Contents

1 Introduction

For the course 'Digital 3D Measurement, Computing and Design' we got an assignment to create our own program with Visual Basic. The first 3 weeks of the period we have had time to learn to work with Microsoft Visual Basic 2005 Express. We had a few small assignments to become familiar with the program. After these three weeks it was time to get to the point.

The final assignment was to create a working program to measure the length of a boat in a canal lock. We had 7 weeks time to finish the project. The basic goal of the project was to learn the basic principles of programming. This means to learn the basic syntax and get familiar with the structure to create a program. For us, as designers, this is important so if we get involved with programming in a design project, we are able to understand and communicate with professional programmers.

For this project we had to design two separate products, the Actual Product (AP) and the Ultimate Product (UP). The AP is the software product we eventually created; the UP is the product in its best form, as it should be as it was actually in use.

During our project we got assistance from a staff member, Tjamme Wiegers. That's how we got to name our project team "Tjamme's Angels". He gave us very useful advice every now and then, so therefore he was more of an angel to us.

1.1 Definition of the problem

Ships that pass a canal lock have to pay a certain fee. This fee is based on the length of the boat. This means that the length should be measured somehow to determine its fee. Because its labour intensive to measure the length with a normal ruler, the canal lock needs an automated system to measure the length. Our task is to design such a system.

Most of the canal locks are designed for more than just one ship at a time, which means that the system should be capable of calculating the length of more ships on 1 photo. Besides the boats won't lie in a straight line, so that will give problems during the calculation of its length. Furthermore the system should work for different kind and sizes of canal locks.

2 Design of the Software

2.1 Approach

First we had to decide on which aspects we going to focus on for our program. Our main goal is to create a program that will be easy to use. People without much knowledge of computers should be able to operate the program. Therefore the user interface of the program should be very clear, which means a few clear options with logic handlings so the operator is aware of what he is doing and fully understands what he's doing.

Secondly the program needs to locate the ship. Because programming languages can only operate on objects that you have previously defined, we first have to 'explain' to the program how it can find a boat on the picture.

A bitmap picture essentially consists of pixels. These are the building blocks of the picture. Our program needs to make a distinction between which pixels belong to the boat and which don't. The picture needs to be converted into a contrasted image of 2 colours, in which all the pixels belonging to the boat are white and the environment consists of black pixels.

Assuming we are able to convert into a contrasted image of 2 colours, we can run a check on all the pixels in the picture and split the white pixels from the black pixels. Next the program needs to locate the two outer white pixels so we have the front and rear of the boat. Then the program counts the pixels in between the two outer pixels, so we have the size of the boat in pixels. Now it's only a matter of translating the pixels into meters.

Initially we will try to create a program that can measure the length of 1 ship that lays straight. If this works we will try more complicated pictures, with more ships or ships that don't lay straight.

After completing the program we will evaluate our version. We will let some people use the program and see if this gives complications or useful advice. If so, we will need to revise the program, to let it work as optimal as possible.

2.2 Requirements

Before we could start designing a product we had to set ourselves some requirements as guidelines for the project. Some of these requirements are essential and some are optional. We were aware of the fact that we wouldn't be able to fulfil all requirements in the AP. That's why we first thought of all possible requirements and then sorted them in requirements for the AP and UP.

2.2.1 Overall requirements

- Ability to locate a ship on a full colour picture;
- Ability to measure a ships length;
- Ability to calculate its price;
- Ability to upload a new picture;
- Ability to locate 2 or more ships;
- Ability to measure the lengths of 2 or more ships;
- Ability to calculate the prices of 2 or more ships;
- Ability to adjust the price per meter;
- Ability to adjust the camera height;
- Ability to locate slanted laying ship;
- Ability to measure the lengths of a slanted laying ship(s);
- Ability to calculate the prices of a slanted laying ship(s);
- Ability to locate 2 or more overlapping ships;
- Ability to measure the lengths of 2 or more overlapping ships;
- Ability to calculate the prices of 2 or more overlapping ships;
- Clear interface;
- Easy to use.

2.2.2 Requirements for AP

- Ability to locate a ship on a contrasted image;
- Ability to measure a ships length;
- Ability to calculate its price;
- Ability to upload a new picture;
- Ability to locate 2 or more ships;
- Ability to measure the lengths of 2 or more ships;
- Ability to calculate the prices of 2 or more ships;
- Ability to adjust the price per meter;
- Ability to adjust the camera height;
- Clear interface;
- Easy to use;

2.2.3 Requirements for UP

- Ability to convert a full color picture into a contrasted image;
- Ability to measure the lengths of a ship laying in every possible orientation;
- Ability to calculate the prices of a ship laying in every possible orientation;
- UP has to be one integrated system;
- UP has to be extremely easy to use;

3 Actual Product

The item we were most focused on during this project was the Actual Product. We had to create a working program that could approach the ultimate product as it could be used for real. Before we could start on programming the program we had to define a few things. First we were bound to certain limits, because in the time we were given we couldn't create an "ultimate" product. After we defined these limits we could start designing the product. The technical documentation of the AP explains how our program works and discusses its interface. At last we had to evaluate the program, as explained in the evaluation on AP.

3.1 Research on Practical Boundaries AP

For creating the actual product we are limited in several ways. Therefore we need to consider what is possible for our AP. First of all as said it's the actual product which we will design and not the UP (ultimate product) which should be the system that could actually be used in a canal lock for measuring a boat's length.

The UP is a system that involves a camera for taking a picture, the program that converts a full colour picture into a contrasted image and calculates the boat length(s) and price(s), and places the output (the boat price) on a screen.

The AP is actually only the program itself that imports the contrasted image into an array and calculates the price for each ship. For our AP we use a pre-written program that converts a picture from a bmp-file to a ASCII-file. In the UP this piece of programming is integrated in the complete software.

So in a way we are limited in designing our product, because we don't have an existing camera (as described in the UP) that takes pictures of boats in the canal lock. Neither we have written the software ourselves that converts the picture into a readable file for our program.

Furthermore time is a very big issue; we have a total of 7 weeks to complete this project, so we can't do all the things we'd like to do or could do. Image 6.1 is a schematic representation of the borders of the program as we concieved them at the start of the project.

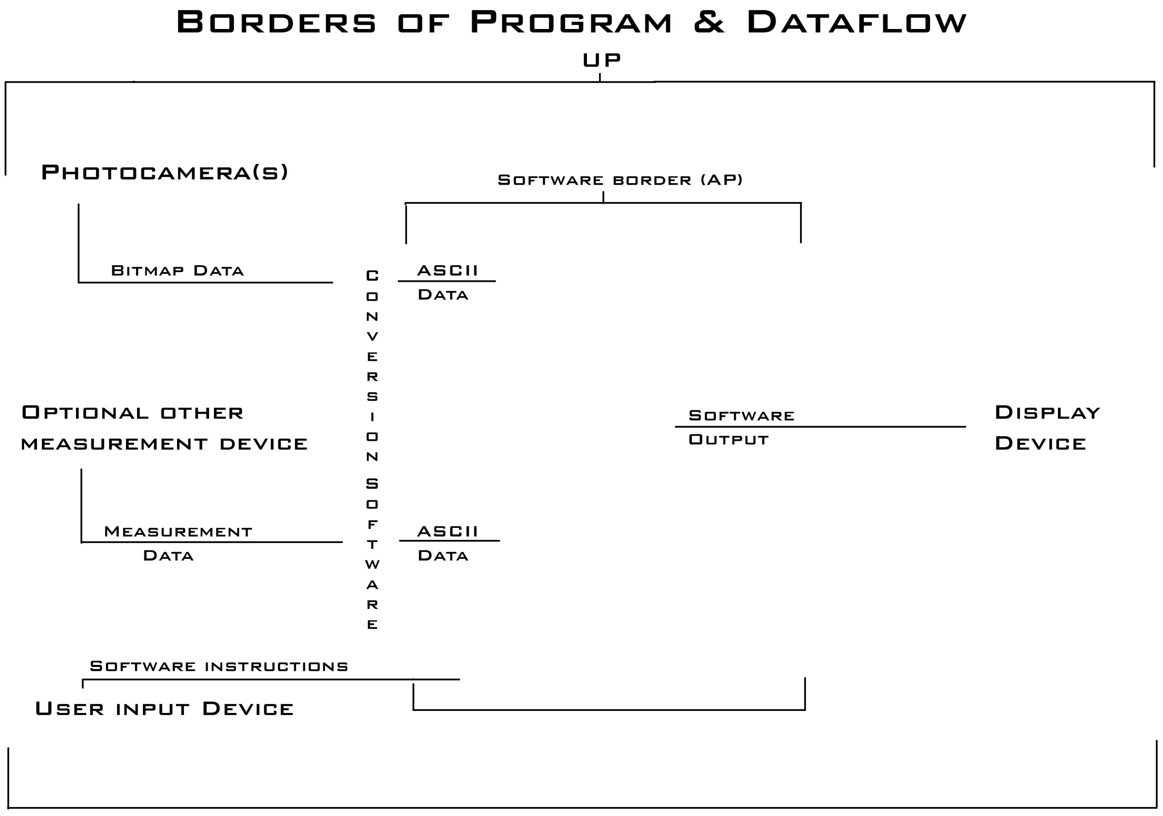


Figure 3.1 Borders of Program and Dataflow

3.2 Technical Documentation of the AP

This will describe the technical principles of the Actual Product; we will outline all the essential steps of our program. Such as input, output, user interface, the main variables of the program and its methods.

This is basically how the program works; it imports an ASCII-file into an array and reads the array for white pixels. It searches for the pixel with the smallest x-value and the largest x-value and counts the number of pixels between these two x-value's. These pixels are translated into meters and the price of the boat can be calculated. Below you will find an expanded version of our program worked out to every detail.

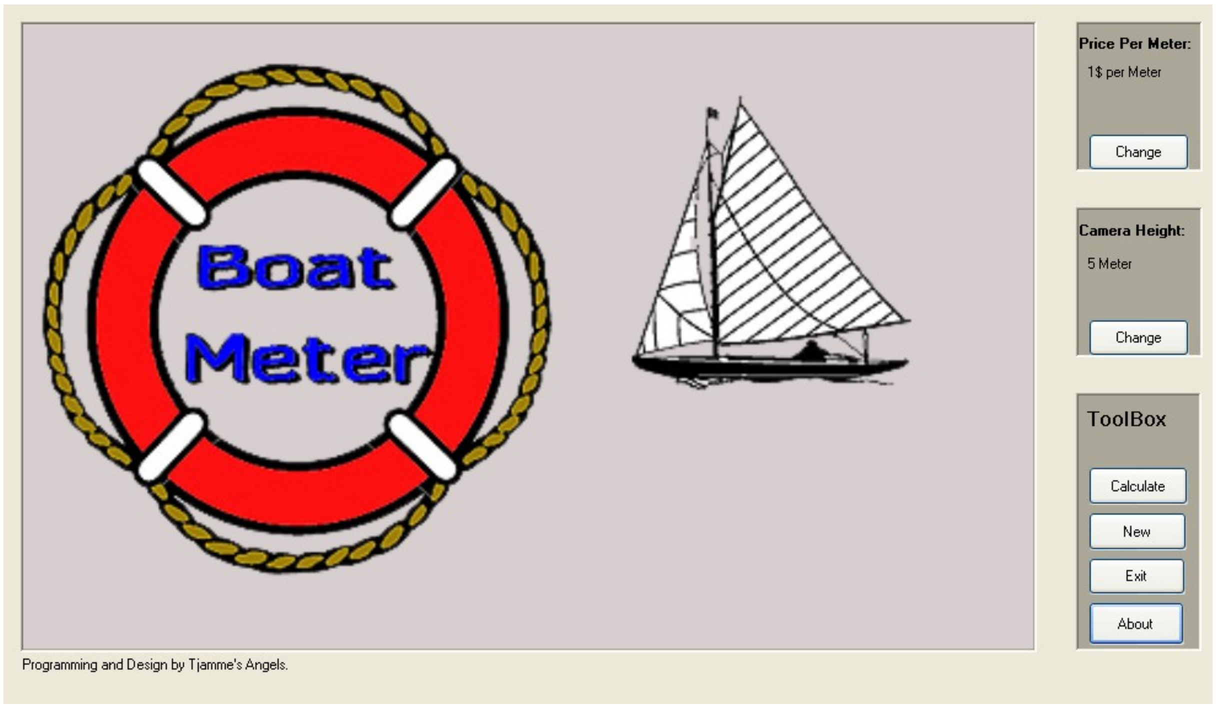


Figure 3.2 Standard screen of boatmeter

3.2.1 Handling pictures containing one boat

The input of our program is a contrasted image. This image is converted into an ASCII-file. This means that the picture as seen in pixels with different color values is converted into text file. So basically it's a text-file with the information on the actual (visible) image. The program used to convert a bitmap to ASCII has been provided by the project staff. The ASCII file contains a list of numbers, arranged in five rows. The first row represents the x-value of the pixels the second represents the y-value of the pixels and the final three rows contain brightness values for the colors red, green and blue.

I The next thing we do is, we load the ASCII-file into a two-dimensional array. A subroutine will read each line of the ASCII-file and splits each line into individual strings, and stores them in the array. All the strings now will be converted into integers. **(*As defined in sub-routine Load ASCII)***

II All the strings stored in array will now be divided within 2 groups. All the lines with information on the white pixels will be stored in a second array (WhiteArray). Since the boat on the image is white, we only need the white pixels. Each pixel in the full array is checked on its color value. For white the color value is 255 for red, green and blue. Since we work with contrasted images that have only black and white pixels, we can easily check each pixel's third value to see whether it equals 255. if so, the pixel is white and it's x and y value will be stored in WhiteArray. In order to let WhiteArray be as large as the number of white pixels, we change the length of the array every time a white pixel is found. ***(As defined in subroutine Sort Arrays)***

III For each of the white pixels in the array the x-values are stated. So it's just a matter of comparing all the x-values in the array, to calculate its size. The program runs a check on all the white pixels to find the pixel with the smallest x-value. Same goes for the white pixel with the largest x-value. Now we are able to calculate the boat's length because we have the white pixel with the smallest x-value, which is the rear of the boat, and we have the white pixel with the largest x-value, which is the front of the boat. ***(As defined in subroutines ValGrootsteX and ValKleinsteX)***

IV In principle this is all the information we need to calculate the size of the boat. We take the smallest x-value (as found in ValKleinsteX) and subtract this from the largest x-value (as found in ValGrootsteX). The program calculates the number of pixels between the 2 outer pixels of the boat. For our program we've determined that the size of 1 pixel equals 0,10 meter, so to calculate the length of the ship we multiply the numbers of pixels with 0,1 to get the actual size of the ship. The actual size times the price per meter gives us the fare for using the canal lock. ***(As defined in subroutine Calculate)***

3.2.2 Handling pictures containing more than one boat

Considering the program has to work on locks, it has to be able to handle pictures containing several boats. In order to do this we first have to determine the borders we are going to use in the AP. We decided to restrict the options in the AP in a couple of ways:

- **The AP only needs to handle a maximum of 2 boats**
  (if we can make it for 2 boats, it takes little effort to expand it to more boats.)
- **The AP only needs to handle boats that are stacked in a vertical direction, without overlapping. Therefore the AP will only be able to handle boats that are at least one row of pixels (in y direction) apart.**
  (We chose the vertical direction, because we don't have the time to implement both vertically and horizontally stacked boats. However, finding two horizontally stacked boats works in the same way as with the vertically stacked boats. Only with x values instead of y values.)
- **The AP only needs to handle boats that are exactly leveled with the horizon**.

If we take these restrictions in consideration it gives us some simple methods of handling several boats.

I In principle the program works in the same way as it does for one boat until step 2 of handling pictures containing one boat. From that point it will change and follow the next 3 steps.

II First of all we want the program to determine whether it has to deal with one or two boats. To do this we have written a subroutine that looks at every pixel in the WhiteArray and compares the value of every second entry (which represents the y coordinates) with the second entry of the previous pixel. If the second entry is higher than the previous pixel +1 there is a gap in the white pixels and therefore there has to be more than one boat. If this occurs we change a new variable (NumberBoats) to 2,

so that we can use this to tell the program there are 2 boats.
As soon as this happens, all of the following pixels are placed in a second array. Now we have two separate arrays, the first containing all the pixels of the first boat, the second containing all the pixels of the second boat. ***(As defined in subroutine Boatcounter)***

III Next we can calculate the length of both boats in the same way we calculated the length of one boat. Because we have determined there are 2 boats in the picture in step 1, we can easily ad a couple of lines of code to the calculate subroutine that checks whether it has to calculate 1 or two boats and displays the correct length and price for one or two boats. ***(As defined in subroutine Calculate)***

Output: The output of the program is actually the pop-up screen that tells us the price to pay for the boat. Above you can read how our program actually calculates its length and price.

User Interface: the interface of our program is very important; the picture below shows an actual view of our program. All the options in our program will be discussed in this part.

As you can see the program has diverse options; listed under ToolBox: Calculate, New, Exit and About. Each of the function of these options will be discussed below, and for each option we will show a screenshot to show the given situation.

New: To upload a new picture. Pushes calculate will give you the length and the price of the new uploaded picture.

Calculate: This button will give you the calculation of the length of the ship and as well its price.

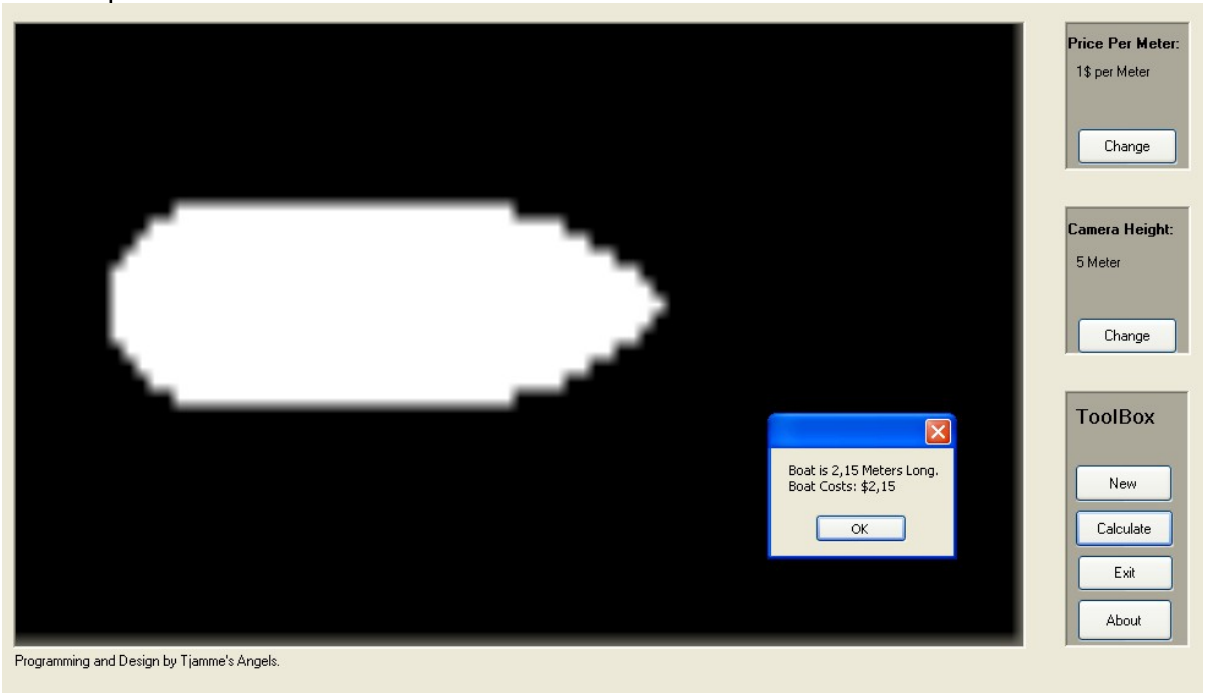
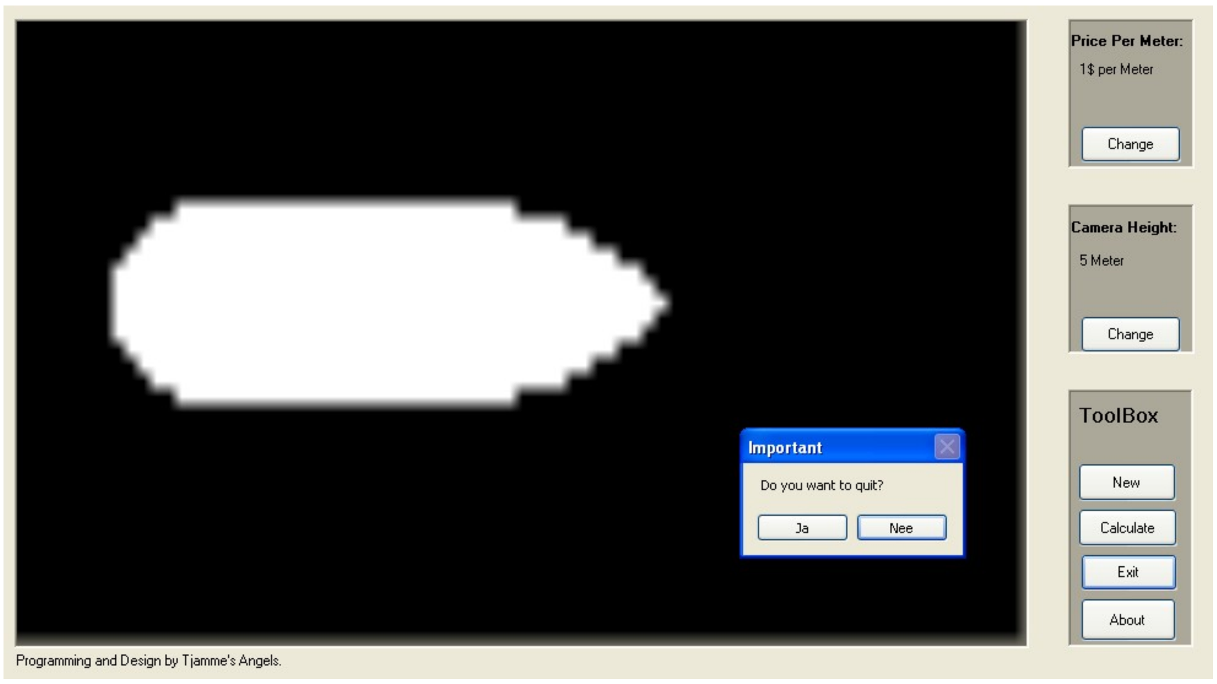

Price Per Meter:
1$ per Meter
Change
Camera Height:
5 Meter
Change
ToolBox
New
Calculate
Exit
About
Boat is 2,15 Meters Long.
Boat Costs: $2,15
OK
Programming and Design by Tjamme's Angels.

Figure 3.3 Calculate the length of the boat

Exit: To exit the program.

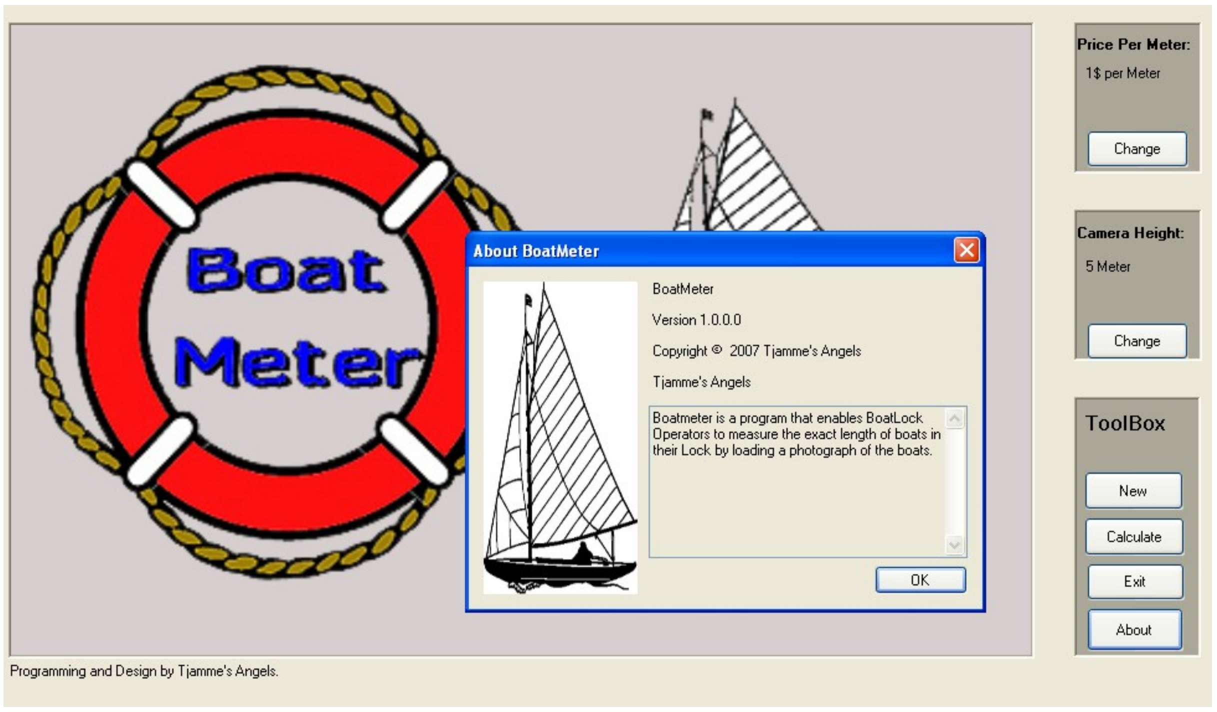


Do you want to quit?

Programming and Design by Tjamme's Angels.

Figure3.4 Exit the program

About: To show the about box, with all the credits of this program



BoatMeter

Version 1.0.0.0

Copyright © 2007 Tjamme's Angels

Tjamme's Angels

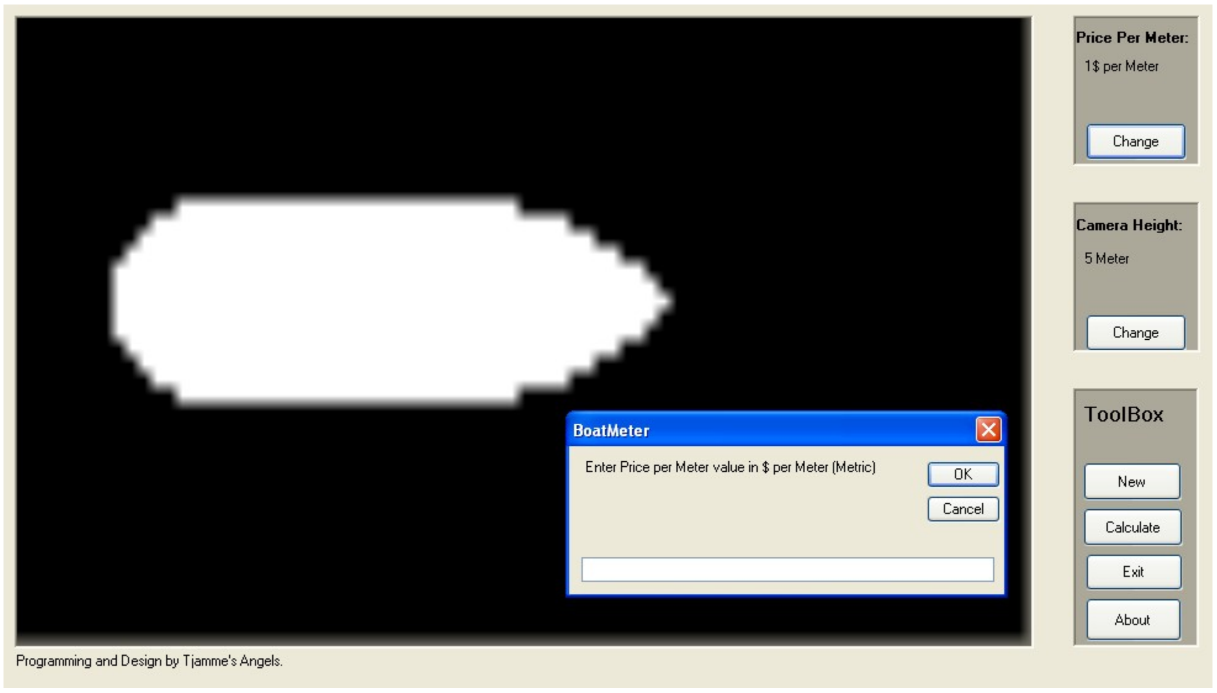
Boatmeter is a program that enables BoatLock Operators to measure the exact length of boats in their Lock by loading a photograph of the boats.

Programming and Design by Tjamme's Angels.

Figure 3.5 Aboutbox

Main variables
Our program contains several variables, first of all our program consists 2 variable

options. It is able to change the price per meter; this means you can change the price for different kinds of ships and different kinds of locks. One ship may be more expensive than the other, or one canal lock may be more expansive than the other.

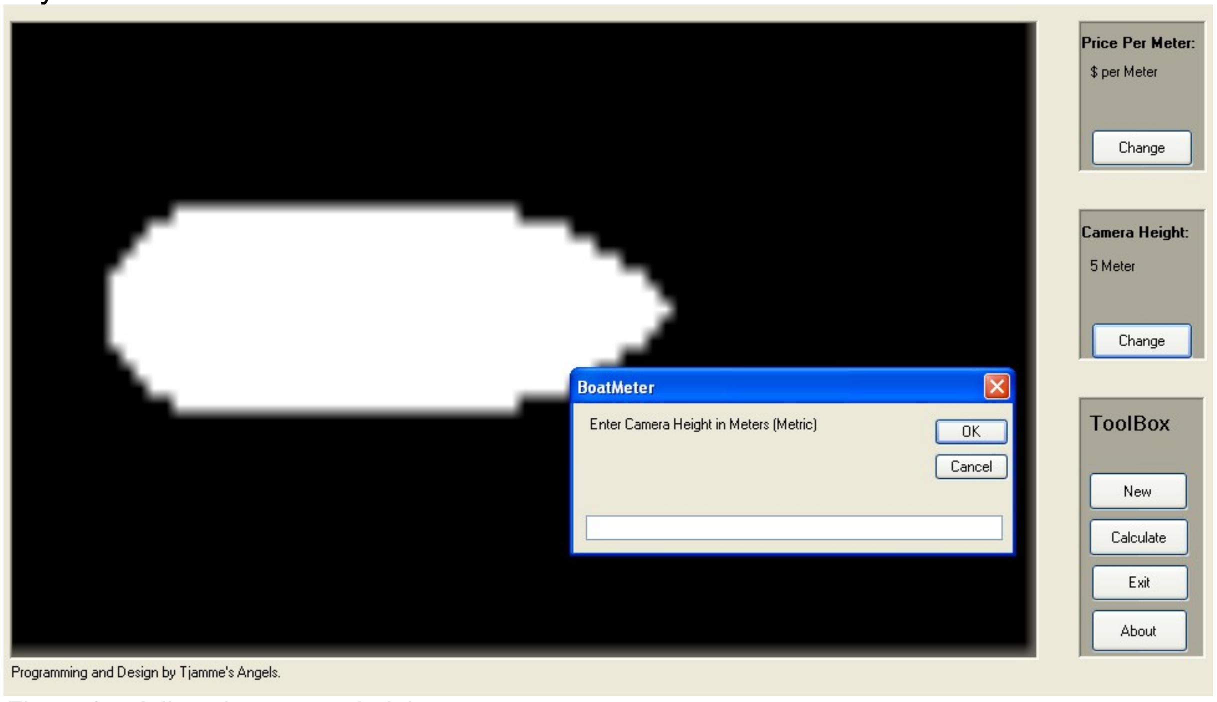


Enter Price per Meter value in $ per Meter (Metric)

Programming and Design by Tjamme's Angels.

Figure 3.6 Adjust the price per meter

It is also able to change the camera height, so take pictures from different heights with different scales. Therefore our program can be used for canal locks of all kinds of sizes, because you can vary the camera height so you can always take a full shot of your canal lock.



Enter Camera Height in Meters (Metric)

Programming and Design by Tjamme's Angels.

Figure 3.7 Adjust the camera height

Second our input is also a variable, because ships aren't always the same. They vary in size and shape. Besides a photo can contain more ships so we need to consider that as well.

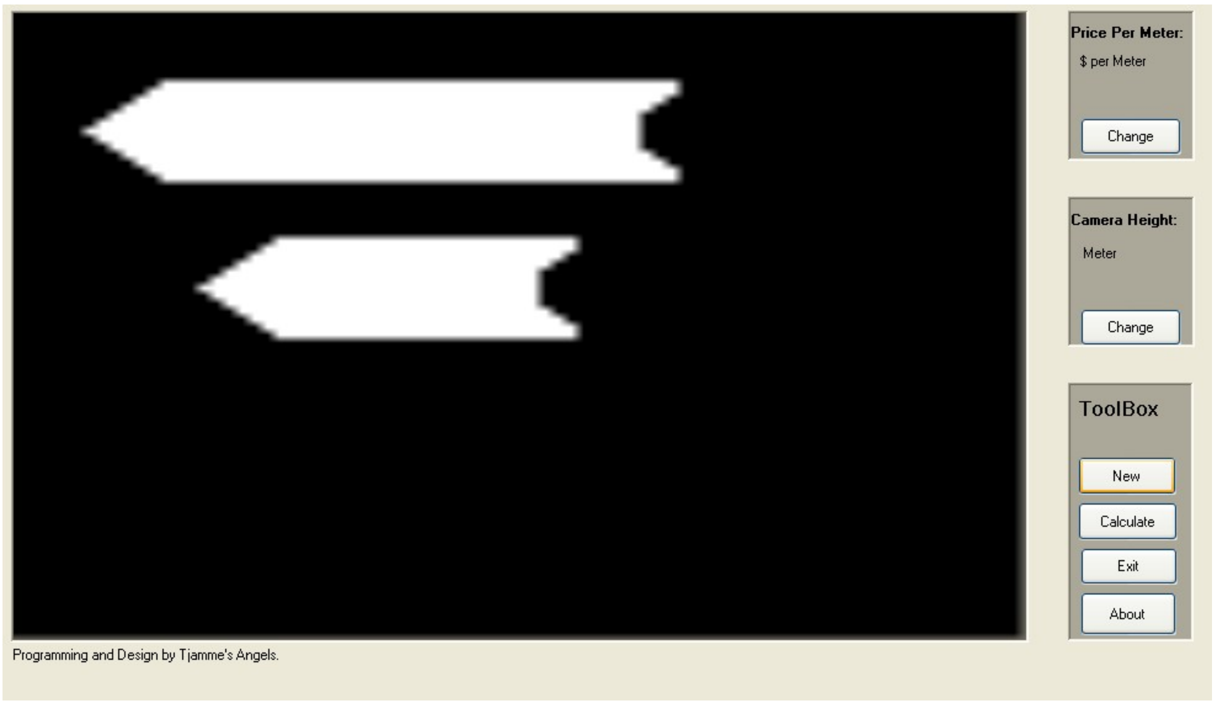


Figure 3.8 Situation for more than one ship

4 Evaluation of the AP

4.1 Instruction on our program

First we will explain each step of our program to make clear how it actually works.

A picture is taken by a camera. This camera hangs on top of the canal lock and takes a picture from the top view of the boats inside the lock. A piece of software is implemented in the program that converts the picture into an ASCII-file. This file is readable for the piece of software that we will design.
Our program contains a few options: Calculate boat price, Define price per meter, New, Exit. Each of these options we will further explain:

Calculate Boat Price: this function is the actual function that will be the main output of the program. After the length of the boat is measured and the price per meter is decided, the program will be able to calculate the price of the boat(s) in the lock.

Define Price Per Meter: with this function as said you can define the price per meter. In order to make possible price changes or different prices for different types of boats.

New: this function will activate the camera to take a new picture, which will directly be uploaded on the screen in the program.

Exit: to shut down the program.

4.2 Method of evaluation

At first we will test if our AP will work. We will go through our manual and see if everything is right. That if the output is in coherence with the input. If not we will either adapt our manual or the program itself. If everything is corrected, we will start with questioning people about our program. We will make a note with instructions on it so the people will know what they have to do. We will make a questionnaire with questions about our program that the testers can answer after they used our program. We need to question at least 10 persons to see if our program is easy to use. We will first let them test our program without the manual, and after that with the manual. We will do this to see if our manual is useful. We should also try to question one or more lock operators. We will also question some of our fellow students who are known with Visual Basic. This way we could see if they will understand our program and see if they will find it useful. Our program does not require people to manually convert bmp files into ASCII, we have implemented a set of images and matching ASCII that will be rotated in a set sequence. The test person will answer to a few questions about the use of the program and the program itself. We will write down the results, so we can see for each person their opinion on the program.

For the evaluation of our AP we need to consider 2 things. First of all we need to evaluate the technical side of the program. Does it fulfill the technical requirements? Is it working properly? Does the output match with the input, which means that the length of the ship is measured correctly and the price attached to the length is corresponding to the price per meter. Therefore we need to implement some mathematic formulas in our program.

To evaluate the technical side of our program we can let it calculate a ship of which the length and price is pre-determined. So we will be able to check if our program is working properly.

Secondly we need to evaluate the usability of the program. The interface of the program should be clear, so an inexperienced user can work with the program at first sight. This means that the program should not contain too many functions so the user will not be puzzled when he tries to operate the program. In general it's necessary to keep the program as easy and basic as possible, so that people that don't have much knowledge of computers also can use it.

4.3 Actual evaluation on the AP

Eleven people have tested our program. They had to fill in two different forms. The first consisted of the instructions on what they had to do with the program. On that particular form, they also had to give the answers to the calculations that the program made, so that we could see if they filled in the right variables. The other form consists of evaluation questions. What did they think about the program, what problems did they run into, etcetera. These forms can be found in the appendices.

Issues pointed out by the testers.

The new and calculate button should be switched around. It gave some confusion to some users because they pushed the calculate button instead of the new button, before they even had uploaded a photo. If we change these two buttons people might press new first instead of calculate.

There are too many boxes popping up during the test. There are quite some popup boxes used in our program. Each step you take is verified by a popup box. It ensures that people won't make mistakes, but to some people it works annoying.

The ultimate price should be outputted somewhere on screen. To avoid some of the popup boxes the price and length of the boat could be showed in a label instead of a popup box. This should be clearer to the user and more user friendly.

These were the most important suggestions or comments the testers had. Most people had these comments, so they're quite universal. One person even said that the program looked like it was made for Windows 95, but that is just a comment on the looks of the program and doesn't further affect the usability of the program. But it should be reconsidered for our UP, because a nice looking program is much more fun to work with.

5 Ultimate Product

5.1 Hardware in the UP

This paragraph is going to cover the considerations we've made to determine of which hardware our ultimate product should consist. We cover a number of possible options and conclude with a proposal.

Let us start off by stating two definitions, taken from Wikipedia. This is necessary to prevent confusion later on.

- An **embedded system** is a special-purpose computer system designed to perform one or a few dedicated functions, sometimes with real-time computing constraints. It is usually *embedded* as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. [1]
- **Embedded software** is computer software or firmware which plays an integral role in the electronics it is supplied with. Embedded software's principal role is not Information Technology, but rather the interaction with the physical world. It's written for machines that are not, first and foremost, computers. Embedded software is 'built in' to the electronics in cars, telephones, audio equipment, robots, appliances, toys, security systems, pacemakers, televisions and digital watches, for example. Embedded software is usually written for special purpose hardware: that is computer chips that are different from general purpose CPUs. [2]

To determine which components suited our system best, we started by listing the functions that had to be covered by these components. This is something we have covered to certain extent in Figure 5.1: Borders of program & data flow. Considering this information, we can list the following main functions that must exist in our ultimate product. Image capturing, data processing, user interaction and device interfaces. We can translate these main functions into five main components.

- Camera and camera mount;
- Processing device;
- User input device(s);
- Display device;
- Communication (e.g. cabled or wireless)

In order to limit the number of components the user would have to deal with, we decided to place the processing device, user input devices and the display device in one casing. We believe this will improve the overall image and user friendliness of the UP by reinforcing the image of one unified product instead of a system consisting of loose components. It will also simplify the set-up and reduce possible errors due to loose cables etc. Next we will list our considerations per main component.

For the camera the main consideration we had to make was between the two most used image sensors, the charge-coupled device (CCD [3]), used in most consumer digital cameras, and the active pixel sensor (APS [4]), used widely in security camera applications. In our application it seems logical to use the APS, because it's cheaper, faster, consumes less power and performs better in low-light conditions. The disadvantage is a less aesthetically pleasing image, because the color balance is less accurate and it generates slightly more noise then a CCD. The camera mount is very canal lock dependent, so we can't go into any detail on this. The most accurate definition we can give at this point is: the camera mount is a mechanical device which keeps the camera in a fixed position with respect to the canal lock. For fastening the camera to the mount we will use nuts and bolts.

The processing device is the sum of all hardware components on which our software runs. This is where all data processing and calculations take place. We have the choice of using two different types of hardware. The first being an embedded system consisting of components optimized to perform calculations and data manipulation that we use in our software. Cost is mainly determined by the number of functions the hardware has to perform and the costs of setting up a production line. The second option is using modular hardware, popular in modern desktop computers. Advantages are compatibility with other components and the lack of production and research costs. Embedded systems are generally cheaper then modular hardware when you're dealing with a great production series. Because we are only dealing with a moderate production series, we propose to use modular computer hardware.

We will use two user input devices: a touchscreen and a numeric keypad (or numpad). Our software can be controlled by using a pointing device and numerical input. The more common choice for a pointer is of course a mouse, because it is much more cost efficient than a touchscreen. The reason we chose a touchscreen instead is that a touchscreen can be used more intuitively, especially by people with less than average experience with computers. It's also not possible to integrate a mouse into the casing, so it will always be viewed as a loose component by the user. There is another pointing device that could be integrated into the casing: the trackball. Since this has become quite an exotic device these days, we haven't considered it to be an option. The only alternative for the numpad would be to use a full keyboard, but since our program doesn't require any text input it would be a useless addition.

As a display device we chose a Thin Film Transistor Liquid Crystal Display (TFT LCD). This is integrated with the previously mentioned touchscreen. The only alternative we considered was to use a Cathode Ray Tube (CRT) monitor. We chose for the TFT LCD because it needs less space then a CRT monitor, it produces a 'quieter' image and it's generally perceived as a more modern device then a CRT monitor. There are no significant cost differences between the two standards. For the size we find 15” to be large enough for this application.

There are a lot of possible communication options available. We were looking for two-way communication between the processing device and the camera. This excludes analog image transmission options. Because it's much more practical to use existing technology than to create our own, we chose for standard computer networking technology. We still had the choice between a wired and wireless network, but we chose for wired because no moving components are used and a wired network is generally more stable. This means we will use a twisted pair cable with standard RJ-45 connectors. We also have to use a camera with computer networking capability (IP-camera).

For the overview we have listed all the proposed components in Table 6.1. Not only function and component are listed, but also an approximation of the cost price. This approximation was made by comparing consumer prices of similar components and subtracting 30% (bulk discount). Final amounts were rounded to the nearest 100, because we don't want to give the appearance of an accuracy that is not present.

5.2 Software in the UP

First of all this paragraph needs to address the gap between the AP and UP requirements. Certain functions that we feel are necessary to implement in a successful UP we couldn't implement in our AP, mainly for time considerations. We will discuss the following issues: the gap between hard- and software, converting a full color bitmap photograph into a contrasted image, converting this contrasted image into an ASCII file, dealing with ships in any orientation.

What we mean by the gap between hard- and software is, that we have defined the hardware of the processing device in paragraph 5.1, but we have not defined how the hardware will load our software program. We will not go into great technical detail, because the focus of this paper lies elsewhere. When our UP is powered up, a chip (on the motherboard of the processing device) runs firmware code, called the Basic Input/Output System (BIOS [1]). The BIOS initializes the hardware components of the processing device and loads a boot loader, generally of an operating system. Because we only want to use a single piece of software it is unnecessary for us to run a whole operating system. The only operating system component we need to run our software is a so-called kernel [2]. The kernel is responsible for communication between hard- and software. Any additional functions of the operating system (such as a graphical interface and integrated applications) are not necessary for us. Finally the kernel loads device drivers and runs our software.

Converting a full color image into a 'contrasted image' (as we call it for clarity) is done in many applications and existing products. A lot of the operations we are going to describe could therefore possibly be done by existing algorithms, but we will focus on programming these ourselves to avoid losing ourselves in the legal depths of software licensing.

Our main task will be segmenting the image. This means we will change our image from a pixel-based representation to a region based representation of reality [3]. We will use the Hue, Saturation and Brightness (HSB) color space instead of the RGB color space to simplify the used operations. Next we list the operations that are necessary to achieve image segmentation and thus lead to what we call a contrasted image. To do this, we need to assume a reference photo of an empty canal lock is taken at set intervals (for instance five times per day).

- Conversion of an RGB bitmap into HSB [4];
- Create a histogram listing frequency of appearance of each brightness value [5];
- Subtracting a histogram of the reference photograph from this;
- Now you are left with 'bumps' in the resulting histogram. These bumps represent the brightness values of the pixels that 'belong' to the boat;
- Apply tresholding algorithms to make all the pixels that fall within these areas white and all those that do not fall within the areas black. What is left is an image with the general bodies of the boats in white and the background in black. There is still some noise that has to be removed because it is essential for our calculations that the only white pixels in the image actually belong to a boat [5];
- To remove the noise we need to check for each white pixel if the surrounding pixels are white too. If not all surrounding pixels are also white, this pixel becomes black. This operation needs to be repeated a number of times to make sure all groups of white pixels that do not belong to a boat become black. This causes the borders of our boats to shrink one row per repetition. To end up with a boat that is the same size as you started with, this operation can be reversed and executed the same number of times the first operation was repeated [6]; Small groups of white pixels (noise) do not grow back, because they have become completely black;
- The image left is completely segmented, showing white regions (boats) and a black background without any noise.

Now this contrasted image has to be converted into ASCII, using the same principles as the conversion program we used for the AP. The only difference being that the ASCII file will not be RGB, but HSB based. Because HSB values are a decimal between 0 and 1, we will check for white pixels as brightness being 1. The other principles of our software remain the same.

To deal with ships in any orientation, we need to figure out the graphical centre of gravity of the picture, so we can decide the centerline of the boat. The method we will use to determine this centerline is as follows:

- We take the average x-value of all the white pixels in the contrasted picture;
- We take the average y-value of all the white pixels in the contrasted picture;
- Now we have the x-and y values for the graphical centre of gravity (the averages for both x-and y values);
- Most boats have some sort of a triangle shape the centre of gravity (figure 5.1) will lay more to the rear of the ship (because in a triangle shaped picture the thicker part of the triangle contains more pixels)
- We draw a line from the centre of gravity towards the furthest white pixel, this will give us the centerline (figure 5.1)
- We turn the picture until the centerline lays parallel to the line y = 0, this will give us a ship that lays horizontal.
- Now we can measure and calculate the length of the ship as we used to do in the AP. This means that if there are more than one ship on the picture it will run the subroutine for each ship separately.

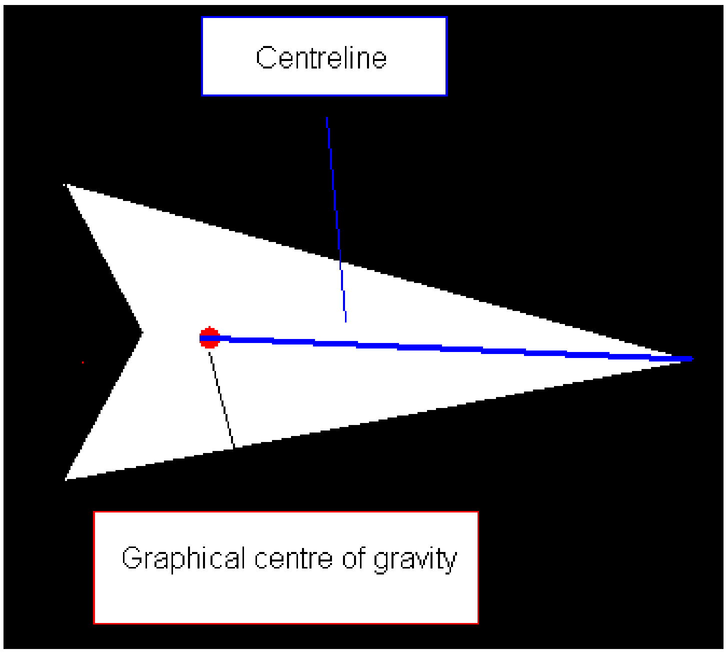


Figure 5.1 Method for measuring the length of a ship in any orientation

Finally to avoid the number of popup boxes we need to place labels on the boats with their length and corresponding price. The centre of gravity of the boat will be used as a reference where the label should be projected on the boat.

5.3 Sources

Sources used for the report of the UP are numbered and listed for each of the chapters it is used for. The number of the source correspond with the numbers in the text.

Hardware of UP

[1] http://en.wikipedia.org/wiki/Embedded_system
[2] http://en.wikipedia.org/wiki/Embedded_software
[3] http://en.wikipedia.org/wiki/Charge-coupled_device
[4] http://en.wikipedia.org/wiki/Active_pixel_sensor

Software of UP

[1] http://en.wikipedia.org/wiki/BIOS
[2] http://en.wikipedia.org/wiki/Kernel_%28computer_science%29
[3] http://en.wikipedia.org/wiki/Image_segmentation
[4] http://en.wikipedia.org/wiki/HSI_color_space
[5] Course Information Processing (ET3125) College sheets (digitally available on blackboard or from Alex Olieman on request)
[6]http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT3/node3.html
We have also used the following book as reference. Because it was not in our posession at the time of writing, we could not refer to page numbers so instead referred to similar information on the internet. R.C. Gonzales and R.E. Woods (2002). Digital Image Processing. Addison-Wesley.

6 Evaluation on the UP

We did not really test people for our UP. We did asked people for recommendations and comments on our program. We also asked them what they thought of the improvements we were going to add in our UP. They all reacted positive on the improvements we were about to program in our UP.

The improvements for the UP should contain the ability to calculate the length of any/all ship(s) in any position. It should also contain the ability to convert a full colour picture into a contrasted image. The price and length of a ship should be projected on the ship, to avoid the huge amount of popup boxes.

Furthermore the UP will contain just as the reviewed AP the recommendations given in the evaluation of the AP. The “calculate” and “new” buttons are switched so the user hopefully uploads a new photo first before he starts to calculate the length of the ship. Some of the popup boxes in the program are already deleted to reduce the number of popup boxes.

The predicted costs of our program are split into two groups; the cost prediction of the hardware and the cost prediction of the overall development of the program. The costs made for the hardware applications are recurring cost for any system. The costs for the overall development however are onetime costs.

Cost prediction Hardware

Function	**Component(s)**	**Approx. costs**
Image capturing	Camera with APS and IP	€400
	Camera mount	€0 - €400 (on average €200)
Data processing	Modular computer hardware	€300
User interaction	15” TFT LCD Touchscreen	€500
	Numeric keypad	€0
Device interfaces	UTP cable + installation	€100
	Total approx. hardware	**€1500**

Table 6.1 Cost prediction Hardware

Cost prediction overall development

Function	**Expected Time**	**Avg. wage per hour**	**Total Costs**
Research	20 hours	€20	€400
Programming	30 hours	€40	€1200
Testing	5 hours	€40	€200
Evaluation	3 hours	€12	€36
		Total approx. software	**€1836**

Table 6.2 Cost predictions Software; Wage is estimated according www.salarischecker.be for the most matching job found. The expected time is based on the time we used for our program comparing to experienced people who are familiar with every aspect.

7 Recommendations for improvements

These are the recommendations pointed out by the testers of our program, these were also the improvements we thought of ourselves.

The new and calculate button should be switched around. It gave some confusion to some users because they pushed the calculate button instead of the new button, before they even had uploaded a photo. If we change these two buttons people might press new first instead of calculate.

There are too many boxes popping up during the test. There are quite some popup boxes used in our program. Each step you take is verified by a popup box. It ensures that people won’t make mistakes, but to some people it works annoying.

The ultimate price should be outputted somewhere on screen. To avoid some of the popup boxes the price and length of the boat could be showed in a label instead of a popup box. This should be clearer to the user and more user friendly.

8 Prospects of the software in future products or services

In principle the product can be used in any system to calculate the length of any object. These are some of the applications our program could be used for.

For example: measuring the length of a car before entering a parking garage. If the car would be too long, the program could give a sort of alarm, so the driver would know not too drive further into the garage. This method could also be used for airplanes which need to enter a hangar.

Trucks have to pay a fee sometimes at a tollbooth. Our program can calculate the price of the fee that has to be paid according to the length of the truck. In general it would be the same as with the ships, only this time it’s on the road instead of water.

Packages which are sent by mail can also be charged for the size they are. Our program could also calculate the price of the package by measuring the size of it.

9 Conclusion

At the end of each project follows a conclusion. After several weeks we managed to create a program that is able to measure the length of a ship and calculate the fee it should pay to the operator.

Our Actual Product is able to calculate several situations containing boats. First of all we wanted to design a program that could measure the length of only one ship that lays straight. When that worked out properly, it was made possible that we could measure two boats at one time. But, they have to be in one line and may not be turned into different angles. The buttons of our program are easy to use. There are buttons to upload a new photo, one to calculate the price of the boat, one to exit the program and one that opens up the aboutbox. Furthermore there are two lines where you can vary the price per meter and the height of the camera.

In this way we can be satisfied about our AP, we got our program working for one boat, which was our main goal in the first place. Then we also got the program working for more than one ship. Our user interface is very easy and clear, because of the predicted users are assumed not to be very easygoing with computers. Therefore it is neccesary that the program doesn't contain difficult options or calculations made by the operator and the operator is aware of what he is doing and fully understands his actions.

Our Ultimate Product will be able to measure and calculate the price of more than just two boats and boats which lie in different angles. This wasn't possible to realise for the AP, just because there wasn't enough time. When the boat has been measured, the price will appear in the boat(s) on the screen as a label instead of in a popup box. This will avoid some of the numerous popup boxes. The buttons will remain the same as with the AP. The UP is the ultimate program for the lock operators. It is very simple to use, likewise the user interface in the AP.

Our product can be used for a great number of other applications, in principle for any product that measures a length of an object. Products such as the seize of trucks for a tollbooth. Or cars, planes and packages are some of other future possibilities.

The course itself was very interesting, we learned a lot about programming of software. We think the practicum should need some changes for future students, because it was very basic comparing to the project. We did not learn enough of Visual Basic to make a start in the beginning of the project. Besides the complete practicum was explained on the instruction forms, so we didn’t really had to think for ourselves.

Appendix A: Project organization

3D Digital Measurement and Design
Opdracht: Boat Port

Opdracht Beschrijving

Schrijf een stuk software dat aan de hand van een foto van een sluis met boten (van bovenaf genomen) de lengte van de bo(o)t(en) bepaald.

Plan van Aanpak

Taken:

Alex:
-Research on functionalities in VB
-methods of getting program working
-UP

Joris:
-General Programming
-Overseeing Project
-Assigning Tasks
-AP

Guimar:
-Beta testing
-Exterior Design (UP) User Interface (AP & UP)
-UP

Werner:
-General Programming
-Writing of Report
-AP

TimeTable

Week 3
Deadline Do 12.00
-Generating Workplan (Joris)
-Defining Borders of Program (devision soft/hardware) (Alex, Guimar)
-finishing VB practicum (Werner, Guimar)

Week 4
Deadline Do 12.00
-Hand in Workplan
-Research on existing products (Joris)
- research on practical boundries AP (Werner)
- research on VB and other usable software (Alex)

-Research on Environment (guimar)

Week5
Deadline Do 12.00
-Requirements of actual Product (Joris)
-User Manual of Actual Product (Alex)
-Method of evaluation of AP and UP (Guimar & Werner)

Week 6
Deadline Do 12.00
-Principal Design of AP (all)

Week 7
Deadline Do 12.00
-Initial Software Solution of AP (Alex & Joris)
-initial technical documentation of AP (Guimar & Werner)

Week 8
Deadline Do 12.00
-Evaluation (Guimar)
-Final Report and Presentation Preparations (Werner & Alex)
-Final Software Improvements (Joris)

Week 9
-Final Report (Werner)
-Presentation (All)

Appendix B: Time registration table

Werner van Huffelen

Week 1-3	VB practicum	20 hours
Week 4	Research on practical boundries AP	2 hours
Week 5	Method of evaluation of AP and UP	3 hours
Week 6	Principal Design of AP	2 hours
Week 7	Technical documentation of AP	10 hours
Week 8	Final report `	8 hours
Week 9	Final report	10 hours
Week 10	Finishing report	10 hours
Total		**65 hours**

Guimar Letsoin

Week 1-3	VB Practicum	20 hours
Week 3	Graphical development borders of program	2 hours
Week 4	Research on environment	3 hours
Week 5	Method of evaluation of AP and UP	3 hours
Week 6	Principal Design of AP	2 hours
Week 7	Initial technical documentation of AP	4 hours
Week 8	Evaluation	8 hours
Week 9	Processing evaluation forms	5 hours
Week 10	Making 3d model of housing of UP	4 hours
	Finishing report	5 hours
	Making front	1 hour
	Graphical design of the program	3 hours
Total		**60 hours**

Alex Olieman

Week 1-3	VB Practicum	20 hours
Week 3	Defining borders of program	3 hours
Week 4	Research on digital image processing	10 hours
Week 5	Research on digital image processing	4 hours
Week 5	Writing the import ASCII sub	4 hours
Week 6	General programming	6 hours
Week 7	General programming	8 hours
Week 8	Design UP	4 hours
Week 9	Design UP	12 hours
Week 10	Design UP	10 hours
Total		**81 hours**

Joris van't Ende

Week 1-2	VB practicum	20 hours
Week 3	Generated Workplan	1 hour
Week 4	Research on existing Products	4 hours
Week 5	Requirements of Actual Product	4 hours
Week 6	Principle design of AP	2 hours
Week 7	Programming	12 Hours
Week 8	Programming	15 hours
Week 9	Presentation and final report	14 hours
Total		72 hours

Appendix C: User manual of the AP

Manual BoatMeter

1. Measuring Length and Price of Boat(s)
2. Changing the Price/Meter Value
3. Changing CameraHeight
4. Starting with a new Picture
5. Exiting the Program

1. Measuring Length and Price
On the main screen, you will now see the photo you are currently using.
To start measurements on the Boat(s), simply press the **Calculate** Button, From the **Tools**
Menu.
The Boatmeter will now calculate both the length and price for each ship, and display those
values on the correct ship.

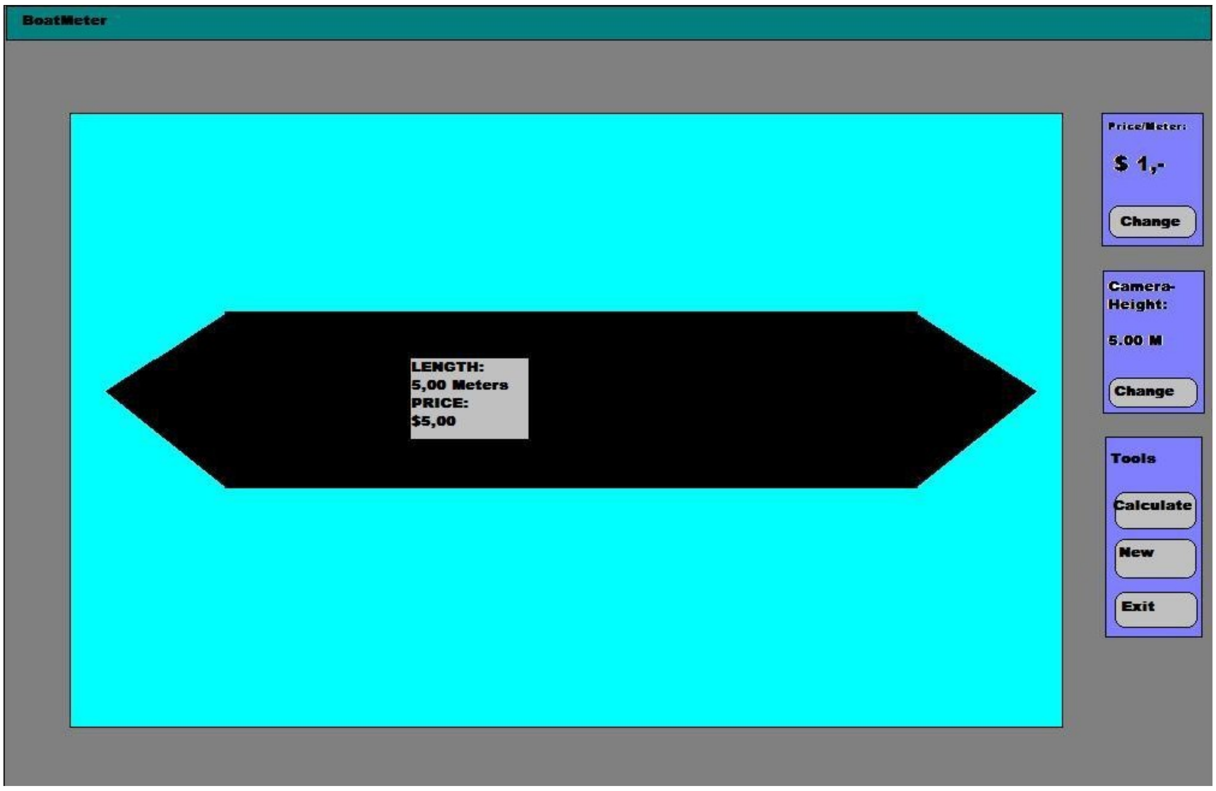
2. Changing Price/Meter value
To change the value of the Price per Meter value, simply press the **Change** Button directly
beneath the currently displayed Price/Meter Value.
A pop-up box will appear, prompting you to enter the new Price per Meter Value.
Remember: the Price is in US Dollars, The length is in Meters (Metric)
After you entered the new Price per Meter value and pressed the **OK** Button, the pop-up box
will disappear and the new price will be displayed on the correct boats.

3. Changing the CameraHeight value
To change the CameraHeight value (the distance between the camera and the waterline),
simply press the **Change** Button directly beneath the currently displayed CameraHeight.
A Pop-up Box will appear, prompting you to enter the new CameraHeight value.
Remember: the CameraHeight value is in Meters (Metric)
After you entered the new CameraHeight value and pressed the **OK** button, the Pop-up Box
will disappear and the CameraHeight value will be changed.
The new length and price will be displayed on the correct Boat(s).

4. Starting with a new Picture
To start measurement on a new picture, simply press the **New** Button from the **Tools** menu.
The program will restart, asking for the new picture to be used (if you are using the AP
version, you will have to manually convert the picture to ASCII value).
The new picture will be displayed on the main screen.

5. Exiting the Program

To exit the program, simply press the **Exit** Button from the **Tools** Menu.
A Pop-up Box will appear, asking you to confirm your order to shut down the program.
Press **OK** to confirm your order, the program will terminate instantly.
Press **Cancel** to cancel your order, the pop-up box will disappear and your session will
continue.



LENGTH:
5,00 Meters
PRICE:
$5,00

Appendix D: VB software Listing

BoatMeter 1.0 Code

Form1.VB

```
Imports System.IO
Public Class BoatMeter
    'Declare everything here!
    Dim fullArray(6399, 4) As Integer
    Dim stPriceMeter As String = 1.0
    Dim stCamHeight As String = 5.0
    Dim stPictureNumber As Integer
    Dim CurrentDataFile As String
    Dim CurrentPicture As String
    Dim n As Integer
    Dim WhiteArray(1, n) As Integer
    Dim BoatLength1 As Double
    Dim BoatLength2 As Double
    Dim BoatCost As Double
    Dim BoatCost1 As Double
    Dim BoatCost2 As Double
    Dim stBoatLength1 As Integer
    Dim stBoatLength2 As Integer
    Dim GrootsteX As Integer
    Dim GrootsteX2 As Integer
    Dim KleinsteX As Integer = 1000
    Dim KleinsteX2 As Integer = 1000
    Dim m As Integer
    Dim k As Integer
    Dim x As Integer
    Dim y As Integer
    Dim z As Integer
    Dim BoatNumber As Integer = 1
    Dim Boat1(1, n) As Integer
    Dim Boat2(1, n) As Integer
    Dim BoatArray As Integer

    Private Sub Form_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
        'the famous Form Load event.
        UpdateLabels()
    End Sub
    Private Sub UpdateLabels()
        'Subroutine that Updates the content of the PriceperMeter and Camheight Labels
        PriceMeterlbl.Text = stPriceMeter & "$ per Meter"
        CamHeightlbl.Text = stCamHeight & " Meter"
    End Sub
    Private Sub BoatCounter()
        'this subroutine checks if there are 2 boats. if there are, it will divide the WhiteArray into one
separate array for each boat.
        'the subroutinge checks the presence of 2 boats by scanning the WhiteArray for any gaps in
white pixels.
        'if a pixel is found who's y-value is higher then it's previous pixels y value +1 there is a gap in
white pixels. therefore a second boat must
        'be present.
        x = 1   'this variable starts at one because it is used to compare a pixel to it's previous pixel. if we
start x at 0, the routine is trying to
        y = 0   'compare a pixel (at 0) to a non existing pixel (-1). by making x start at 1 the routine
compares pixel (at 1) to the pixel (at 0), wich exists.
        z = 0
        BoatNumber = 1
        Do While x < ((WhiteArray.Length) / 2)
            If WhiteArray(1, x) > (WhiteArray(1, (x - 1)) + 1) Then
                BoatNumber = BoatNumber + 1
            End If
            If BoatNumber = 1 Then
                ReDim Preserve Boat1(1, y)
                Boat1(0, y) = WhiteArray(0, x)
                Boat1(1, y) = WhiteArray(1, x)
                y = y + 1
            ElseIf BoatNumber = 2 Then
                ReDim Preserve Boat2(1, z)
                Boat2(0, z) = WhiteArray(0, x)
                Boat2(1, z) = WhiteArray(1, x)
                z = z + 1
            End If
            x = x + 1
        Loop
    End Sub
    Private Sub ValGrootsteX()
        'calculates largest x value by comparing each x value in BoatArrays to the last found largest x
value
        GrootsteX = 0
        GrootsteX2 = 0
        m = 0 'we use variable m to make sure we don't get interference from other calculations
        If BoatNumber = 1 Then
            Do While m < ((Boat1.Length) / 2)
                If GrootsteX < Boat1(0, m) Then
                    GrootsteX = Boat1(0, m)
                End If
                m = m + 1
            Loop
        ElseIf BoatNumber = 2 Then
            Do While m < ((Boat1.Length) / 2)
                If GrootsteX < Boat1(0, m) Then
                    GrootsteX = Boat1(0, m)
                End If
                m = m + 1
            Loop
            m = 0
            Do While m < ((Boat2.Length) / 2)
                If GrootsteX2 < Boat2(0, m) Then
                    GrootsteX2 = Boat2(0, m)
                End If
                m = m + 1
            Loop
        End If
    End Sub
    Private Sub ValKleinsteX()
        'calculates smallest x value by comparing each x value in Boatarrays to the last found smallest x
value
        KleinsteX = 1000 'startingvalue is 1000 because we now for sure there is no x or y value larger
than that value at any time.
        KleinsteX2 = 1000
        m = 0 'we use variable m to make sure we don't get interference from other calculations
        If BoatNumber = 1 Then
            Do While m < ((Boat1.Length) / 2)
```

```
                If KleinsteX > Boat1(0, m) Then
                    KleinsteX = Boat1(0, m)
                End If
                m = m + 1
            Loop
        ElseIf BoatNumber = 2 Then
            Do While m < ((Boat1.Length) / 2)
                If KleinsteX > Boat1(0, m) Then
                    KleinsteX = Boat1(0, m)
                End If
                m = m + 1
            Loop
            m = 0
            Do While m < ((Boat2.Length) / 2)
                If KleinsteX2 > Boat2(0, m) Then
                    KleinsteX2 = Boat2(0, m)
                End If
                m = m + 1
            Loop
        End If
    End Sub
    Private Sub Calculate()
        'the actual calculation part of the program
        LoadASCII()
        SortArrays()
        BoatCounter()
        ValGrootsteX()
        ValKleinsteX()
        'calculates length of boats bij taking first and last pixel of BoatArrays and distracting them
        If BoatNumber = 1 Then
            stBoatLength1 = GrootsteX - KleinsteX
            BoatLength1 = stCamHeight * (stBoatLength1 * 0.01)
            BoatCost = (BoatLength1 * stPriceMeter)
            If BoatLength1 > 0 Then
                MessageBox.Show("Boat is " & CStr(BoatLength1) & " Meters Long. " & vbCrLf & "Boat
Costs: $" & CStr(BoatCost))
            Else
                MessageBox.Show("No boat found on Photograph. please upload new Photograph or try
again.")
            End If
        ElseIf BoatNumber = 2 Then
            stBoatLength1 = GrootsteX - KleinsteX
            BoatLength1 = stCamHeight * (stBoatLength1 * 0.01)
            BoatCost1 = BoatLength1 * stPriceMeter
            stBoatLength2 = GrootsteX2 - KleinsteX2
            BoatLength2 = stCamHeight * (stBoatLength2 * 0.01)
            BoatCost2 = BoatLength2 * stPriceMeter
            MessageBox.Show("Upper Boat is " & CStr(BoatLength1) & " Meters Long. " & vbCrLf &
"Upper Boat Costs: $" & CStr(BoatCost1) & vbCrLf & vbCrLf & "Lower Boat is " & CStr(BoatLength2)
& " Meters Long. " & vbCrLf & "Lower Boat Costs: $" & CStr(BoatCost2))
        End If

    End Sub

    Private Sub LoadASCII()
        'the subroutine that let's the user load the converted image (in ascii value) into a two-dimensional
array.
        Dim stream As StreamReader
        Dim dataLine As String
        Dim lineArray() As String
        'Two dimensional integer array
        Dim i As Int32
        Dim j As Int32 = 0
        If stPictureNumber > 0 Then
            stream = New StreamReader(CurrentDataFile)
            'read text line by line
            While Not stream.EndOfStream
                dataLine = stream.ReadLine()
                'split each line into individual strings and store them in an array
                lineArray = dataLine.Split()
                For i = 0 To lineArray.Length - 1
                    Try
                        fullArray(j, i) = Int32.Parse(lineArray(i)) 'Convert string to integer
                    Catch ex As Exception
                        MessageBox.Show(ex.Message)
                    End Try
                Next
                j = j + 1
            End While
            stream.Close()
        Else : MessageBox.Show("Please upload a Photo")
        End If
        'http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=2233216&SiteID=1
    End Sub
    Private Sub SortArrays()
        'this subroutine will sort the fullarray to WhiteArray by excluding all black pixels (wich we don't
need).
        ' this way we only have to work with the array containing all the white pixels, since the boats on
the pictures will be
        'white.
        n = 0
        k = 0
        Do While n < 6400
            'we check each pixel in fullarray for colorvalue 255(white) if true, the x and y value of the pixel
is placed in whitearray.
            'the length of whitearray is defined by the number of entries. so that we only have true x values
in whitearray and no empty fields
            '(this would interfere with our calculations)
            'we use variable k to make sure no gaps occur in the whitearray (gaps would be 0, and we
don't want that)
            If fullArray(n, 2) = 255 Then
                ReDim Preserve WhiteArray(1, k)
                WhiteArray(0, k) = fullArray(n, 0)
                WhiteArray(1, k) = fullArray(n, 1)
                k = k + 1
            End If
            n = n + 1
        Loop
    End Sub

    Private Sub DisplayPicture()
        'displays current picture in picturebox (PictureBox is set to autoscale so that every picture loaded
into it will be automaticly
        'displayed covering the entire picturebox.
        If stPictureNumber = 6 Then
            stPictureNumber = 1
        End If
        CurrentPicture = My.Computer.FileSystem.CurrentDirectory & "\Picture" & CStr(stPictureNumber)
& ".bmp"
        CurrentDataFile = My.Computer.FileSystem.CurrentDirectory & "\data" & CStr(stPictureNumber)
& ".asc"
        'this enables us to change the pictures and datafiles in a pre arranged order.
        'first picture will be Picture1, the second Picture2 etc.
        'same goes for datafile
        PictureBox1.ImageLocation = CurrentPicture
    End Sub

    Private Sub PriceMeterbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles PriceMeterbtn.Click
        'Invokes Pop-up box for user to enter new Price per Meter Value
        stPriceMeter = InputBox("Enter Price per Meter value in $ per Meter (Metric)")
        UpdateLabels()
    End Sub

    Private Sub CamHeightbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles CamHeightbtn.Click
        'Invokes Pop-up box for user to enter new Camera Height Value
        stCamHeight = InputBox("Enter Camera Height in Meters (Metric)")
        UpdateLabels()
    End Sub

    Private Sub Calculatebtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Calculatebtn.Click
        'Intitializes the calculate subroutine
        Calculate()
    End Sub

    Private Sub Newbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Newbtn.Click
        'Enables user to load a new picture and datafile
        stPictureNumber = stPictureNumber + 1
        DisplayPicture()

    End Sub

    Private Sub Exitbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Exitbtn.Click
        'exits the program
        'first prompts user to confirm exit
        Dim msg As String
        Dim title As String
        Dim style As MsgBoxStyle
        Dim response As MsgBoxResult
        msg = "Do you want to quit?"
        style = MsgBoxStyle.DefaultButton2 Or _
           MsgBoxStyle.YesNo
        title = "Important"
        response = MsgBox(msg, style, title)
        If response = MsgBoxResult.Yes Then
            End
        End If
    End Sub

    Private Sub Aboutbtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Aboutbtn.Click
        'Displays the About box, containing information on the program
        AboutBox1.Show()
    End Sub

End Class
```

Splashscreen1.VB

```
Public NotInheritable Class SplashScreen1

    'TODO: This form can easily be set as the splash screen for the application by going to the
"Application" tab
    '  of the Project Designer ("Properties" under the "Project" menu).


    Private Sub SplashScreen1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load
        'Set up the dialog text at runtime according to the application's assembly information.

        'TODO: Customize the application's assembly information in the "Application" pane of the project
        '  properties dialog (under the "Project" menu).

        'Application title
        If My.Application.Info.Title <> "" Then
            ApplicationTitle.Text = My.Application.Info.Title
        Else
            'If the application title is missing, use the application name, without the extension
            ApplicationTitle.Text =
System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.AssemblyName)
        End If

        'Format the version information using the text set into the Version control at design time as the
        '  formatting string.  This allows for effective localization if desired.
        '  Build and revision information could be included by using the following code and changing the
        '  Version control's designtime text to "Version {0}.{1:00}.{2}.{3}" or something similar.  See
        '  String.Format() in Help for more information.
        '
        '    Version.Text = System.String.Format(Version.Text, My.Application.Info.Version.Major,
My.Application.Info.Version.Minor, My.Application.Info.Version.Build,
My.Application.Info.Version.Revision)

        Version.Text = System.String.Format(Version.Text, My.Application.Info.Version.Major,
My.Application.Info.Version.Minor)

        'Copyright info
        Copyright.Text = My.Application.Info.Copyright
    End Sub


End Class
```

AboutBox1.VB

```
Public NotInheritable Class AboutBox1

    Private Sub AboutBox1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        ' Set the title of the form.
        Dim ApplicationTitle As String
        If My.Application.Info.Title <> "" Then
            ApplicationTitle = My.Application.Info.Title
        Else
            ApplicationTitle =
System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.AssemblyName)
        End If
        Me.Text = String.Format("About {0}", ApplicationTitle)
        ' Initialize all of the text displayed on the About Box.
        ' TODO: Customize the application's assembly information in the "Application" pane of the project
        '    properties dialog (under the "Project" menu).
        Me.LabelProductName.Text = My.Application.Info.ProductName
        Me.LabelVersion.Text = String.Format("Version {0}", My.Application.Info.Version.ToString)
        Me.LabelCopyright.Text = My.Application.Info.Copyright
        Me.LabelCompanyName.Text = My.Application.Info.CompanyName
        Me.TextBoxDescription.Text = My.Application.Info.Description
    End Sub

    Private Sub OKButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles OKButton.Click
        Me.Close()
    End Sub

End Class
```

Appendix E: Evaluation Forms