Computer Science - QT and C++

PIXEL ART Project Presentation

Olivier AGBOHOUI

Centre Universitaire Condorcet - Bachelor in Computer Vision - BSCV 2016/2017

May 22, 2017

Overview

This presentation will describe how to create an pixel art application in Qt.

- Load, display and save an image locate anywhere on a computer hard drive.
- Pixelised an image using the average method.
- Load any database of provided set of images from a drive find the best match sample image using the summation method and compute a PIXEL ART application.

Project Purpose

The main purpose of this project is:

- Ability to develop graphical applications using Qt, C++ and good oriented object practices.
- Focus on OOP practices in particular constructors, destructors, access rights, inheritances, templates and classes.
- Focus on simplicity of the interface.
- Focus on your ability to use numerous Qt widgets and objects and be illustrative or your ability to construct rich GUI applications with menus, drawing, shortcuts, tabs, proper signal and slot connections, etc.
- Be fully commented and documented for clarity and conciseness.

What is Pixel Art ?

Definition

Pixel art is a form of digital art wherein images are created and edited at the pixel level using a graphics editing software. What defines pixel art is its unique visual style, where individual pixels serve as the building blocks that make up the image.

Load save and display an image

Project Structure - Part 1

- Design and draw my interface.
  - Create tab widget to save and display my pixelised image.
- Include all directives needed to load and display my image.
  - In the header file define all attributes.
  - In the Mainwindow class define the behaviour of the load and display image process.

Include Include Qt standard libraries

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

// Libraries
#include <QMainWindow>
#include <QVBoxLayout>
#include <QScrollArea>
#include <QFileDialog>
#include <QLabel>
#include <QMenu>
#include <QAction>
#include <iostream>
```

Include Include Qt standard libraries

In Qt these are in-built libraries which are already coded and ready for usage .

- The QVBoxLayout class lines up widgets vertically.
- The QScrollArea class provides a scrolling view onto another widget.
- The QFileDialog class provides a dialog that allow users to select files or directories.
- The QLabel class provides a text or image display.
- The QMenu class provides a menu widget for use in menu bars.
- The QAction class provides an abstract user interface action that can be inserted into widgets.
- The QTabBar class The QTabBar class provides a tab bar, e.g. for use in tabbed dialogs.

Pointers

// Define pointers for every action to be taken later.

```
// Define attributs for action
Ui::MainWindow *ui;
QScrollArea *scroll_original, *scroll_final;
QLabel *label_original, *label_final;
QMenuBar *menuBar;
QMenu *menuFile;
QAction *actionOpen, *actionSave, *actionExit;
```

Define functions and syntaxes to load an image

```
public slots:
// define function that will load my image
void load_image();
```

mainwindow.cpp

```
//Constructing labels
label_original = new QLabel();
label_final = new QLabel();
//Constructing scrollarea
// Same with the scroll_original
scroll_original = new QScrollArea();
scroll_original->setWidget(label_original);

//Constructing Menus
menuBar = new QMenuBar();
// New menue file
menuFile = new QMenu();
// create a new menu file
actionOpen = new QAction();// to be added to my menu
 menuFile->addAction(actionOpen);
```

How to connect my Menu to my load Image function

```
// Define Menu options
connect(ui->actionOpen, SIGNAL(triggered()), this, SLOT(
    load_image()));
```

Load image function

```
// Load and display image function
void MainWindow::load_image() {
// The QFileDialog class provides a dialog that allow
    users to select files or directories.
// So a variable filename of type QString is created in
    which the process of getting the image in
// the directories is saved
QString filename = QFileDialog::getOpenFileName(this, tr(
    "Open"),
QDir::currentPath());
```

How to display my image

```
// Display the loaded image
// The QPixmap class is used for painting the Image in
    the predefinded display widget .
ui->label_original->setPixmap(QPixmap::fromImage(image));
ui->label_original->adjustSize();
```

Fine ! Let us run our program

Interface output

Project Structure - Part 2

- Second Important step : Define in the class " improcess " a function to pixelised an image.
- Process image pixelation.
- Paint pixelised image in display widget.

Illustration

Now that we can upload an image . It is time to find a way to pixelised it.

Let us start !!!!!!!

Header file

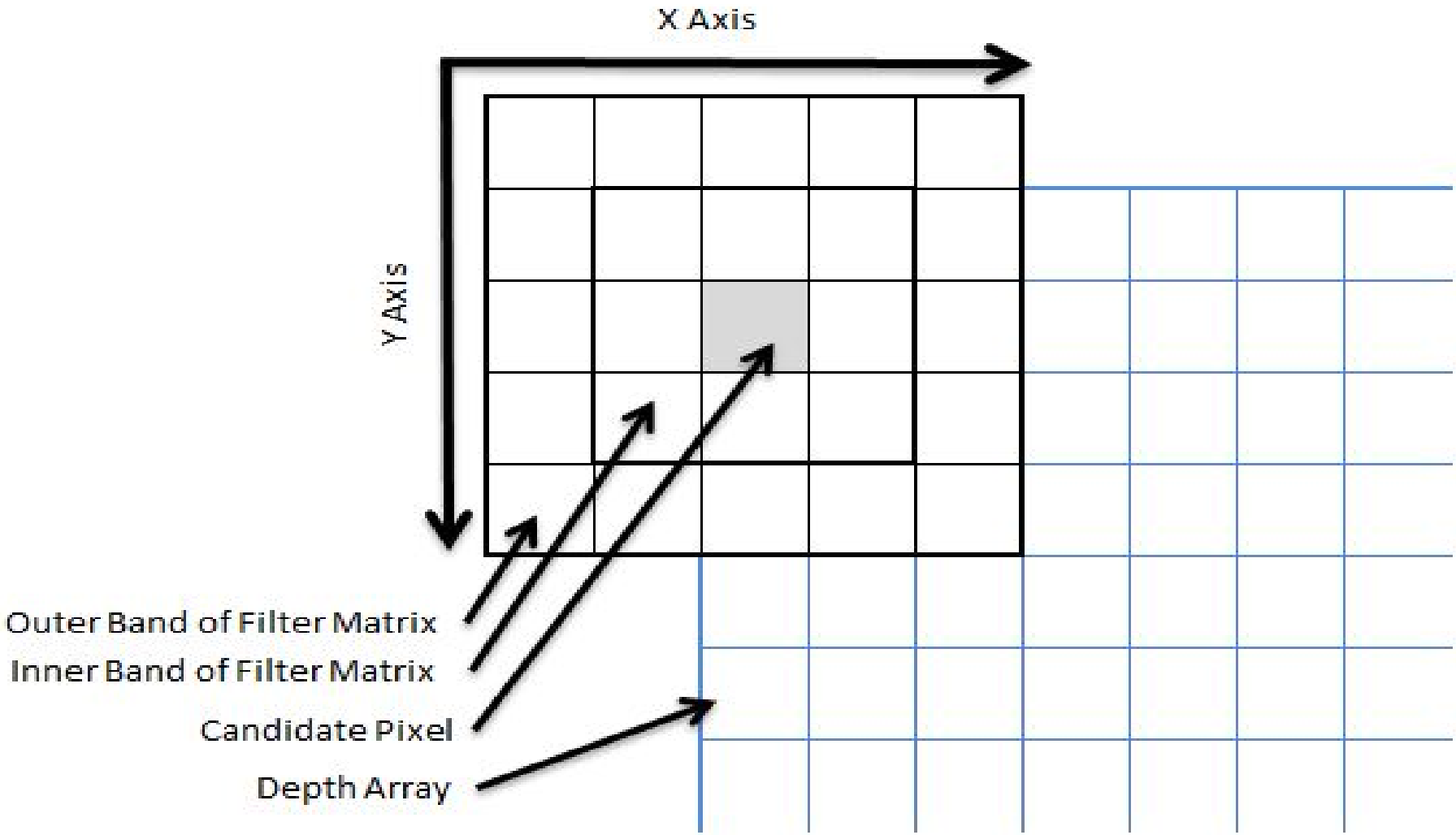
```
// Defining a class improcess for the pixelisation
    process
class imgprocess
{
public:
imgprocess();
~ imgprocess();

public slots:
QImage pixeliseFunction (QImage img, int window);
};

#endif // IMGPROCESS_H
```

Average method of Image Pixelating

Pixelating an im-
age consist of dividing an image into pixel so that it is break up into visible pixels.

The Average method consist of parsing the uploaded image and collecting information of this image which will later be used to compute the average of each pixel in the Image. Pixel in the image will then be replaced by their average pixel value to obtain my new pixelised image.

How the mask move over my image



The mask goes over each pixel in an image and collect information.

Average method

The average method, consist of retrieving all the pixels values of the neighbourhood pixels, calculate the average value and replace the main pixel by its average.

What happen in my Pixelation function?

This function is composed of 3 for loops.

- The first one define an array where the 3 RGB pixel values are saved, and a variable to save the total number of pixel in the image.
- The second one walk on the image and compute the average of each pixel value on the image.
- The last one change each pixel value on my image by its average value.

Pixelation function

```
QImage pixeliseFunction (QImage img, int window);
```

Please refer to the program to see the behaviour of the pixelation process.

Waoh ! Let us run our program

Interface output

PIXEL ART rendering

We are now ready to move to the most important part of this project which is the implementation of the pixel art.

Project Structure - Part 3

- Describe the behaviour of the pixel art function.
- Define the process for finding the best match image in folder.
- Display and save the pixel art image.

How it works ?

The pixelation process is defined as follow:

A statement will be define to go over each pixel block in an image and replace them by their best match images in a set of images using the absolute difference summation method.

- Extract the mean colour of the sample image.
- Calculate the distance of the mean colour of the sample image and the mean colour of the original image.

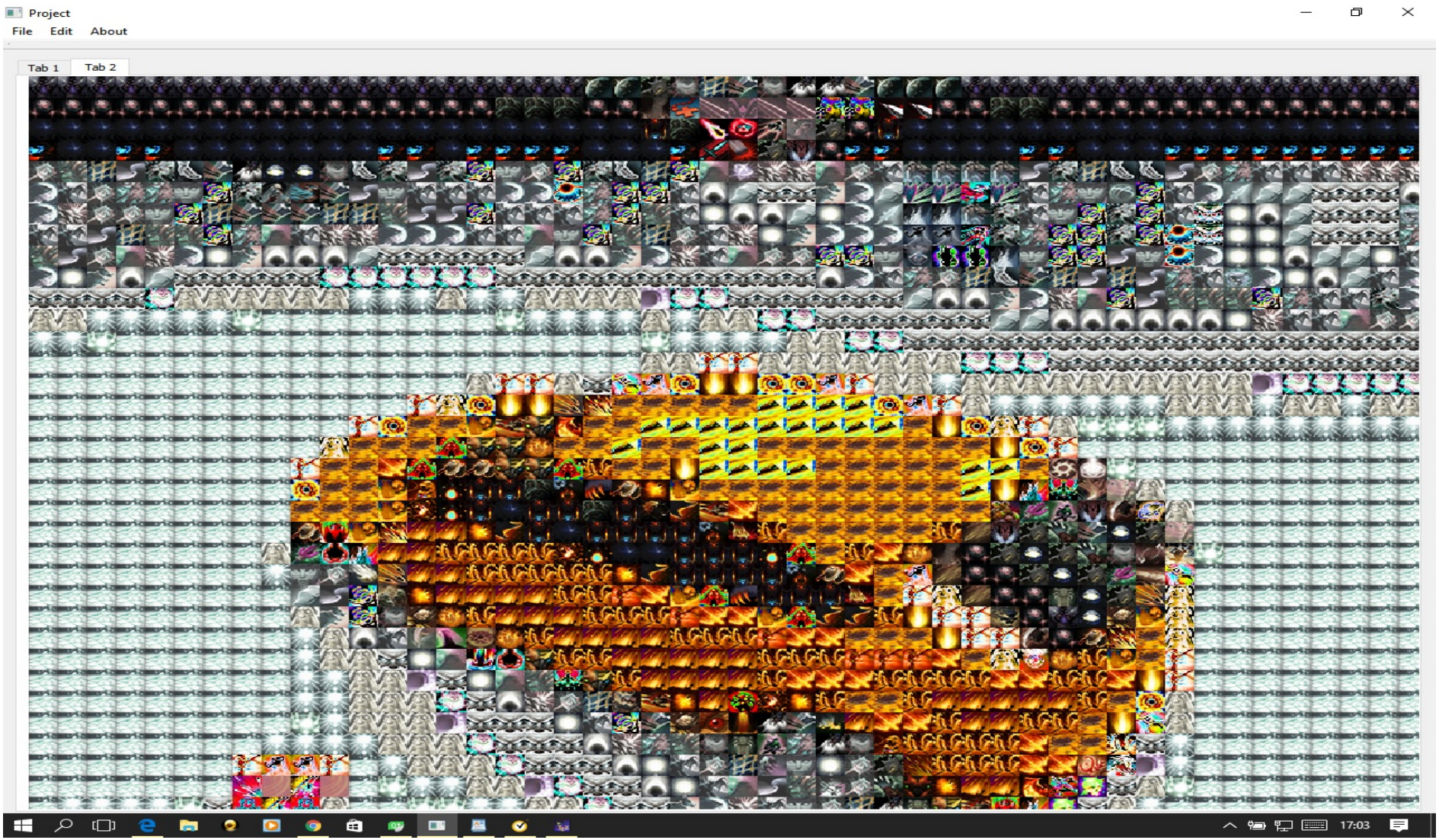
Knowing the summation value we can compare and get the best match sample image.

Functions definition

```
QImage bestMatch (std::vector<QImage> vecImages, std::
    vector<double> distance, int window, QImage
    img_original);

QImage pixelArt (QImage img_original, int window, std::
    vector<QImage> vecImages);
```

Please refer to the program to see the behaviour of the best-match and the pixel art process.

Pixel art image



Final program

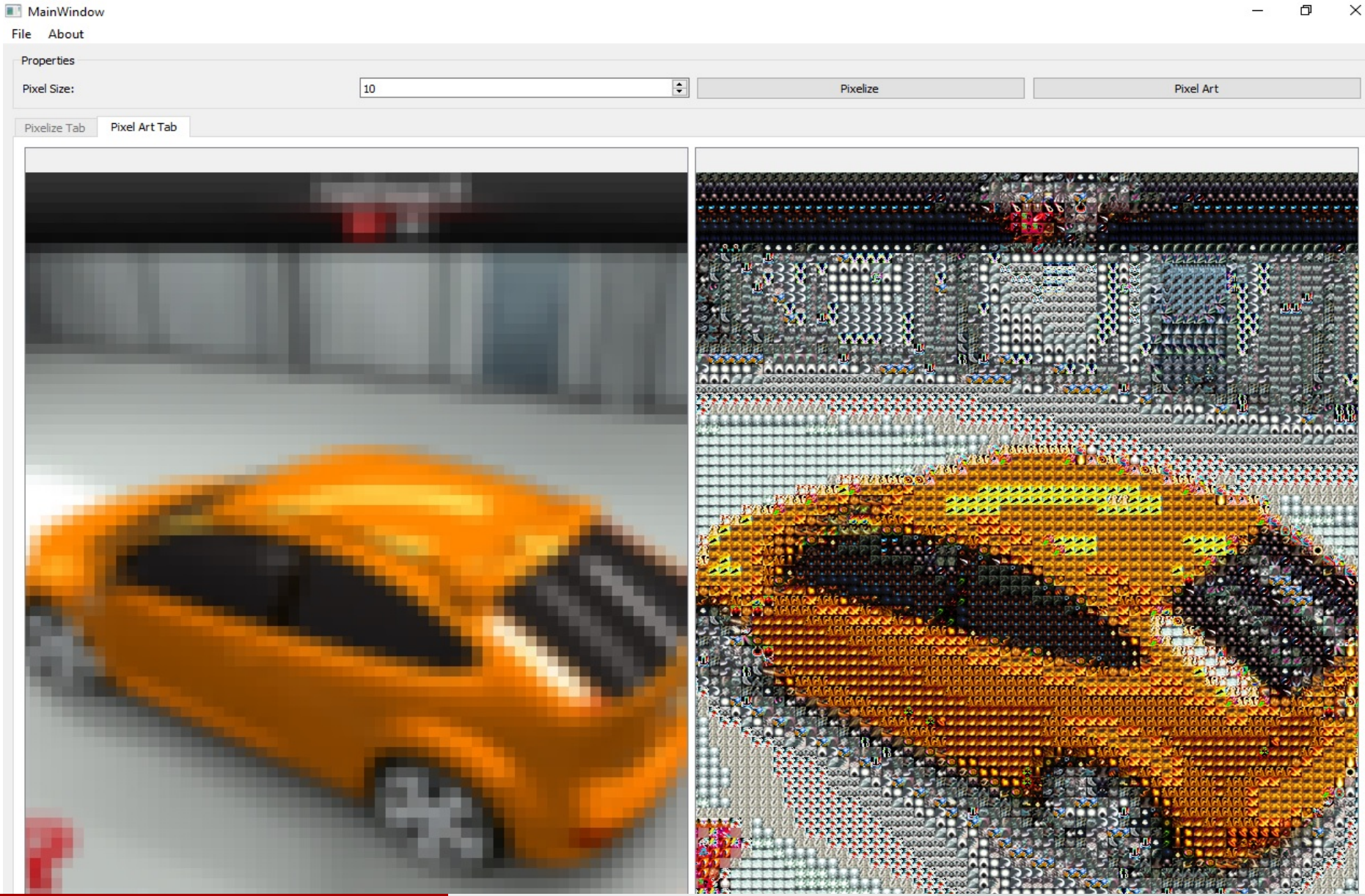
After the understanding of the pixel art process. Few changes has been added such as:

- A spinBox widget to input pixel blocks size.
- A pixelisation button.
- A Pixel Art button.

Few changes

Apart from using the "Go to slot function " and redefining my pixelisation and my pixel art functions in the mainwindow, the main change here is to add the spinBox to define the pixel block size and pass this value to my pixelisation function to set pixel cube size.

Final program



Conclusion

This project promotes the development of two pixel manipulation approaches: pixelize and pixel art. We presented information about how to pixelize an image and to create a pixel art. This program has not been fully optimize, the program takes sometime to run. An optimize version may be send later.

References

- https://github.com/search?utf8=
- https://github.com/agboliver
- //github.com/rafaelasouza/Pixelation
- http://en.cppreference.com/w/cpp/memory/c/malloc
- http://www.ijcaonline.org/archives/volume153/number10/swaroop-2016-ijca-912165.pdf
- http://stackoverflow.com/questions/8099591/variable-block-size-sum-of-absolute-difference-calculation-in-c
- https://www.qt.io/
-
- https://www.codeproject.com/KB/audio-video/317974/filter.JPG
- Doctor Yohan Fougerolle/ Centre Universitaire Condorcet/ Yohan.Fougerolle@u-bourgogne.fr