University of Burgundy

Computer Science project

PIXEL ART using QT

Danie Jianah SONIZARA

supervised by
Pr.Yohan FOUGEROLLE

Centre Universitaire Condorcet - Le Creusot
May 23rd , 2017

Contents

1 Introduction

The way to "Pixel Art " an image also called photomosaic of an image is obtained by assembling a large number of smaller unrelated photographs called tiles, so that each tile approximates a small block of the image.

Objectives

The aim of this project is to propose and implement a Qt/C++ application related to color in the general sense. The project is composed of a common set of tasks which concern the Pixelisation of an image and the Pixel Art rending. The common functionalities of the project can be summarized as follows:

Pixelization of an image:
Pixelise an image, requires:
- The ability to load and display an image located anywhere on the hard drive, and to save any processed image.
- The provided software should allow to transform the loaded image into a second one such that the pixels color of the second image is computed according to various methods (average, median, most represented color, etc.) so that the image is pixelized. For instance, the image below shows an example of the expected result :

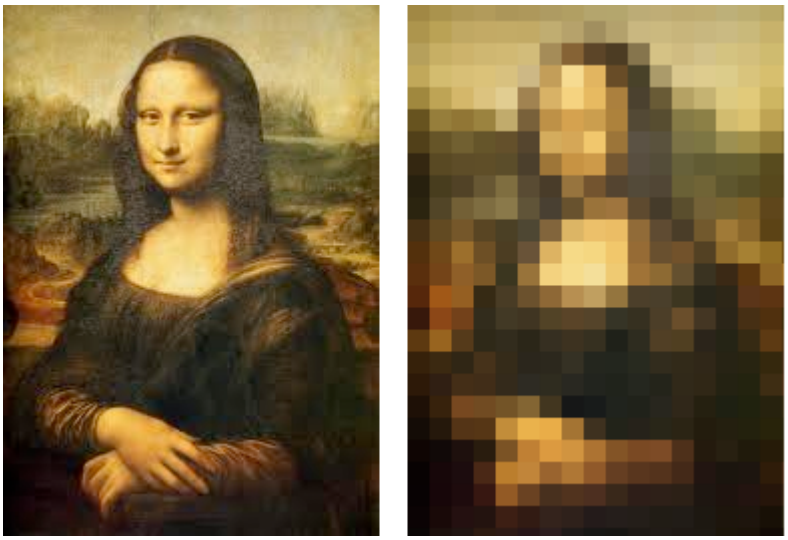

Figure 1: Example of pixelised image

Pixel Art rending:
In this context, the pixelized image has to be transformed into a third one in which its big and blurred pixels are now represented with images from a set of images of our own choice as many images as we want. The only restriction being that these images should not be included as resources of the project, so that the application can automatically load any database of provided images from your the without recompiling then entire program.

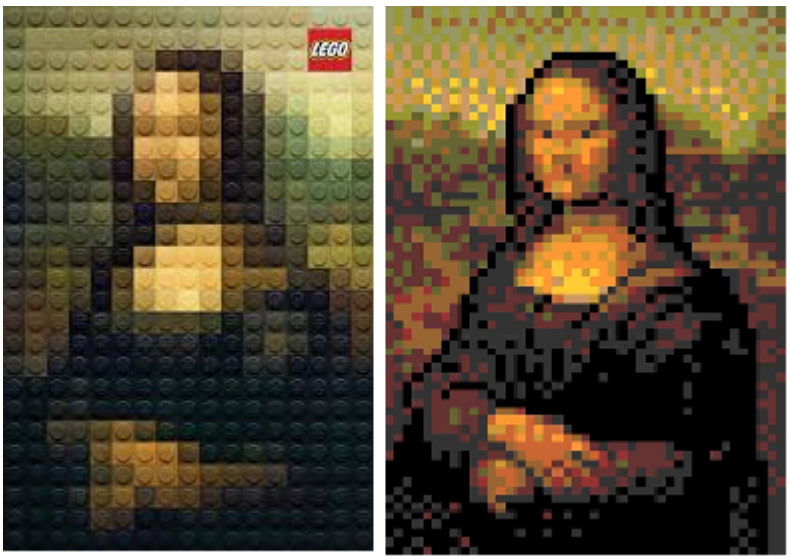

Figure 2: Example of Pixel Art using Lego and and set of images

2 Pixelization

2.1 Load and Display an image

First of all, we need to be able to load an display an image underqt because it is mandatory for the restof the project.
For that we use a QFileDialog and we assign a path to it, with this command we will be able to load an image from any location of the computer.
Carefull when displaying the image : be sure to keep size of the original image and set into QPixmap the you want to display.

Below the sample of original image that we will use:

Figure 3: Original image

2.2 Working with sub-images

In this part, in order to divide our image into many blocks of small images, we assign a value which will be the dimensions of one block.
For example if we assign a pixelationAmount =5, it means that we want to split our image into many blocks of 5*5 matrix.
Next we use 4 for loops :

- a for loop to go through every pixel of the x coordinate of the image (Width);
- a for loop to go through the y coordinate (Height);
- a for loop to go through every pixel of the xx coordinate of the block (Width of the block = 5);
- a for loop to go through every pixel of the xx coordinate of the block (Height of the block = 5);

In the 2 last loop we calculate the value of the colored pixel. Then we calculate the mean of each RGB channel ny dividing it by the dimension of the blocks

Finally we set setPisel the image before displaying the final result which we can see in the figure 4 . .

3 Pixel Art rending or Photographic mosaic

A photographic mosaic is an image that is created from many smaller images. The effect is to recreate a picture by replacing small portions of that picture with another images (which we will call tiles) that has the same average color. At a distance, the mosaic will look like the original picture, and ,the individual tiles can be seen. An example of a photographic mosaic can be seen in the result we will see later. Using the notion of sub-images discussed earlier, we can split a picture into sub-images and compare each ones average color value with a tile-set. The original pictures sub-image will be replaced with a tile from the tile-set that has the closest average color value. While it should give you a good result, typically mosaics that contain a tile-set with a theme have a more profound impact artistically.

3.1 Formatting the tiles

In order to make the mosaic, the tiles size should be adjust into the size of the sub-images or the blocks (5*5 matrix for our example).
Before resizing all the images, we need to load them at the same time. For that we use a vector, so then we do the same task again again inside a for loop.Use the same for loop to resize all the set of images.

3.2 Getting the best matched color

Once the formating finished, we do again the 2 first loop like we did before for the pixelization. The changes comes with the third and the fourth loops which are to go through each sample image size. As before, we calculate the RGB channels in order to get the means after.
The aim of this part is to get the best matched color from the tiles. To do that we need to calculate the difference between the RGB colors of the original image and the means of the RGB channels from the tiles. The expected result is to get the nearest image corresponding to the color.

4 Result

In the figure 4 as we can see, each blocks of 5*5 display only the mean value of the 4 channels (red, green ,blue). It is the result of the pixelization of the original image.
In the figure 5, all the block of pixel cube were replaced by the best image similar to the color of the pixel inside the original image.

Figure 4: Result of the pixelization of an image

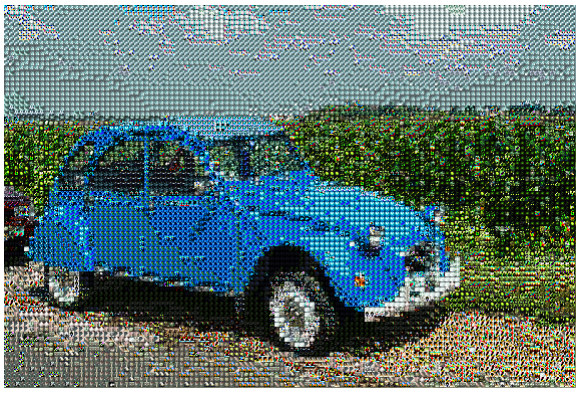

Figure 5: Result of the Pixel Art

5 Conclusion

The aim of this project was to create a mosaic image of a pixelized image from a set of images. These objectives were achieved despite some imperfections. My code is usable under qt. By a user having knowledge of the operation of the software. The code will be improveable since there are still point to be reviewed.

6 Reference

http://ssli.ee.washington.edu/courses/ee299/labs/Lab4.pdf http://urbanhonking.com/ideasfordo photomosaics-in-processing/

List of Figures