Assignment 1: Image Pixelization

Overview

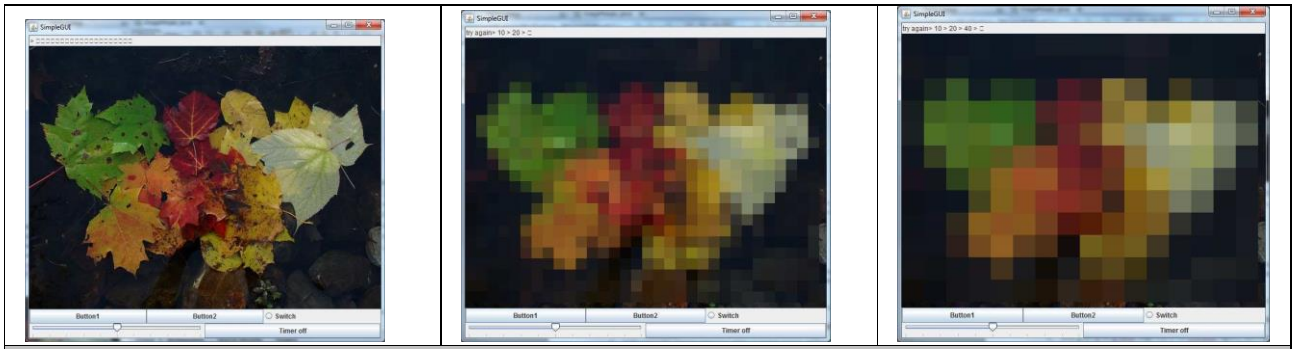
Welcome to the world of Image Processing!

Your task is to read an image, and create its "pixelized" version.
Pixelization is the image processing technique of lowering the image resolution.
How does it work?
The image is divided into non-overlapping square regions of same size. Each square is then represented by the average color of its original pixels (i.e. image dots, "pixel" stands for picture element).

Don't panic. You are able to do this with the knowledge you gained from previous courses, and with the appropriate tools to read and manipulate images.
The tool to do so is SimpleGUI, which is introduced in the lab.

What will you learn in this assignment?

- handling of the software package SimpleGUI, and working with it
- basic object handling (using methods of objects in SimpleGUI)
- to model software from a non-trivial task description
- lots of nested for-loops (just look at the description above: for each square... for each pixel...)
- splitting a seemingly large task into smaller, very simple parts
- creating software that is of actual use. Or at least enjoyable.



Pixelization: left: original image, center: pixelization with grid size 16, right: pixelization with grid size 32

I admit this task seems ambitious for a first assignment. You will get help in the lab and in class. Also, SimpleGUI contains demo programs dealing with image processing (yes, we start with the last section in the manual). Have a look at these demos.

Your task in steps.

1. read an image of your liking (no bonus points for pets, cats on roomba, or faces of faculty members)
2. create a SimpleGUI object of appropriate size and show the image
3. With g being the gridsize (edge-size of the pixelated region), starting with row=0 and column=0, step through every g-th row and column (nested for-loops!) and draw a filled square (SimpleGUI: drawFilledBox(.)) of edge size g with a certain color C . If we would know that color, we would be done. For now, set the color to black. Hence, for now, after this step the image is covered with black squares.

4. write a method to determine the correct color. It is the average color of all pixels of the square-image region starting at (row,column), edge size g.
5. in step 3, set color C to the color you computed in step 4.

That's it.

Detailed Description

1. read an image

We assume that you have created a Netbeans project for this assignment, and have added the SimpleGUI jar-file to it. The manual (section 1) explains how, you will also learn this in the lab. SimpleGUI contains a class that handles images. It is called "DrwImage". To read an image (just read, nothing is displayed yet), instantiate an object of this class, with the image-filename as parameter (e.g. DrwImage im = new DrwImage("myFirstPet.jpg")). The object "im" holds the image information, its methods allow you to retrieve this information.
Every digital image is a rectangular arrangement of pixels with certain color values, see the figure below:

Right: magnified view of marked area (left): the image is a collection of pixels.

The pixels are defined solely by their color (apart from their position, of course). The color is usually described using the RGB model, i.e. each color is a sum of a red, green and blue part.
In SimpleGUI, you can retrieve the rgb-value of a pixel at position (x,y) with the command:

```
RGB rgb = im.getPixelRGB(x, y);
```

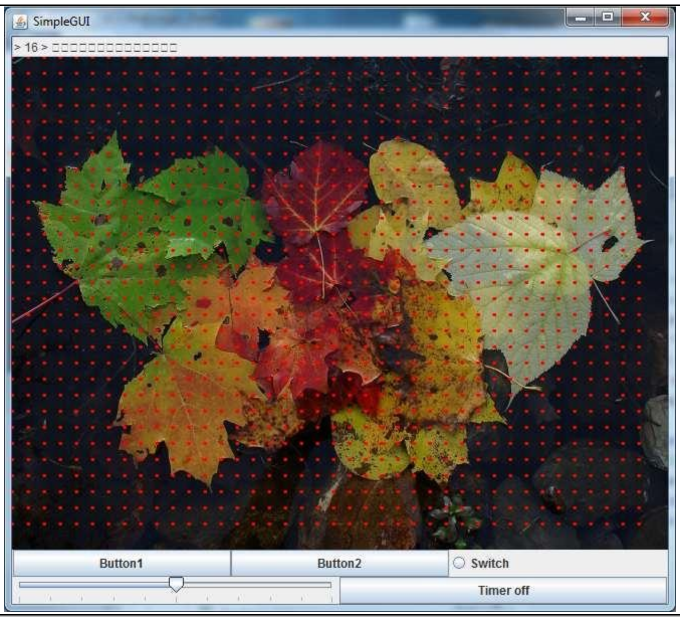
(where im is your previously instantiated DrwImage object). The RGB object contains fields for red (r), green(g) and blue(b), each of them integers in 0..255.

2. Create a SimpleGUI object of appropriate size and show the image

You need to get width and height of the image (im.getWidth(), im.getHeight()), and call the SimpleGUI constructor with these values. That's it. You should see your image!

3. Step through the Image, hitting every g-th row and column. This is obviously a nested for loop, creating the row indices (0, g, 2g, 3g, ..., ng<im.height), and the respective column indices.

TEST YOUR FOR LOOP by plotting a dot on each of your positions! (sg.drawDot(x,y,1)); This should look like the following image:



Testing your for-loops: print some dots.

But you don't want to faint in the middle --- here comes the pixelization. Now that we know that our positions are correct, we can draw boxes of size g instead of the dots. If we knew correct box color, we would be done.

4. Determine the correct color for each box.
Write a method (yes! do NOT fill this into your nested loops directly, the code would become ugly. Ugly code: -2 points!) that takes as input the row and column, and returns an RGB object with the required color in its fields r,g,b. To determine the color, you need to step through each pixel of the the square starting at (column,row), ending at (column+gridsize-1, row+gridsize-1). That's another nested for loop! For each pixel you hit, get the RGB value. Compute the average of R,G and B of all these pixels, and return it.

5. Use the color in step 3. You are done.

Enjoy, good luck!

The assignment will be due at 11:50pm, Thursday, September 3.