CImg<unsigned char> image = CImg<>(file_i);

  // Create a display window
  CImgDisplay main_disp(image,"Input Image");

  // Enter event loop. This loop ends when the display window is closed or when the keys 'ESC' or 'Q' are pressed.
  while (!main_disp.is_closed &&
   !main_disp.is_keyESC &&
   !main_disp.is_keyQ) {
    cimg::wait(20);
  }

  // Navigate pixels
  // for (int y=0 ; y<myImage.dimx() ; y++){
  //      for (int x=0 ; x<myImage.dimy() ; x++){
  //              ...
  //      }
  // }
  cimg_forXY(image,x,y){
    std::cout << "[ "<< (float)image(x,y,0) << "," << (float)image(x,y,1) << "," << (float)image(x,y,2) << "]"<< std::endl;
  }
  // Save the image in BMP format
  image.save_bmp(file_o);

  return 0;
}
```