Image Cropping Application

A Project by

John Vincent Lloyd O. Bermudez  
Lester T. Lim

Submitted to

Luisito L. Agustin  
Instructor, ELC 152

In Partial Fulfillment of the Requirements for the Course  
ELC 152: Signal Processing

Department of Electronics, Computer and Communications Engineering  
School of Science and Engineering  
Loyola Schools  
Ateneo de Manila University  
Quezon City, Philippines

October 2010
ABSTRACT

The Image Cropping Application is a cropping application that has the capability of cropping images at an angle. The program is written in C++ and compiled using wxDevC++ in order to employ a reliable GUI. Multiple document interface (MDI) was used in its user interface to be able to handle and view multiple images in the same main frame. The program is able to load, crop at an angle, and save images in BMP, JPG, and PNG formats.
Acknowledgments

This project would not have been fruitful without His guidance.

To the learners who came before us - Joshoua Esmenda, Mark Mateo, and Jerome Paez - and made doing DSP a tad easier to do, our heartfelt gratitude.

To our family and friends who never wavered in their moral support, thank you. We owe all of you one.

And finally to Sir Lui for all the guidance and tenacity in seeing us through, our sincerest thanks.
# Table of Contents

1. Introduction ............................................................................................................... 6  
   1.1. Images ................................................................................................................. 6  
   1.2. Image Cropping .................................................................................................. 6  
   1.3. Bilinear Interpolation ....................................................................................... 6  
2. Project Overview ............................................................................................................ 8  
   2.1. Objectives ........................................................................................................... 8  
   2.2. Significance of the Project ................................................................................. 8  
   2.3. Scope and Limitations ....................................................................................... 9  
   2.4. Implementation Summary .................................................................................. 9  
3. Bilinear Interpolation Algorithm .................................................................................. 10  
4. Conclusions and Recommendations ............................................................................ 13  
   4.1. Conclusions ........................................................................................................ 13  
   4.2. Recommendations ............................................................................................ 13  
Appendix 1. User's Manual ............................................................................................. 14  
   A1.1. Software Overview .......................................................................................... 14  
      A1.1.1. Minimum System Requirements ................................................................. 14  
      A1.1.2. Features ..................................................................................................... 14  
   A1.2. Availability ........................................................................................................ 14  
      A1.3.1. Using the Software ..................................................................................... 15  
      A1.3.2. Loading Other Image Formats .................................................................... 15  
      A1.3.3. Loading Multiple Images .......................................................................... 15  
      A1.3.4. Cropping the Image .................................................................................. 15  
      A1.3.5. Getting Help ............................................................................................ 15  
      A1.3.6. Saving the Image ...................................................................................... 16  
      A1.3.7. Exiting the Program ................................................................................ 16  
Appendix 2. Source Code ................................................................................................ 17  
   A2.1. Source Files ..................................................................................................... 17  

Page 4
1. Introduction

1.1. Images

An image is made up of thousands of pixels arranged in a two dimensional array of a certain number of rows and columns. The pixels or picture elements are represented by three numbers corresponding to the intensities of the three primary colors: red, green, and blue.\(^1\) A mixture of varying intensity of each color will produce the desired pixel color for that specific point. An image \(f(i,j)\) is divided into \(i\) columns and \(j\) rows, of which each intersection is called a pixel.\(^2\)

1.2. Image Cropping

Image cropping involves selecting a particular area of the base image with the option of choosing an angle by which to crop the base image. The resulting cropped image will have a smaller dimension than the base image and will appear rotated by the number of degrees the user chooses. Rotation and cropping of the base image entails computing for the unknown pixel values of the resulting image. Computation for these values employs the use of bilinear interpolation.

1.3. Bilinear Interpolation

Bilinear interpolation is used for interpolating functions of two variables. It can be thought of as linear interpolation done twice: 1. First, separate interpolation between the two upper points and two lower points, 2. Then, interpolate between the two resulting points from the first two interpolations. Consider the point \((l+i, m+j)\) whose value needs to be interpolated. See Figure 1 below. The 2 x 2 neighboring pixels, i.e., the four pixels with known values immediately surrounding \((l+i, m+j)\) are \(f(l, m+1)\), \(f(l+1, m)\), \(f(l, m)\), and \(f(l+1, m+1)\). See Figure 1 in the succeeding page.

---

Computing for the interpolated value means taking the weighted average of the four surrounding pixels:

\[
\begin{align*}
    f(l+i, m+j) &= (1-i)(1-j)f(l, m) + i(1-j)f(l+1, m) + (1-i)jf(l, m+1) \\
    & \quad + ijf(l+1, m+1).
\end{align*}
\]

\[\text{Figure 1. Bilinear Interpolation}\]

\[\text{Bracewell, Ronald N.} \ \text{Two-Dimensional Imaging.} \ USA: \ Prentice-Hall \ Inc., 1995. pp. 248-250.\]
2. Project Overview

2.1. Objectives

The main objective of this project is to come up with an image cropping application that is capable of cropping base images at a certain angle. The project is to be written in C++ as it is the most accessible to the project members and more importantly, it supports object-oriented programming. The program is to be compiled using wxDevC++ as it has the same features as DevC++ needed for the image cropping program but with the added features of an environment for developing graphical user interfaces and displaying images.

The user interface should make it convenient for the user to manipulate images. It should utilize MDI to be able to view multiple images, e.g., the base image and the resulting cropped image, in the same window. It should also be able to open, load, and save images of different file formats. As it is an image cropping application, the program should also have fold panel sidebars that contain entry fields for the user to input the different parameters to manipulate the base image, e.g., angle of rotation, height and width of resulting image, etc., and a panel or portion that allows the user to view the base image’s properties. An error message should also appear in cases when the input parameters are outside the base image boundary or when the base image is not a supported file type.

2.2. Significance of the Project

Displaying, processing and interpretation of images are widely used in many industrial and commercial applications. Photo manipulation almost always includes image cropping functions. The capability to crop images is important in refining photos for web content, presentations and the like. Moreover, the additional capability of

---

cropping images at an angle makes it more convenient to correct photos whose alignment might be skewed by a few degrees. In other words, the image cropping application with rotation is useful for straightening out tilted photos or portions of the photo. Its compatibility with a lot of popular image formats such as JPEG, PNG, GIF, and BMP makes it convenient to use.

2.3 Scope and Limitations

The Image Cropping Application can handle JPG, GIF, BMP, and PNG image files. It can crop images by a positive or negative input angle. It allows the user to determine the height and width of the output image as well as select where to crop.

The program cannot handle RGB text files and can’t crop areas that will go beyond the picture, i.e., cropping takes place within the area of the picture only. The program doesn’t have a real-time cropping selector.

2.4 Implementation Summary

The Image Cropping Application used the image class to handle image files and wxDevC++ functions such as wxBitmap to make the user interface easier to use. The graphical user interface allows the user to open and load a base image to crop. It also allows the user to see the resulting cropped image in a new child frame under the same main window and has options for the user to save the output image in any directory and in various image file formats. The user interface has fold panel bars that contain entry fields for the user to input the desired height and width of the output image and the number of degrees by which the cropped image will be rotated. The fold panel bar will also contain text inputs to determine the starting point of cropping, i.e., an X and Y value. Finally, there will also be a fold panel bar displaying the properties of the base image.

The user interface is graphical in nature and accepts the file name of the input image. The main window has a pane which shows the current image's properties and cropping options. The main feature of the program is its ability to crop images at a user-defined angle. The cropped image can be saved to various image formats like JPG, BMP, and PNG.
3. Bilinear Interpolation Algorithm

The necessary formula to obtain the interpolated pixel value from the four neighboring base pixels is given by:

\[
f(l+i, m+j) = (1-i)(1-j)f(l, m) + i(1-j)f(l+1, m) + (1-i)m f(l, m+1) + ij f(l+1, m+1)
\]

There are several variables used in this formula. Variables \( l \) and \( m \) are the coordinates of the base_matrix where the values of the neighboring base pixels are located. Variables \( i \) and \( j \) correspond to the respective x and y distance of the interpolated pixel from \( l \) and \( m \). Suppose that the interpolated pixel value has four corner base pixels from which it gets its correct value. \( f(l,m) \) corresponds to the pixel value at the upper left corner of interpolated value while \( f(l+1,m) \) corresponds to the pixel value at the upper right corner of interpolated value. Similarly, \( f(l,m+1) \) corresponds to the pixel value at the lower left corner of interpolated value while \( f(l+1,m+1) \) corresponds to the pixel value at the lower right corner of interpolated value. Finally, \( f(l+i,m+j) \) corresponds to the interpolated pixel value. The respective pixel values will be assigned as pixel values Q1, Q2, Q3, Q4, and P. Reviewing the bilinear interpolation formula, it can be seen that the interpolated pixel value is the sum of a specific percentage of pixel value from each base pixel.

Before obtaining each of the variables, we must understand what each variable represents. For example, the missing pixel value P has the coordinates (76.5, 82.3). We must first obtain the coordinates of Q1 since it corresponds to the first pixel that appears in the array. From Q1, we can determine the coordinates of Q2, Q3, and Q4 by adding 1 to \( l \), \( m \), and \( l \) and \( m \) respectively. As we can see, the x-coordinate of Q1 is actually the greatest integer of the x-coordinate of pixel value P.

The same applies also to the y-coordinate of Q1 in that it is the greatest integer of the y-coordinate of P. The greatest integer of the coordinates of the interpolated pixel is given by:

```
for (l=0; e>=1; l++)
{
 e = e-1;
}
for (m=0; f>=1; m++)
{
 f = f-1;
}
```

Where (e,f) are the coordinates of the interpolated pixel. To obtain the greatest integer, l will increment by 1 until e is less than zero. The same goes for m and f as well. Now that the greatest integer has been calculated, next step is obtaining the value of i and j. It must be clear that i and j are the fractional parts which would determine the percentage of pixel value each base pixel will contribute to the final interpolated pixel value. To obtain i, we simply get the modulus of e when divided by l. The same step will be done to f and m to obtain j. Once all values have been obtained, the interpolated pixel value can then be calculated for that specific pixel.

The function `CropAtAnAngle_BLinterpolation()` returns an image object from the `temp_matrix` where the interpolated values are temporarily stored. Its parameters are the base image object (`input_image`), (x,y) origin (upper leftmost of the crop rectangle) (`x_start`, `y_start`), crop rectangle height and width (`input_height`, `input_width`), and crop rectangle rotation in degrees (`input_angle`). Initially, the function extracts the pixel value matrix from the base image object. Then, a new temporary array is allocated with the input crop rectangle height and width as its dimensions. When the temporary array is allocated, each pixel is accessed via two `for` loops. The outer loop increments until it is equal to the input crop rectangle height. The inner loop increments until it is equal to the input crop rectangle height. Basically, each pixel coordinate is first accessed from left to right before it progresses to the next line.
Before accessing each pixel coordinate, it is shifted by a certain crop rectangle rotation angle using rotation matrices.

```c
//if rotation is counter-clockwise, angle is positive
if (input_radian >= 0){
    e= x_start + (u*cos(input_radian) - t*sin(input_radian));
    f= y_start + (t*cos(input_radian) + u*sin(input_radian));
}
//if rotation is clockwise, angle is negative
else {
    e= x_start + (u*cos(input_radian) + t*sin(input_radian));
    f= y_start + (t*cos(input_radian) - u*sin(input_radian));
}
```

The variable `input_radian` is the converted value of `input_angle` from degrees to radians. As seen in the lines of code above, two rotation matrices are used for cases when the angle is positive or negative. In any case, the origin `x_start` and `y_start` are added before the rotation matrix.

Bilinear interpolation is then performed on each interpolated pixel to obtain its pixel value. This is done after obtaining values for `l, m, i, and j`.

```c
//linear interpolation between the two upper base pixels
//and two lower base pixel
temp1 = (1-i)* base_matrix[l][m] + i* base_matrix[l+1][m];
temp2 = (1-i)* base_matrix[l][m+1]
    + i * base_matrix[l+1][m+1];
//linear interpolation between the two interpolated points
//from the previous linear interpolations
temp_matrix[u][t] = (1-j)*temp1 + j*temp2;
```

Using the `make_image()` function, we can now obtain the temporary image object which will be returned eventually.
4. Conclusions and Recommendations

4.1. Conclusions

The Image Cropping Application is able to crop at an angle certain base image file types like JPG, BMP, and PNG. The program used bilinear algorithm to compute for unknown pixel values in the output image. The graphical user interface, developed using wxDevC++, allows the user to open, load, and save images in the aforementioned image file formats. The program also allows users to view multiple images in the same main window. It also contains fold panel bars that have entry fields for the user to input image parameters and angle of crop rectangle rotation as well as a panel or portion to view the properties of the base image.

4.2. Recommendations

A better interpolation algorithm such as bicubic interpolation could be used to obtain a sharper or more defined output image. The user interface could also be enhanced to allow for real time viewing of the crop rectangle as well as allow for preview features.
Appendix 1
User's Manual

A1.1. Software Overview

A1.1.1. Minimum System Requirements

- For binary/executable
  - Windows 98/XP/Vista
  - 256 MB RAM
  - 30 MB free hard disk space (4MB for executable, extra for image)
  - 256 or more colors compatible display

A1.1.2. Features:

The *Image Cropping Application* allows the user to:

- Load BMP, JPEG, PNG and GIF images
- View BMP, JPEG, PNG and GIF images
- View multiple images
- Choose the starting point of the leftmost pixel of the cropping selector
- Input the desired width and height of the output image
- Input the number of degrees by which the crop rectangle will be rotated
- View the base image’s properties in the fold panel bar
- Save BMP, JPEG, and PNG images

A1.2. Availability

The *Image Cropping Application* is available through request from the authors.

---


A1.3.1. Using the Software

The *Image Cropping Application* has a graphical user interface designed to run in Microsoft Windows. To run the program, open the “Image Cropping Application.exe” from the directory “<CD root directory>\program\software”.

A1.3.2. Loading Images

Click on the “Open File” toolbar button or menu. You can also click on the open file button located on the toolbar.

A1.3.3. Loading Multiple Images

You can open another image aside from the current image by repeating the process in A1.3.2.

A1.3.4. Cropping the Image

An image should be open at this point. In the fold panel bar on the left side of the main window, input the starting point of the leftmost pixel of the crop rectangle or the upper leftmost point from which you want to start cropping. Next, input the desired height and width dimensions of the output image in the entry fields found in the same fold panel bar. Also enter the number of degrees by which you want to rotate the crop rectangle.

A1.3.5. Getting Help

Clicking on the Image Cropping Application Help in the Help menu, the Help button on the toolbar or pressing the F1 key shall open the HTML help file included with the program.
A1.3.6. Saving the image

Click on the Save button on the toolbar, File->Save Image menu or pressing CTRL+S, shall produce a save file dialog if an image is opened. You can now save the image to the desired file name and image type.

A1.3.7. Exiting the program

There are several ways to exit the program. You can click on the X button on the title bar, press the Alt-X shortcut key in windows or click on the “X” button on the toolbar.
Appendix 2
Source Code

A2.1. Source Files

A2.1.1. app.cpp

//---------------------------------------------------------------------
//
// Name: app.cpp
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Modified by: Lester Lim and John Vincent Bermudez
// Description: Implementation for the cropper_app class
// Part of Image Cropping Application - Elc152
//
//---------------------------------------------------------------------
#include "app.h"
#include "main_frame.h"
extern void InitXmlResource();
IMPLEMENT_APP(cropper_app) //cropper_app is called

//the "main()" counterpart, called when cropper_app is implemented
bool cropper_app::OnInit()
{
    wxInitAllImageHandlers(); //to be able to open any image format
    wxXmlResource::Get()->InitAllHandlers(); //embedding resources to
    //binary
    InitXmlResource(); //same as above
    _window_number=0;

    //Creates the main_frame, the main window
    //and shows it as the top window
    main_frame *frame = new main_frame(NULL,wxID_ANY,
        _T("Image Cropping Application"),wxDefaultPosition,
        wxSize(800,600),wxDEFAULT_FRAME_STYLE|wxHSCROLL|
            wxVSCROLL);
    frame->Maximize(true);
    frame->Show(true);
    SetTopWindow(frame);
    return true;
}

int cropper_app::OnExit()
{
    return 0;
}
void cropper_app::increase(void)
{
    _window_number++;
}

void cropper_app::decrease(void)
{
    _window_number--;
}

int cropper_app::number_of_windows(void)
{
    return _window_number;
}

A2.1.2. app.h

/*-------------------------------------------------------------------------------
//
// Name:   app.h
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Modified by: Lester Lim and John Vincent Bermudez
// Description: Declaration for the cropper_app class
//              Part of Image Cropping Application - Elc152
//
/*-------------------------------------------------------------------------------
#ifndef app_h
#define app_h

//If compiler supports precompilation
#ifndef WX_PRECOMP
    #include <wx/wx.h>
#else
    #include <wx/wxprec.h>
#endif

class cropper_app : public wxApp
{
    private:
        int _window_number; //number of opened images, for checking
    public:
        bool OnInit(void);
        int OnExit(void);
        void increase(void);
        void decrease(void);
        int number_of_windows(void);
};
#endif
A2.1.3. app.rc

//---------------------------------------------------------------------
//
// Name:     app.rc
// Author:   Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Used by:  Lester Lim and John Vincent Bermudez
// Description: Resource file for Image Cropping Application
//              Part of Image Cropping Application - Elc152
//
//---------------------------------------------------------------------
#include <wx/msw/wx.rc>

A2.1.4. child_frame.cpp

//---------------------------------------------------------------------
//
// Name:     child_frame.cpp
// Author:   Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Modified by: Lester Lim and John Vincent Bermudez
// Description: Implementation for the child_frame class
//              Part of Image Cropping Application - Elc152
//
//---------------------------------------------------------------------

//header files
#include "app.h"
#include "main_frame.h"
#include "child_frame.h"
#include "my_canvas.h"

//wxWidgets libraries included
#include <wx/app.h>
#include <wx/init.h>
#include <wx/settings.h>

DECLARE_APP(cropper_app)

//event handlers
BEGIN_EVENT_TABLE(child_frame, wxMDIChildFrame)
    EVT_MENU(CHILD_QUIT, child_frame::OnQuit)
    EVT_SIZE(child_frame::OnSize)
    EVT_CLOSE(child_frame::OnClose)
    EVT_ACTIVATE(child_frame::OnActivate)
END_EVENT_TABLE()

child_frame::child_frame(main_frame *parent, const wxString& title,
                         const image& the_image)
    :wxMDIChildFrame(parent,wxID_ANY,title,wxDefaultPosition,
                     wxDefaultSize,wxDEFAULT_FRAME_STYLE |
                     wxNO_FULL_REPAINT_ON_RESIZE)
    {_display_canvas = new my_canvas(this,title,the_image);}
_image_name = title;
_my_parent = parent;
this->SetClientSize(_display_canvas->GetClientSize());
this->SetBackgroundColour(wxColour(_T("DIM GREY")));
Refresh();
}

child_frame::~child_frame(void)
{
}

void child_frame::OnSize(wxSizeEvent& event)
{
_display_canvas->SetSize(this->GetClientSize());
_display_canvas->Refresh();
}

void child_frame::OnQuit(wxCommandEvent& WXUNUSED (event))
{
Close(true);
}

void child_frame::OnClose(wxCloseEvent& event)
{
wxGetApp().decrease();
_my_parent->set_image_properties(wxEmptyString);
_my_parent->set_height_label(wxEmptyString);
_my_parent->set_width_label(wxEmptyString);
event.Skip();
}

void child_frame::OnActivate(wxActivateEvent& event)
{
//Sets the main_frame item labels to reflect the activated child
wxString height_label,width_label,temp_labels;
height_label.Printf(_T("%d"),_display_canvas->access_height());
width_label.Printf(_T("%d"),_display_canvas->access_width());
temp_labels = _T("Filename: ") + _image_name;
temp_labels = temp_labels + _T("\n")
_T("Height in Pixels: ") + height_label;
temp_labels = temp_labels + _T("\n")
_T("Width in Pixels: ") + width_label;
temp_labels = temp_labels + _T("\n\n");
_my_parent->set_image_properties(temp_labels);
_my_parent->set_height_label(height_label);
_my_parent->set_width_label(width_label);
_my_parent->Refresh();
event.Skip();
}
my_canvas* child_frame::return_canvas(void)
{
    return _display_canvas;
}

wxString child_frame::return_image_name(void)
{
    return _image_name;
}

main_frame* child_frame::return_my_parent(void)
{
    return _my_parent;
}

A2.1.5. child_frame.h

#ifndef child_frame_h
#define chlid_frame_h

//wxWidgets libraries included
#include <wx/wx.h>
#include <wx/mdi.h>
#include <wx/string.h>

//standard C++ libraries
#include <cstdlib>
#include <cstring>

//other includes
#include "image.h"
#include "my_canvas.h"
#include "main_frame.h"

class child_frame : public wxMDIChildFrame
{
    private:
        my_canvas *display_canvas;
        main_frame *my_parent; //pointer to parent, for access
        wxString image_name, char_starting_x, char_starting_y;
        int starting_x, starting_y;

    public:
        //Constructor/Deconstructor
child_frame(main_frame *parent, const wxString& title,
    const image& the_image);
~child_frame(void);

//Event functions
void OnSize(wxSizeEvent& event);
void OnActivate(wxActivateEvent& event);
void OnClose(wxCloseEvent& event);
void OnQuit(wxCommandEvent& event);

//Accessor functions
my_canvas* return_canvas(void);
main_frame* return_my_parent(void);
wxString return_image_name(void);

DECLARE_EVENT_TABLE();
);
#endif

A2.1.6. image.cpp

//---------------------------------------------------------------------
//
// Name:   image.cpp
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Used by: Lester Lim and John Vincent Bermudez
// Description: Image Class Implementation
//              Part of Image Cropping Application - Elc152
//
//---------------------------------------------------------------------

#include "image.h"

//-----------------fixed point implementations-----------------

fixed_point::fixed_point()
{
    _value = 0;
}

fixed_point::fixed_point(int new_value)
{
    _value = new_value << (8);
}

fixed_point::fixed_point(string hex_value)
{
    _value = hexToFixedPoint(hex_value);
}
fixed_point& fixed_point::operator = (const fixed_point& a)
{
    _value = a._value;
    return *this;
}

fixed_point operator + (const fixed_point& a, const fixed_point& b)
{
    fixed_point temp;
    temp._value = a._value + b._value;
    return temp;
}

fixed_point operator * (int a, const fixed_point& b)
{
    fixed_point temp;
    temp._value = a*b._value;
    return temp;
}

fixed_point operator / (const fixed_point& a, int b)
{
    fixed_point temp;
    temp._value = a._value/b;
    return temp;
}

int fixed_point::decimal_value() const
{
    int temp;
    temp = _value >> (8);
    return temp;
}

///////////////////////////////////////////////////////////////////////
// Created on: 12-18-07
//
// Description: Converts a character into its hex counterpart in
//              fixed point
// Parameters: char cTemp
//
// Last modified: 12-21-07
//
///////////////////////////////////////////////////////////////////////

int fixed_point::charToHex(char cTemp)
{
    switch (cTemp)
    {
    case '0': return 0x00000000;
    case '1': return 0x00000001;
    case '2': return 0x00000002;
    case '3': return 0x00000003;
    case '4': return 0x00000004;
    }
case '5': return 0x00000005;
case '6': return 0x00000006;
case '7': return 0x00000007;
case '8': return 0x00000008;
case '9': return 0x00000009;
case 'A': return 0x0000000A;
case 'B': return 0x0000000B;
case 'C': return 0x0000000C;
case 'D': return 0x0000000D;
case 'E': return 0x0000000E;
case 'F': return 0x0000000F;
case 'a': return 0x0000000A;
case 'b': return 0x0000000B;
case 'c': return 0x0000000C;
case 'd': return 0x0000000D;
case 'e': return 0x0000000E;
case 'f': return 0x0000000F;
default: return 0x00000000;
}
}

///////////////////////////////////////////////////////////////////////////
// Created on: 12-18-07
//
// Description: Converts a hex string into fixed point integer
//
// Parameters: string Hex, bool Signed
//
// Last modified: 01-08-08
//
///////////////////////////////////////////////////////////////////////////

int fixed_point::hexToFixedPoint(string Hex)
{
    int Value = 0;
    for(int i=0; i<Hex.length(); i++)//adds the hex to the leftmost
        {
            Value = Value << 4;
            Value = Value | charToHex(Hex[i]);
        }
    Value = Value << (8);//adjusts towards the middle
    return (Value);
}

//--------------pixel implementations--------------
pixel::pixel()
{
}
pixel::pixel(fixed_point r, fixed_point g, fixed_point b)
{
    _R = r;
    _G = g;
    _B = b;
}
pixel::~pixel()
{
}

pixel& pixel::operator = (const pixel& a)
{
    _R = a._R;
    _G = a._G;
    _B = a._B;
    return *this;
}

pixel operator + (const pixel& a, const pixel& b)
{
    pixel temp;
    temp._R = a._R + b._R;
    temp._G = a._G + b._G;
    temp._B = a._B + b._B;
    return temp;
}

pixel operator * (int a, const pixel& b)
{
    pixel temp;
    temp._R = a*b._R;
    temp._G = a*b._G;
    temp._B = a*b._B;
    return temp;
}

pixel operator / (const pixel& a, int b)
{
    pixel temp;
    temp._R = a._R/b;
    temp._G = a._G/b;
    temp._B = a._B/b;
    return temp;
}

//----------------------image implementations----------------------
image::image()
{
    _height = 0;
    _width = 0;
    _image_matrix = new pixel*[_height];
    for(int h = 0; h < _height; h++)
    {
        _image_matrix[h] = new pixel[_width];
    }
}

image::~image()
{
    delete2Darray (_image_matrix, _height);
}
image& image::operator = (const image& a)  
{
    delete2Darray(_image_matrix, _height);
    _height = a._height;
    _width = a._width;
    _image_matrix = new pixel*[_height];
    for (int i = 0; i < _height; i++)
    {
        _image_matrix[i] = new pixel[_width];
        for (int j = 0; j < _width; j++)
        {
            _image_matrix[i][j] = a._image_matrix[i][j];
        }
    }
    return *this;
}

void image::make_image(pixel** new_matrix, int new_height, int new_width)
{
    delete2Darray(_image_matrix, _height);
    _height = new_height;
    _width = new_width;
    _image_matrix = new pixel*[_height];
    for (int i = 0; i < _height; i++)
    {
        _image_matrix[i] = new pixel[_width];
        for (int j = 0; j < _width; j++)
        {
            _image_matrix[i][j] = new_matrix[i][j];
        }
    }
}

pixel image::make_pixel(fixed_point r, fixed_point g, fixed_point b)
{
    pixel temp_pixel(r, g, b);
    return temp_pixel;
}

fixed_point image::make_fixed_point(string hex_value)
{
    fixed_point temp_fixed_point(hex_value);
    return temp_fixed_point;
}

fixed_point image::make_fixed_point(int new_value)
{
    fixed_point temp_fixed_point(new_value);
    return temp_fixed_point;
}
void image::delete2Darray(pixel** array, int size)
{
    for (int i = 0; i < size; i++)
    {
        delete [] array[i];
    }
    delete [] array;
}

pixel** image::image_matrix()const
{
    pixel** temp;
    temp = new pixel*[_height];
    for (int i = 0; i < _height; i++)
    {
        temp[i] = new pixel[_width];
        for (int j = 0; j < _width; j++)
        {
            temp[i][j] = _image_matrix[i][j];
        }
    }
    return temp;
}

A2.1.7. image.h

#ifndef image_h
#define image_h

#include <cstdlib>
#include <string>
#include <fstream>
#include <cstdio>
#include <iostream>

using namespace std;

class fixed_point
{
private:
    int _value;
public:
    fixed_point();

Page 27
fixed_point(int);
fixed_point(string);

//operator overloading
fixed_point& operator = (const fixed_point&);
friend fixed_point operator + (const fixed_point&, const fixed_point&);
friend fixed_point operator * (int, const fixed_point&);
friend fixed_point operator / (const fixed_point&, int);

//accessor functions
int value() const {return _value;}
int decimal_value() const;

//general utility functions
int charToHex(char);
int hexToFixedPoint(string);
};

class pixel
{
private:
    fixed_point _R;
    fixed_point _G;
    fixed_point _B;
public:
    pixel();
    pixel(fixed_point,fixed_point,fixed_point);
    ~pixel();

    //operator overloading
    pixel& operator = (const pixel&);
    friend pixel operator + (const pixel&, const pixel&);
    friend pixel operator * (int, const pixel&);
    friend pixel operator / (const pixel&, int);

    //accessor functions
    fixed_point R()const {return _R;}
    fixed_point G()const {return _G;}
    fixed_point B()const {return _B;}
};

class image
{
private:
    pixel** _image_matrix;
    int _height;
    int _width;
public:
    image();
    ~image();
    image& operator = (const image&);//assignment operator overloading

    //accessor functions
int height() const {return _height;}
int width() const {return _width;}
pixel** image_matrix() const;

// input file parsing functions
int read_file(string);
// main parsing function
bool get_keyword(ifstream&, string);
// checks for the expected keyword
bool get_dimensions(ifstream&);
// gets the dimensions of the image
bool check_depth(ifstream&);
// checks if the bit depth for each r, g, b values equals eight
bool get_values(ifstream&);
// constructs the pixel matrix by getting the corresponding values

// general utility functions
pixel make_pixel(fixed_point, fixed_point, fixed_point);
// returns a pixel object on three fixed point object inputs
fixed_point make_fixed_point(string);
// returns a fixed point object on a string input
fixed_point make_fixed_point(int);
// returns a fixed point object on an integer input
void delete2Darray(pixel**, int);
// deallocates memory for the pixel matrix

// mutator function for the image object
void make_image(pixel**, int, int);

#endif

A2.1.8. main_frame.cpp

//---------------------------------------------------------------------
// Name:   main_frame.cpp
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Modified by: Lester Lim and John Vincent Bermudez
// Description: Partial implementation for the main_frame class
//              Part of Image Cropping Application - Elc152
//  
//---------------------------------------------------------------------

// Header files
#include "app.h"
#include "main_frame.h"
#include "child_frame.h"
define pi 3.14159265

DECLARE_APP(cropper_app)// for access to resizer_app class
wxMenuBar *CreateMenuBar();
wxMenu *file_menu = new wxMenu;
wxMenu *about_menu = new wxMenu;

file_menu->Append(MDI_OPEN, _T("&Open Image	CTRL+O"),
                  _T("Opens an image file"));
file_menu->Append(MDI_SAVE, _T("&Save Image	CTRL+S"),
                  _T("Saves the image file"));
file_menu->AppendSeparator();
file_menu->Append(CHILD_QUIT, _T("Close Image 	CTRL+X"),
                  _T("Close active image"));
file_menu->AppendSeparator();
file_menu->Append(MDI_QUIT, _T("&Quit Program	ALT+X"),
                  _T("Quit the application"));
about_menu->Append(MDI_ABOUT, _T("Image Cropping Application 
Help\tF1"),
                  _T("Image Cropping Application Help"));

wxMenuBar *menu_bar = new wxMenuBar;

menu_bar->Append(file_menu, _T("&File"));
menu_bar->Append(about_menu, _T("&Help"));
return menu_bar;
}

wxTextValidator *validator()
{
    //Checks input, if input not in array, error is sounded
    //Accepts only 0-9
    wxTextValidator *input_validator
        = new wxTextValidator(wxFILTER_INCLUDE_CHAR_LIST,NULL);
    wxArrayString array;
    array.Add("0"); array.Add("1"); array.Add("2"); array.Add("3");
    array.Add("4"); array.Add("5"); array.Add("6"); array.Add("7");
    array.Add("8"); array.Add("9"); array.Add("-");
    input_validator->SetIncludes(array);
    return input_validator;
};

BEGIN_EVENT_TABLE(main_frame,wxMDIParentFrame)
    EVT_SIZE(main_frame::OnSize)
    EVT_SASH_DRAGGED(SIDEBAR,main_frame::OnSashDrag)
    EVT_UPDATE_UI(wxID_ANY,main_frame::OnUpdate)
    EVT_MENU(wxID_ANY,main_frame::OnMenu)
    EVT_BUTTON(CROP_IMAGE,main_frame::OnApply)
    EVT_CLOSE(main_frame::OnClose)
END_EVENT_TABLE()
//Initialize Member Variables
_height_label = NULL;
_width_label = NULL;
_angle_label = NULL;
start_x = NULL;
start_y = NULL;
_height_input = NULL;
_width_input = NULL;
_angle_input = NULL;
crop_image = NULL;
tool_panel = NULL;
_image_properties = NULL;
_sidebar = NULL;
_image_opened = false;

//Creates the sidebar
_sidebar = CreateSash();
tool_panel = CreateToolPanel(_sidebar,wxEmptyString);
_sidebar->SizeWindows();

//Sets Program Icon
wxBitmap icon;
icon = wxBitmap(wxXmlResource::Get()->LoadBitmap(wxT("icon")));
wxIcon the_icon;
the_icon.CopyFromBitmap(icon);
SetIcon(the_icon);

//Creates keyboard shortcuts
wxAcceleratorEntry shortcuts[7];
shortcuts[0].Set(wxACCEL_CTRL, (int) 'o', MDI_OPEN);
shortcuts[1].Set(wxACCEL_CTRL, (int) 's', MDI_SAVE);
shortcuts[2].Set(wxACCEL_ALT, (int) 'x', MDI_QUIT);
shortcuts[3].Set(wxACCEL_NORMAL, WXK_F1, MDI_ABOUT);
shortcuts[4].Set(wxACCEL_CTRL, (int) 'x', CHILD_QUIT);
wxAcceleratorTable key_shortcuts(5,shortcuts);
SetAcceleratorTable(key_shortcuts);

//Create Status bar and toolbar
CreateToolBar(wxNO_BORDER | wxT_FLAT | wxT_HORIZONTAL);
CreateStatusBar();
InitToolBar(GetToolBar());

//Create Menu bar
SetMenuBar(CreateMenuBar());

main_frame::~main_frame(void) {
    DestroyChildren();
}
wxSashLayoutWindow* main_frame::CreateSash(void)
{
    wxSashLayoutWindow* temp = new wxSashLayoutWindow(this,SIDEBAR,
    wxDefaultPosition,wxSize(200,30),
    wxBORDER_NONE|wxSW_3DSASH |
    wxCLIP_CHILDREN);
    temp->SetDefaultSize(wxSize(200,30));
    temp->SetOrientation(wxLAYOUT_VERTICAL);
    temp->SetAlignment(wxLAYOUT_LEFT);
    temp->SetSashVisible(wxSASH_RIGHT,true);
    temp->SizeWindows();
    return temp;
}

wxFoldPanelBar* main_frame::CreateToolPanel(wxWindow *parent, wxString msg)
{
    //Creates the wxFoldPanelBar from the GUI items listed in the
    //class declaration
    wxFoldPanelBar *temp = new wxFoldPanelBar(parent, wxID_ANY,
    wxDefaultPosition, wxDefaultSize,
    wxFPB_DEFAULT_STYLE |wxFPB_VERTICAL,
    wxFPB_COLLAPSE_TO_BOTTOM);
    wxCaptionBarStyle cs;
    cs.SetCaptionStyle(wxCAPTIONBAR_RECTANGLE);
    wxFoldPanel panel_temp1 =
    temp->AddFoldPanel(_T("Properties:"),true,cs);
    _image_properties =
    new wxStaticText(panel_temp1.GetParent(),IMAGE_PROPERTIES,msg);
    temp->AddFoldPanelWindow(panel_temp1,
    _image_properties,wxFPB_ALIGN_WIDTH,5,20);
    wxFoldPanel panel_temp2 =
    temp->AddFoldPanel(_T("Cropping Options"),true,cs);
    start_x = new wxTextCtrl(panel_temp2.GetParent(),
    START_X,wxEmptyString,
    wxDefaultPosition,wxDefaultSize,wxTE_RIGHT,
    wxDefaultValidator,_T("x");
    start_y = new wxTextCtrl(panel_temp2.GetParent(),
    START_Y,wxEmptyString,
    wxDefaultPosition,wxDefaultSize,wxTE_RIGHT,
    wxDefaultValidator,_T("y");
    start_x_label = new wxStaticText(panel_temp2.GetParent(),
    wxID_ANY,_T("X Coordinate:"),wxDefaultPosition,
    wxDefaultSize,0,_T("X Coordinate");
    start_y_label = new wxStaticText(panel_temp2.GetParent(),
    wxDefaultPosition,wxDefaultSize,0,_T("Y Coordinate"));
wxID_ANY, _T("Y Coordinate:"), wxDefaultPosition, wxDefaultSize, 0, _T("Y Coordinate"));
_height_label = new wxStaticText(panel_temp2.GetParent(),
wxID_ANY, _T("Height:"), wxDefaultPosition, wxDefaultSize, 0, _T("Height"));
_width_label = new wxStaticText(panel_temp2.GetParent(),
wxID_ANY, _T("Width:"), wxDefaultPosition, wxDefaultSize, 0, _T("Height"));
_angle_label = new wxStaticText(panel_temp2.GetParent(),
wxID_ANY, _T("Angle in Degrees:"), wxDefaultPosition, wxDefaultSize, 0, _T("Angle"));
_height_input = new wxTextCtrl(panel_temp2.GetParent(),
HEIGHT_INPUT, wxEmptyString, wxDefaultPosition, wxDefaultSize, wxTE_RIGHT, wxDefaultValidator, _T("Height Input"));
_width_input = new wxTextCtrl(panel_temp2.GetParent(),
WIDTH_INPUT, wxEmptyString, wxDefaultPosition, wxDefaultSize, wxTE_RIGHT, wxDefaultValidator, _T("Width Input"));
_angle_input = new wxTextCtrl(panel_temp2.GetParent(),
ANGLE_INPUT, wxEmptyString, wxDefaultPosition, wxDefaultSize, wxTE_RIGHT, wxDefaultValidator, _T("Angle Input"));
start_x->SetValidator(*validator());
start_y->SetValidator(*validator());
_height_input->SetValidator(*validator());
_width_input->SetValidator(*validator());
_angle_input->SetValidator(*validator());
_crop_image = new wxButton(panel_temp2.GetParent(),
CROP_IMAGE, _T("Crop Image"), wxDefaultPosition, wxDefaultSize, 0);

temp->AddFoldPanelWindow(panel_temp2, start_x_label, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, start_x, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, start_y_label, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, start_y, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, _height_label, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, _height_input, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, _width_label, wxFPB_ALIGN_WIDTH, 5, 20);
temp->AddFoldPanelWindow(panel_temp2, _width_input, wxFPB_ALIGN_WIDTH, 5, 20);
void main_frame::OnUpdate(wxUpdateUIEvent& event)
{
    if(event.GetId()== MDI_SAVE)
        event.Enable(_image_opened && (wxGetApp().number_of_windows()));
    if(event.GetId()== CHILD_QUIT)
        event.Enable(_image_opened && (wxGetApp().number_of_windows()));
    if(event.GetId()== CROP_IMAGE)
        event.Enable(_image_opened && (wxGetApp().number_of_windows()));
}

void main_frame::InitToolBar(wxToolBar* toolBar)
{
    wxBitmap bitmaps[4];
    //Loads resources from resources.cpp
    bitmaps[0] = wxBitmap(wxXmlResource::Get()->LoadBitmap(wxT("open")));
    bitmaps[1] = wxBitmap(wxXmlResource::Get()->LoadBitmap(wxT("save")));
    bitmaps[2] = wxBitmap(wxXmlResource::Get()->LoadBitmap(wxT("about")));
    bitmaps[3] = wxBitmap(wxXmlResource::Get()->LoadBitmap(wxT("close")));

    int x, y;
    x=bitmaps[0].GetWidth();
y=bitmaps[0].GetHeight();

    toolBar->SetToolBitmapSize(wxSize(x,y));
    toolBar->AddTool(MDI_OPEN,_T("Open"),
        bitmaps[0],wxNullBitmap,
        wxITEM_NORMAL,_T("Open File"), _T("Opens an image file"));
    toolBar->AddTool(MDI_SAVE,_T("Save"),
        bitmaps[1],wxNullBitmap,
        wxITEM_NORMAL,_T("Save File"), _T("Saves the image"));
    toolBar->AddSeparator();
    toolBar->AddTool(MDI_ABOUT,_T("About"),bitmaps[2],wxNullBitmap,
        wxITEM_NORMAL,_T("Help"), _T("Help on the program"));
    toolBar->AddTool(MDI_QUIT,_T("Close"),bitmaps[3],wxNullBitmap,
        wxITEM_NORMAL,_T("Close Application"), _T("Quit the application"));

    toolBar->Realize();
}

wxString main_frame::get_x_input(void)
{
wxString main_frame::get_y_input(void)
{
    return start_y->GetValue();
}

wxString main_frame::get_height_input(void)
{
    return _height_input->GetValue();
}

wxString main_frame::get_width_input(void)
{
    return _width_input->GetValue();
}

wxString main_frame::get_angle_input(void)
{
    return _angle_input->GetValue();
}

void main_frame::set_height_label(wxString label)
{
    _height_input->ChangeValue(label);
}

void main_frame::set_width_label(wxString label)
{
    _width_input->ChangeValue(label);
}

void main_frame::set_angle_label(wxString label)
{
    _angle_input->ChangeValue(label);
}

void main_frame::set_image_properties(wxString label)
{
    // Error occurred using _image_properties->SetLabel()
    // Solution was to recreate tool panel with the
    // _image_properties label as parameter
    _sidebar->DestroyChildren();
    _tool_panel = CreateToolPanel(_sidebar,label);
    _sidebar->SizeWindows();
    wxFoldPanel temp(0);

    for(size_t i = 0; i<_tool_panel->GetCount();i++)
    {
        temp = _tool_panel->Item(i);
        _tool_panel->Expand(temp);
    }
}
void main_frame::OnSize(wxSizeEvent& event)
{
    wxLayoutAlgorithm layout;
    layout.LayoutMDIFrame(this);
}

void main_frame::OnSashDrag(wxSashEvent& event)
{
    if (event.GetDragStatus() == wxSASH_STATUS_OUT_OF_RANGE)
        return;
    _sidebar->SetDefaultSize(wxSize(event.GetDragRect().width, 1000));
    wxLayoutAlgorithm layout;
    layout.LayoutMDIFrame(this);
    GetClientWindow()->Refresh();
}

void main_frame::OnMenu(wxCommandEvent& event)
{
    //selects handler function for the specific menu ID
    switch(event.GetId())
    {
    case MDI_OPEN:
        menu_OpenFile();
        break;
    case MDI_SAVE:
        menu_Save();
        break;
    case MDI_ABOUT:
        menu_About();
        break;
    case MDI_QUIT:
        menu_Quit();
        break;
    }
}

void main_frame::menu_OpenFile()
{
    wxString wildcards;
    //allowed image formats
    wildcards = _T("Image Files (*.bmp;*.jpeg;*.jpg;*.png;*.gif)")
                        _T("|*.bmp;*.jpeg;*.jpg;*.png;*.gif")
                        _T("|BMP files (*.bmp)|*.bmp")
                        _T("|JPEG files (*.jpeg;*.jpg)|*.jpeg;*.jpg")
                        _T("|PNG files (*.png)|*.png")
                        _T("|GIF files (*.gif)|*.gif");
    wxFileDialog* open_file_dlg = new wxFileDialog(this,wxT("Choose image file"),wxEmptyString,wxEmptyString,wildcards<wxFD_OPEN|wxFD_FILE_MUST_EXIST|
                            wxFD_CHANGE_DIR|wxCENTRE);
if(open_file_dlg->ShowModal()==wxID_OK)
{
    wxString filename,path;
    filename=open_file_dlg->GetFilename();
    path = open_file_dlg->GetPath();
    wxFileName the_file(path,filename);
    if(!open_image(filename))
        show_error(_T("Error opening image file"));
}
open_file_dlg->Destroy();
Refresh();
}

void main_frame::menu_Save()
{
    wxString wildcards;
    wildcards =  _T("|BMP file *.bmp|*.bmp")
        _T("|JPEG file *.jpeg;*.jpg|*.jpeg;*.jpg")
        _T("|PNG file *.png|*.png");
    wxFileDialog* save_file_dlg = new wxFileDialog(this,wxT("Save Image file"), wxEmptyString,wxEmptyString,
    wildcards,wxFD_SAVE |wxFD_OVERWRITE_PROMPT|
    wxFD_FILE_MUST_EXIST|wxFD_CHANGE_DIR|
    wxCENTRE);
    if(save_file_dlg->ShowModal()==wxID_OK)
    {
        wxString filename,path;
        filename=save_file_dlg->GetFilename();
        path = save_file_dlg->GetPath();
        wxFileName the_file(path,filename);
        if(!save_image(filename))
            show_error(_T("Save Error"));
    }
    save_file_dlg->Destroy();
}

void main_frame::menu_About()
{
    //opens the default html browser of the system to load index.html
    //which should be on the same directory as the executable
    wxString path =
    ((wxStandardPaths&)wxStandardPaths::Get()).GetDataDir();
    wxString html = path+wxT("\index.html");
    wxLaunchDefaultBrowser(html);
}

void main_frame::menu_Quit()
{
    Close();
}
void main_frame::OnClose(wxCloseEvent& event)
{
    if(event.CanVeto() && (wxGetApp().number_of_windows() > 0))
    {
        wxString message;
        message.Printf(_T("%d windows still open, proceed with exit?") ,
                wxGetApp().number_of_windows());
        if(wxMessageBox(message, _T("Please confirm") ,
                wxICON_QUESTION|wxYES_NO) != wxYES)
        {
            event.Veto();
            return;
        }
        else
        {
            // For closing all child windows
            wxMDIChildFrame *pChild = GetActiveChild();
            int nChildren = GetChildren().GetCount();
            int nCount = 0;

            while (nCount < nChildren-1)
            {
                if (pChild->Close( false ) == false)
                {
                    return;
                }
                nCount++;
                ActivateNext();
                pChild = GetActiveChild();
            }
        }
    }
    event.Skip();
}

void main_frame::OnApply(wxCommandEvent& event)
{
    image to_be_cropped, cropped;
    wxString name_of_image;
    int input_height, input_width, input_angle, x_start, y_start,
        base_height, base_width, c, d, g, h, s, t;
    double input_radian;

to_be_cropped = get_active_image();
base_height = to_be_cropped.height();
base_width = to_be_cropped.width();

    x_start = wxAtoi(get_x_input());
y_start = wxAtoi(get_y_input());
input_height = wxAtoi(get_height_input());
input_width = wxAtoi(get_width_input());
input_angle = wxAtoi(get_angle_input());
//Convert angle in degrees to radians
input_radian = input_angle*(pi/180);

if(!(input_height == 0 || input_width == 0))
{
    //Boundary checking
    //if rotation is positive, counterclockwise
    if (input_radian >= 0)
    {
        //upper rightmost pixel
        c = x_start + input_width*cos(input_radian);
        d = y_start + input_width*sin(input_radian);
        //lower leftmost pixel
        g = x_start - input_height*sin(input_radian);
        h = y_start + input_height*cos(input_radian);
        //lower rightmost pixel
        s = x_start + (input_width*cos(input_radian) -
                      input_height*sin(input_radian));
        t = y_start + (input_height*cos(input_radian) +
                      input_width*sin(input_radian));
    }
    //if rotation is clockwise, angle is negative
    else {
        //upper rightmost pixel
        c = x_start + input_width*cos(input_radian);
        d = y_start - input_width*sin(input_radian);
        //lower leftmost pixel
        g = x_start + input_height*sin(input_radian);
        h = y_start + input_height*cos(input_radian);
        //lower rightmost pixel
        s = x_start + (input_width*cos(input_radian) +
                      input_height*sin(input_radian));
        t = y_start + (input_height*cos(input_radian) -
                      input_width*sin(input_radian));
    }

    if ((c>=0)&&(c<=base_width-1)&&(d>=0)&&(d<=base_height-1)
        &&(g>=0)&&(g<=base_width-1)&&(h>=0)&&(h<=base_height-1)
        &&(s>=0)&&(s<=base_width-1)&&(t>=0)&&(t<=base_height-1))
    {
        wxMDIChildFrame* active_frame = this->GetActiveChild();
        child_frame *active_child =
            wxDynamicCast(active_frame,child_frame);
        name_of_image = _T("Cropped ") + active_child->
            return_image_name();
        cropped = CropAtAnAngle_BLinterpolation(to_be_cropped,input_height,
                                                input_width, input_radian, x_start, y_start);
        open_new_child(name_of_image,cropped);
    }
    else
    {
        show_error(_T("Parameters are outside the image boundary"));
    }
    else
    {
        show_error(_T("A zero in the input dimension is not allowed"));
    }
}
#ifndef main_frame_h
#define main_frame_h

//wxWidgets libraries included
#include <wx/wx.h>
#include <wx/toolbar.h>
#include <wx/foldbar/foldpanelbar.h>
#include <wx/laywin.h>
#include <wx/mdi.h>
#include <wx/menuitem.h>
#include <wx/app.h>
#include <wx/init.h>
#include <wx/icon.h>
#include <wx/valtext.h>
#include <wx/filedlg.h>
#include <wx/progdig.h>
#include <wx/filename.h>
#include <wx/xrc/xmlres.h>
#include <wx/msw/helpchm.h>
#include <wx/utils.h>
#include <wx/stdpaths.h>

//other includes
#include "image.h"

//global functions
wxMenuBar *CreateMenuBar();
wxTextValidator *validator();

//enumeration of wxIDs (identifies GUI items)
enum {
    SIDEBAR=100,
    IMAGE_PROPERTIES,
    HEIGHT_INPUT,
    WIDTH_INPUT,
    ANGLE_INPUT,
    CROP_IMAGE,
    START_X,
    START_Y,
class main_frame : public wxMDIParentFrame
{
private:
  //bool flags
  bool _image_opened;

  //GUI items
  wxTTextCtrl *start_x;
  wxTTextCtrl *start_y;
  wxStaticText *image_properties;
  wxFoldPanelBar *tool_panel;
  wxStaticText *start_x_label;
  wxStaticText *start_y_label;
  wxStaticText *height_label;
  wxStaticText *width_label;
  wxStaticText *angle_label;
  wxTTextCtrl *height_input;
  wxTTextCtrl *width_input;
  wxTTextCtrl *angle_input;
  wxButton *crop_image;
  //End GUI Items

protected:
  wxSashLayoutWindow* _sidebar;
public:
  //Constructor/Deconstructor
  main_frame(wxWindow *parent, const wxWindowID id,
               const wxString& title, const wxPoint& pos,
               const wxSize& size, const long style);
  ~main_frame(void);

  //GUI related functions
  void OnUpdate(wxUpdateUIEvent& event);
  void InitToolBar(wxToolBar* toolBar);
  wxSashLayoutWindow* CreateSash(void);
  wxFoldPanelBar* CreateToolPanel(wxWindow *parent, wxString msg);

  //User Input Related functions -GUI related
  void OnSize(wxSizeEvent & event);
  void OnSashDrag(wxSashEvent & event);

  //Accessor functions to this class
  wxString get_x_input(void);
  wxString get_y_input(void);
  wxString get_height_input(void);
  wxString get_width_input(void);
}

//Menu Items
MDI_OPEN,
MDI_SAVE,
MDI_ABOUT=wxID_ABOUT,
MDI_QUIT =wxID_EXIT,
}
wxString get_angle_input(void);
void set_height_label(wxString label);
void set_width_label(wxString label);
void set_image_properties(wxString label);

// User Input Related functions
void OnApply(wxCommandEvent& event);
void OnMenu(wxCommandEvent& event);
void menu_About();
void menu_OpenFile();
void menu_Save();
void menu_Quit();
void OnClose(wxCloseEvent& event);

// Utility functions
void show_error(wxString msg);
void open_new_child(const wxString& title,
                    const image& the_image);
bool open_image(wxString filename);

// Error handling is in function
bool save_image(wxString filename);
image CropAtAnAngle_BlInterpolation(const image& input_image,
                                   int input_height, int input_width,
                                   double input_angle, int x_start,
                                   int y_start);

image& get_active_image(void);
DECLARE_EVENT_TABLE()}

#endif

A2.1.10. main_frame_contd.cpp

//---------------------------------------------------------------------
// Name: main_frame_contd.cpp
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Modified by: Lester Lim and John Vincent Bermudez
// Description: Partial Implementation for the main_frame class
//               Part of Image Cropping Application - E1c152
//
//---------------------------------------------------------------------

// Header files
#include "app.h"
#include "main_frame.h"
#include "child_frame.h"
#include "image.h"

DECLARE_APP(cropper_app)
//Error Dialog
void main_frame::show_error(wxString msg)
{
    wxString error_dialog = new wxMessageDialog(this,
        msg, wxT("ERROR!!!!"), wxOK|wxCENTRE|wxICON_ERROR);
    error_dialog->ShowModal();
    error_dialog->Destroy();
}

//Open New Child Window
void main_frame::open_new_child(const wxString& title,
    const image& the_image)
{
    child_frame *sub_frame = new child_frame(this,title,the_image);
    wxGetApp().increase();
    sub_frame->Show(true);
    _image_opened = true;
    wxBitmap icon;
    icon = wxBitmap(wxXmlResource::Get()->LoadBitmap(wxT("child")));
    wxIcon the_icon;
    the_icon.CopyFromBitmap(icon);
    sub_frame->setIcon(the_icon);
    wxFoldPanel temp(0);
    for(size_t i = 0; i<_tool_panel->GetCount();i++)
    {
        temp = _tool_panel->Item(i);
        _tool_panel->Expand(temp);
    }
    Refresh();
}

//opens an image
bool main_frame::open_image(wxString filename)
{
    wxImage temp_image(filename, wxBITMAP_TYPE_ANY,-1);
    if(temp_image.IsOk())
    {
        image extracted_image;
        int height, width;
        pixel** matrix;
        pixel temp_pixel;
        fixed_point r,g,b;
        int int_R,int_G,int_B;
        height = temp_image.GetHeight();
        width = temp_image.GetWidth();
        matrix = new pixel*[height];

        for(int s = 0; s<height; s++)
        {
            matrix[s] = new pixel[width];
        }

        for(int i=0; i<height;i++)
        {
            for(int j=0; j<width;j++)
            {
                int_R = temp_image.GetRGBValue(i,j,0);
                int_G = temp_image.GetRGBValue(i,j,1);
                int_B = temp_image.GetRGBValue(i,j,2);
                r = static_cast<fixed_point>(int_R);
                g = static_cast<fixed_point>(int_G);
                b = static_cast<fixed_point>(int_B);
                temp_pixel = (pixel)(r,g,b);
                matrix[i][j] = temp_pixel;
            }
        }
    }
}
for(int j=0; j<width;j++)
{
    int_R = temp_image.GetRed(j,i);
    int_G = temp_image.GetGreen(j,i);
    int_B = temp_image.GetBlue(j,i);
    r = extracted_image.make_fixed_point(int_R);
    g = extracted_image.make_fixed_point(int_G);
    b = extracted_image.make_fixed_point(int_B);
    temp_pixel = extracted_image.make_pixel(r,g,b);
    matrix[i][j] = temp_pixel;
}
}

extracted_image.make_image(matrix,height,width);
open_new_child(filename,extracted_image);
return true;
}
else
    return false;
}

//Save an image
bool main_frame::save_image(wxString filename)
{
    wxMDIChildFrame* active_frame = this->GetActiveChild();
    child_frame *active_child = wxDynamicCast(active_frame,child_frame);
    wxBitmap* temp = active_child->return_canvas()->return_bitmap();
    wxImage temp_image = temp->ConvertToImage();
    return temp_image.SaveFile(filename);
}

image& main_frame::get_active_image(void)
{
    //returns the image of the active window
    wxMDIChildFrame* active_frame = this->GetActiveChild();
    child_frame *active_child = wxDynamicCast(active_frame,child_frame);
    return active_child->return_canvas()->return_image();
}

image main_frame::CropAtAnAngle_BLinterpolation(const image& input_image, int input_height, int input_width,
    double input_radian, int x_start, int y_start)
{
    //create temporary image and get input image information
    image temp;
    pixel** base_matrix;
    int base_height, base_width;
    base_matrix = input_image.image_matrix();
    base_height = input_image.height();
    base_width = input_image.width();

    //allocate temporary pixel array according to new dimensions
    pixel** temp_matrix;
    temp_matrix = new pixel*[input_height];
    for(int s = 0; s<input_height; s++)
temp_matrix[s] = new pixel[input_width];

//bilinear interpolation algorithm
int i, j, e, f, l, m;
pixel temp1, temp2;

for(int t = 0; t<input_height; t++)
{
    for(int u = 0; u<input_width; u++)
    {
        //if rotation is counter-clockwise, angle is positive
        if (input_radian >= 0){
            e = x_start + (u*cos(input_radian) - t*sin(input_radian));
            f = y_start + (t*cos(input_radian) + u*sin(input_radian));
        }
        //if rotation is clockwise, angle is negative
        else {
            e = x_start + (u*cos(input_radian) + t*sin(input_radian));
            f = y_start + (t*cos(input_radian) - u*sin(input_radian));
        }

        //to obtain the greatest integer of the rotated coordinates
        for (l=0; e>=1; l++)
        {
            e = e-1;
        }
        for (m=0; f>=1; m++)
        {
            f = f-1;
        }

        //to obtain the fractional part
        i = e/l;
        j = f%m;

        //linear interpolation between the two upper base pixels
        //and two lower base pixel
        temp1 = (1-i) * base_matrix[l][m] + i * base_matrix[l+1][m];
        temp2 = (1-i) * base_matrix[l][m+1] + i * base_matrix[l+1][m+1];

        //linear interpolation between the two interpolated points
        //from the previous linear interpolations
        temp_matrix[u][t] = (1-j)*temp1 + j*temp2;
    }
}

//mutate temporary image and return it
temp.make_image(temp_matrix, input_height, input_width);
return temp;
A2.1.11. my_canvas.cpp

//---------------------------------------------------------------------
//
// Name:   my_canvas.cpp
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Used by: Lester Lim and John Vincent Bermudez
// Description: Implementation for the my_canvas class
//          Part of Image Cropping Application - Elc152
//
//---------------------------------------------------------------------

//Header files
#include "my_canvas.h"
#include "image.h"
#include "child_frame.h"
#include <wx/dcbuffer.h>
#include <wx/dcclient.h>
#include <wx/dcmemory.h>
#include <wx/image.h>
#include <wx/wx.h>

my_canvas::my_canvas(wxWindow *parent, const wxString& title,
                      const image& temp_image)
    : wxScrolledWindow(parent,wxID_ANY,wxDefaultPosition,
                      wxDefaultSize,wxNO_FULL_REPAINT_ON_RESIZE)
{
    _the_parent = parent;
    _the_image = temp_image;
    initialize_buffer();
    this->SetClientSize(wxSize(_image_buffer->GetWidth(),
                                _image_buffer->GetHeight()));
    this->SetScrollbars(20, 20, 20, 20);
    //For scrollbars to know when scrollbars are needed
    //If this is larger than SetSize, scrollbars are produced
    this->SetVirtualSize(GetClientSize());
    this->SetSize(GetBestSize());
}

my_canvas::~my_canvas(void)
{
}

//For display of the image
void my_canvas::OnDraw(wxDC& dc)
{
    int width = 0, height = 0;
    int x = 0, y = 0, image_height, image_width;
    this->GetClientSize(&width,&height);
    image_height = _the_image.height();
    image_width = _the_image.width();

    //Algorithm to calculate so that image is always at the center of window

if(width < image_width)
    x = 0;
else
    x = (width - image_width)/2;

// Algorithm to calculate so that image is always at the center of window
if(height < image_height)
    y = 0;
else
    y = (height - image_height)/2;

child_frame *parent_frame = wxDynamicCast(_the_parent, child_frame);

// Quicker render of display as the bitmap is drawn as a whole
dc.DrawBitmap(*_image_buffer, x, y, false);
}

void my_canvas::initialize_buffer()
{
    // Creates an wxImage from the image object
    // Access each RGB values of image object in decimal form
    // and input it as the wxImage's RGB values
    wxImage *temp_image = new wxImage(_the_image.width(),
    _the_image.height());
    pixel** temp_matrix;
    fixed_point r, g, b;
    temp_matrix = _the_image.image_matrix();
    for(int i=0; i<_the_image.height(); i++)
    {
        for(int j=0; j<_the_image.width(); j++)
        {
            r = temp_matrix[i][j].R();
            g = temp_matrix[i][j].G();
            b = temp_matrix[i][j].B();
            temp_image->
                SetRGB(j, i, r.decimal_value(), g.decimal_value(), b.decimal_value());
        }
    }
    _image_buffer = new wxBitmap(*temp_image, -1);
}

int my_canvas::access_height(void)
{
    return _the_image.height();
}

int my_canvas::access_width(void)
{
    return _the_image.width();
}

image& my_canvas::return_image(void)
{

Page 47
return _the_image;
}

wxBitmap* my_canvas::return_bitmap(void)
{
    return _image_buffer;
}

A2.1.12. my_canvas.h

//-------------------------------------------------------------------------------
//
// Name:   my_canvas.h
// Author: Joshua Esmenda, Mark Andrew Mateo, Jerome Paez
// Used by: Lester Lim and John Vincent Bermudez
// Description: Declaration for the my_canvas class
//               Part of Image Cropping Application - Elc152
//
//-------------------------------------------------------------------------------

#ifndef my_canvas_h
#define my_canvas_h

#include <wx/wx.h>
#include <wx/scrolwin.h>
#include <wx/object.h>
#include <wx/bitmap.h>
#include <wx/icon.h>
#include "image.h"

class my_canvas : public wxScrolledWindow
{
    private:
        wxWindow * _the_parent; // pointer to parent, for quick access
        image _the_image;
        wxBitmap* _image_buffer; // for quicker rendering

    public:
        // Constructor/Deconstructor:
        my_canvas(wxWindow *parent, const wxString& title,
                   const image& temp_image);
        ~my_canvas(void);
        // GUI related
        // display the image upon construction of class:
        virtual void OnDraw(wxDC& dc);
        void initialize_buffer(); // makes wxBitmap
        // Accessor functions:
        int access_height(void);
        int access_width(void);
        image& return_image(void);
        wxBitmap* return_bitmap(void);
    }
#endif
A2.1.13. resources.cpp

// Name: resources.cpp
// Author: Joshua Esmeenda, Mark Andrew Mateo, Jerome Paez
// Used by: Lester Lim and John Vincent Bermudez
// Description: Resource file: for embedding images to binary
// Part of Image Cropping Application - Elc152

// This file was automatically generated by wxrc, do not edit by hand.

#include <wx/wxprec.h>

#ifndef __BORLANDC__
  #pragma hdrstop
#endif

#include <wx/filesys.h>
#include <wx/fs_mem.h>
#include <wx/xrc/xmlres.h>
#include <wx/xrc/xh_all.h>

#if wxCHECK_VERSION(2,8,5) && wxABI_VERSION >= 20805
  #define XRC_ADD_FILE(name, data, size, mime) \
    wxMemoryFSHandler::AddFileWithMimeType(name, data, size, mime)
#else
  #define XRC_ADD_FILE(name, data, size, mime) \
    wxMemoryFSHandler::AddFile(name, data, size)
#endif

static size_t xml_res_size_0 = 2070;
static unsigned char xml_res_file_0[] = {
  137,80,78,71,13,10,26,10,0,0,0,13,73,72,68,82,0,0,0,32,0,0,0,32,8,6,0,0, 
  0,115,122,122,244,0,0,0,4,115,66,73,84,8,8,8,124,8,100,136,0,0,0,9,11, 
  2,72,89,115,0,0,27,175,0,0,27,175,1,94,26,145,28,0,0,0,25,116,69,88,116,8, 
  3,111,102,116,119,97,114,101,0,119,119,119,46,105,110,107,115,99,97,112,1, 
  01,46,111,114,103,155,238,60,26,0,0,7,147,73,68,65,84,88,133,197,151,125,7, 
  6,213,215,25,199,63,231,229,7,8,162,192,5,189,112,17,185,82,152,216,130,1, 
  04,78,55,9,174,91,77,103,199,236,40,171,53,179,203,218,180,46,113,51,173,9, 
  3,172,21,59,99,135,214,249,22,95,8,5,223,136,213,59,169,92,177,32,47,242, 
  254,114,239,221,31,191,23,238,197,85,93,255,217,47,57,255,252,114,206,2, 
  49,
230,233,89,153,164,140,29,69,75,71,47,191,120,253,99,123,122,133,221,174,
37,75,200,74,79,26,78,42,66,32,184,19,64,10,34,242,69,198,184,120,86,47,
154,66,158,215,69,48,24,162,116,219,1,90,219,187,192,108,201,54,59,224,
95,215,148,22,60,152,78,249,139,139,153,152,154,64,32,16,100,119,213,5,
62,60,124,9,169,70,36,24,59,210,173,178,27,19,165,153,95,48,129,217,185,
110,164,20,180,220,238,163,244,205,3,28,60,126,17,238,214,148,134,65,56,
109,121,108,140,193,75,171,139,248,94,81,30,82,10,154,219,123,56,117,193,
79,221,213,175,104,235,236,167,171,119,16,195,80,36,143,141,99,124,82,28,
15,122,147,121,200,235,98,76,108,20,161,80,136,35,181,141,172,127,109,15,
45,102,146,187,119,91,62,194,18,206,195,228,241,130,73,188,186,102,33,233,
227,198,68,204,179,215,138,17,81,217,220,222,195,111,222,57,204,222,170,
90,251,215,253,63,76,194,32,238,120,154,205,204,203,96,241,220,41,20,228,
122,72,26,27,75,66,124,12,2,232,232,238,167,189,179,143,154,203,77,236,
61,124,158,163,103,175,216,219,124,179,167,217,8,144,255,207,227,244,191,
128,216,207,243,37,64,46,144,6,216,189,88,35,166,143,47,2,123,249,31,158,
231,255,1,116,233,162,174,95,160,148,171,0,0,0,73,69,78,68,174,66,96,130};

static size_t xml_res_size_1 = 2140;
static unsigned char xml_res_file_1[] = {
137,80,78,71,13,10,26,10,0,0,0,13,73,72,68,82,0,0,0,32,0,0,0,32,8,6,0,0,0,
115,122,122,244,0,0,0,8,35,73,68,65,84,120,218,237,151,107,112,84,103,25,
199,255,231,178,151,179,201,18,178,201,38,155,128,104,135,41,23,41,147,
64,24,101,80,26,2,185,192,52,66,33,182,37,69,32,130,98,219,65,138,45,99,
85,132,218,33,29,219,177,227,104,173,86,109,189,116,196,15,126,161,21,79,
90,178,101,91,74,65,170,88,166,78,33,14,5,90,104,2,52,23,178,247,61,187,
231,230,255,61,27,58,140,206,50,249,226,232,7,119,230,100,79,118,223,243,
190,191,231,121,254,207,101,37,199,113,240,223,124,73,255,115,0,181,53,
53,234,156,219,230,182,70,106,107,166,154,166,105,23,87,73,255,254,224,
13,159,75,238,187,184,28,233,198,175,193,189,85,143,87,61,125,230,221,183,
79,189,125,234,47,19,2,232,88,177,188,98,211,221,247,30,173,145,180,185,
217,108,134,123,144,193,118,184,35,247,148,199,23,137,115,248,185,101,243,
130,3,177,131,101,241,222,41,222,75,252,222,226,122,211,50,17,169,139,224,
228,197,254,95,239,220,245,221,205,19,2,104,93,222,81,217,189,98,101,180,
60,145,111,202,229,178,144,5,128,101,127,108,152,56,66,60,34,14,183,5,128,
101,193,52,12,152,182,69,0,113,48,217,80,4,40,228,243,168,173,175,197,0,
50,207,61,254,196,247,183,76,8,96,105,123,123,229,166,59,187,250,106,10,
static unsigned char xml_res_file_3[] = {
137,80,78,71,13,10,26,10,0,0,0,13,73,72,68,82,0,0,0,32,0,0,0,32,8,6,0,0,
0,115,122,122,244,0,0,144,73,68,65,84,120,218,205,151,121,112,148,229,
29,199,63,123,239,102,119,179,185,19,72,64,36,17,53,92,38,38,165,80,14,
5,180,10,104,128,170,180,83,15,180,210,58,56,142,211,209,142,224,53,83,
188,102,80,28,116,164,118,170,162,82,41,99,145,10,11,37,160,218,69,68,72,
131,7,4,18,174,208,36,155,132,100,143,236,125,188,251,94,125,226,216,76,
223,26,248,129,135,225,255,69,192,120,97,163,46,240,94,29,194,90,206,37,
224,6,93,215,189,23,244,164,6,195,141,226,177,245,188,2,234,231,150,18,
215,204,68,67,58,214,108,22,255,137,16,229,181,165,88,242,173,184,92,6,
252,7,253,232,186,1,79,153,3,205,100,167,189,201,143,42,41,228,184,44,2
04,
7,
199,242,234,3,155,190,155,128,185,247,253,24,41,199,68,232,116,154,76,9
5,
47,33,95,12,57,107,228,162,154,66,130,9,3,69,110,3,201,80,146,164,12,11
8,
131,145,104,123,138,234,105,149,88,29,25,74,242,114,176,149,20,16,75,23
5,
152,226,42,189,221,33,170,38,143,227,243,86,31,45,175,238,249,110,2,110,
91,121,11,93,61,167,40,27,93,130,210,27,228,189,109,173,160,155,153,48,
115,4,190,147,17,102,150,217,200,143,38,249,236,120,20,95,105,17,11,102,
8,
249,246,26,252,194,59,41,33,184,247,253,246,239,38,96,193,147,139,56,21
7,
210,134,51,207,202,216,138,114,62,218,190,139,21,197,9,42,107,174,35,57,
231,97,170,159,152,206,113,145,74,7,127,114,19,214,189,219,80,175,28,19
8,
6,171,141,96,159,134,197,166,227,52,106,52,109,108,7,19,92,181,184,154,
222,176,68,70,202,210,185,173,235,127,11,208,52,205,123,255,27,203,104,
220,188,133,162,66,27,229,195,75,232,63,216,198,6,123,15,7,18,53,20,223,
189,140,220,39,22,17,172,155,134,165,102,58,193,21,79,177,107,220,48,2,
227,28,68,50,98,243,162,2,92,109,49,62,223,237,99,32,162,80,85,91,66,24
1,
172,82,34,89,153,247,86,29,194,104,52,158,95,128,170,170,222,197,203,23
1,
16,140,133,144,146,126,210,201,12,29,190,12,197,106,150,235,47,154,202,
245,161,102,122,251,163,244,143,159,195,176,3,219,217,239,44,166,207,14
6,
9,
89,91,18,231,101,110,140,69,133,124,178,169,29,147,201,116,126,1,138,16
2,
120,199,212,122,48,219,33,45,153,209,36,21,57,37,33,169,26,30,35,220,10
1,
9,
164,12,22,2,207,63,77,56,16,224,244,37,35,80,210,1,12,121,78,18,113,5,1
27,
71,148,183,55,108,102,254,252,249,79,138,125,254,249,95,123,139,96,210,
50,36,64,150,101,239,228,91,39,18,238,56,77,36,32,17,108,79,94,72,44,12,
static size_t xml_res_size_4 = 1474;
static unsigned char xml_res_file_4[] = {

};

Page 60

static size_t xml_res_size_5 = 997;
static unsigned char xml_res_file_5[] = {
137,80,78,71,13,10,26,10,0,0,0,13,73,72,68,82,0,0,0,32,0,0,0,32,8,6,0,0,

static size_t xml_res_size_6 = 1083;
static unsigned char xml_res_file_6[] = {
60, 63, 120, 109, 108, 32, 118, 101, 114, 115, 105, 111, 110, 61, 34, 49, 46, 48, 34, 32, 101,
110, 99, 111, 100, 105, 110, 103, 61, 34, 85, 84, 70, 45, 56, 34, 63, 62, 10, 60, 114, 101,
115, 115, 61, 34, 119, 120, 66, 105, 116, 109, 97, 112, 34, 32, 110, 97, 109, 101, 61, 34,
16, 116, 105, 110, 103, 115, 95, 74, 86, 83, 80, 95, 68, 101, 115, 107, 116, 111, 112, 95, 80,
114, 111, 106, 101, 99, 116, 115, 95, 84, 72, 69, 32, 73, 77, 65, 71, 69, 32, 82, 69, 83, 73,
90, 69, 82, 95, 98, 105, 116, 109, 97, 112, 115, 95, 97, 98, 111, 117, 116, 46, 112, 110, 110,
108, 97, 115, 115, 61, 34, 119, 120, 66, 105, 116, 109, 97, 112, 34, 32, 110, 97, 109, 101,
112, 112, 36, 67, 95, 95, 68, 111, 99, 117, 109, 101, 110, 116, 115, 32, 97, 110, 100, 32,
112, 95, 80, 114, 111, 106, 101, 99, 116, 115, 95, 84, 72, 69, 32, 73, 77, 65, 71, 69, 32, 82,
69, 83, 73, 90, 69, 82, 95, 98, 105, 116, 109, 97, 112, 115, 95, 99, 104, 105, 108, 100, 46,
10, 100, 32, 83, 101, 116, 116, 105, 110, 103, 115, 95, 74, 86, 83, 80, 95, 68, 101, 115, 107,
116, 111, 112, 95, 80, 114, 111, 106, 101, 99, 116, 115, 95, 84, 72, 69, 32, 73, 77, 65, 71,
69, 32, 82, 69, 83, 73, 90, 69, 82, 95, 98, 105, 116, 109, 97, 112, 115, 95, 99, 108, 111, 115,
10, 101, 46, 112, 110, 103, 60, 47, 111, 98, 106, 101, 99, 116, 62, 10, 32, 32, 60, 111, 98, 106,
void InitXmlResource()
{
    // Check for memory FS. If not present, load the handler:
    wxMemoryFSHandler::AddFile(wxT("XRC_resource/dummy_file"), wxT("dummy one")));
    wxFileSystem fsys;
}
wxFSFile *f =
fsys.OpenFile(wxT("memory:XRC_resource/dummy_file"));
wxMemoryFSHandler::RemoveFile(wxT("XRC_resource/dummy_file"));
if (f) delete f;
else wxFileSystem::AddHandler(new wxMemoryFSHandler);
}

XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$C__Documents and
Settings_JVSP/Desktop_Projects_THE IMAGE RESIZER_bitmaps_about.png"),
xml_res_file_0, xml_res_size_0, _T("image/png"));
XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$C__Documents and
Settings_JVSP/Desktop_Projects_THE IMAGE RESIZER_bitmaps_child.png"),
xml_res_file_1, xml_res_size_1, _T("image/png"));
XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$C__Documents and
Settings_JVSP/Desktop_Projects_THE IMAGE RESIZER_bitmaps_close.png"),
xml_res_file_2, xml_res_size_2, _T("image/png"));
XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$C__Documents and
Settings_JVSP/Desktop_Projects_THE IMAGE RESIZER_bitmaps_icon.png"),
xml_res_file_3, xml_res_size_3, _T("image/png"));
XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$C__Documents and
Settings_JVSP/Desktop_Projects_THE IMAGE RESIZER_bitmaps_open.png"),
xml_res_file_4, xml_res_size_4, _T("image/png"));
XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$C__Documents and
Settings_JVSP/Desktop_Projects_THE IMAGE RESIZER_bitmaps_save.png"),
xml_res_file_5, xml_res_size_5, _T("image/png"));
XRC_ADD_FILE(wxT("XRC_resource/resources.cpp$._resources.xrc"),
xml_res_file_6, xml_res_size_6, _T("text/xml"));
wxXmlResource::Get()->Load(wxT("memory:XRC_resource/resources.cpp$._resources.xrc"));
}
Appendix 3

The Image Object

The image object was developed using the image class. The image object was specifically designed for a file that follows the RGBtext file format, which uses RGB coordinates as the representation domain of the color for a given pixel. As the discussion goes further, it will be seen that image object was also specifically designed for bilinear interpolation. The elements of the object could be divided into the groups. However, before understanding the image object, it is important to know two objects created for the image class itself, the classes, fixed_point and pixel.

A3.1. Fixed Point

Conceptually, a fixed point is a value that represents a digital signal, which is basically a fraction, one that has a minimum and a maximum value. The class fixed_point is designed for the purpose of standardizing the decimal point of any digital signal that has bit depth ranging from 1-16. This standardized value is stored in a private variable named as _value. The fixed_point class has three constructors. The first is the default constructor, which sets _value to zero, the other two are overloaded constructors, which obtain the fixed point equivalent of an input hexadecimal byte in string format and an integer type value.

In order to use this class well and efficient, certain operators must be overloaded. For the particular use of the bilinear interpolation algorithm, the following overloading were done; the assignment operator “=”, plus operator “+” for two fixed point objects, the multiplier operator “*” for an int type variable followed by a fixed point object, and the division multiplier “/” for a fixed point object followed by an int type variable.

The remaining elements are just accessor and general utility functions. The decimal equivalent of the fixed point object is returned by the decimal_value() function.

The charToHex() and hexToFixedPoint() are functions which are used by the overloaded constructor of the fixed point object.

A3.2. Pixel

The class pixel can be just thought of as a container of three fixed point objects, namely _R, _G, and _B. For simplicity of calculation and efficiency, the pixel object also contains overloaded operators specifically chosen for the bilinear interpolation. As a result, the same operators and their corresponding arguments mentioned for the fixed point object were overloaded for this class.

The pixel object contains two constructors, a default one, which sets _R, _G, and _B values to zero, and an overloaded one, which sets _R, _G and _B values respectively according the order of its arguments. The pixel object also contains a destructor for the purpose of saving memory in runtime. Finally, the pixel object contains accessor functions for the fixed point objects _R, _G, and _B. The functions are R(), G(), and B() respectively.

A3.3. Elements of the Image Object

The image class has only three private variables. The first two are the int types _height and _width, which are the values for the dimensions of the image itself. The last private variable is of type pixel** named _image_matrix, which is used to create a two-dimensional array of pixel objects. Each of these three private variables has its own accessor functions. The functions are height(), width(), and image_matrix() respectively.

A3.4. Constructors, Destructors, and Assignment Operator Overloading

The image object only has one constructor, which sets _height and _width to zero and allocates zero memory spaces for _image_matrix. The destructor is necessary to deallocate memory used for _image_matrix as an image object goes out of scope. Also, the assignment operator “=” was overloaded for the image object.
A3.5. General Utility Functions

These functions were used by all other functions. The make_fixed_point() is an overloaded function just returns a fixed point object from an input hexadecimal byte in string format or an integer type value. The make_pixel() function just returns a pixel created from three fixed point parameters, which sets _R, _G, and _B values respectively. Finally, the delete2Darray() function just deallocates the memory used for _image_matrix.

A3.6. The make_image() Function

This function was specifically made to create an image object besides from parsing an RGBtext file. This will be used by the bilinear interpolation algorithm to return an image object. It takes a pixel** type as its first parameter to which _image_matrix gets its value. The last two parameters sets the _height and _width values respectively.
Appendix 4

The User Interface

The GUI of the Image Cropping Application was developed using wxDevC++. This GUI can load and save multiple images and crop at an angle the supported file types (Bitmap, JPEG, and PNG files). It has a main window, a menu bar, a toolbar, a sidebar with fold panel bars, and client windows.

![Image Cropping Application Graphical User Interface](image)

Figure 2. Image Cropping Application Graphical User Interface

The GUI involves the following source codes: app.rc, app.h, app.cpp, main_frame.h, main_frame.cpp, main_frame_contd.cpp, child_frame.h, child_frame.cpp, my_canvas.h, my_canvas.cpp, resources.xrc and resources.cpp.

app.h and app.cpp contains the implementation of the cropper_app. Under this class, the main_frame class declared in main_frame.h and implemented in main_frame.cpp and main_frame_contd.cpp is called. The main_frame contains all the objects in the application as mentioned above. resources.xrc is an xrc list of the images used for the program while the resources.cpp is a generated file produced when the wxrc.exe (Please consult appendix) compiles the resources.xrc and the listed resources. resources.cpp is used to embed the images to the binary of the application.

The main_frame class calls upon a child_frame object in the instance an image file is opened. This child_frame is the client window as shown above. The child_frame contains the display object which is the my_canvas object. The my_canvas object is a wxScrolledWindow derived class which also contains the object of the class image produced from the file opened.

The program offers quick access to the user with keyboard shortcuts, a menu bar and toolbar. The menu bar has options to close the image and the program, as well as a link to the documentation. The menu items show their corresponding shortcuts beside them. The tool bar also provides access to certain functions.

The first fold panel bar shows the image of the active client window. The second fold panel bar includes the necessary parameters needed for image cropping. The first two inputs are text inputs for the x and y coordinate. The user can determine the upper leftmost pixel coordinate which is the starting point of the crop rectangle by defining its exact x and y coordinates. Next, there are two text input controls which correspond to the height and width of the crop rectangle. The last text input control correspond to the angle of rotation of the crop rectangle. Then, the Crop Image button can be used to crop the image when the desired area has been selected. See Figure 3 in the succeeding page.
The program also has the ability to view several pictures at once using the wxWidgets MDI or Multiple Document Interface. See Figure 4 in the succeeding page.

From the image files, a display image is produced using the wxWidgets Device Contexts (DC). A wxDC is a device context onto which graphics and text can be drawn. The device context is intended to represent a number of output devices in a generic way, with the same API being used throughout.

Using the program interface, user can open a file using a file dialog prompting to select an image file of the supported formats. Once done, the program detects what type of file is opened. The program uses the wxImage, a image container from wxWidgets, to open the file, then a image class object is constructed by extracting information from the wxImage (RGB values and dimensions).

The wxImage contains color values as char. To import this char to the fixed points of the image class, the main_frame::decimal_to_fixed() is called. This function bit shifts the integer value from the wxImage R, G or B char by 8. The return value of this function is then stored as the fixed point values of the pixel of the image. If errors are present, error dialogs indicating the error shall prompt the user.

If no errors are shown at this point, a client window of the class child_frame is constructed with the filename and the image as parameter. The child_frame class stores
the filename and constructs the my_canvas class which has the image object as one of its parameters. In the construction of the calls the OnDraw(wxDC& dc) function, which shall display the image in the my_canvas of the client window.

Figure 4. Multiple Document Interface

The OnDraw(wxDC& dc) function retrieves the integer equivalent of the fixed point objects in the image object and displays the RGB values using the following code which is in my_canvas.cpp:

```c++
pixel** temp_matrix;
fixed_point r,g,b;
temp_matrix=_the_image.image_matrix();
for(int i=0;i<_the_image.height();i++)
{
    for(int j=0;j<_the_image.width();j++)
    {
        r=temp_matrix[i][j].R();
        g=temp_matrix[i][j].G();
        b=temp_matrix[i][j].B();
        dc.SetPen(wxColour(r.decimal_value(),
                        g.decimal_value(),b.decimal_value()));
        dc.DrawPoint(wxPoint(x+j,y+i));
    }
}
```
A wxBitmap image buffer is created similar to the OnDraw function, and stores the RGB values to the wxBitmap. The wxBitmap is then drawn at a whole usind dc.DrawBitmap().

After the image window has been constructed and displayed, the user can now crop or save the image. The image intended to be cropped should be the active window (the child_frame on top). The user can select the x and y coordinates of the starting point of the crop rectangle, the height and width of the crop rectangle, and the angle of rotation of the crop rectangle.

If the user decides to use Crop Image, the image object is accessed from the my_canvas and sent to the main_frame::CropAtAnAngle_BLinearInterpolation function with the necessary parameters. This function outputs another image as discussed and this image is then used to create a new child_frame as discussed above.

If the user saves the image, a file dialog shall appear prompting the user to input a filename. It calls upon member functions of the wxWidgets library called image handlers to handle saving. As of writing this application, wxWidgets is limited to only reading GIF files not saving it as the patent for GIF has just recently expired.


Esmenda, Joshua C., Mark Andrew S. Mateo, and Jerome Vergil S. Paez. *Image Resizer*. Quezon City: ECCE Department, Ateneo de Manila University, 2008.