Image Resizer Using Bicubic Interpolation

A Project by

Isadelle Mara de Guzman
Elaine Valerie Ramos
Jose Paolo Talusan

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

An image scaler is implemented using the C++ language. The images were scaled using the method of bicubic interpolation. Included in the project is a graphic user interface (GUI), in which both the input image and the rescaled output image can be viewed. Image types such as BMP, JPG, PNG, and GIF can be opened and scaled in the project. Furthermore, multiple images can be loaded and rescaled in the same GUI. The images can then be saved in the form of BMP files, JPG files, or PNG files.

Acknowledgements

God

ECE Batch 2011 – for their support and being there for us when the road looked tough ahead.

Sir Lui – for this intense challenge, greater than whatever we have encountered in 4 years of existence here in the Ateneo.

Table of Contents

1.Introduction

1.1. Images

Mathematically speaking, an image is a set of points called pixels within a two-dimensional grid. This is a set of finite data points stored in a function such as $f(i, j)$ where i and j are the number of columns and rows respectively and each intersection is called a pixel. Pixels or picture elements are containers which store the values of the image depending on what color space it uses.

1.2. RGB

The most used color space is red, green and blue or RGB. Pixels store the intensity level of these values by assigning a number within 0-255 to it. The closer the values to 255 the more intense and more white the pixel become. The values of red, green and blue are combined to reproduce an array of different other colors ranging from black, zero intensity to white.

1.3. Image Resizing

Image resizing is the process of changing one or both of the dimensions of an image such that the output image will be of a different size, either larger or smaller than the original image. Since there would be a change in dimensions there would be new pixels present that were not in the original image. These values are obtained using interpolation.

There is a need for interpolation since in order to resize the images it would need to calculate a number of unknown pixels, obtaining their values from the pixels surrounding them. unknown pixels are obtained using interpolation.

1.4. Interpolation[1]

Interpolation is a process of fitting a continuous function to a discrete set of points. This results in being able to obtain any other unknown points in between or beyond the said discrete set of points. There are many functions being used to interpolate points such as polynomials, rational functions and also trigonometric functions. The higher the order of the function is used to model the set of points, the more accurately it depicts it, although, a function with sharp corners is more accurately depicted by a lower order of function. As a final note, there is no new data added by any interpolation scheme, it will only draw all the values from the original grid.

1.5. Interpolation on Images

In the case of images, it has to be interpolated along a grid in multi-dimensions. Instead of dealing with points in a line, it deals with pixels, as described above, arranged in a two dimensional grid. It has to model a pixel with respect to the intensity of the other pixels surrounding it to obtain an interpolated value. In these cases there are two types of interpolation categories, adaptive and non-adaptive. Adaptive algorithms produce optimal results because they are designed to adapt to the properties of the pixels surrounding the point to be interpolated. Meanwhile, non-adaptive algorithms such as those described here produce sub-optimal results by their inability to take into account how it treats other pixels depending on the pixel they are interpolation. The up-side to these is their simplicity of implementation.

1.5.1. Nearest Neighbor

The nearest neighbor function is the simplest form of interpolation. This is where the value of a point to be interpolated is obtained from the values of the old coordinate

1 *Numerical Recipes The Art of Scientific Computing Third Edition.*

point which is the value nearest to it. This is the simplest of the interpolation techniques as it has the least ability to reproduce sharp image details especially at larger sizes. This results in the resized image having excessive artifacts and degradation.

Shown below are the outputs of the various interpolation methods being used to recreate the discrete sinusoid wave in the figure 1.

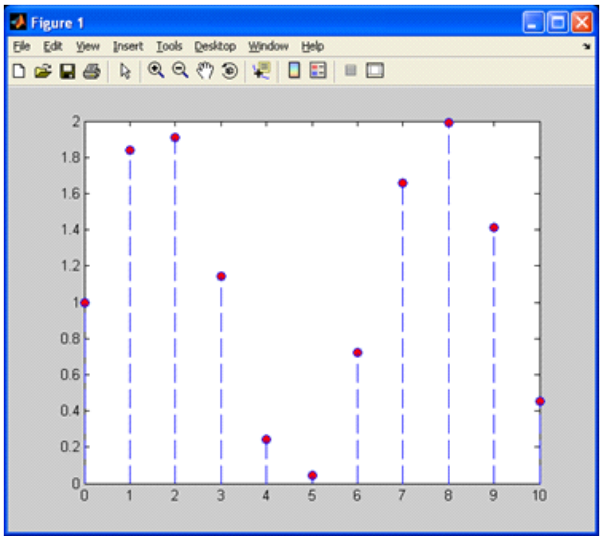

Figure 1: Discrete Decimated Sinusoidal Wave[2]

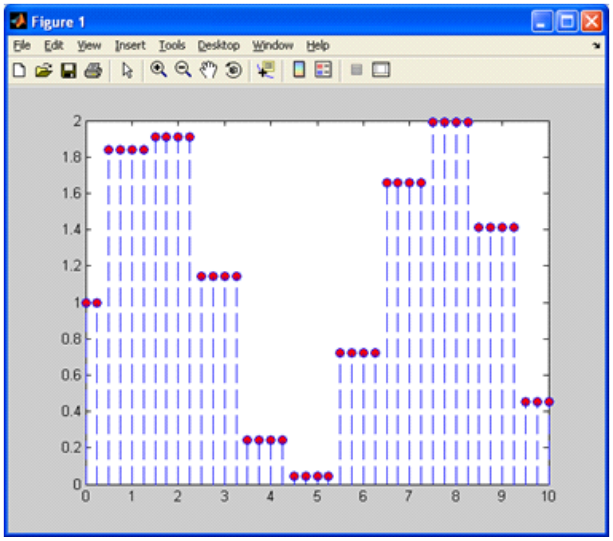

Figure 2: Reconstruction of wave using Nearest Neighbor

2 *Engineering, Medical Research, and 3D Image Processing (as with subsequent graphs)*

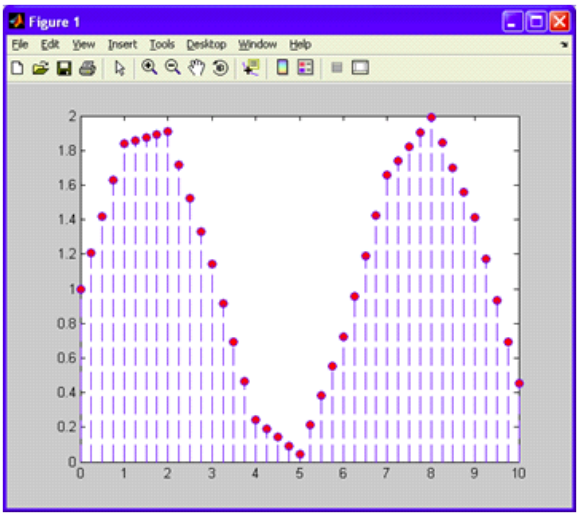

Figure 3: Reconstruction using Bilinear Interpolation

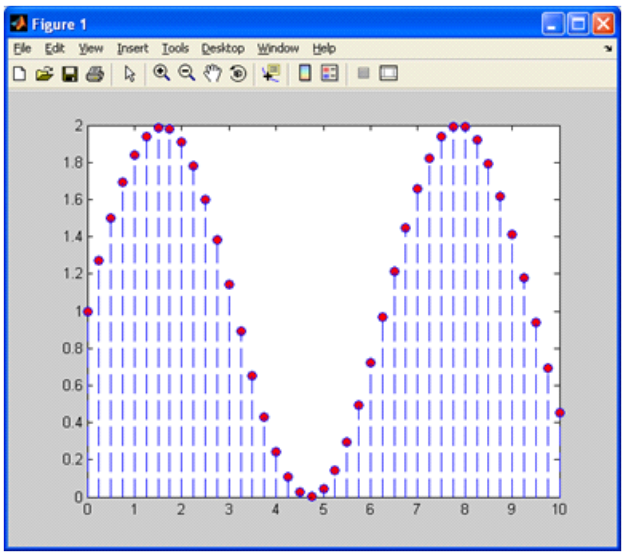

Figure 4: Reconstruction using Bicubic Interpolation

1.5.2. Bilinear Interpolation[3]

Bilinear interpolation stems from the one dimensional linear interpolation where the output of the function based on two known points $f(i)$ and $f(i+1)$ on a single axis is given by: $f(i+p)=(1-p)f(i)+pf(i+1)$. Moving on to two dimensions require this same equation done on the other axis. Thus this requires four

3 *Image Resizer.*

points

surrounding the pixel to be interpolated. Instead of the initial two points

$$f(i)$$

$$f(i+1)$$

there would now also be

$$f(j)$$

$$f(j+1)$$

As seen from the figure below.

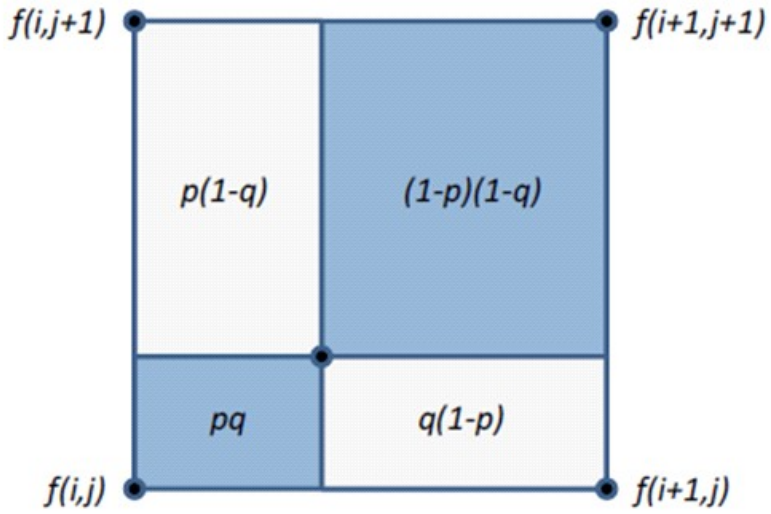

Figure 5: Grid Square used for Bilinear Interpolation

As such the equation therefore would become:

$$f(i+p,j+q)=[(1-p)(1-q)f(i,j)]+[(p)(1-q)f(i+1,j)] \\ +[(q)(1-p)f(i,j+1)]+[(p)(q)f(i+1,j+1)] \quad [1]$$

Bilinear interpolation is often called "close enough for government work."[2] Since it produces an output image that is a good enough representation of the original image, without the artifacts and blocks present in the nearest neighbor algorithm. This algorithm has a good balance of speed and quality for most any use of resizing. From this there are two distinct routes in order to improve the quality of interpolation.

These are two use higher-order methods, remember that bilinear is simply a zero-order method. One of these methods is called the bicubic interpolation method, not to be confused with the bicubic spline method. From here on the term “bi-“refers to operations in two dimensions, i.e. the image grid.

1.5.3. Bicubic Splines

Bicubic splines is a lot like bilinear interpolation but instead of fitting all the data points along a certain line, it is fitted along a curve of third degree,

$$y = f(x) = a + bx + cx^2 + dx^3$$

hence a cubic spline. As discussed before this would result in a more optimal output compared to lower degree methods. Of course one curve cannot be the model for all of the points in the data set.

A cubic spline is defined as a function that provides a value for any point between the data points j and k (start and end of the data set being interpolated). It must pass through all the original points in the set and it must be a number of piecewise continuous functions over several sub-intervals which simply means that these different functions must be connected with each other.

The difficulty begins when the cubic splines must also have the same first derivative value on both sides of a point this also goes for the second derivatives.

Finally there is the case for the cubic spline to be either clamped or natural. The natural boundary imagines the curve to pass through all points while the clamped attempts to extrapolate data from the end points. The natural boundary condition is the one being satisfied by the curve in the future discussions.

Moving on to the computational aspect of bicubic splines, much like bilinear

interpolation it needs to identify the four nearest neighbor pixels of the pixel to be interpolated,

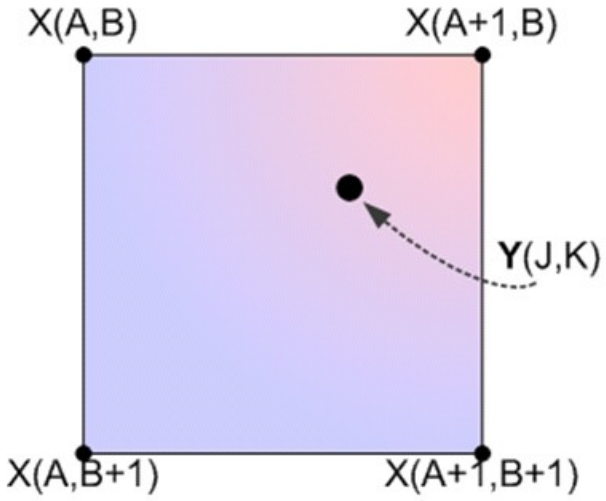

Figure 6: Grid Square for Bicubic Splines

but again a curve is needed to fit all the data points, as shown by the figure below.

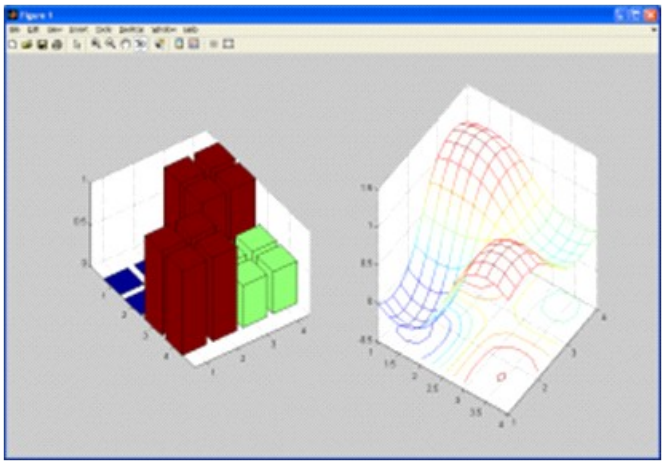

Figure 7: Output of the bicubic splines

From this the values of the four points as well as their derivatives along any of the axes will be obtainable.

The original intensities of the surrounding pixels contribute four values:

$$f(A,B),\ f(A+1,B),\ f(A,B+1) \quad \text{and} \quad f(A+1,B+1)$$

and the partial derivatives:

$$\frac{\partial f}{\partial x} = f_x \quad \text{and} \quad \frac{\partial f}{\partial y} = f_y$$

and the unused 4 cross derivatives:

$$\frac{\partial^2 f}{\partial x \partial y} = f_{xy}$$

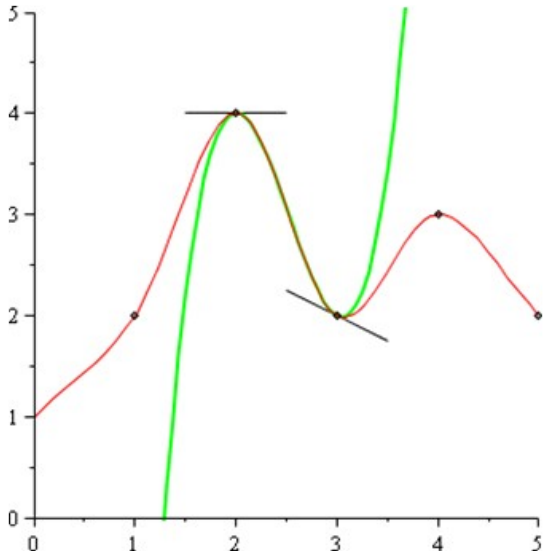

Figure 8: Placement of the points for bicubic splines

1.5.4. Bicubic Interpolation[4]

Since bicubic splines are just a special case of bicubic interpolation, it lacks some functionality of bicubic interpolation that makes it more accurate. While bicubic splines also make use of sixteen values, the same as bicubic interpolation, it does not make use of cross-derivative values

This trade-off for accuracy results in much more complicated algorithm than the bicubic splines. As said before the generalized bicubic interpolation now uses all of sixteen points discussed in the bicubic spline interpolation, as well as the grid-square. It

4 *Engineering, Medical Research, and 3D Image Processing*

then attempts to reconstruct the exact surface between the four initial pixels by extracting from it the sixteen other pieces of information.

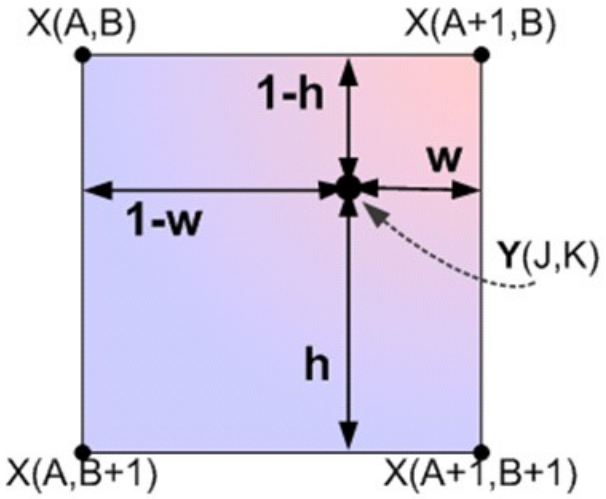

Figure 9: Grid Square for the Bicubic Interpolation

Calculating these partial and cross derivatives based on the grid square above are as follows:

$$f_x(A,B)=\frac{f(A+1,B)-f(A-1,B)}{(A+1)-(A-1)}=\frac{f(A+1,B)-f(A-1,B)}{2}$$

$$f_y(A,B)=\frac{f(A,B+1)-f(A,B-1)}{(B+1)-(B-1)}=\frac{f(A,B+1)-f(A,B-1)}{2}$$

$$f_{xy}(A,B)=\frac{[f(A-1,B-1)+f(A+1,B+1)]-[f(A+1,B-1)+f(A-1,B+1)]}{[(A+1)-(A-1)]\times[(B+1)-(B-1)]}$$

$$f_{xy}(A,B)=\frac{[f(A-1,B-1)+f(A+1,B+1)]-[f(A+1,B-1)+f(A-1,B+1)]}{4}$$

Using these values will enable the algorithm to find any interpolated value in the image output $p(x,y)$ and going back to figure 9 for references will give 16 various points of information depicted by the following.

$$p(J,K)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{mn}(1-w)^m(1-h)^n$$

$$p_x(J,K)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{mn}(1-w)^{m-1}(1-h)^n$$

$$p_y(J,K)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{mn}(1-w)^m(1-h)^{n-1}$$

$$p_{xy}(J,K)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{mn}(1-w)^{m-1}(1-h)^{n-1}$$

These values again result in another unknown this time these are the coefficients, a_{mn} this is obtainable by assuming that w and h are continuous over a unit square and redefine the four corners of the grid square in figure 9 as, starting from the upper left hand, clockwise as $f(0,0), f(1,0), f(0,1), f(1,1)$ this results in the sixteen coefficients. Matching the interpolation surface, $p(x,y)$, with the function values of the four corners, the following four equations

$$f(0,0)=a_{00}$$

$$f(1,0)=a_{00}+a_{10}+a_{20}+a_{30}$$

$$f(0,1)=a_{00}+a_{01}+a_{02}+a_{03}$$

$$f(1,1)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{00}+a_{01}+a_{02}+a_{03}+a_{10}+a_{11}+a_{12}+a_{13}$$
$$+a_{20}+a_{21}+a_{22}+a_{23}+a_{30}+a_{31}+a_{32}+a_{33}$$

are obtained. Applying the same procedure with the horizontal and vertical derivatives, the following eight equations, the first four are for the horizontal derivatives and the latter are for the vertical derivatives,

$$f_x(0,0)=a_{10}$$

$$f_x(1,0)=a_{10}+2a_{20}+3a_{30}$$

$$f_x(0,1)=a_{10}+a_{11}+a_{12}+a_{13}$$

$$f_x(1,1)=\sum_{m=1}^{3}\sum_{n=0}^{3}a_{mn}m$$

$$f_y(0,0)=a_{01}$$

$$f_y(1,0)=a_{01}+a_{11}+a_{21}+a_{31}$$

$$f_y(0,1)=a_{01}+2a_{02}+3a_{03}$$

$$f_y(1,1)=\sum_{m=0}^{3}\sum_{n=1}^{3}a_{mn}n$$

are derived. Lastly, four other equations are obtained from the cross derivatives.

$$f_{xy}(0,0)=a_{11}$$
$$f_{xy}(1,0)=a_{11}+2a_{21}+3a_{31}$$
$$f_{y}(0,1)=a_{11}+2a_{12}+3a_{13}$$
$$f_{y}(1,1)=\sum_{m=1}^{3}\sum_{n=1}^{3}a_{mn}mn$$

Combining the coefficients a_{mn} with the function values and the previously obtained partial and cross-derivatives function values, the output can be expressed in a matrix form

$$M\,\alpha=\beta$$

where α is the matrix, which contains the sixteen a_{mn} coefficients, β is the matrix, which has the function values, and its partial and cross derivatives, and M is the coefficient matrix for 'a'-values.

$$\alpha=(a_{00}\,a_{10}\,a_{20}\,a_{30}\,a_{01}\,a_{11}\,a_{21}\,a_{31}\,a_{02}\,a_{12}\,a_{22}\,a_{32}\,a_{03}\,a_{13}\,a_{23}\,a_{33})^{T}$$

$$\beta=\left[f(0,0)\,f(1,0)\,f(0,1)\,f(1,1)\,f_{x}(0,0)\,f_{x}(1,0)\,f_{x}(0,1)\,f_{x}(1,1)\,f_{y}(0,0)\,f_{y}(1,0)\,f_{y}(0,1)\,f_{y}(1,1)\,f_{xy}\right.$$

$$M=\begin{matrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&2&3&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
0&1&2&3&0&1&2&3&0&1&2&3&0&1&2&3\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&2&0&0&0&3&0&0&0\\
0&0&0&0&1&1&1&1&2&2&2&2&3&3&3&3\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&2&3&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&2&0&0&0&3&0&0\\
0&0&0&0&0&1&2&3&0&2&4&6&0&3&6&9
\end{matrix}$$

To compute for the interpolated value, the sixteen a_{mn} coefficients or the α is needed. From $M\,\alpha=\beta$, it is rewritten such that

$$\alpha=M^{-1}\beta\quad.$$

The transposed matrix M, shown below.

$$M^{-1}=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & -3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\
-3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\
9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\
-6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\
2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
-6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\
4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1
\end{bmatrix}$$

The transposed 16x16 matrix, which when multiplied to β, provides the values of all the α. Thus, giving the path to obtain the pixel to be interpolated which is given by this final equation.

$$p(J,K)=\sum_{m=0}^{3}\sum_{n=0}^{3}a_{mn}(1-w)^m(1-h)^n$$

2.Project Overview

2.1. Objectives

The main objective of this project is to improve upon the image resizing program that was created two years ago by modifying their code, which was made using the bilinear interpolation algorithm, to a bicubic interpolation method.

The project is to be written in C++ for the same reasons as the previous groups as well as to make use of the wxWidgets within the compiler wxDev-C++. This is an open-source compiler which combines the Bloodshed Dev-C++ with wxWidgets for creating graphical user interfaces. This has the added capability of manipulating images through its wx classes such as wxImage and wxBitmap.

The previous' groups code created their own image classes with which to display and operate the image for their bilinear interpolation, to be discussed later, the group aims to modify this to remove certain redundancies in their code which improves certain parts of their program in exchange for some trade-offs.

The graphical user interface will be the same as before although it would make things simpler for future modification of the code. This will still be able to open, save and resize images all while giving the user a view of the original and resized images.
Compatibility with the most often used formats is to be achieved, this includes and is not limited to JPG, BMP, PNG and GIF.

2.2. Significance of the Project

Resizing is a frequently used process by programs and users alike that handle

images. This is specifically needed when creating or transferring images for viewing to smaller or larger mediums such as HDTVs or mini-sized LCDs such as digital cameras and other portable instruments.

The efficiency of any image resizing algorithm is measured by two main factors which are output image quality and computational complexity. When resizing images it will be crucial to have a balance of the two, having acceptable computational complexity and speed while having less blocks, artifacts and blurring.

This project will be useful for other people needing to resize their images for their own program's uses such as image stitching, or image cropping.

2.3. Scope and Limitations

The program since it was built on top of the previous groups' programming will be able to open and resize image formats BMP, JPG, PNG and GIF. While being able to save them with the exception of GIF since its handlers are limited to opening only due to licensing. Also worth noting is the absence of the RGBtext format since it is an obsolete image format which is severely limited in its abilities, the only use it serves is for extensive testing and no more.

Also the removal of their image classes found in their "image.h" and "image.cpp" brings about certain limitations such as the absence of a dedicated function for interpolation. But with the removal of the redundant classes, it become easier to follow and track what is happening within the code since their classes are replace by simple wxImage class, which are built in the compiler.

The scope of this project only covers bicubic interpolation and no other interpolation method.

3.Code

3.1. Removing the image class

In removing their image classes the group simply built a program from scratch taking snippets of their code while at the same time replacing functions and data types within the code from the old image class to the corresponding counterpart within the wx classes.

The basic flow of the past programming was to load an image and assign the RGB pixel values to their own fixed point class which was created by shifting the int data type obtained from getRed, getBlue, getGreen wx classes, eight places to the left. It was then placed unto a container labeled “pixel” this grouped the three int values into one container.

As it follows other parts of their code will also be modified such as the way it displays images, rather than in their own class, images will be displayed either using wxImage or wxBitmap. By doing this the previous group's code has been made simpler to modify and translate for other projects since there has been no more need to wade through their image classes

After removing the image class of the previous group, the following functions were added to the resizer_Frm class in order to perform bicubic interpolation:

```
wxImage bicubic_interpolation(wxImage &source, int newx, int newy);
float bicubicSpline(float x);
```

where the first function performs bicubic interpolation and the second computes for the bicubic spline to be used in the bicubic interpolation.

3.2. Bicubic Interpolation[5]

The bicubic interpolation implementation, wxImage* resizable_image::bicubic_interpolation, is found within the resizer_Algo.cpp.

The ratio of the the old dimension with the new dimension is computed with

```
xScale = (float)source.GetWidth()  / (float)new_width;
yScale = (float)source.GetHeight() / (float)new_height;
```

where source.GetWidth() and source.GetHeight() obtains the dimensions of the original image, and new_width and new_height are the new dimensions inputted by the user. To get the data of each pixel, the following function is used,

```
wxUint8 *src = source.GetData();
```

The wxImage::GetData returns the image data as an array in RGBRGBRGB... format. Since the data retrieved is in unsigned char data type, wxUint8 is used.

The derivatives are computed by multiplying the location of the pixel in the new image to the ratio of the old dimension to the new dimension. Then, the largest integer value of f_y is subtracted to the value of f_y.

```
f_y = (float) y * yScale;
i_y = (int) floor(f_y);
a   = f_y - (float)floor(f_y);
f_x = (float) x * xScale;
i_x = (int) floor(f_x);
b   = f_x - (float)floor(f_x);
```

Then the obtained values are passed through resizer_Frm::b3spline, which performs bicubic spline. If the derivative obtained is less than zero, then it is set that the output would be 0.

5 *wxImageResampler.cpp: implementation of the wxImageResampler class.*

```
if((x + 2.0f) <= 0.0f) a = 0.0f;
else a = (float)pow((x + 2.0f), 3.0f);
if((x + 1.0f) <= 0.0f) b = 0.0f;
else b = (float)pow((x + 1.0f), 3.0f);
if(x <= 0) c = 0.0f; else c = (float)pow(x, 3.0f);
if((x - 1.0f) <= 0.0f) d = 0.0f;
else d = (float)pow((x - 1.0f), 3.0f);
return (0.16666666666666666667  (a - (4.0f * b) + (6.0f * c) - (4.0f *
d)));
```

To get the location of the source pixel, the width of the original image is multiplied to the height of the original image and then added to the width of the orignal image. Since the wxImage::GetData is used, it is multiplied by three.

```
srcPos = (yy * source.GetWidth() + xx) * 3;
```

To finally perform bicubic interpolation, the interpolated value is computed by multiplying the pixel value to the bicubic spline with respect to height and width.

```
redPixel += src[srcPos]   * r1 * r2;
greenPixel += src[srcPos+1] * r1 * r2;
bluePixel += src[srcPos+2] * r1 * r2;
```

4.User Interface

The user interface of the resizer program was a modified program from the previous group's code and was again designed under the wxWidgets cross-platform C++ GUI toolkit. The graphical user interface of the program allows the user to open, view and resize the following supported image files (BMP, JPG, PNG, and GIF files). The main components of the graphical user interface include the status bar, the menu bars, the side-bar and finally the main parent window.

Once an image has been selected and opened, a child window opens from within the parent window, displaying the name of the file while opening the child frame in its original size. At the same time the properties of the image namely the file name and the dimensions in pixels. Also it automatically loads and prints the values of the said picture unto the resizing options part of the side-bar, shown below:

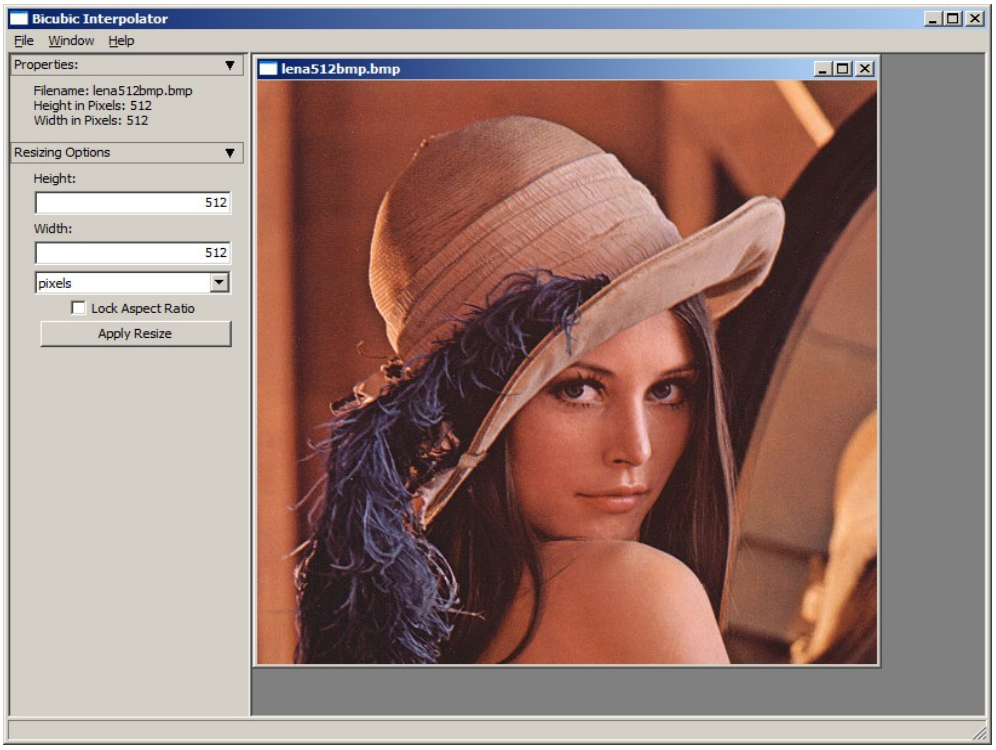

Figure 10: Main window of the Program with image opened

The main of the program for the GUI is found in the **`resizer_App.cpp`** and resizer_App.h. This controls the windows for the child frames as well as to initialize the

main window for the application. The **`resizer_Frm.cpp`** and its accompanying header files control all the events within the main frames, ranging from the open, close, save and resize. This also creates all the windows within the main application such as the menu bars, status bars and the side-bars. Also this contains the main definition and declaration for the **`resizer_Frm`** which is the main application.

Also worth noting is that there is a wxFoldPanelBar present within this **`resizer_Frm.cpp`** that is not readily available within wxDev-C++.

Also within the **`resizer_Frm`** is the instantiation of the child frame which occurs when you open any image. This calls upon the **`set_image_properties`** and displays the dimensions of the image within the sidebar.

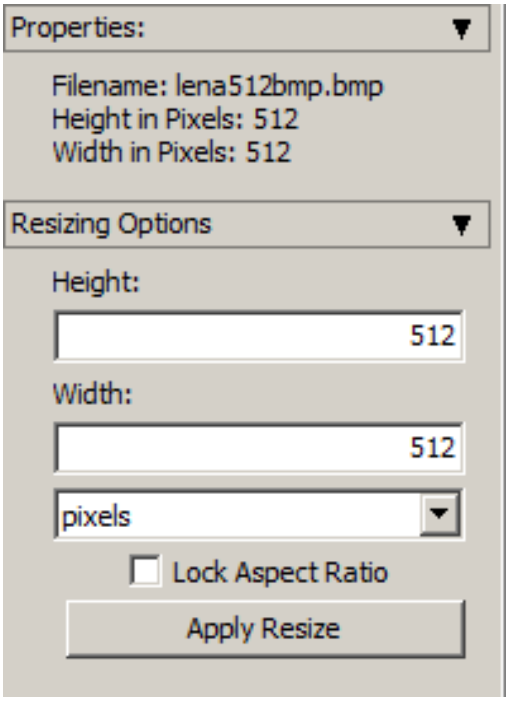

Figure 10: side bar of the Program with image opened

This side-bar also allows the user to set the new dimension to which the currently opened image will be resized to. There is also the option of resizing using pixels as well as with the percent size of the current image. There is also the option to lock the aspect ratio of so that when either the height or width is set, the other dimension will be automatically set to keep the same aspect ratio as the original image.

This **resizer_Frm** opens the child_frame which contains the display object which is the my_canvas object. The **my_canvas** object is in the **my_canvas.cpp** which controls the display of the image by creating a buffer using wxBitmap. This makes the program work and display image faster than the previous group.

This also has the capability to be able to open multiple child windows at the same time using the wxWidgets MDI or multiple document interfaces. These are accessed by the **resizer_Chl.cpp** which controls the properties of the children windows such as the names, its parents, and the events when an image is closed.

Also this includes keyboard shortcuts for opening, saving and closing windows and there is a help window in the about menu bar which details the steps in resizing an image.

4.1 Program Flow

Using the program interface, in the menubar, the user can open a file using the file open dialogue which prompts the user to select an image of the supported formats. It just displays the image formats that are supported by the program and outputs and error if it is an invalid image.

Once an image is passed, the program creates a wxImage container for the said image and it calls the open_new_child function as well as the my_canvas as well as the **resizer_Chl.cpp**. This opens the image with the parameters, filename and the wxImage. While at the same time the functions set_image_properties and get_"dimensions"_labels are called to pass the values to the **OnActivate** in the **resizer_Chl.cpp** and passes these to the sidebar to display and take note for further use.

Once opened, the image can now be resized. When the dimensions have been edited, while the lock aspect ratio button is checked, there would be a call to the various functions to take into account the initial dimensions in inputting the new dimensions.

Using pixels or percentages of the current dimension the user selects new ones.

Once the apply resize button is pressed, the program calls these new dimensions as well as active image which is the current image window opened and passes it into the bicubic interpolation function.

In the same function, once these variables are passed the program computes for the bicubic interpolation of every pixel on the image and stores them in a container,

```
redPixel   += src[srcPosition]   * vertSpline * horSpline;
greenPixel += src[srcPosition+1] * vertSpline * horSpline;
bluePixel  += src[srcPosition+2] * vertSpline * horSpline;
```

which are then passed to the next function which places the new interpolated pixels unto a new wxImage.

```
// set the RGB values using the values obtained
resized_image.SetRGB(u,t,R,G,B);
```

this wxImage is then returned and displayed

```
return resized_image;
open_new_child(name_of_image, resized_image);
```

storing the values to a wxBitmap for quick display when called by the function **`dc.DrawBitmap()`**, where dc means device context. These are all handled by the **`resizer_Chl.cpp`** file.

Once the new resized image is displayed, the user has the option to save the image into the different formats supported by the program. The program then saves a copy into the directory of the user's choice while retaining the dimensions of the resized image.

Saving and opening are handled by the image handlers in the wxWidgets library and saving is not available for GIF since the patent for that has recently expired.

5.Conclusions and Recommendations

5.1. Conclusions

The group was able to develop an image resizer which performs using bicubic interpolation while using only the entire wxWidgets library. The project was implemented with a graphical user interface that is able to open, view, save and resize the images of various formats. This allows easier debugging and easier modification in the future.

5.2. Recommendations

The program created removes the image classes of the previous group in exchange for the wxWidgets libraries only; this led to an easier to follow program and code for future modifications. Further modifications could be made to feature other interpolation methods such as sinc and Lanczos[6] interpolation methods and fractal algorithms like those used by *Genuine Fractals.* Finally the code could be simply improved to perform faster.

6 Comparison of Interpolating Methods for Image Resampling.

Appendix 1 User's Manual

A.1. Software Overview

A1.1.1. Minimum System Requirements

For modification

- WxDev-C++
- Updating of the current packages
- Addition of the lib libwxmsw28_foldbar.a to the compiler in the project options (since the foldpanelbar used by the previous group is not found in the default settings of wxDev-C++)

For binary/executable

- Windows 98/XP/Vista
- 256 MB RAM
- 30 MB free hard disk space (5.75MB for executable, extra for image)
- 256 or more colors compatible display

A1.1.2. Features

The *Image Resizer* allows the user to:

- Load Bitmap, JPEG, PNG and GIF images;
- View Bitmap, JPEG, PNG and GIF images;
- View multiple images;
- Resize an image and view the resized image;
- Save Bitmap, JPEG, and PNG images.

A1.1.3. Availability

A1.2. User's Guide

A1.2.1. Using the Software

The program, Image Resizer, is designed to run on Microsoft Windows. To run the program, the executable file named Image Resizer is opened.

A1.2.2. Loading Image Formats

Go to the File Menu then, click Open Image. Also, the keyboard shortcut Ctrl + O could be used for loading the image on the program. Upon loading the image, the filename and dimensions of the image is seen in the side panel.

A1.2.3 Loading Multiple Images

To open another image, the same process as discussed in A1.3.2 is repeated.

A1.2.4 Resizing the Image

After loading the image, which is either in bitmap, jpeg, png or gif format, the image could be resized by typing the desired width and height in the side panel.

The user has an option to tick the "Lock Aspect Ratio." If this option is checked, the program automatically calculates the other dimension when one dimension is changed.

The user can also input the desired dimensions using percentage.

Finally, in order to apply the new dimensions to the image, press the "Apply Resize" button then, the program resizes the image with accordance to the user's specifications.

A1.2.5. Saving the Image

To save the image, click the option *Save Image* in the File menu. Also, the keyboard shortcut Ctrl + S could be used. Then, a dialog box, where the user can input the filename and image type, appears.

A1.2.6. Closing the image window

The user can click the X button on the image window. Also, the user can go to the File menu, then click the *Close Image* option. A third option is to use the keyboard

shortcut Ctrl + F4.

A1.2.7 Closing the program

The user can click the X button on the program window. Also, the user can go to the File menu, then click the *Quit Application* option. A third option is to use the keyboard shortcut Alt + F4.

Appendix 2

Source Code

A2.1. Source Files

A2.1.1. *my_canvas.cpp*

```
//---------------------------------------------------------------------------
//
// Name:        my_canvas.cpp
// Author:
// Description: my_canvas class implementation
//
//---------------------------------------------------------------------------

//Header files
#include "my_canvas.h"
#include "resizer_Ch1.h"

//wxWidgets libraries included
#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 wxImage& temp_image)
         : wxScrolledWindow(parent,wxID_ANY,wxDefaultPosition,
                     wxDefaultSize,wxNO_FULL_REPAINT_ON_RESIZE)
{
    the_parent = parent;
    image_bitmap = new wxBitmap(temp_image,-1);
    this->SetClientSize(wxSize(image_bitmap->GetWidth(),
                           image_bitmap->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 bitmap 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 = image_bitmap->GetHeight();
```

```
    image_width = image_bitmap->GetWidth();

     //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;
    }
    // Draws the image
    dc.DrawBitmap(*image_bitmap,x,y,false);
}

int my_canvas::access_height(void)
{
    return image_bitmap->GetHeight();
}

int my_canvas::access_width(void)
{
    return image_bitmap->GetWidth();
}

wxBitmap* my_canvas::return_image_bitmap(void)
{
    return image_bitmap;
}
```

A2.1.2. my_canvas.h

```
//-----------------------------------------------------------------------
//
// Name:        my_canvas.h
// Author:
// Description: Declaration for the my_canvas class
//              Part of the Image Resizer project for ELC 152
//
//-----------------------------------------------------------------------

#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>

class my_canvas : public wxScrolledWindow
{
    private:
        wxWindow* the_parent;       //pointer to parent, for quick
access
        wxBitmap* image_bitmap;     //for quicker rendering
    public:
        //Constructor/Deconstructor:
        my_canvas(wxWindow *parent, const wxString& title,
                                    const wxImage& temp_image);
        ~my_canvas(void);

        // Display the bitmap image from wxImage:
        virtual void OnDraw(wxDC& dc);
        //Accessor functions:
        int access_height(void);
        int access_width(void);
        wxBitmap* return_image_bitmap(void);
};
#endif
```

A2.1.3. resizer_Algo.cpp

```
//-----------------------------------------------------------------------
//
// Name:        resizer_Algo.cpp
// Author:
// Description: Declaration for the my_canvas class
//              Part of the Image Resizer project for ELC 152
//
//-----------------------------------------------------------------------

//acts like main_frame_contd.cpp
#include "resizer_App.h"
#include "resizer_Frm.h"
#include "resizer_Chl.h"

DECLARE_APP(resizer_app)

void resizer_Frm::OnApply(wxCommandEvent& event)
{
    wxImage temp_image = get_active_image()->ConvertToImage();
    wxImage resized_image;
    wxString name_of_image;

    int new_height, new_width,
        input_height, input_width,
        old_height, old_width;

    old_height = temp_image.GetHeight();
    old_width = temp_image.GetWidth();
```

```
    // users choices
    input_height = wxAtoi(get_height_input());
    input_width  = wxAtoi(get_width_input());

    if(!(input_height == 0 || input_width == 0) && !(input_height ==
old_height && input_width == old_width))
    {
        wxMDIChildFrame* active_frame = this->GetActiveChild();
        resizer_Chl *active_child =
wxDynamicCast(active_frame,resizer_Chl);
        name_of_image = _T("Resized ") + active_child-
>return_image_name();

        // for user input in pixels
        if(_in_pixels)
        {
            new_height = input_height;
            new_width = input_width;
        }
        // for user input in percentage
        else
        {
            new_height = (input_height*old_height)/100;
            new_width = (input_width*old_width)/100;
        }

        resized_image = Bicubic(temp_image, new_width, new_height);
        open_new_child(name_of_image, resized_image);
    }

    else if (input_height == old_height && input_width == old_width)
    {
        wxMDIChildFrame* active_frame = this->GetActiveChild();
        resizer_Chl *active_child =
wxDynamicCast(active_frame,resizer_Chl);
        name_of_image = _T("Resized ") + active_child-
>return_image_name();
        open_new_child(name_of_image, temp_image);
    }
    else
    {
        show_error(_T("A zero in the input dimension is not allowed"));
    }
}

wxImage resizer_Frm::Bicubic(wxImage &source, int new_width, int
new_height)
{
      wxCHECK_MSG(source.Ok(), source, wxT("Invalid source wxImage
passed to wxImageResampler::Bicubic."));
    wxImage resized_image(new_width,new_height,true);

    float xScale, yScale;
      xScale = (float)source.GetWidth()  / (float)new_width;
      yScale = (float)source.GetHeight() / (float)new_height;

    wxUint8 *src = source.GetData();
      long srcPosition;
```

```
      // bicubic interpolation by Blake L. Carlson <blake-
carlson(at)uiowa(dot)edu
    float scaledWidth, scaledHeight,
          diffHeight, diffWidth,
          redPixel, greenPixel, bluePixel,
          vertSpline, horSpline;

      int   floorScaledWidth, floorScaledHeight, xx, yy;

    for(long y = 0; y < new_height; y++)
      {

            scaledHeight = (float) y * yScale;
            floorScaledHeight = (int) floor(scaledHeight);
            diffHeight   = scaledHeight - (float)floor(scaledHeight);
            for(long x = 0; x < new_width; x++)
        {
                  scaledWidth = (float) x * xScale;
                  floorScaledWidth = (int) floor(scaledWidth);
                  diffWidth   = scaledWidth - (float)floor(scaledWidth);

                  redPixel = greenPixel = bluePixel = 0;
                  for(int m = -1; m < 3; m++)
            {
                        vertSpline = bicubicSpline((float) m -
diffHeight);
                        for(int n = -1; n < 3; n++)
                {
                              horSpline = bicubicSpline(-1.0F*((float)n
- diffWidth));
                              xx = floorScaledWidth + n + 2;
                              yy = floorScaledHeight + m + 2;

                              if (xx < 0)
                    {
                        xx=0;
                    }

                              else if (xx >= source.GetWidth())
                    {
                        xx = source.GetWidth()-1;
                    }

                    if (yy < 0)
                    {
                        yy = 0;
                    }

                              else if (yy >= source.GetHeight())
                    {
                        yy = source.GetHeight()-1;
                    }


                              srcPosition = (yy * source.GetWidth() +
xx) * 3;
```

```
redPixel   += src[srcPosition]   * vertSpline * horSpline;
greenPixel += src[srcPosition+1] * vertSpline * horSpline;
bluePixel  += src[srcPosition+2] * vertSpline * horSpline;
                        }
                }
             resized_image.SetRGB(x, y, redPixel, greenPixel, bluePixel);
             }
      }
      return resized_image;
}

float resizer_Frm::bicubicSpline(float x)
{
      float a, b, c, d;

      if((x + 2.0f) <= 0.0f)
    {
        a = 0.0f;
    }

      else
    {
        a = (float)pow((x + 2.0f), 3.0f);
    }

      if((x + 1.0f) <= 0.0f)
    {
        b = 0.0f;
    }

      else
    {
        b = (float)pow((x + 1.0f), 3.0f);
    }

      if(x <= 0)
    {
        c = 0.0f;
    }

    else
    {
        c = (float)pow(x, 3.0f);
    }

      if((x - 1.0f) <= 0.0f)
    {
        d = 0.0f;
    }

      else
    {
        d = (float)pow((x - 1.0f), 3.0f);
    }

      return (0.16666666666666666667f * (a - (4.0f * b) + (6.0f * c) -
(4.0f * d)));
}
```

```
void resizer_Frm::OnText(wxCommandEvent& event)
{
    //For calculating the other dimension when aspect locked
    if(_aspect_locked)
    {
        wxString label;
        int old_height, old_width;
        old_height = get_active_image()->GetHeight();
        old_width = get_active_image()->GetWidth();
        if(event.GetId() == WIDTH_INPUT)
        {
            if(!_in_pixels)
            {
                set_height_label(get_width_input());
            }
            else
            {
                int input_width = wxAtoi(get_width_input());
                int new_height;
                new_height = (input_width*old_height)/(old_width);
                label.Printf("%d",new_height);
                set_height_label(label);
            }
        }
        if(event.GetId() == HEIGHT_INPUT)
        {
            if(!_in_pixels)
            {
                set_width_label(get_height_input());
            }
            else
            {
                int input_height = wxAtoi(get_height_input());
                int new_width;
                new_width = (input_height*old_width)/old_height;
                label.Printf("%d",new_width);
                set_width_label(label);
            }
        }
    }
}

void resizer_Frm::OnCheck(wxCommandEvent& event)
{
    if(_aspect_ratio->GetValue())
    {
        _aspect_locked = true;
    }
    else
    {
        _aspect_locked = false;
    }
}

void resizer_Frm::show_error(wxString msg)
{
    wxMessageDialog* error_dialog = new wxMessageDialog(this,
```

```
            msg,wxT("ERROR!!!!"), wxOK|wxCENTRE|wxICON_ERROR);
    error_dialog->ShowModal();
    error_dialog->Destroy();
}

void resizer_Frm::open_new_child(const wxString& title,
                            const wxImage the_image)
{
    resizer_Chl *sub_frame = new resizer_Chl(this,title,the_image);
    wxGetApp().Increase();
    sub_frame->Show(true);
    image_ok=true;
    wxFoldPanel temp(0);
    for(size_t i = 0; i<_tool_panel->GetCount();i++)
    {
        temp = _tool_panel->Item(i);
        _tool_panel->Expand(temp);
    }
    Refresh();
}

// Open an image
bool resizer_Frm::open_image(wxString filename)
{
    wxImage temp_image(filename,wxBITMAP_TYPE_ANY,-1);
    if(temp_image.IsOk())
    {
        open_new_child(filename,temp_image);
        return true;
    }
    else
        return false;
}

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

wxBitmap* resizer_Frm::get_active_image(void)
{
    wxMDIChildFrame* active_frame = this->GetActiveChild();
    resizer_Chl *active_child = wxDynamicCast(active_frame,resizer_Chl);
    return active_child->return_canvas()->return_image_bitmap();
}
```

A2.1.4. *resizer_App.cpp*

```
//---------------------------------------------------------------------------
//
// Name:        resize_App.cpp
// Author:      Administrator
// Description:
//
//---------------------------------------------------------------------------

#include "resizer_App.h"
#include "resizer_Frm.h"

IMPLEMENT_APP(resizer_app)

bool resizer_app::OnInit()
{
    wxInitAllImageHandlers();
    window_count = 0;
    resizer_Frm* frame = new resizer_Frm(NULL,wxID_ANY,_T("Bicubic
Interpolator"),
    wxDefaultPosition,wxSize(800,600),wxDEFAULT_FRAME_STYLE|wxHSCROLL|
wxVSCROLL);
    SetTopWindow(frame);
    frame->Show(true);
    frame->Maximize(true);
    return true;
}

int resizer_app::OnExit()
{
      return 0;
}

void resizer_app::Increase(void)
{
    window_count++;
}

void resizer_app::Decrease(void)
{
    window_count--;
}

int resizer_app::number_of_windows(void)
{
    return window_count;
}
```

A2.1.5. *resizer_App.h*

```
//-----------------------------------------------------------------------
//
// Name:        resizer_App.h
// Author:
// Description: Declaration for the resizer_app class
//              Part of the Image Resizer project for ELC 152
//
//-----------------------------------------------------------------------

#ifndef __RESIZER_APP_H__
#define __RESIZER_APP_H__

#ifndef WX_PRECOMP
      #include <wx/wx.h>
#else
      #include <wx/wxprec.h>
#endif

class resizer_app : public wxApp
{
    private:
        int window_count;
      public:
            bool OnInit();
            int  OnExit();
            void Increase(void);
            void Decrease(void);
            int number_of_windows(void);
};
#endif
```

A2.1.6. *resizer_Chl.cpp*

```
//-----------------------------------------------------------------//
// Name:        resizer_Chl.cpp
// Author:
// Description: resizer_Chl class implementation
//
//-----------------------------------------------------------------//

#include "resizer_Frm.h"
#include "resizer_App.h"
#include "resizer_Chl.h"
#include "my_canvas.h"

//Do not add custom headers between
//Header Include Start and Header Include End
//wxDev-C++ designer will remove them
////Header Include Start
#include <wx/app.h>
#include <wx/init.h>
#include <wx/settings.h>
////Header Include End
```

```
DECLARE_APP(resizer_app)
//event handlers
BEGIN_EVENT_TABLE(resizer_Chl, wxMDIChildFrame)
    EVT_SIZE(resizer_Chl::OnSize)
    EVT_CLOSE(resizer_Chl::OnClose)
    EVT_ACTIVATE(resizer_Chl::OnActivate)
END_EVENT_TABLE()

resizer_Chl::resizer_Chl(resizer_Frm *parent, const wxString& title,
                                          const wxImage&  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();
}

resizer_Chl::~resizer_Chl(void)
{
}

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

void resizer_Chl::OnClose(wxCloseEvent& event)
{
    wxGetApp().Decrease();
    my_parent->set_image_properties(wxEmptyString);
    event.Skip();
}

void resizer_Chl::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("\n")
                    _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");

    if(my_parent->is_in_pixels())
    {
        my_parent->set_image_properties(temp_labels);
        my_parent->set_height_label(height_label);
```

```
        my_parent->set_width_label(width_label);
    }
    else
    {
        my_parent->set_image_properties(temp_labels);
        my_parent->set_height_label(_T("100"));
        my_parent->set_width_label(_T("100"));
    }
    my_parent->Refresh();
    event.Skip();
}

my_canvas* resizer_Chl::return_canvas(void)
{
    return display_canvas;
}

wxString resizer_Chl::return_image_name(void)
{
    return image_name;
}

resizer_Frm* resizer_Chl::return_my_parent(void)
{
    return my_parent;
}
```

A2.1.7. resizer_Chl.h

```
//-----------------------------------------------------------------------
//
// Name:        resizer_Chl.h
// Author:
// Description: Declaration for the resizer_Chl class
//              Part of the Image Resizer project for ELC 152
//
//-----------------------------------------------------------------------

#ifndef __RESIZER_CHL_H__
#define __RESIZER_CHL_H__

//wxWidgets libraries included
#include <wx/wx.h>
#include <wx/mdi.h>
#include <wx/string.h>
//standard C++ libraries
#include <cstdlib>
#include <string.h>
//other includes
#include "my_canvas.h"
#include "resizer_Frm.h"

class resizer_Chl : public wxMDIChildFrame
{
      private:
            my_canvas* display_canvas;
        resizer_Frm* my_parent; //pointer to parent, for access
```

```
        wxString image_name;

      public:
        //Constructor/Deconstructor
            resizer_Chl(resizer_Frm *parent, const wxString& title,
                                        const wxImage& the_image);
            virtual ~resizer_Chl();

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

            //Accessor functions
            my_canvas* return_canvas(void);
        resizer_Frm* return_my_parent(void);
        wxString return_image_name(void);
      DECLARE_EVENT_TABLE();
};
#define CHILD_QUIT 4
#endif
```

A2.1.8. *resizer_Frm.cpp*

```
//------------------------------------------------
//
// Name:        resizer_Frm.cpp
// Author:
// Description: Resizer_Frm class implementation
//
//------------------------------------------------

#include "resizer_App.h"
#include "resizer_Frm.h"
#include "resizer_Chl.h"

DECLARE_APP(resizer_app)

wxMenuBar *CreateMenuBar()
{
    wxMenu* file_menu  = new wxMenu;
    wxMenu* about_menu = new wxMenu;
      file_menu->Append(MDI_OPEN, _T("&Open Image\tCtrl+O"),
                                  _T("Opens an image File."));
      file_menu->Append(MDI_SAVE, _T("&Save Image\tCtrl+S"),
                                  _T("Saves the Image File."));
      file_menu->AppendSeparator();
      file_menu->Append(CHILD_QUIT, _T("&Close Image\tCtrl+X"),
                                   _T("Close Active Image."));
      file_menu->AppendSeparator();
      file_menu->Append(MDI_QUIT, _T("&Quit Application\tAlt+F4"),
```

```
                                    _T("Quit the Application"));
    about_menu->Append(MDI_ABOUT, _T("&About\tF1"),
                                    _T("More Information"));
    wxMenuBar* menu_bar = new wxMenuBar;
    menu_bar->Append(file_menu,  wxT("&File"));
    menu_bar->Append(about_menu, wxT("&Help"));
    return menu_bar;
}

wxTextValidator *validator()
{
    // 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");
    input_validator->SetIncludes(array);
    return input_validator;
};

////Event Table Start
BEGIN_EVENT_TABLE(resizer_Frm,wxMDIParentFrame)
    EVT_UPDATE_UI(wxID_ANY,resizer_Frm::OnUpdate)
    EVT_MENU(wxID_ANY,resizer_Frm::OnMenu)
      EVT_CLOSE(resizer_Frm::OnClose)
      EVT_SIZE(resizer_Frm::OnSize)
    EVT_SASH_DRAGGED(SIDEBAR,resizer_Frm::OnSashDrag)
    EVT_CHECKBOX(ASPECT_RATIO,resizer_Frm::OnCheck)
    EVT_CHOICE(DIMENSION_TYPE,resizer_Frm::OnChoice)
    EVT_BUTTON(APPLY_TO_IMAGE,resizer_Frm::OnApply)
    EVT_TEXT(wxID_ANY,resizer_Frm::OnText)
END_EVENT_TABLE()
////Event Table End

resizer_Frm::resizer_Frm(wxWindow *parent, const wxWindowID id,
                         const wxString &title, const wxPoint& position,
                         const wxSize& size, const long style)
        : wxMDIParentFrame(parent, id, title, position, size, style)
{
      //initialize member variables (for sidebar)
    _aspect_ratio = NULL;
    _height_label = NULL;
    _width_label = NULL;
    _height_input = NULL;
    _width_input = NULL;
    _dimensions = NULL;
    _apply_resize = NULL;
    _tool_panel = NULL;
    _sidebar = NULL;
    _image_properties = NULL;
    image_ok = false;

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

```
      // Applying keyboard shortcuts to the actions in the menu
    wxAcceleratorEntry shortcuts[4];
    shortcuts[0].Set(wxACCEL_CTRL, (int)'o', MDI_OPEN);
    shortcuts[1].Set(wxACCEL_CTRL, (int)'s', MDI_SAVE);
    shortcuts[2].Set(wxACCEL_ALT, WXK_F4, MDI_QUIT);
    shortcuts[3].Set(wxACCEL_CTRL, WXK_F4, CHILD_QUIT);
    shortcuts[4].Set(wxACCEL_NORMAL, WXK_F1, MDI_ABOUT);
    wxAcceleratorTable key_shortcuts(5,shortcuts);
    SetAcceleratorTable(key_shortcuts);

    //Create Status bar and toolbar
    CreateStatusBar();
    SetMenuBar(CreateMenuBar());
}

resizer_Frm::~resizer_Frm(void)
{
    DestroyChildren();
}

wxSashLayoutWindow* resizer_Frm::CreateSash(void)
{
    wxSashLayoutWindow* temp = new wxSashLayoutWindow(this,SIDEBAR,
                               wxDefaultPosition,wxSize(200,30),
                               wxBORDER_THEME|wxSW_3DBORDER |
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* resizer_Frm::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);
    wxCaptionBarStyle cs;
    cs.SetCaptionStyle(wxCAPTIONBAR_RECTANGLE);

    //problem here
    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("Resizing Options"),true,cs);
```

```
      _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"));
      _aspect_ratio = new wxCheckBox(panel_temp2.GetParent(),
                      ASPECT_RATIO,_T("Lock Aspect Ratio"),
                      wxDefaultPosition,wxDefaultSize,0,
                      wxDefaultValidator,_T("Aspect Ratio"));
      _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"));
    _height_input->SetValidator(*validator());
    _width_input->SetValidator(*validator());

    wxString choices[] =
    {
        _T("pixels"),_T("% percent")
    };

    _dimensions = new wxChoice(panel_temp2.GetParent(),
                      DIMENSION_TYPE,wxDefaultPosition,
                      wxDefaultSize,2,choices,0,
                      wxDefaultValidator,_T("dimensions in:"));
    _dimensions->SetSelection(0);
    _in_pixels=true;
    _apply_resize = new wxButton(panel_temp2.GetParent(),APPLY_TO_IMAGE,
                                 _T("Apply Resize"),wxDefaultPosition,
                                 wxDefaultSize,0);
    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);
    temp-
>AddFoldPanelWindow(panel_temp2,_dimensions,wxFPB_ALIGN_WIDTH,5,20);
    temp-
>AddFoldPanelWindow(panel_temp2,_aspect_ratio,wxFPB_ALIGN_WIDTH,5,50);
    temp-
>AddFoldPanelWindow(panel_temp2,_apply_resize,wxFPB_ALIGN_WIDTH,5,25);
    return temp;
}

void resizer_Frm::OnUpdate(wxUpdateUIEvent& event)
{
    if (event.GetId()== MDI_SAVE)
        event.Enable(image_ok && (wxGetApp().number_of_windows()));
    if (event.GetId()== CHILD_QUIT)
        event.Enable(image_ok && (wxGetApp().number_of_windows()));
```

```
    if (event.GetId()== APPLY_TO_IMAGE)
        event.Enable(image_ok && (wxGetApp().number_of_windows()));
    if(event.GetId()== ASPECT_RATIO)
    {
        _aspect_ratio->SetValue(_aspect_locked);
        wxMenuBar *menu_bar = this->GetMenuBar();
        wxMenu *menu = menu_bar->GetMenu(1);
        menu->Check(ASPECT_RATIO,_aspect_locked);
        SetMenuBar(menu_bar);
    }
}

bool resizer_Frm::is_aspect_locked(void)
{
    return _aspect_locked;
}


bool resizer_Frm::is_in_pixels(void)
{
    return _in_pixels;
}

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

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

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

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

void resizer_Frm::OnSize(wxSizeEvent& event)
{
    wxLayoutAlgorithm layout;
    layout.LayoutMDIFrame(this);
}

void resizer_Frm::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 resizer_Frm::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_QUIT: menu_Quit(); break;
        case MDI_ABOUT: menu_About(); break;
    }
}

void resizer_Frm::menu_Aspect_Ratio(wxCommandEvent& event)
{
    _aspect_locked = event.IsChecked();
}

void resizer_Frm::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,extension;
        filename=open_file_dlg->GetFilename();
        path = open_file_dlg->GetPath();
        wxFileName the_file(path,filename);
        extension = the_file.GetExt();
        if(!open_image(filename))
        {
            show_error(_T("Error opening image file"));
        }
    }
    open_file_dlg->Destroy();
    Refresh();
}

void resizer_Frm::menu_Save()
{
    wxString wildcards;
    wildcards = _T("BMP file (*.bmp)|*.bmp")
                _T("|JPEG file (*.jpg)|*.jpg")
                _T("|PNG file (*.png)|*.png");
    wxFileDialog* save_file_dlg = new wxFileDialog(this,wxT("Save Image
```

```
file"),
                                        wxEmptyString,wxEmptyString,wildc
ards,
                                        wxFD_SAVE |wxFD_OVERWRITE_PROMPT|
                                        wxFD_FILE_MUST_EXIST|
                                        wxFD_CHANGE_DIR|wxCENTRE);
    if(save_file_dlg->ShowModal()==wxID_OK)
    {
        wxString filename,path,extension;
        filename=save_file_dlg->GetFilename();
        path = save_file_dlg->GetPath();
        wxFileName the_file(path,filename);
        extension = the_file.GetExt();
        if(!save_image(filename))
        {
            show_error(_T("Save Error"));
        }
    }
    save_file_dlg->Destroy();
}

void resizer_Frm::menu_Quit()
{
    Close();
}

void resizer_Frm::set_image_properties(wxString label)
{
    //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 resizer_Frm::menu_About()
{
    wxString message;
    message.Printf(_T("Welcome to The Image Resizer using \n "
                    "Bicubic Interpolation by \nElaine Ramos \n"
                    "JP Talusan \nIsadelle De Guzman"));
    wxMessageBox(message,_T("About"));
}

void resizer_Frm::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 resizer_Frm::OnChoice(wxCommandEvent& event)
{
    if(_dimensions->GetSelection()==0)
    {
        _in_pixels=true;
    }
    else
    {
        _in_pixels=false;
        set_height_label(_T("100"));
        set_width_label(_T("100"));
    }
}
```

A2.1.9. *resizer_Frm.h*

```
//-----------------------------------------------------------------------
//
// Name:        resizer_Frm.h
// Author:
// Description: Declaration for the resizer_Frm class
//              Part of the Image Resizer project for ELC 152
//
//-----------------------------------------------------------------------

#ifndef __RESIZER_FRM_H__
#define __RESIZER_FRM_H__

//wxWidgets libraries included
#include <wx/wx.h>
#include <wx/toolbar.h>
#include <wx/panel.h>
#include <wx/foldbar/foldpanelbar.h>
```

```
#include <wx/laywin.h>
#include <wx/mdi.h>
#include <wx/file.h>
#include <wx/filefn.h>
#include <wx/menuitem.h>
#include <wx/app.h>
#include <wx/init.h>
#include <wx/icon.h>
#include <wx/defs.h>
#include <wx/valtext.h>
#include <wx/filedlg.h>
#include <wx/progdlg.h>
#include <wx/filename.h>
#include <wx/xrc/xmlres.h>
#include <wx/msw/helpchm.h>
#include <wx/utils.h>
#include <wx/stdpaths.h>
#include <wx/wfstream.h>

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


//enumeration of wxIDs (identifies GUI items)
enum
{
    SIDEBAR=100,
    HEIGHT_INPUT,
    WIDTH_INPUT,
    DIMENSION_TYPE,
    APPLY_TO_IMAGE,
    ASPECT_RATIO,
    IMAGE_PROPERTIES,
    //Menu Items
    MDI_OPEN,
    MDI_SAVE,
    MDI_DISPLAY,
    MDI_ABOUT,
    MDI_QUIT =wxID_EXIT,
};

class resizer_Frm : public wxMDIParentFrame
{
    private:
        //bool flags
        bool image_ok;
        bool _in_pixels;
        bool _aspect_locked;


        //GUI items
        wxFoldPanelBar *_tool_panel;
        wxCheckBox *_aspect_ratio;
        wxStaticText *_height_label;
        wxStaticText *_width_label;
        wxStaticText *_image_properties;
        wxTextCtrl *_height_input;
```

```
        wxTextCtrl *_width_input;
        wxChoice *_dimensions;
        wxButton *_apply_resize;
        //End GUI items

    protected:
        wxSashLayoutWindow* _sidebar;

    public:
        //Constructor/Deconstructor
        resizer_Frm(wxWindow *parent, const wxWindowID id,const
wxString& title,
                    const wxPoint& position,const wxSize& size, const
long style);
        virtual ~resizer_Frm();
        //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
        void set_image_properties(wxString label);
        bool is_aspect_locked(void);
        bool is_in_pixels(void);
        wxString get_height_input(void);
        wxString get_width_input(void);
        void set_height_label(wxString label);
        void set_width_label(wxString label);

        // dimensions of the image
        int input_height, input_width,
        old_height, old_width,
        new_height, new_width;
        int** matrix;

        //User Input Related functions
        void OnApply(wxCommandEvent& event);
        void OnText(wxCommandEvent& event);
        void OnCheck(wxCommandEvent& event);
        void OnChoice(wxCommandEvent& event);
        void OnMenu(wxCommandEvent& event);
        void menu_About();
        void menu_OpenFile();
        void menu_Save();
        void menu_Quit();
        void menu_Aspect_Ratio(wxCommandEvent& event);
        void menu_Display(wxCommandEvent& event);
        void OnClose(wxCloseEvent& event);
        //Utility functions
        void show_error(wxString msg);
        void open_new_child(const wxString& title,
                            const wxImage my_image);
        bool open_image(wxString filename);
```

```
        //Error handling is in function
        bool save_image(wxString filename);
        wxBitmap* get_active_image(void);

          wxImage bicubic_interpolation(wxImage &source, int newx, int
newy);
          float bicubicSpline(float x);
    DECLARE_EVENT_TABLE()
};

#endif
```

References

[1] Giassa, Matthew. *Engineering, Medical Research, and 3D Image Processing.* 2009. http://www.giassa.net/?page_id=371 (accessed September 2010).

[2] J. Anthony Parker, Robert V. Kenyon and Donald E. Troxel. "Comparison of Interpolating Methods for Image Resampling." *IEEE TRANSACTIONS ON MEDICAL IMAGING VOL. MI-2, NO.1, MARCH*, 1983.

[3] Joshoua C. Esmenda, Mark Andrew S. Mateo and Jerome Vergil S. Paez. *Image Resizer.* Quezon City: Department of Electronics, Computer and Communications Engineering, 2008.

[4] Lancaster, Don. *Welcome to Don Lancaster's Guru Lair.* 2007. www.tinaja.com/glib/pixintpl.pdf (accessed September 2010).

[5] Michael Unser, Philippe Thevanaz and Leonid Yaroslavsky. "Convolution-Based Interpolation for Fast, High-Quality Rotation of Images." *IEEE TRANSACTIONS ON IMAGE PROCESSING VOL.4, NO.10, OCTOBER*, 1995.

[6] William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery. *Numerical Recipes The Art of Scientific Computing Third Edition.* New York: Cambridge University Press, 2007.

[7] Blake L. Carlson. *wxImageResampler.cpp: implementation of the wxImageResampler class.* The University of Iowa. http://www.koders.com/cpp/fid944571151037ADD80C4E890A042F44CADBACDF23.aspx?s=%22Blake+Carlson%22#L36 (accessed October 2010)