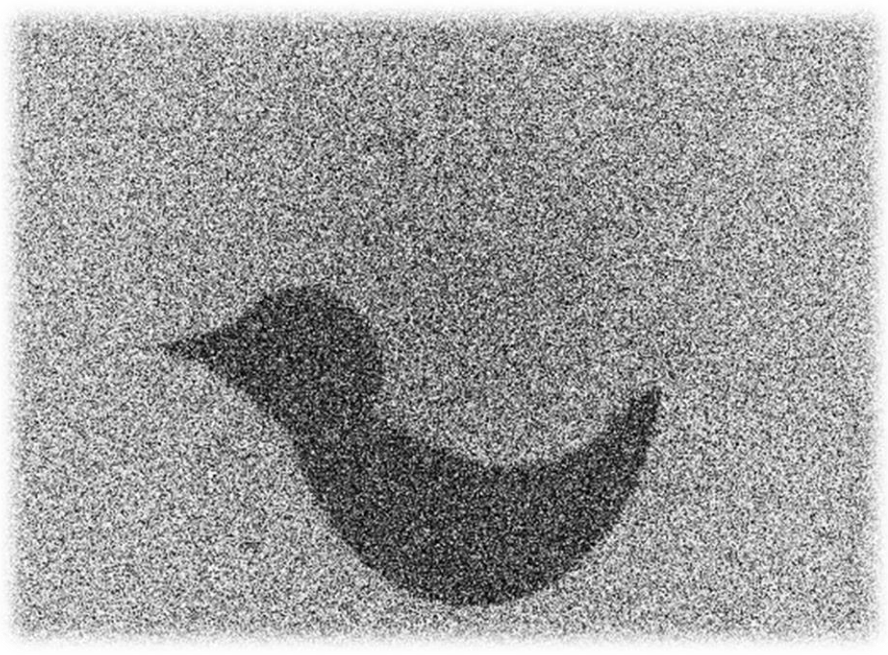
Documentation of the graded exercise in the course “Practical C++“

Author and Programmer: Marco Geils (Matr.Nr.: 375736)



Inhalt

1. General user guideline

Requirements to start the program in the QT Creator:

- You need to have QT Creator 5.7
- The Creator has to have the QT Charts module installed

When these requirements are met, the program should compile without an error and start. If this is not the case, like it was ion the development, a click on "clean" the project and then "build new" can help.

When the program starts up you can see the menu bar in the upper left of the opening window. The Options are split into "File", "View" and "Tools". When no picture is loaded only the menu points under the "File" menu are clickable. In the file menu you can choose between opening an image in various formats (I tested .png and .jpg). After the image is opened, the title and size is displayed in the status bar on the lower right corner of the screen, all the optional tools in the tool menu and view menu become available. In the view menu you can zoom in and out of the image, scale it to the normal size and scale it to the size of the applications window. In the tool menu you can perform calculations over its basic indicators and get a histogram for the image. Additionally you can also smooth and threshold the image, all you need to do is to set the N for that functions. Also you can choose to show the gradient image of the image by choosing one of the options under the "Gradient calculations" sub menu. Also there exists a tool for calculating size. This calculates the size of the darker areas in the image as I explain in an extra document. With that you can calculate the size of that bird like object of the picture shown on the first side.

2. General architecture and class overview

In general the structure of the program is not very deep. The main opens the main window which is defined in the window class. There everything according to the menu is build and linked with functionality. From there the different functions called through menu calls create the needed class, which then does the needed calculation or image manipulation. Existing classes are: *main.cpp*, *window.cpp*, *basicindicatorcalculator.cpp, histogram.cpp, imagesmoother.cpp* and *thresholder.cpp*. The size calculation is done in the window.cpp as well, because the algorithm is using only other classes like the smoothing and thresholding and also to keep the complexity as low as possible.

3. Basic indicators calculations

The basic indicator calculator gets the image. After that it can loop over the all the pixels and find the minimum and maximum as it compares it with the previous minimum and maximum. While it does that it also sums the pixel together for further calculating the average. With this it then calculates the biased standard deviation. This calculation was done using a step by step tutorial from http://www.wikihow.com/Calculate-Standard-Deviation. The program than shows the results as a message in the status bar and prints it out to the console. Normally I would provide a normal window or dialog to show this friendlier for the user. Why I accept such a shortcoming now you can see under point 10.

4. Histogram

The histogram also gets the image and loops over all pixels and counts up a previous created array at the specified point. When the histogram is created it gets shown in a bar chart which is displayed in a chart view for the user. The histogram gets automatically saved as a .txt file in the folder where the QT executable is located. In the real world this also would have to be done in a better way with a real user interface where the user can choose a location for the file.

5. Image smoothing

Image smoothing also gets the image first. It then creates another image where the pixels are replaced by an average from the NxN neighbourhood. The N is chosen by the user via a user input dialog in the window class. I decided to adjust even Ns to one number below, because even numbers in an NxN neighbourhood make the middle of this neighbourhood a 2x2 pixel area. I wanted to avoid that so even numbers get scaled down. Also the corners are just treated that way, that the pixels in the corners or at the limiting line of the image are just calculating the existing neighbours. After that new image is calculated the window class gets this image and replaces the previous image with the smoothed one.

6. Gradient calculations

The gradient calculation class is creating a new image for every gradient image. They get then created using the formula provided the lecture. It should be noted that I use the concrete colour instead of using change in Intensity. That creates images which are black and dark grey instead of greyish images you can find throughout the web. I decided to do that because in my understanding a change in Intensity should calculate the difference from the mid of 0 to 255 to the used colour. Because the new colour uses integer it would be off by half an integer. So I just used the difference between the colours itself. The borders of the pixel arrays are treated just like in the image smoothing class.

7. Thresholding of the image

This works as all the other classes which replace the actual image, with the difference that the new image is only two colours: black and white. The User also puts in an N and all pixels which are under or are that exact value get replaced with black and all others with white. That way a binary image is created and replaces the old image.

8. Size calculation

This calculation is explained in the “explanation of size calculation” document.

9. Known Bugs

In development I noticed some bugs in the program:

- It needs to be noted that I always get the red value when I ask a pixel what colour it is. That will be buggy if a user puts in coloured images and uses the tools.
- Some .png files a user can find in the web have a different format but are grey scaled. The program can show them and calculate the basic indicators as well as the histogram but cannot smooth or get the gradient image for those images. This happened to me with one picture I found in the web but has not happened once with any other grey scaled image.
- When there are more images opened the same session it puts the information from the new image next to the information from the old image in the status bar. This creates a bug where the window grows to the left as a user opens more and more images the same session.

10. Explanation of shortcomings

As you surly noticed some parts of my code could be made better by some adjustments and also the image size calculation is not one of the fanciest ones. Also some information could have been better shown in an extra dialog or some other interfaces in the program. Why are these shortcomings existent? I don't want to excuse myself, but I just want you to understand why my work will not be one of the best works you will see this semester. Normally a 6 cp course should need a maximum of 120 hours in the "Bremer Model". I had a course which took up an estimate of 400 hours this semester. This should have been undoable for me because I had another 5 courses to do (including this course). But somehow I managed to achieve all I needed. In the same time I had to care about an internship and because I got that, a flat in Berlin. Which also killed a lot of time (I travelled back and forth to Berlin because it was nearly impossible to get a flat with an 8 months "non-existent payment" contract) , that also killed around 120 hours (11 hours travel time for one visit / day where I looked at flats. The bus sadly was so uncomfortable that I was not able to work in there. While all that was going on one person who is very close and important to me got into the hospital… four times. Anyway, somehow I managed everything and I am still sane.

I just felt the need to tell you all of this, because I am not very happy with the quality of my work. But I hope you can understand it a little bit, that I do not have the time to look into details a lot, between all what's going on and that my work is enough for a 3 cp course.