CS – 4802
Digital Image Processing
Lab #1

Application Environment

Submitted by: Jiri Sumbera
Submitted to: Dr. Taylor
Submitted on: 02-06-01

Introduction

The main goal of the first lab was to create an application framework for displaying of images. Due to my zero-level experience with Java and some experience with Borland C++ for Windows, C++ was chosen as the programming language. I decided to develop the application within the MFC framework as this was suggested to me as the most convenient way to do so.

Problem Statement
Develop an application for displaying images using C++ and MFC, which is able to read/write PNG images. In addition design other functions and classes to facilitate the implementation of future labs.

Secondary Goals:
- develop C++ class capable of image processing techniques with as little dependence on proprietary base classes as possible thus allowing for easy portability
- provide the author with OOP experience using C++
- provide the author with basic understanding of MS Windows programming techniques using MFC and Microsoft Visual C++ ver.6.0

Procedure

I did encounter many difficulties while developing this lab especially regarding programming using MFC and displaying images under Windows[1]. Consequently not all the goals of the lab have been met.
The developed application suffers from the following imperfections:
- the only graphic formats supported are PPM (24 bit color format) and PGM (8 bit grayscale format). This is believed to be minor inefficiency as any image to be processed can be converted in and out of those formats quite easily. The only disadvantage associated with the above formats is their lack of compression.
- despite considerable effort, I did not manage to display the image within my application. Consequently the compromise was made to use third-party external image viewer to display the images after each processing step as well as when desired. The application is hard coded to use excellent image viewer Irfan Viewer[2], but this can be easily changed to any PPM-and PGM-supporting application.
- the originally intended purity (i.e. absence of non-standard classes) of the kernel image class was violated (e.g. the class uses a CString object as one of its data members.) The reasons for this were simply time pressure and ease of use of some of the non-standard classes

Design Steps:
1, First a simple windows application (with Single Document interface) called 'Nothing' was created using MFC Wizard.
2, Next, the image class called 'My' was designed. The initial design include very little apart from loading and saving functions, which were primarily based on functions accompanying the book 'A Simplified Approach to Image Processing (see Resources).
3, The essential application framework input/output code was adopted from Open Source FreeImage[3] Library demo application FI_Demo. This demo application is available as a full source code project for MFC. The essential file-handling functions (Open, Save, Save As) were replicated without the full knowledge of the underlying framework[4].

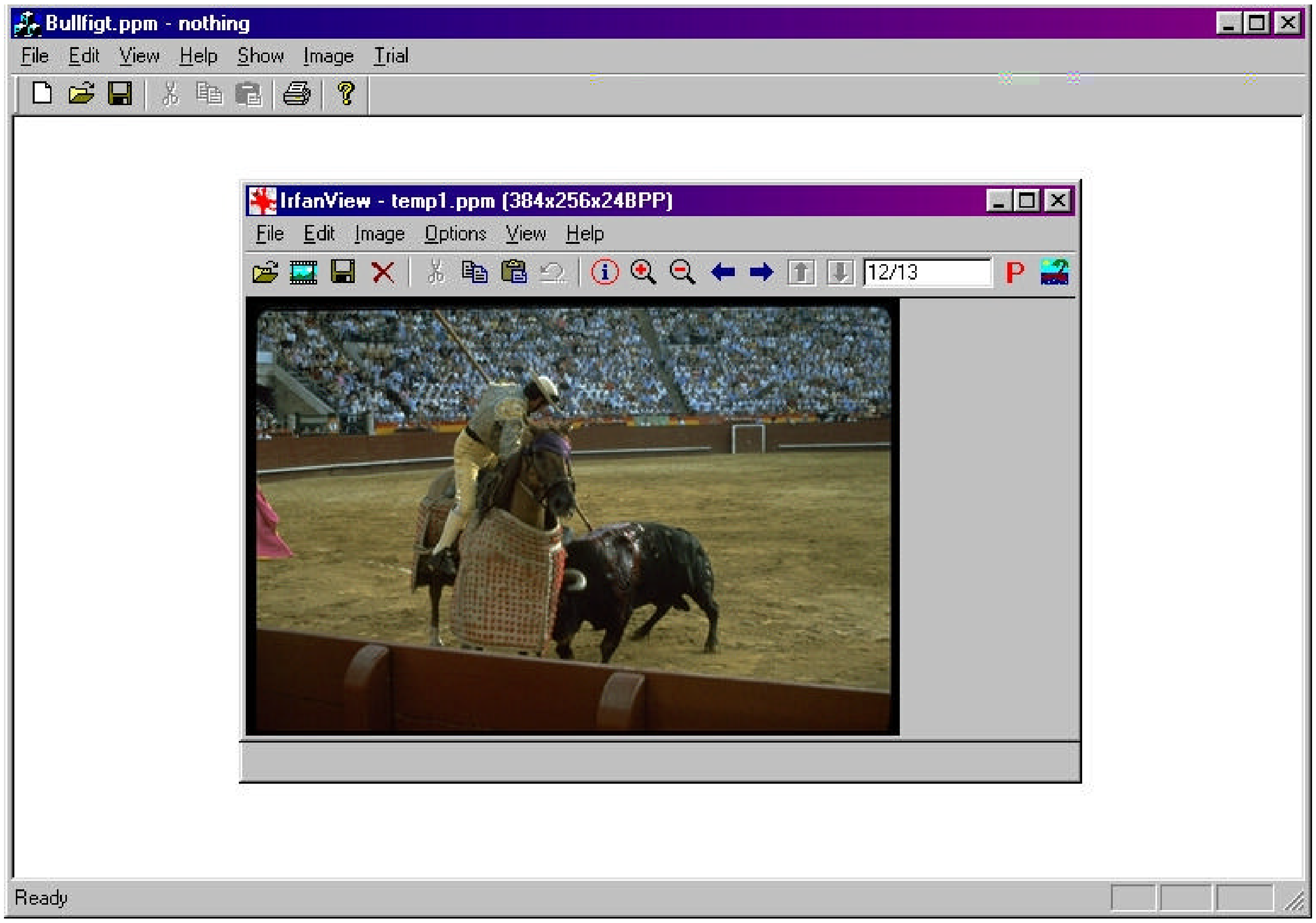
[1] The main reason for those was probably my initial unawareness of the complexity of the problem for a beginner. By the time I realized I should have full understanding of the MFC document framework prior to beginning my design, I was already running out of time. As the primary goal of my efforts was image processing and not OOP, the produced application may appear 'crippled' to say the least.
[2] Copyright Irfan Skiljan 1996-2000
[3] Copyright 2000 Floris van den Berg (freeimage@wxs.nl)
[4] But I believe that is the consequence of OOP – programmers using classes developed by someone else without knowing their insides. I just wish there was more on-line documentation available such as which

4, The following step was to add the ability view the image, at least externally. This was accomplished by saving the image into a temporary file and calling Irfan Viewer to display it. The call is facilitated by the ‘spawn’ function, which suspends the application until the external viewer is closed.

Discussion

I believe at this point it is appropriate to explain, why the application lacks some of the requirements. First of all I was not successful with including any of several PNG supporting libraries due to severe difficulties with compilation of the source code for these libraries. The reasons for the failures could be several, the most probable ones being my insufficient experience with the Microsoft Visual C++ developing environment and the fact that I had never done this before even though I understand the theory behind static libraries. [5]When I finally found a library distributed as a binary dll, I was not able to copy data in and out of its data structures.
The second source of problem was the Windows API itself and especially its rather confusing support of images (RGB stored as BGR, padding with zeros every 4 bytes).

Below is the screen capture of the application, showing the image displayed by Irfan Viewer.


Bullfigt.ppm - nothing
File Edit View Help Show Image Trial
IrfanView - temp1.ppm (384x256x24BPP)
File Edit Image Options View Help
12/13
Ready

message handling function handles the File-Open command and passes the name of the open file to the function OnOpenDocument.
[5] Never the less, I believe it not to be a good practice to require the use of other yet-to-be-compiled libraries for compiling of the target library as the chance of an error grows at least O(n^3) with the volume of the code.

Known bugs
Sometimes, the _spawnv() function used to call the external viewer Irfan Viewer, does not execute the executable as it should. I believe the error lies probably within the _spawnv() function as Irfan Viewer does not even start, which it would still do if there was e.g. an error in the command line passed to it (which is pretty much all the 'Show' function does.
This does not happen frequently and only when the first image after the application has started is loaded.

Resources used:

Irfan Viewer: http://stud4.tuwien.ac.at/~e9227474/
Irfan Viewer is a free image viewer supporting various image formats
Copyright Irfan Skiljan 1996-2000

FreeImage: http://home.planet.nl/~flvdberg/
FreeImage is an Open Source image library for multimedia and game programmers
Copyright 2000 Floris van den Berg (freeimage@wxs.nl)

A Simplified Approach to Image Processing:
Classical and Modern Techniques in C
by Randy Crane (Hewlett-Packard Company)
Published by Prentice Hall PTR
Upper Saddle River, New Jersey 07458
Copyright: Hewlett-Packard Company 1997

Activity Log
The following are very gross estimates:

Design: 1hr
Coding: 5hrs
Debug: 4hrs
Test: 1hr
Documentation: 2 hrs
Other - research: 10hrs

Conclusions

This lab proved itself to be more useful with providing insight into Windows API, Microsoft Foundation Classes and Object Oriented Programming in general than with digital image processing, however this insight was interesting enough.
If I were to do it again, I would probably start with fully reading some MFC-explaining online resource before attempting to write the application as this would save me a lot of time and allow to improve both my image class as well the way it interfaces with the Windows application.

Please note that the source code submitted with this application performs more than described in the report above. That is because it already incorporates all functions for the second lab assignment.