3D Histogram Display II

A Project by

Ricson O. Chua
Sherico Paulo D. dela Cruz
Jose L. Jimenez III

Submitted to

Luisito Agustin
Instructor, CE101

In Partial Fulfillment of the Requirements for the Course
CE 101: Techniques in Signal Processing

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

January 2011

Abstract

This project displays a conventional image histogram in 3D format. The three axes represent the three main colors of an image – red, green, and blue. A histogram displays the frequency of the colors by a cube representation in the graph – the more frequent the color appears, the larger the cube is displayed on the graph. The cubes' colors share the same color as they represent.

The software includes sliders which can change the background color of the graph for a better viewing of the colors of the cubes. Using the mouse, the user can move, zoom, and pan the graph for a better viewing in all angles. The graph may also be viewed given the range of values for the relative frequency of the colors. In this way, cubes represent a range of colors and not just a single one. A report can be generated to display the frequency at which the colors appeared in the histogram, depending on the desired number of bit display. Finally, a distribution slider is also included in the system. The user supplies the value of the distribution percentage and whatever that value is, only frequency percentage higher than or equal to that value will be displayed on the histogram.

The software is created using wxDev-C++ with additional libraries, OpenCV and OpenGL for the opening of files and creating the 3D graph respectively. The software opens all common image types and displays it side by side with the graph.

Table of Contents

Page

Abstract…….... 2

Chapter 1. Introduction ……..…….... 4

1.1. Objectives …….. 4

1.2. Scope and Limitations ……... 4

1.3. Significance …… 4

Chapter 2. Theoretical Background …….. 5

2.1. Image …..... 5

2.2. Histogram …..... 5

Chapter 3. Software Requirements 7

3.1. Previous Software …….. 7

3.1.1. Installations and other Requirements ……. 7

3.1.2. Software features……. 8

Chapter 4. Implementation …… 12

4.1. Parts of the Program…….. 12

4.1.1. MyGLCanvas.cpp……..... 12

4.1.2. Project1Frm.cpp…….. 16

Chapter 5. Conclusion..... 19

Appendix 1. Source Code…….. .20

A1.1. Project1Frm.cpp……. .20

A1.2. drawingaxes.h……. 29

A1.3. drawingcubes.h…….. 30

A1.4. MyGLCanvas.h…….. 32

A1.5. MyGLCanvas.cpp……. . 33

A1.6. Project1App.cpp……. 40

A1.7. Project1Frm.h…….. 41

A1.8. Compiler options additional parameters…… 43

Bibliography 44

1. Introduction

1.1. Objectives

This project aims to construct a 3D diagram display that is more improved than the previous project with the help of wxDev-C++, OpenCV, and OpenGL graphics. It opens and displays all common types of images. Using sliders, the user can change the color of the background of the graph and using the mouse, the user can move, zoom, and pan the graph as desired. A distribution slider is aimed to display only the higher frequency values for precise viewing.

1.2. Scope and Limitations

The software accepts images and auto-allocates the range levels for variable frequency distribution for plotting in the graph. However, it does accept almost every type of image file.

1.3. Significance

Histograms are generally used to determine if an image has a correct exposure, which when graphed gives a bell-shaped curve (low values on either end of the graph and increasing values to the center). When plotted on a 3D graph for the three colors, clusters of cubes (large sizes) are found on the center of the cubic space.

2. Theoretical Background

2.1. Image

Images are representations of any actual object. It is a rectangular grid of pixels that has a definite height and width. A digital image can be counted in pixels which are obtained by dividing the image into rows and columns. Each pixel is a square and has a fixed size on a given display. Each pixel has a color which is represented by a 32-bit integer. The first eight bits determine the redness of a pixel, the next eight bits determine the greenness, the next eight bits determine the blueness and the remaining eight bits determine the transparency with 255 being completely opaque and 0 being completely transparent. A pixel with 0 in all color channels is black while a pixel with 255 in all color channels is white. By varying the values of the color bits, we can produce a total of 16,777,216 colors.

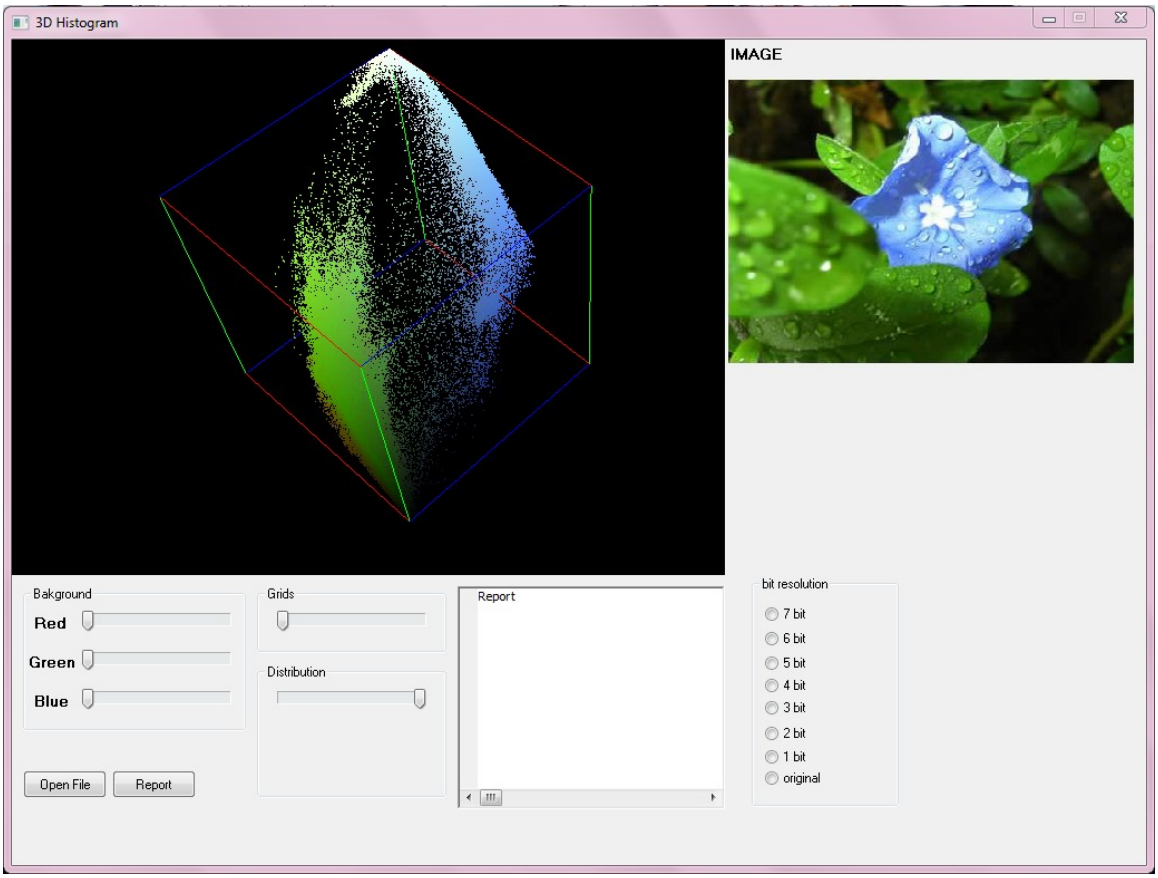
An example of an image

2.2. Histogram

A histogram is a chart that depicts the distribution of a set of data. This kind of chart does not reflect process performance over time rather it displays a frequency distribution of values. Normally, histograms are used to summarize large data sets graphically and to compare frequency of values easily.

As a different case of the histogram, the 3D histogram displays the frequencies of pixels of different colors that are present in an image. The 3D histogram is made up of cubes that represent the RGB values of each color. The x-axis, y-axis and the z-axis represent

the redness, greenness and the blueness of a color, respectively. All axes range from a value of 0 to 255.



Screenshot of the custom software

3. Software Requirements

3.1. Previous Software

Initially, a similar project was made but was not ported into a widget. Aside from not being a widget, there are a lot of shortcomings with the previous iteration of the software, like its disability to open different image format which we believe was crucial.

3.1.1. Installations and Other Instructions

This software requires an installation of WxDevC++ 7.3.1.3 if users plan on recompiling the code. Other requirements are as follows.

Download openCV_1.0 for windows and install.

Download glew-1.5 and glut 3.7.6+ devpak and install the packages.

Open WxDev-C++. In compiler options of tools tab, type -lhighgui -lcv -lcxcore

-lcvaux -lglut32 -lglu32 -lopengl32 -lwinmm -lgdi32 to linker command line.

Switch to the Directories tab, and in the Libraries sub-tab add to directories:

C:\Program Files\OpenCV\lib

Switch to the C includes tab. Add the following to include directories:

C:\Program Files\OpenCV\cxcore\include

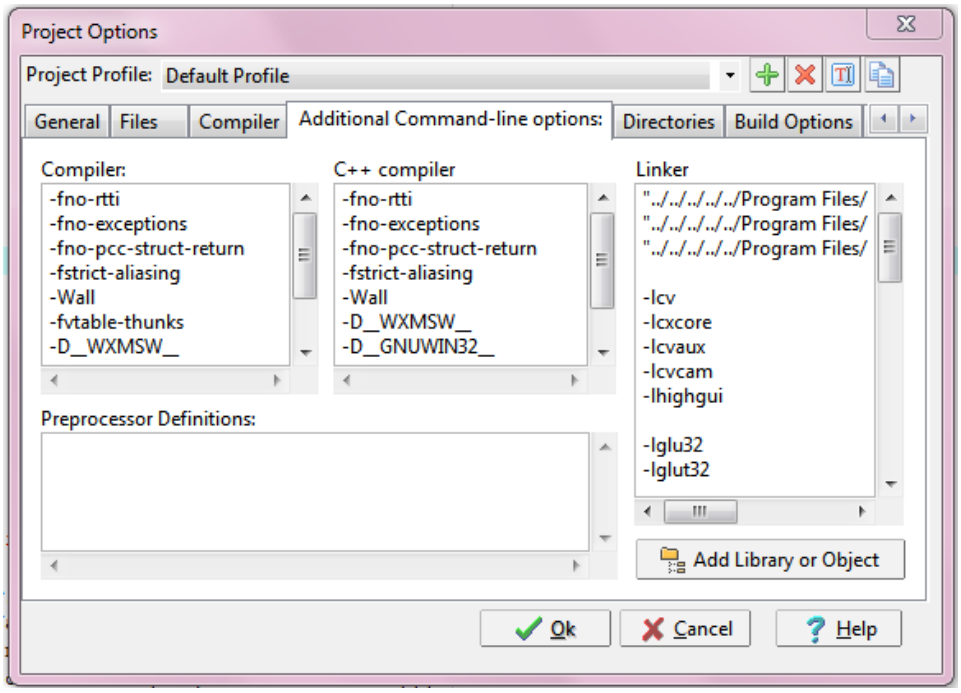
C:\Program Files\OpenCV\cv\include

C:\Program Files\OpenCV\otherlibs\highgui

C:\Program Files\OpenCV\cvaux\include

C:\Program Files\OpenCV\otherlibs\cvcam\include

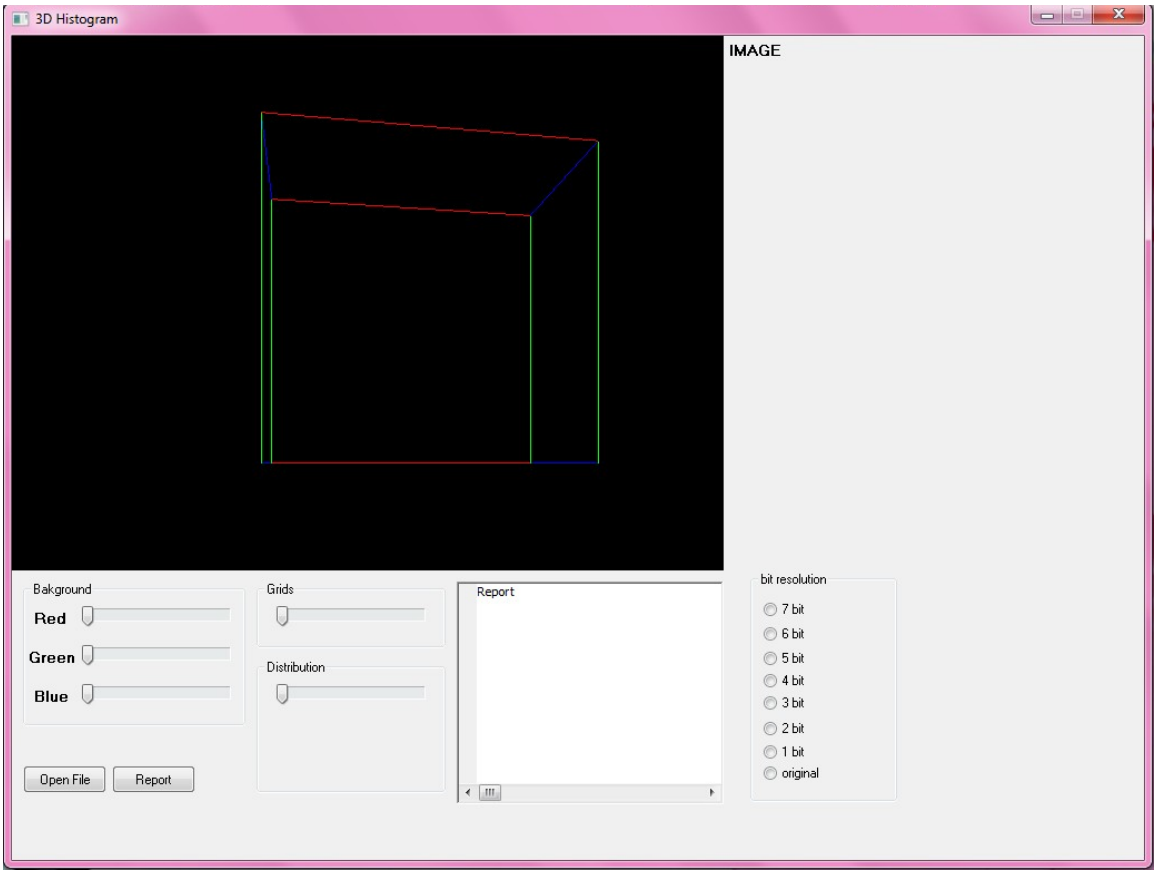
After all the settings have been added if the user goes to project -> project options the windows will look like as the one below



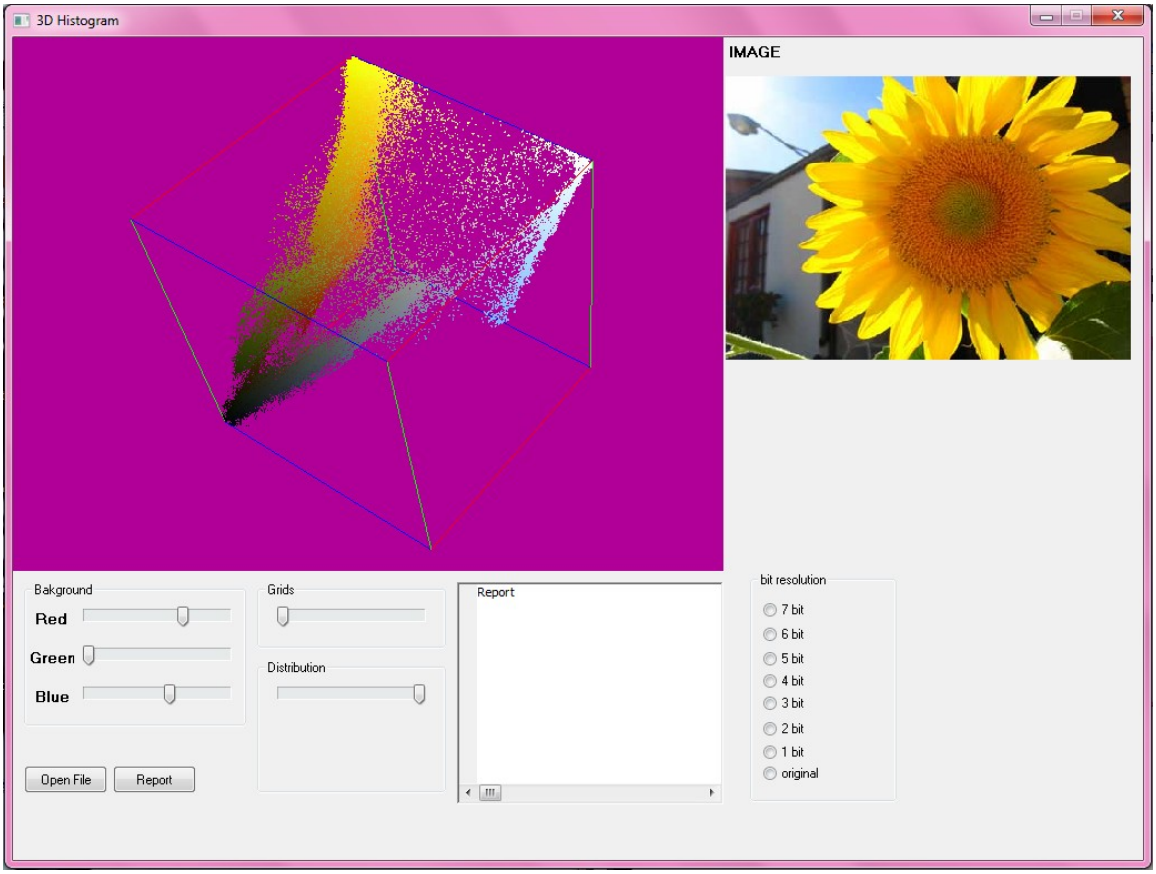
The other files included are can be searched at the Appendix section.

3.1.2. Software Features

Upon running the program a new window opens and a blank cube can be seen in the left corner

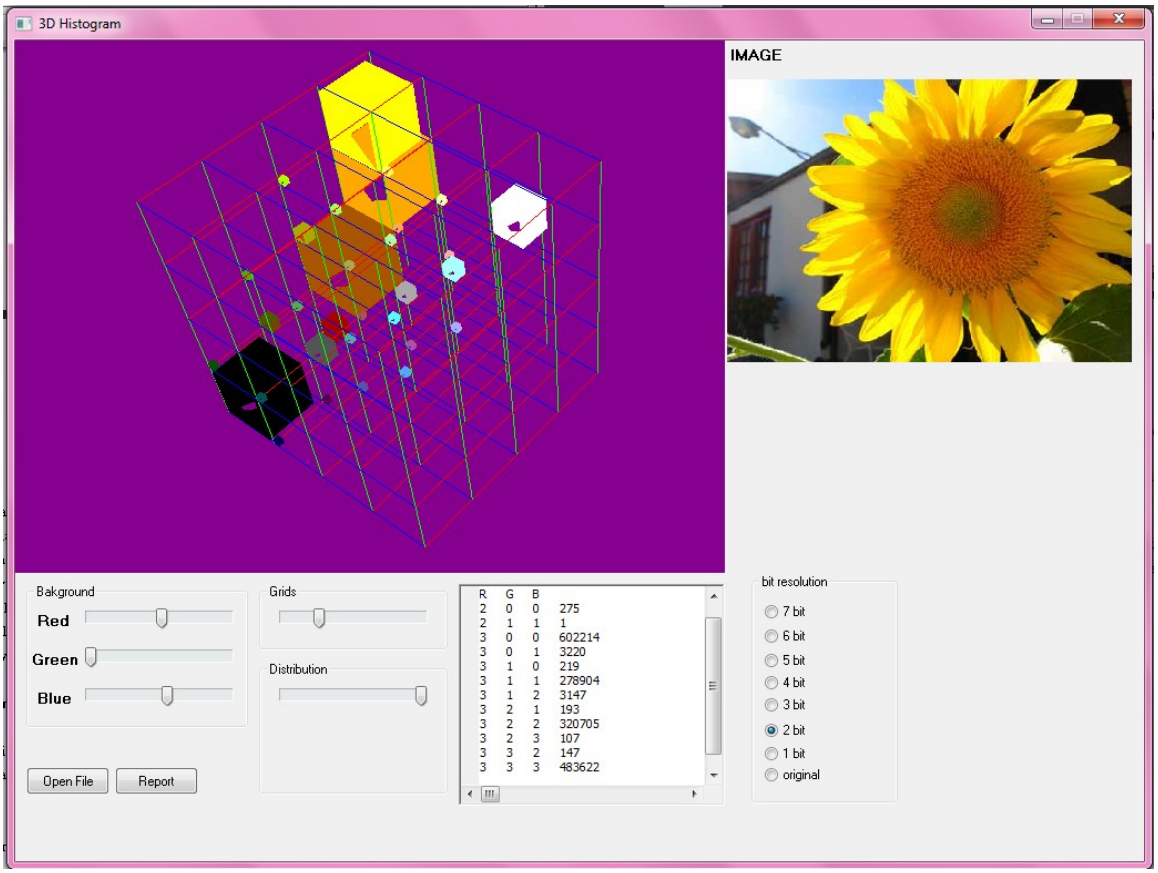


Clicking Open File opens an *Open File* dialog menu. Users can select various image files they want analyzed. Formats include .rgbtext; .bmp, .jpeg, .jpg, .png, .gif and all the variations of the said file types.



Upon opening a file the image will be displayed beside the 3D histogram. All images opened will be scaled to 365×255 pixels. Users may also adjust the color of the background using the sliders under the Background static box. The distribution slider is initially set to 100% upon opening a file. The percentage is a relative frequency of the image. The program searches for the color with the highest amount of pixels and sets it as the relative maximum. All the other color will now have a relative frequency based on the relative maximum by getting the ratio of the amount of pixels and the relative maximum. Another essential feature of this program is the ability to pan, tilt and zoom the 3D histogram. The controls are left click for panning, right click for zooming and left double click to rotate.

As it is right now, the 3D histogram is not very informative. With that problem in mind, this program allows the users to reduce the color bits and compresses results to a smaller resolution, thereby aggregating the results.



As seen here the results are compressed to 2 bits. That is (0, 0, 0) in a 2 bit result array will assume the values of (0, 0, 0) to (63, 63, 63) of the 8 bit results array. Aggregation of values was done to allow for a bigger sized cube to be made because initially the red, green and blue axes are divided into 256 divisions when aggregated to 2 bits the division are now just 4. Therefore a cube can be as large as 64 units as opposed to just 1 unit in the 256 division when an 8 bit results is plotted. To allow users to gauge the differences in size of the cubes the Grids slider is there add wireframes in between the axes. The program is capable of showing the grids to show all the 256 divisions.

Finally the program also has a Report button where users are allowed to generate reports of the current analysis. The report shows the number of pixels of the said color in the image file.

4. Implementation

4.1. Parts of the program

Most of the functions used by the program are members of MyGLCanvas class. This documentation will discuss the members of the said class and how they are used in Project1Frm.cpp

4.1.1. MyGLCanvas.cpp

The program was made possible through two popular image handling/processing libraries. Those are OpenGL and OpenCV, both of which are more or less a standard when it comes to 3D rendering and image processing respectively.

```
IplImage* img;
…
img = cvLoadImage(filename, 1);
```

`IplImage *img` creates an instance of the `IplImage` class of OpenCV. The image is loaded by taking the filename being returned by the open file dialog box. `cvLoadImage` takes in two parameters the first one is the filename and the other one is the mode. 1 denotes a pixel by pixel reading.

Starting from the upper left most pixel, the color stored in a pixel is checked and the corresponding RGB value stored in the pixel is incremented by one. The next pixel to the right is then checked and until the last pixel in the uppermost row is checked before moving to the next row. The process continues until all pixels are checked.

```
uchar* ptr;
for(int y=0; y<img->height; y++){
      for(int x=0; x<img->width; x++){
            ptr = cvPtr2D(img, y, x, NULL);
            pixel[ptr[2]][ptr[1]][ptr[0]]+= pixel[ptr[2]][ptr[1]]
```

```
            [ptr[0]]
            }
      }
```

After copying all the pixels still in the `InitGL()` function the program now proceeds to look for the relative maximum `maxval`.

```
      for(int red=0; red<256; red++){
            for(int green=0; green<256; green++){
                  for(int blue=0; blue<256; blue++){
                              if(pixel[red][green][blue]>maxval){
                              maxval=pixel[red][green][blue];
                        }
                  }
            }
      }
```

Once all the initializations are done a call to `Draw()` is made by the widget within Draw is the `wxClientDC` and `Paint(wxDC &dc)`. The paint function stores some of the features discussed awhile ago.

```
      glClearColor(x/255, y/255, z/255, 1.0);
```

This line is responsible for the background of the `wxPanel`. `glClearColor()` takes in 4 floats as parameters where the 1st 3 corresponds to RGB with 0 being the lowest possible value and 1 being the highest possible value. The values in our implementation, `x`, `y` and `z`, are bounded to the 3 sliders in the Background box which allows users to gain background colors accurate up to 8 bits. The painting algorithm itself is divided into two – one where in the results in its 8 bit form are plotted and everything else (please see A1.5. for the complete listing of the function `Paint(wxDC &dc))`

```
Paint(wxDC &dc)
{
      …
      if (factor<2)
      {
            …
            if((pixels/maxval)<=(Dist_ceil/100.0))
            {
                  drawingCubes(float (red)/10, float (green)/10, float
```

```
                (blue)/10);//draws cube for corresponding RGB value
                pixelcount++;
            }
        }
        Else
        {
            …
            if(current/maxval  <  0.1)  drawingCubesT(float  (red)/10,
        float (green)/10, float (blue)/10,factor, 0.1);
            …
        }
}
```

There are two methods of the drawing the cubes, which are in `drawingcubes.h`. `drawingCubesT` takes in the RGB values which also corresponds to the coordinates of the cube and a factor or the value which is in powers of 2, being divided to 256 to get a lowered bitrate and the percentage which is the relative frequency of the cube about to be plotted. (see A1.3 for complete listing of `drawingCubesT()`)

Another crucial function of the `MyGLCanvas` class is the `Compress()` function. The function is called frequently as each time the user select a radio button an array has to be reinitialized and recalculated. This is also the reason why another 3D array was used instead of just one. The 1st array `int ***pixel` holds the original 8 bit result for the other array `***agregate` to access repeatedly.

```
int bits = 256/factor;
    maxval = 0;
    for (int i=0; i<256/factor; i++){
        agregate[i]=new int* [256/factor];
        for (int j=0; j<256/factor; j++){
            agregate[i][j]=new int [256/factor];
            for (int k=0; k<256/factor; k++){
                agregate[i][j][k]=0;
            }
        }
    }
```

As seen here the 1st thing that `Compress()` does is to re-initialize an array to its needed size. Then it proceeds to sum the 8 bit values that share the same n-bit index

```
for(int red=0; red<256; red++){
          for(int green=0; green<256; green++){
               for(int blue=0; blue<256; blue++){
```

```
                                                    if(pixel[red]
                        [green][blue]>maxval)     maxval=pixel[red]
                        [green][blue];
                        agregate[red/factor][green/factor]
                        [blue/factor]+=pixel[red][green][blue];
                }
            }
        }
```

Finally `Compress()` looks for a new relative maximum which the class will later use to determine the size of the cube.

```
for(int red=0; red<bits; red++){
          for(int green=0; green<bits; green++){
               for(int blue=0; blue<bits; blue++){
                         if(agregate[red][green][blue]>maxval)
maxval=agregate[red][green][blue];
               }
          }
}
```

Other functions available in the class are as follows: event handlers, used in mouse movements, are handled inside the `OnMouseMove(wxMouseEvent& event)` function.

```
if (MouseButton == 1)
  {
          RotX=RotX + (GLfloat)(Y - Ycoord)/2;
          RotY=RotY + (GLfloat)(X - Xcoord)/2;
          Xcoord = X;
          Ycoord = Y;
  }
  if (MouseButton == 2)
  {
         Depth = Depth - (GLfloat)(Y-Zcoord)/5;
         Zcoord = Y;
  }
  if (MouseButton == 3)
  {
          PanX = PanX - (GLfloat)(Y - Ycoord)/10;
          PanY = PanY - (GLfloat)(-X + Xcoord)/10;
          Xcoord = X;
          Ycoord = Y;
  }

  Draw();
```

The `MouseButton` gets its values through events. A value of 1 means a left double click, a value of 2 means a right click and finally a value of 3 means a left click. Most of the other OpenGL functions used in the class are arbitrary like

```
glShadeModel(GL_SMOOTH);                    // Enables Smooth Color
```

```
Shading
glClearDepth(10.0);                          // Depth Buffer Setup
glEnable(GL_DEPTH_TEST);                    // Enable Depth Buffer
glDepthFunc(GL_LESS);                          // The Type Of Depth
Test To Do
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);   //Realy Nice
perspective calculations
```

Other functions are used to carry out the pan, zoom and tilt functions.

```
glTranslated(PanY,PanX, Depth);
glRotatef(RotX, 1.0, 0.0, 0.0);
glRotatef(RotY, 0.0, 1.0,0.0);
```

As seen above the variables `PanX`, `PanY`, `RotX`, `RotY` and `Depth` are all variable in the `OnMouseMove` function which is responsible for the pan tilt and zooming.

4.1.2. Project1Frm.cpp

This section of the documentation will explain how the functions in MyGLCanvas.cpp are used and how the class was bound to the `WxPanel`.

The `WxPanel` has various potential depending on how it is used it can be used to contain images, graphs and possibly animation. The constructor of `MyGLCanvas` takes in a `WxPanel` which then allows the class to perform various imaging tasks on the `WxPanel`. In the constructor of `Project1Frm`, we see how an instance of `MyGLCanvas` was created through the variable `pGLCanvas`.

```
pGLCanvas = new MyGLCanvas(WxPanel2, ID_WXPANEL2, wxPoint(0,0),
wxSize(1024,768));
```

The size of the panel, the name of the panel and the top-left coordinates are placed within the constructor of `MyGLCanvas`.

A crucial function in `Project1Frm.cpp` is the Open File button, which uses mostly functions built in the `wxWidgets`.

```
wxImage imagetest(loadimage1,wxBITMAP_TYPE_ANY,-1);
wxBitmap temp_img1(imagetest, -1);
image1 = temp_img1;
temp1 = image1.ConvertToImage();
image1 = wxBitmap(temp1.Rescale(365,255),-1);
sb1 = new wxStaticBitmap(WxScrolledWindow1,-1,image1);
int image1Width = image1.GetWidth();
int image1Height = image1.GetHeight();
WxScrolledWindow1->SetScrollbars(0,0,image1Width,image1Height);
WxScrolledWindow1->Scroll(1,1);
sb1->Show(true);
WxSlider5->SetValue(100);
```

Difficulty was encountered as there were two types of image handling functions in the `wxWidgets`. Most of the control codes are just routines. One function is the `WxSlider5Scroll`.

```
void Project1Frm::WxSlider5Scroll(wxScrollEvent& event)
{
    // insert your code here
    pGLCanvas->Dist_ceil = WxSlider5->GetValue();
    pGLCanvas->Draw();
}
```

Most of the slider functions pass values from the slider to `pGLCanvas`. The one above is the one responsible for the distribution slider.

As for the radio buttons, it is coded explicitly, nothing out of the ordinary instantation.

```
void Project1Frm::WxRadioButton1Click(wxCommandEvent& event)
{
      // insert your code here
      pGLCanvas->factor = 2;
      pGLCanvas->Compress();
      pGLCanvas->Draw();
}
```

This is where `Compress()` gets called a lot because for every change in bit size, `Compress()` has to be executed.

4. Conclusion and Recommendation

The implementation of the 3D Histogram II in wxDev-C++ turned out well. The image was displayed along with the graph and the software achieved its full functionality. It also had its added functionalities such as the distribution bar, availability of ranges, and the menu bar which serves as the GUI of the software, eliminating the hassle of typing everything from the command prompt as do a simple Bloodshed Dev-C++.

Some revision that the group recommends is as follows. One is to change all the public variables into private variable and use getter and setter functions instead. Another recommendation is to have the textbox show partial results as opposed to waiting for everything to print to the textbox and make the program somewhat hang in image with high amounts of color availability. Finally a better knowledge of OpenGL may be necessary as some features are not maximized very well like the drawing of the cubes, the group is not entirely sure if hardware discrepancy was the reason for the holes in the box or if it was just an optimization problem but both the previous groups agree that it is a hardware discrepancy as most monitoring software warns us of extreme hardware usage when the cubes are being drawn.

Appendix 1. Source Code

Project1Frm.cpp

```
//-----------------------------------------------------------------------------
//
// Name:        Project1Frm.cpp
// Author:      Ross
// Created:     15/04/2005 2:23:03 PM
//
//-----------------------------------------------------------------------------

#include "Project1Frm.h"
#include <vector>
#include <list>
#include <string>
using namespace std;
//Do not add custom headers.
//wx-dvcpp designer will remove them
////Header Include Start
////Header Include End


//-----------------------------------------------------------------------------
// Project1Frm
//-----------------------------------------------------------------------------
   //Add Custom Events only in the appropriate Block.
   // Code added in  other places will be removed by wx-dvcpp
  ////Event Table Start
BEGIN_EVENT_TABLE(Project1Frm,wxFrame)
       ////Manual Code Start
       ////Manual Code End

       EVT_CLOSE(Project1Frm::Project1FrmClose)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON8,Project1Frm::WxRadioButton8Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON7,Project1Frm::WxRadioButton7Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON6,Project1Frm::WxRadioButton6Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON5,Project1Frm::WxRadioButton5Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON4,Project1Frm::WxRadioButton4Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON3,Project1Frm::WxRadioButton3Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON2,Project1Frm::WxRadioButton2Click)
       EVT_RADIOBUTTON(ID_WXRADIOBUTTON1,Project1Frm::WxRadioButton1Click)
       EVT_BUTTON(ID_WXBUTTON2,Project1Frm::WxButton2Click)
       EVT_BUTTON(ID_WXBUTTON1,Project1Frm::WxButton1Click)
       EVT_SLIDER(ID_WXSLIDER5,Project1Frm::WxSlider5Slider)
       EVT_COMMAND_SCROLL(ID_WXSLIDER5,Project1Frm::WxSlider5Scroll)

       EVT_COMMAND_SCROLL(ID_WXSLIDER4,Project1Frm::WxSlider4Scroll)

       EVT_COMMAND_SCROLL(ID_WXSLIDER3,Project1Frm::WxSlider3Scroll)

       EVT_COMMAND_SCROLL(ID_WXSLIDER2,Project1Frm::WxSlider2Scroll)

       EVT_COMMAND_SCROLL(ID_WXSLIDER1,Project1Frm::WxSlider1Scroll)

       EVT_UPDATE_UI(ID_WXPANEL2,Project1Frm::WxPanel2UpdateUI1)

       EVT_UPDATE_UI(ID_WXPANEL1,Project1Frm::WxPanel1UpdateUI)
END_EVENT_TABLE()
  ////Event Table End
```

```
Project1Frm::Project1Frm( wxWindow *parent, wxWindowID id, const wxString
&title, const wxPoint &position, const wxSize& size, long style )
    : wxFrame( parent, id, title, position, size, style)
{
    CreateGUIControls();
         pGLCanvas = new MyGLCanvas(WxPanel2, ID_WXPANEL2, wxPoint(0,0),
wxSize(1024,768));
    wxInitAllImageHandlers();

}


Project1Frm::~Project1Frm() {}

void Project1Frm::CreateGUIControls(void)
{
    wxString wildcards;
    wildcards = _T("Image Files(*.rgbtext;*.bmp;*.jpeg;*.jpg;*.png;*.gif)")
                _T("|*.rgbtext;*.bmp;*.jpeg;*.jpg;*.png;*.gif")
                _T("|RGBtext files (*.rgbtext)|*.rgbtext")
                _T("|BMP files (*.bmp)|*.bmp")
                _T("|JPEG files (*.jpeg;*.jpg)|*.jpeg;*.jpg")
                _T("|PNG files (*.png)|*.png")
                _T("|GIF files (*.gif)|*.gif");

    //Do not add custom Code here
    //wx-devcpp designer will remove them.
    //Add the custom code before or after the Blocks
    ////GUI Items Creation Start

       WxPanel1 = new wxPanel(this, ID_WXPANEL1, wxPoint(0, 0), wxSize(1008,
730));
       WxPanel1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       wxAnimationCtrl1 = new wxAnimationCtrl(WxPanel1, ID_WXANIMATIONCTRL1,
wxNullAnimation, wxPoint(762, 190), wxSize(2, 1) );
       wxAnimationCtrl1->Enable(false);
       wxAnimationCtrl1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

       WxPanel2 = new wxPanel(WxPanel1, ID_WXPANEL2, wxPoint(0, 0), wxSize(640,
480));
       WxPanel2->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       Bakground = new wxStaticBox(WxPanel1, ID_BAKGROUND, wxT("Bakground"),
wxPoint(10, 490), wxSize(200, 130));
       Bakground->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       Grids = new wxStaticBox(WxPanel1, ID_GRIDS, wxT("Grids"), wxPoint(220,
490), wxSize(170, 60));
       Grids->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS Sans
Serif")));

       Distribution = new wxStaticBox(WxPanel1, ID_DISTRIBUTION,
wxT("Distribution"), wxPoint(220, 560), wxSize(170, 120));
       Distribution->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

       WxSlider1 = new wxSlider(WxPanel1, ID_WXSLIDER1, 0, 0, 255, wxPoint(55,
```

```
510), wxSize(150, 35), wxSL_HORIZONTAL | wxSL_SELRANGE , wxDefaultValidator,
wxT("WxSlider1"));
       WxSlider1->SetRange(0,255);
       WxSlider1->SetValue(0);
       WxSlider1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       WxSlider2 = new wxSlider(WxPanel1, ID_WXSLIDER2, 0, 0, 255, wxPoint(55,
545), wxSize(150, 35), wxSL_HORIZONTAL | wxSL_SELRANGE , wxDefaultValidator,
wxT("WxSlider2"));
       WxSlider2->SetRange(0,255);
       WxSlider2->SetValue(0);
       WxSlider2->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       WxSlider3 = new wxSlider(WxPanel1, ID_WXSLIDER3, 0, 0, 255, wxPoint(55,
580), wxSize(150, 35), wxSL_HORIZONTAL | wxSL_SELRANGE , wxDefaultValidator,
wxT("WxSlider3"));
       WxSlider3->SetRange(0,255);
       WxSlider3->SetValue(0);
       WxSlider3->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       WxSlider4 = new wxSlider(WxPanel1, ID_WXSLIDER4, 1, 1, 9, wxPoint(230,
510), wxSize(150, 35), wxSL_HORIZONTAL | wxSL_SELRANGE , wxDefaultValidator,
wxT("WxSlider4"));
       WxSlider4->SetRange(1,9);
       WxSlider4->SetValue(1);
       WxSlider4->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       WxSlider5 = new wxSlider(WxPanel1, ID_WXSLIDER5, 0, 0, 100, wxPoint(230,
580), wxSize(150, 35), wxSL_HORIZONTAL | wxSL_SELRANGE , wxDefaultValidator,
wxT("WxSlider5"));
       WxSlider5->SetRange(0,100);
       WxSlider5->SetValue(0);
       WxSlider5->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       WxButton1 = new wxButton(WxPanel1, ID_WXBUTTON1, wxT("Open File"),
wxPoint(10, 655), wxSize(75, 25), 0, wxDefaultValidator, wxT("WxButton1"));
       WxButton1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       WxEdit1 = new wxTextCtrl(WxPanel1, ID_WXEDIT1, wxT(""), wxPoint(9, 633),
wxSize(200, 20), 0, wxDefaultValidator, wxT("WxEdit1"));
       WxEdit1->Show(false);
       WxEdit1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

       Red = new wxStaticText(WxPanel1, ID_RED, wxT("Red"), wxPoint(20, 515),
wxDefaultSize, 0, wxT("Red"));
       Red->SetFont(wxFont(10, wxSWISS, wxNORMAL, wxBOLD, false, wxT("MS Sans
Serif")));

       Green = new wxStaticText(WxPanel1, ID_GREEN, wxT("Green"), wxPoint(15,
550), wxDefaultSize, 0, wxT("Green"));
       Green->SetFont(wxFont(10, wxSWISS, wxNORMAL, wxBOLD, false, wxT("MS Sans
Serif")));

       Blue = new wxStaticText(WxPanel1, ID_BLUE, wxT("Blue"), wxPoint(20, 585),
wxDefaultSize, 0, wxT("Blue"));
       Blue->SetFont(wxFont(10, wxSWISS, wxNORMAL, wxBOLD, false, wxT("MS Sans
```

```
Serif")));

	WxButton2 = new wxButton(WxPanel1, ID_WXBUTTON2, wxT("Report"),
wxPoint(90, 655), wxSize(75, 25), 0, wxDefaultValidator, wxT("WxButton2"));
	WxButton2->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false, wxT("MS
Sans Serif")));

	WxStyledTextCtrl1 = new wxStyledTextCtrl(WxPanel1, ID_WXSTYLEDTEXTCTRL1,
wxPoint(400, 490), wxSize(240, 200), 0, wxT("WxStyledTextCtrl1"));
	WxStyledTextCtrl1->AppendText(wxT("Report"));
	WxStyledTextCtrl1->SetFocus();
	WxStyledTextCtrl1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRichTextCtrl1 = new wxRichTextCtrl(WxPanel1, ID_WXRICHTEXTCTRL1,
wxT(""), wxPoint(240, 620), wxSize(130, 35), 0, wxDefaultValidator,
wxT("WxRichTextCtrl1"));
	WxRichTextCtrl1->SetMaxLength(0);
	WxRichTextCtrl1->Show(false);
	WxRichTextCtrl1->AppendText(wxT("colorcount"));
	WxRichTextCtrl1->SetFocus();
	WxRichTextCtrl1->SetInsertionPointEnd();
	WxRichTextCtrl1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxScrolledWindow1 = new wxScrolledWindow(WxPanel1, ID_WXSCROLLEDWINDOW1,
wxPoint(642, 35), wxSize(365, 255), wxVSCROLL | wxHSCROLL);
	WxScrolledWindow1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxStaticText1 = new wxStaticText(WxPanel1, ID_WXSTATICTEXT1, wxT("IMAGE
FILE"), wxPoint(645, 5), wxDefaultSize, 0, wxT("WxStaticText1"));
	WxStaticText1->SetFont(wxFont(10, wxSWISS, wxNORMAL, wxBOLD, false,
wxT("MS Sans Serif")));

	wxArrayString arrayStringFor_WxRadioBox1;
	WxRadioBox1 = new wxRadioBox(WxPanel1, ID_WXRADIOBOX1, wxT("bit
resolution"), wxPoint(664, 481), wxSize(132, 207), arrayStringFor_WxRadioBox1,
1, wxRA_SPECIFY_ROWS, wxDefaultValidator, wxT("WxRadioBox1"));
	WxRadioBox1->SetSelection(0);
	WxRadioBox1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton1 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON1, wxT("7
bit"), wxPoint(676, 506), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton1"));
	WxRadioButton1->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton2 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON2, wxT("6
bit"), wxPoint(676, 528), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton2"));
	WxRadioButton2->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton3 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON3, wxT("5
bit"), wxPoint(676, 550), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton3"));
	WxRadioButton3->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton4 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON4, wxT("4
bit"), wxPoint(676, 570), wxSize(113, 17), 0, wxDefaultValidator,
```

```
wxT("WxRadioButton4"));
	WxRadioButton4->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton5 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON5, wxT("3
bit"), wxPoint(676, 590), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton5"));
	WxRadioButton5->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton6 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON6, wxT("2
bit"), wxPoint(676, 613), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton6"));
	WxRadioButton6->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton7 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON7, wxT("1
bit"), wxPoint(676, 634), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton7"));
	WxRadioButton7->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxRadioButton8 = new wxRadioButton(WxPanel1, ID_WXRADIOBUTTON8,
wxT("original"), wxPoint(676, 653), wxSize(113, 17), 0, wxDefaultValidator,
wxT("WxRadioButton8"));
	WxRadioButton8->SetFont(wxFont(8, wxSWISS, wxNORMAL, wxNORMAL, false,
wxT("MS Sans Serif")));

	WxMessageDialog1 = new wxMessageDialog(this, wxT(""), wxT("Message
box"));

	WxSaveFileDialog1 = new wxFileDialog(this, wxT("Choose a file"),
wxT(""), wxT(""), wxT("*.*"), wxSAVE);

	WxOpenFileDialog1 = new wxFileDialog(this, wxT("Choose a file"),
wxT(""), wxT(""), wxT("*.*"), wxOPEN);

	SetTitle(wxT("3D Histogram"));
	SetIcon(wxNullIcon);
	SetSize(8,8,1024,768);
	Center();

    ////GUI Items Creation End

}

void Project1Frm::Project1FrmClose(wxCloseEvent& event)
{
    // --> Don't use Close with a Frame,
    // use Destroy instead.
    Destroy();
}



/*
 * WxPanel1UpdateUI
 */
void Project1Frm::WxPanel1UpdateUI(wxUpdateUIEvent& event)
{
	// insert your code here
}
```

```
/*
 * WxPanel2UpdateUI1
 */
void Project1Frm::WxPanel2UpdateUI1(wxUpdateUIEvent& event)
{
	// insert your code here
}


/*
 * WxSlider1Scroll
 */
void Project1Frm::WxSlider1Scroll(wxScrollEvent& event)
{
  pGLCanvas->x = WxSlider1->GetValue();
  pGLCanvas->Draw();
}

/*
 * WxSlider2Scroll
 */
void Project1Frm::WxSlider2Scroll(wxScrollEvent& event)
{
	// insert your code here
    pGLCanvas->y = WxSlider2->GetValue();
    pGLCanvas->Draw();
}

/*
 * WxSlider3Scroll
 */
void Project1Frm::WxSlider3Scroll(wxScrollEvent& event)
{
	// insert your code here
	pGLCanvas->z = WxSlider3->GetValue();
    pGLCanvas->Draw();
}

/*
 * WxSlider4Scroll
 */
void Project1Frm::WxSlider4Scroll(wxScrollEvent& event)
{
	// insert your code here
	pGLCanvas->Grid = WxSlider4->GetValue();
	pGLCanvas->Draw();
}

/*
 * WxButton1Click
 */

void Project1Frm::WxButton1Click(wxCommandEvent& event)
{
	 wxString wildcards, loadimage1;
	 wxBitmap image1;
	 wxStaticBitmap *sb1;
	 wxImage temp1;
	    wxFileDialog* WxOpenFileDialog1 = new wxFileDialog(this,wxT("Choose
image	 file"),	wxEmptyString,wxEmptyString,	wildcards,wxFD_OPEN|
wxFD_FILE_MUST_EXIST|wxFD_CHANGE_DIR|wxCENTRE);
	 if (WxOpenFileDialog1->ShowModal()==wxID_OK)
	{
```

```
            loadimage1 = WxOpenFileDialog1->GetFilename();
                        strcpy(pGLCanvas->filename,  wxString(WxOpenFileDialog1-
>GetFilename()).char_str());
            pGLCanvas->InitGL();
            pGLCanvas->Draw();
            wxImage imagetest(loadimage1,wxBITMAP_TYPE_ANY,-1);
            wxBitmap temp_img1(imagetest, -1);
            image1 = temp_img1;
            temp1 = image1.ConvertToImage();
            image1 = wxBitmap(temp1.Rescale(365,255),-1);
            sb1 = new wxStaticBitmap(WxScrolledWindow1,-1,image1);
            int image1Width = image1.GetWidth();
            int image1Height = image1.GetHeight();
            WxScrolledWindow1->SetScrollbars(0,0,image1Width,image1Height);
            WxScrolledWindow1->Scroll(1,1);
            sb1->Show(true);
            WxSlider5->SetValue(100);
        }
        else WxOpenFileDialog1 -> Close();
}

/*
 * WxSlider5Scroll
 */
void Project1Frm::WxSlider5Scroll(wxScrollEvent& event)
{
    // insert your code here
    pGLCanvas->Dist_ceil = WxSlider5->GetValue();
    pGLCanvas->Draw();
}

/*
 * WxEdit2Updated
 */
void Project1Frm::WxEdit2Updated(wxCommandEvent& event)
{
       // insert your code here
}

/*
 * WxButton2Click
 */
void Project1Frm::WxButton2Click(wxCommandEvent& event)
{
    // insert your code here
       char test[16];
       //int bits = 256/pGLCanvas->factor;
       int ***report;
       if(pGLCanvas->factor<2) report = pGLCanvas->GetPixel();
       else report = pGLCanvas->GetAgregate();
    //float max = pGLCanvas->GetMaxval();
    float dist_ceil = WxSlider5->GetValue();
    WxStyledTextCtrl1->ClearAll();
    WxStyledTextCtrl1->AppendText(wxT("Statistics\nR\tG\tB\t\n"));
    for(int m=0; m<256/pGLCanvas->factor; m++){
          for(int n=0; n<256/pGLCanvas->factor; n++){
               for(int o=0; o<256/pGLCanvas->factor; o++){
                    if(report[m][n][o]>0)
                    {
                       sprintf(test,"%d\t%d\t%d\t%d\n",m,n,o,report[m][n][o]);
                       WxStyledTextCtrl1->AppendText(wxT(test));
                    }
               }
```

```
          }
     }
}

/*
 * WxSlider6Scroll
 */
void Project1Frm::WxSlider6Scroll(wxScrollEvent& event)
{
       // insert your code here
       //pGLCanvas->Dist_floor = WxSlider6->GetValue();
    //pGLCanvas->Draw();
}

/*
 * AutoCompSelection
 */
void Project1Frm::AutoCompSelection(wxStyledTextEvent& event)
{
       // insert your code here
}

/*
 * WxSlider5Slider
 */
void Project1Frm::WxSlider5Slider(wxCommandEvent& event)
{
       // insert your code here
       WxRichTextCtrl1->Clear();
    //(*WxRichTextCtrl1) << pGLCanvas->pixelcount;
    (*WxRichTextCtrl1) << pGLCanvas -> Dist_ceil/100.0;

}

/*
 * WxRadioButton1Click
 */
void Project1Frm::WxRadioButton1Click(wxCommandEvent& event)
{
       // insert your code here
       pGLCanvas->factor = 2;
       pGLCanvas->Compress();
       pGLCanvas->Draw();
}

/*
 * WxRadioButton2Click
 */
void Project1Frm::WxRadioButton2Click(wxCommandEvent& event)
{
       // insert your code here
       pGLCanvas->factor = 4;
       pGLCanvas->Compress();
       pGLCanvas->Draw();
}

/*
 * WxRadioButton3Click
 */
void Project1Frm::WxRadioButton3Click(wxCommandEvent& event)
{
       // insert your code here
       pGLCanvas->factor = 8;
```

```
	pGLCanvas->Compress();
	pGLCanvas->Draw();
}

/*
 * WxRadioButton4Click
 */
void Project1Frm::WxRadioButton4Click(wxCommandEvent& event)
{
	// insert your code here
	pGLCanvas->factor = 16;
	pGLCanvas->Compress();
	pGLCanvas->Draw();
}

/*
 * WxRadioButton5Click
 */
void Project1Frm::WxRadioButton5Click(wxCommandEvent& event)
{
	// insert your code here
	pGLCanvas->factor = 32;
	pGLCanvas->Compress();
	pGLCanvas->Draw();
}

/*
 * WxRadioButton6Click
 */
void Project1Frm::WxRadioButton6Click(wxCommandEvent& event)
{
	// insert your code here
	pGLCanvas->factor = 64;
	pGLCanvas->Compress();
	pGLCanvas->Draw();
}

/*
 * WxRadioButton7Click
 */
void Project1Frm::WxRadioButton7Click(wxCommandEvent& event)
{
	// insert your code here
	pGLCanvas->factor = 128;
    pGLCanvas->Compress();
	pGLCanvas->Draw();
}

/*
 * WxRadioButton8Click
 */
void Project1Frm::WxRadioButton8Click(wxCommandEvent& event)
{
	// insert your code here
	pGLCanvas->factor = 1;
	//pGLCanvas->Compress();
	pGLCanvas->Draw();
}
```

drawingaxes.h

```
/*************************************************
* drawingAxes Header:                            *
* Declaration of axis drawing function for histogram *
*************************************************/

#ifndef DRAWINGAXES
#define DRAWINGAXES

#include <GL/glew.h>
int grid=1;
void drawingAxes (void){
    int lines=256;
    switch(grid){
      case 1:lines=256; break;
      case 2:lines=128; break;
      case 3:lines=64; break;
      case 4:lines=32; break;
      case 5:lines=16; break;
      case 6:lines=8; break;
      case 7:lines=4; break;
      case 8:lines=2; break;
      case 9:lines=1; break;
      }
     /******draw red-axis********/
     glBegin(GL_LINES);
           glColor3f(1.0f, 0.0f, 0.0f);
           for(int y = 0; y<257; y++){
                if(y%lines==0){
                     for(int x = 0; x<257; x++){
                          if(x%lines==0){
                               glVertex3f(0.0f, float(x)/10.0f, float(y)/10.0f);
                                    glVertex3f(25.6f,  float(x)/10.0f,
float(y)/10.0f);
                          }
                     }
                }
           }
         /******draw blue-axis********/
           glColor3f(0.0f, 0.0f, 1.0f);
           for(int v = 0; v<257; v++){
                if(v%lines==0){
                     for(int z = 0; z<257; z++){
                          if(z%lines==0){
                               glVertex3f(float(v)/10.0f, float(z)/10.0f, 0.0f);
                                    glVertex3f(float(v)/10.0f,  float(z)/10.0f,
25.6f);
                          }
                     }
                }
           }
       /******draw green-axis********/
           glColor3f(0.0f, 1.0f, 0.0f);
           for(int w = 0; w<257; w++){
                if(w%lines==0){
                     for(int u = 0; u<257; u++){
                          if(u%lines==0){
                               glVertex3f(float(w)/10.0f, 0.0f, float(u)/10.0f);
                                    glVertex3f(float(w)/10.0f,  25.6f,
float(u)/10.0f);
                          }
                     }
```

```
                }
            }
     glEnd();
}
#endif
```

drawingcubes.h

```
/*****************************************************
* drawing cubes Header:                                *
* Declaration of cube drawing function for histogram  *
*****************************************************/
#ifndef DRAWINGCUBES
#define DRAWINGCUBES

#include <GL/glew.h>

void drawingCubes (float red, float green, float blue){
     glBegin(GL_QUADS);

         glColor3f(red/25.5,green/25.5,blue/25.5);
        //Front Face
           glVertex3f(red, green,  blue+0.1f);
           glVertex3f(red, green+0.1f,  blue+0.1f);
           glVertex3f(red+0.1f, green+0.1f ,  blue+0.1f );
           glVertex3f(red+0.1f, green,  blue+0.1f );
             // Back Face
           glVertex3f(red, green,  blue);
           glVertex3f(red, green+0.1f ,  blue);
           glVertex3f(red+0.1f , green+0.1f ,  blue);
           glVertex3f(red+0.1f , green,  blue);
             // Top Face
           glVertex3f(red, green+0.1f ,  blue);
           glVertex3f(red, green+0.1f ,  blue+0.1f );
           glVertex3f(red+0.1f , green+0.1f ,  blue);
           glVertex3f(red+0.1f , green+0.1f ,  blue+0.1f );
             // Bottom Face
           glVertex3f(red, green,  blue);
           glVertex3f(red, green,  blue+0.1f );
           glVertex3f(red+0.1f , green,  blue);;
           glVertex3f(red+0.1f , green,  blue+0.1f );
             // Right face
           glVertex3f(red+0.1f , green+0.1f ,  blue+0.1f );
           glVertex3f(red+0.1f , green+0.1f ,  blue);
           glVertex3f(red+0.1f , green,  blue+0.1f );
           glVertex3f(red+0.1f , green,  blue);
             // Left Face
           glVertex3f(red, green+0.1f ,  blue+0.1f );
           glVertex3f(red, green+0.1f ,  blue);
           glVertex3f(red, green,  blue+0.1f );
           glVertex3f(red, green,  blue);
      glEnd();

}

void drawingCubesT (float red, float green, float blue, int factor, float size)
{
  //glLoadIdentity();
  glBegin(GL_QUADS);
  float bitrate = (25.6/factor)-0.1;
  glColor3f(red/bitrate,green/bitrate,blue/bitrate);
  float R=red*factor;
  float G=green*factor;
```

```
  float B=blue*factor;
  float length = 0.1f*factor*size;
  //Front Face
  glVertex3f(R, G,  B+(length));
  glVertex3f(R, G+(length),  B+(length));
  glVertex3f(R+(length), G+(length),  B+(length));
  glVertex3f(R+(length), G,  B+(length));
  // Back Face
  glVertex3f(R, G,  B);
  glVertex3f(R, G+(length),  B);
  glVertex3f(R+(length), G+(length),  B);
  glVertex3f(R+(length), G,  B);
  // Top Face
  glVertex3f(R, G+(length),  B);
  glVertex3f(R, G+(length),  B+(length));
  glVertex3f(R+(length), G+(length),  B);
  glVertex3f(R+(length), G+(length),  B+(length));
  // Bottom Face
  glVertex3f(R, G,  B);
  glVertex3f(R, G,  B+(length));
  glVertex3f(R+(length), G,  B);

  glVertex3f(R+(length), G,  B+(length));
  // Right face
  glVertex3f(R+(length), G+(length),  B+(length));
  glVertex3f(R+(length), G+(length),  B);
  glVertex3f(R+(length), G,  B+(length));
  glVertex3f(R+(length), G,  B);
  // Left Face
  glVertex3f(R, G+(length),  B+(length));
  glVertex3f(R, G+(length),  B);
  glVertex3f(R, G,  B+(length));
  glVertex3f(R, G,  B);

  glEnd();
} #endif
```

MyGLCanvas.h

```
/*
 *     MyGLCanvas.h
 *     OpenGL Demo
 *
 *     Created by Ross Ashman on Jan 1 2005.
 *     Copyright (c) 2005 Ross Ashman. All rights reserved.
 *
 */

// MyGLCanvas.h: interface for the wxMyGLCanvas class.
//
//////////////////////////////////////////////////////////////////////

#ifndef _MYGLCANVAS_H__INCLUDED_
```

```
#define _MYGLCANVAS_H__INCLUDED_
#include "cv.h"
#include "highgui.h"
#include "cxtypes.h"
#include <wx/glcanvas.h>
#include <wx/gdicmn.h>
#include <GL/glu.h>
class MyGLCanvas : public wxGLCanvas
{
public:
       MyGLCanvas(wxWindow* parent,
                   wxWindowID id = -1,
                   const wxPoint& pos = wxDefaultPosition,
                   const wxSize& size = wxDefaultSize,
                   long style=0,
                   const wxString& name="GLCanvas",
                   int* attribList = 0,
                   const wxPalette& palette = wxNullPalette);
       virtual ~MyGLCanvas();
       // Methods
    float x, y ,z;

    char filename[256], status[16];
    int Grid, Dist_ceil;
    int pixelcount,factor;
       void InitGL();
    void Draw();
    void Compress();
    void Paint(wxDC &dc);
       // Event Handlers (These Functions Should NOT be Virtual)

       void OnScroll(wxScrollEvent& event);
    void OnPaint(wxPaintEvent& event);
       void OnLeftMouse(wxMouseEvent& event);
       void OnLeftMouseUp(wxMouseEvent& event);
       void OnRightMouse(wxMouseEvent& event);
       void OnRightMouseUp(wxMouseEvent& event);
       void OnLeftDMouse(wxMouseEvent& event);
       void OnMouseMove(wxMouseEvent& event);
       void OnSize(wxSizeEvent& event);
       void OnEraseBackground(wxEraseEvent& event);
       void OnIdle(wxIdleEvent& event);

    int ***GetPixel(){return pixel;};
    int ***GetAgregate(){return agregate;};
    float GetMaxval(){return maxval;};

protected:

private:
       //DECLARE_CLASS(wxGLViewCanvas)
       //any class wishing to process wxWindows events must use this macro
       DECLARE_EVENT_TABLE()
      float RotX, RotY,PanX,PanY, Depth,maxval;
      int*** pixel;
      int*** agregate;
      IplImage* img;
      long Xcoord, Ycoord, Zcoord, Dummy, X, Y;
      int MouseButton;
      struct MyColor {
        float R;
        float G;
        float B;
```

```
        } OColor, BColor;
};


#endif // _MYGLCANVAS_H__INCLUDED_
```

MyGLCanvas.cpp

```
/*
 *     MyGLCanvas.cpp
 *     OpenGL Animation Demo
 *
 *     Ross Ashman 4 Jan 2005.
 *     Copyright (c) 2005 Ross Ashman. All rights reserved.
 *
 */

// MyGLCanvas.cpp: implementation of the MyGLCanvas class.
//
//////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////
// Headers
//////////////////////////////////////////////////////////////////////
#include "drawingaxes.h"
#include "drawingcubes.h"
#include <wx/slider.h>
#include "wx/wxprec.h"
#ifndef WX_PRECOMP
       #include "wx/wx.h"
#endif
#include <GL/glu.h>
#include <math.h>
#include <vector>
#include <list>
#include <string>

#include "MyGLCanvas.h"
using namespace std;

//////////////////////////////////////////////////////////////////////
// Constants
//////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////
// Event Tables and Other Macros for wxWindows
//////////////////////////////////////////////////////////////////////

BEGIN_EVENT_TABLE(MyGLCanvas, wxGLCanvas)
       EVT_SIZE(MyGLCanvas::OnSize)
       EVT_PAINT(MyGLCanvas::OnPaint)
       EVT_LEFT_DOWN(MyGLCanvas::OnLeftMouse)
    EVT_LEFT_UP(MyGLCanvas::OnLeftMouseUp)
       EVT_RIGHT_DOWN(MyGLCanvas::OnRightMouse)
    EVT_RIGHT_UP(MyGLCanvas::OnRightMouseUp)
    EVT_LEFT_DCLICK(MyGLCanvas::OnLeftDMouse)
       EVT_MOTION(MyGLCanvas::OnMouseMove)
       EVT_ERASE_BACKGROUND(MyGLCanvas::OnEraseBackground)
       EVT_IDLE(MyGLCanvas::OnIdle)
END_EVENT_TABLE()
```

```
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

MyGLCanvas::MyGLCanvas(wxWindow* parent,
                                wxWindowID id,
                                const wxPoint& pos,
                                const wxSize& size,
                                long style,
                                const wxString& name,
                                int* attribList,
                                const wxPalette& palette)
: wxGLCanvas(parent, id, pos, size, style, name)
{

    InitGL();
       // Set GL Viewport
       int w, h;
       GetClientSize(&w, &h);

       if(GetContext())
       {
              SetCurrent();
              glViewport(0, 0, (GLint)w, (GLint)h);
              glMatrixMode(GL_PROJECTION);
              glLoadIdentity();
              gluPerspective(45.0, (float)w/(float)h, 1.0, 1000.0);
              glMatrixMode(GL_MODELVIEW);
              glLoadIdentity();
       }
}

MyGLCanvas::~MyGLCanvas()
{
}

//////////////////////////////////////////////////////////////////////
// Method Implementations
//////////////////////////////////////////////////////////////////////


void MyGLCanvas::InitGL()
{
    agregate = new int **[256];
      pixel=new int** [256];
    glShadeModel(GL_SMOOTH);                 // Enables Smooth Color Shading
    glClearDepth(10.0);                       // Depth Buffer Setup
    glEnable(GL_DEPTH_TEST);                 // Enable Depth Buffer
    glDepthFunc(GL_LESS);                    // The Type Of Depth Test To Do
        glHint(GL_PERSPECTIVE_CORRECTION_HINT,  GL_NICEST);      //Realy  Nice
perspective calculations
    factor = 1;
    RotX=0;
    RotY=15;
    PanX= 0;
    PanY= -30;
    Depth = -100;
    Dist_ceil = 1;
    for (int i=0; i<256; i++){
        pixel[i]=new int* [256];
        for (int j=0; j<256; j++){
            pixel[i][j]=new int [256];
            for (int k=0; k<256; k++){
```

```
                pixel[i][j][k]=0;
            }
        }
    }
    FILE *fp;
    fp = fopen(filename,"r");
    if(fp!=NULL)
    {
        list<string> pixels;
        list<string>::iterator it;
        strcpy(status,"success");
        fclose(fp);
        img = cvLoadImage(filename, 1); /*loads image as matrix*/

        uchar*    ptr;
        for(int y=0; y<img->height; y++){
              for(int x=0; x<img->width; x++){
                      ptr = cvPtr2D(img, y, x, NULL);
                      pixel[ptr[2]][ptr[1]][ptr[0]] += 1;
              }
        }
        //counts total number of pixels in image
        for(int red=0; red<256; red++){
          for(int green=0; green<256; green++){
               for(int blue=0; blue<256; blue++){
                           if(pixel[red][green][blue]>maxval) maxval=pixel[red]
[green][blue];
               }
            }
        }
    }
    else strcpy(status,"fail");

}

void MyGLCanvas::Compress()
{
    int bits = 256/factor;
    maxval = 0;
    for (int i=0; i<256/factor; i++){
        agregate[i]=new int* [128];
        for (int j=0; j<256/factor; j++){
            agregate[i][j]=new int [128];
            for (int k=0; k<256/factor; k++){
                agregate[i][j][k]=0;
            }
        }
    }
    for(int red=0; red<256; red++){
          for(int green=0; green<256; green++){
               for(int blue=0; blue<256; blue++){
                           if(pixel[red][green][blue]>maxval) maxval=pixel[red]
[green][blue];
                             agregate[red/factor][green/factor][blue/factor] +=
pixel[red][green][blue];
               }
            }
        }
     for(int red=0; red<bits; red++){
          for(int green=0; green<bits; green++){
               for(int blue=0; blue<bits; blue++){
                                          if(agregate[red][green][blue]>maxval)
maxval=agregate[red][green][blue];
```

```
            }
        }
    }


}
void MyGLCanvas::Draw()
{
     //this->Refresh();  //Refresh() seems to chew up more resources??! Just do
a Paint instead!
    wxClientDC dc(this);
    Paint(dc);
}
void MyGLCanvas::Paint(wxDC &dc)
{
    int bits = 256/factor;
    pixelcount = 0;
    if(!GetContext())
              return;

       SetCurrent();
       grid = Grid;
       //glEnable(GL_BLEND);
    //glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);

    glClearColor(x/255, y/255, z/255, 1.0);
       //glDisable(GL_BLEND);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);


    glTranslated(PanY,PanX, Depth);

    glRotatef(RotX, 1.0, 0.0, 0.0);
    glRotatef(RotY, 0.0, 1.0,0.0);

    drawingAxes();

    if (factor<2)
    {
         for(int red=0; red<256; red++){
             for(int green=0; green<256; green++){
                   for(int blue=0; blue<256; blue++){
                       if(pixel[red][green][blue]!=0){
                            float pixels=pixel[red][green][blue];
                            if((pixels/maxval)<=(Dist_ceil/100.0))
                            {
                                                drawingCubes(float (red)/10, float
(green)/10, float (blue)/10);//draws cube for corresponding RGB value
                                       pixelcount++;
                            }
                       }
                  }
             }
        }
    }
    else
    {
        for(int red=0;red<bits;red++)
        {
            for(int green=0; green<bits; green++){
                   for(int blue=0; blue<bits; blue++){
```

```
                    if(agregate[red][green][blue]>0){
                        float current = agregate[red][green][blue];
                                if(current/maxval < 0.1) drawingCubesT(float
(red)/10, float (green)/10, float (blue)/10,factor, float (bits)/10, 0.1);
                                else if (current/maxval >= 0.1 && current/maxval <
0.2) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.2);
                                else if (current/maxval >= 0.2 && current/maxval <
0.3) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.3);
                                else if (current/maxval >= 0.3 && current/maxval <
0.4) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.4);
                                else if (current/maxval >= 0.4 && current/maxval <
0.5) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.5);
                                else if (current/maxval >= 0.5 && current/maxval <
0.6) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.6);
                                else if (current/maxval >= 0.6 && current/maxval <
0.7) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.7);
                                else if (current/maxval >= 0.7 && current/maxval <
0.8) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.8);
                                else if (current/maxval >= 0.8 && current/maxval <
0.9) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 0.9);
                                 else if (current/maxval >= 0.9 && current/maxval <=
1.0) drawingCubesT(float (red)/10, float (green)/10, float (blue)/10,factor,
float (bits)/10, 1);
                            }
                        }
                    }
                }
            }

            // Flush the GL Pipeline
               glFlush();

               // Swap Your Buffers
               SwapBuffers();
}


//////////////////////////////////////////////////////////////////////
// Event Handler Implementations
//////////////////////////////////////////////////////////////////////

void MyGLCanvas::OnPaint(wxPaintEvent& event)
{
       wxPaintDC dc(this);
       Paint(dc);
       event.Skip();
}

void MyGLCanvas::OnSize(wxSizeEvent& event)
{
       // This is necessary to update the context on some platforms
       wxGLCanvas::OnSize(event);

       // Set the GL Viewport (not called by wxGLCanvas::OnSize on all
Platforms)
```

```
        int w, h;
        GetClientSize(&w, &h);
        if(GetContext())
        {
                SetCurrent();
                glViewport(0, 0, (GLint)w, (GLint)h);
                glMatrixMode(GL_PROJECTION);
                glLoadIdentity();
                gluPerspective(45.0, (float)w/(float)h, 1.0, 1000.0);
                glMatrixMode(GL_MODELVIEW);
                glLoadIdentity();
        }
}

void MyGLCanvas::OnLeftDMouse(wxMouseEvent& event)
{
         MouseButton = 1;
         event.GetPosition(&Xcoord, &Ycoord);
}

void MyGLCanvas::OnRightMouse(wxMouseEvent& event)
{
         MouseButton = 2;
         event.GetPosition(&Dummy, &Zcoord);
}

void MyGLCanvas::OnLeftMouse(wxMouseEvent& event)
{
         MouseButton = 3;
         event.GetPosition(&Xcoord, &Ycoord);
}

void MyGLCanvas::OnLeftMouseUp(wxMouseEvent& event)
{
         MouseButton = 0;
}


void MyGLCanvas::OnRightMouseUp(wxMouseEvent& event)
{
         MouseButton = 0;
}



void MyGLCanvas::OnMouseMove(wxMouseEvent& event)
{

    if (event.Dragging() == true)
    {
         event.GetPosition(&X, &Y);

         if (MouseButton == 1)
         {
                  RotX=RotX + (GLfloat)(Y - Ycoord)/2;
                  RotY=RotY + (GLfloat)(X - Xcoord)/2;
                  Xcoord = X;
                  Ycoord = Y;
         }
         if (MouseButton == 2)
         {
                Depth = Depth - (GLfloat)(Y-Zcoord)/5;
                Zcoord = Y;
```

```
        }
        if (MouseButton == 3)
        {
                PanX = PanX - (GLfloat)(Y - Ycoord)/10;
                PanY = PanY - (GLfloat)(-X + Xcoord)/10;
                Xcoord = X;
                Ycoord = Y;
        }

        Draw(); // Causes an event to be generated to repaint the window.

    }
}

void MyGLCanvas::OnEraseBackground(wxEraseEvent& event)
{
       // Do Nothing to Avoid Flashing on MSW
}

void MyGLCanvas::OnIdle(wxIdleEvent& event)
{
       //this->Refresh();   // invoking this chews up lots of cpu cycles, not
needed for static pictures, onpaint is sufficient
}
```

Project1App.cpp

```
//------------------------------------------------------------------------------
//
// Name:        Project1App.cpp
// Author:      Ross
// Created:     15/04/2005 2:23:03 PM
//
//------------------------------------------------------------------------------

#include "Project1App.h"
#include "Project1Frm.h"
#include "MyGLCanvas.h"

IMPLEMENT_APP(Project1FrmApp)

bool Project1FrmApp::OnInit()
{
    Project1Frm *myFrame = new  Project1Frm(NULL);
    SetTopWindow(myFrame);
    myFrame->Show(TRUE);
    return TRUE;
}

int Project1FrmApp::OnExit()
{
       return 0;
}
```

Project1Frm.h

```
//------------------------------------------------------------------------------
//
// Name:        Project1Frm.h
// Author:      Ross
// Created:     15/04/2005 2:23:03 PM
```

```
//
//-----------------------------------------------------------------------------
#ifndef __Project1Frm_HPP_
#define __Project1Frm_HPP_

// For compilers that support precompilation, includes "wx.h".
#include <wx/wxprec.h>

#ifdef __BORLANDC__
#pragma hdrstop
#endif

#ifndef WX_PRECOMP
// Include your minimal set of headers here, or wx.h
#include <wx/wx.h>

#endif

//Do not add custom headers.
//wx-dvcpp designer will remove them
////Header Include Start
#include <wx/filedlg.h>
#include <wx/msgdlg.h>
#include <wx/radiobut.h>
#include <wx/radiobox.h>
#include <wx/scrolwin.h>
#include <wx/richtext/richtextctrl.h>
#include <wx/stc/stc.h>
#include <wx/stattext.h>
#include <wx/textctrl.h>
#include <wx/button.h>
#include <wx/slider.h>
#include <wx/statbox.h>
#include <wx/animate.h>
#include <wx/panel.h>
////Header Include End

#include <wx/frame.h>
#include "MyGLCanvas.h"
////Dialog Style Start
#undef THIS_DIALOG_STYLE
#define  THIS_DIALOG_STYLE  wxCAPTION  |  wxSYSTEM_MENU  |  wxMINIMIZE_BOX  |
wxCLOSE_BOX
////Dialog Style End

class Project1Frm : public wxFrame
{
private:
    DECLARE_EVENT_TABLE()
public:
     Project1Frm( wxWindow *parent, wxWindowID id = 1, const wxString &title =
_T("Project1"),
        const wxPoint& pos = wxDefaultPosition,
        const wxSize& size = wxDefaultSize,
        long style = THIS_DIALOG_STYLE);
    virtual ~Project1Frm();
    MyGLCanvas* pGLCanvas;
public:
  //Do not add custom Control Declarations here.
  //wx-devcpp will remove them. Try adding the custom code
  //after the block.
  ////GUI Control Declaration Start
            wxFileDialog *WxOpenFileDialog1;
```

```
            wxFileDialog *WxSaveFileDialog1;
            wxMessageDialog *WxMessageDialog1;
            wxRadioButton *WxRadioButton8;
            wxRadioButton *WxRadioButton7;
            wxRadioButton *WxRadioButton6;
            wxRadioButton *WxRadioButton5;
            wxRadioButton *WxRadioButton4;
            wxRadioButton *WxRadioButton3;
            wxRadioButton *WxRadioButton2;
            wxRadioButton *WxRadioButton1;
            wxRadioBox *WxRadioBox1;
            wxStaticText *WxStaticText1;
            wxScrolledWindow *WxScrolledWindow1;
            wxRichTextCtrl *WxRichTextCtrl1;
            wxStyledTextCtrl *WxStyledTextCtrl1;
            wxButton *WxButton2;
            wxStaticText *Blue;
            wxStaticText *Green;
            wxStaticText *Red;
            wxTextCtrl *WxEdit1;
            wxButton *WxButton1;
            wxSlider *WxSlider5;
            wxSlider *WxSlider4;
            wxSlider *WxSlider3;
            wxSlider *WxSlider2;
            wxSlider *WxSlider1;
            wxStaticBox *Distribution;
            wxStaticBox *Grids;
            wxStaticBox *Bakground;
            wxPanel *WxPanel2;
            wxAnimationCtrl *wxAnimationCtrl1;
            wxPanel *WxPanel1;
  ////GUI Control Declaration End
public:
       //Note: if you receive any error with these enums, then you need to
    //change your old form code that are based on the #define control ids.
       //It may replace a numeric value in the enums names.
       enum {
////GUI Enum Control ID Start
                    ID_WXRADIOBUTTON8 = 1040,
                    ID_WXRADIOBUTTON7 = 1039,
                    ID_WXRADIOBUTTON6 = 1038,
                    ID_WXRADIOBUTTON5 = 1037,
                    ID_WXRADIOBUTTON4 = 1036,
                    ID_WXRADIOBUTTON3 = 1035,
                    ID_WXRADIOBUTTON2 = 1034,
                    ID_WXRADIOBUTTON1 = 1032,
                    ID_WXRADIOBOX1 = 1031,
                    ID_WXSTATICTEXT1 = 1030,
                    ID_WXSCROLLEDWINDOW1 = 1029,
                    ID_WXRICHTEXTCTRL1 = 1028,
                    ID_WXSTYLEDTEXTCTRL1 = 1026,
                    ID_WXBUTTON2 = 1020,
                    ID_BLUE = 1019,
                    ID_GREEN = 1018,
                    ID_RED = 1017,
                    ID_WXEDIT1 = 1008,
                    ID_WXBUTTON1 = 1007,
                    ID_WXSLIDER5 = 1009,
                    ID_WXSLIDER4 = 1006,
                    ID_WXSLIDER3 = 1005,
                    ID_WXSLIDER2 = 1004,
                    ID_WXSLIDER1 = 1003,
```

```
                    ID_DISTRIBUTION = 1013,
                    ID_GRIDS = 1012,
                    ID_BAKGROUND = 1010,
                    ID_WXPANEL2 = 1002,
                    ID_WXANIMATIONCTRL1 = 1022,
                    ID_WXPANEL1 = 1001,
////GUI Enum Control ID End
   ID_DUMMY_VALUE_ //Dont Delete this DummyValue
   }; //End of Enum
public:
    void Project1FrmClose(wxCloseEvent& event);
    void CreateGUIControls(void);
             void WxPanel2UpdateUI1(wxUpdateUIEvent& event);
             void WxPanel1UpdateUI(wxUpdateUIEvent& event);
             void WxSlider1Scroll(wxScrollEvent& event);
             void WxSlider2Scroll(wxScrollEvent& event);
             void WxSlider3Scroll(wxScrollEvent& event);
             void WxButton2Click0(wxCommandEvent& event);
             void WxSlider4Scroll(wxScrollEvent& event);
             void WxButton1Click(wxCommandEvent& event);
             void WxSlider5Scroll(wxScrollEvent& event);
             void WxEdit2Updated(wxCommandEvent& event);
             void WxButton2Click(wxCommandEvent& event);
             void WxStyledTextCtrl1AutoCompSelection(wxStyledTextEvent& event);
             void WxStyledTextCtrl1UpdateUI(wxUpdateUIEvent& event);
             void Project1FrmActivate(wxActivateEvent& event);
             void WxSlider6Scroll(wxScrollEvent& event);
             void AutoCompSelection(wxStyledTextEvent& event);
             void WxRichTextCtrl1BufferReset(wxRichTextEvent& event);
             void WxRichTextCtrl1UpdateUI(wxUpdateUIEvent& event);
             void WxSlider5Slider(wxCommandEvent& event);
             void WxRadioButton1Click(wxCommandEvent& event);
             void WxRadioButton2Click(wxCommandEvent& event);
             void WxRadioButton3Click(wxCommandEvent& event);
             void WxRadioButton4Click(wxCommandEvent& event);
             void WxRadioButton5Click(wxCommandEvent& event);
             void WxRadioButton6Click(wxCommandEvent& event);
             void WxRadioButton7Click(wxCommandEvent& event);
             void WxRadioButton8Click(wxCommandEvent& event);


};

#endif
```

Compiler options additional parameters

"../../../../../Program Files/Dev-Cpp/lib/libwxmsw28_stc.a"

"../../../../../Program Files/Dev-Cpp/lib/libwxmsw28_scintilla.a"

"../../../../../Program Files/Dev-Cpp/lib/libglu32.a"

-lcv, -lcxcore, -lcvaux. –lcvcam, -lhighgui, -lglu32, -lglut32, -mwindows, -lwxmsw28, -lwxmsw28_gl, -lwxtiff, -lwxjpeg, -lwxpng, -lwxzlib, -lwxregex, -lwxexpat, -lkernel32, -luser32, -lgdi32, -lcomdlg32, -lwinspool, -lwinmm, -lshell32, -lcomctl32, -lole32, -loleaut32, -luuid, -lrpcrt4, -ladvapi32, -lwsock32, -lodbc32, -lopengl32,

Bibliography

"HISTOGRAM", *Basic Tools for Process Improvement*. 27 January 2011.

<http://www.saferpak.com/histogram_articles/howto_histogram.pdf >

"Digitized image and its properties: Image digitization", *Digital Image Processing*. 2003. 27 January 2001.
<http://www.icaen.uiowa.edu/~dip/LECTURE/ImageProperties2.html >

Elliotte Rusty Harold. "What is an Image?", *Cafe au Lait*. 2006. 27 January 2001.
<http://www.cafeaulait.org/course/week9/23.html>

Dr. Ross Ashman, OpenGL with wxDev-C++**.**
<http://wxdsgn.sourceforge.net/?q=node/14>