Assignment I5

CS 5389 Graphical User Interface

Spring 2015

Submitted to: Dr. Dan E Tamir

Department of Computer Science

Texas State University

Submitted by: Vimmi Taneja

Functional Description

Features supported:

Visualization Methods

1. **Triangles** – Shows triangles with number equal to first sample in ssm file.
   Calculation:
   Triangles are drawn in 12 by 12 matrix according to screen coordinates.
   Two types of triangles displayed:
   - Green triangles with value = 1.
   - Red triangles with value = 100.
     How is count calculated: Sample value/100 = count of red triangles.
     Remainder of above division = count of green triangles.
     If value is < 12*12=144 then only green triangles are displayed otherwise both are displayed depending on calculation. Also legend is displayed.

2. **Bar chart** – Shows bar chart of all values in ssm file.

   Calculation:

   All values in ssm file are taken in array, viewport width is divided into equal sized regions. Number of regions equal to number of samples. Quads are drawn in the regions calculated. Height of quad = sample value – y axis area – title area. If sample value > viewport height – y axis area – title area then it is scaled down by following:

   We get maximum value from ssm file.

   Difference = Maximum value – viewport height

   Difference % is calculated and all values are scaled down by difference %.

   Then bars or quads are drawn and X axis, Y axis, tick marks.

   Tick marks - Y axis is divided equally into 5 portions between 0 and maximum value and tick marks are drawn.

   Values on Y axis – Maximum value is divided by 5 and we get 5 values equally spaced.

   X axis – Line is drawn from point of y axis till viewport width.

   Value on X axis – X axis is divided equally into portions = number of samples. Then value from 1 till number of samples is written below X axis.

3. **Graph** – A line graph is drawn corresponding to ssm file sample values.

   Same logic as bar chart except that instead of drawing quads we draw a line joining all sample values.

   Also points are drawn for sample values. Values are displayed on those points.

4. **Image** – Image is read from ppm file. User selects ppm file from file dialog. Message is displayed if file is empty or user selects this option and did not open PPM file.
   Parsing of ppm file: file is read line by line and image width, height, max value are recorded (stored in variables to be used later). Memory is allocated for image with size = 3 * width * height. It is multiplied by 3 since every pixel in image consists of 3 components - Red, Green, Blue, each component = 1 byte. Information in file is written as R, G, B, R, G, B and so on. We put all RGB values in array in reverse order. E.g. first read values from file are put at last position in array.
   Drawing image: in paintGL function, we use glpixeltransfer to change default pixel transfer modes to ratio of 255 and max value in ppm file. Then we change pixel store mode to unpack with bytes swapped since R, G, B bytes are stored in reverse order in memory. Then we draw image using glDrawPixels with origin of image as 0, 0 and width, height from ppm file.

   Save image: PPM Image can be saved from file menu.

Image size

Multiplying viewport size with size selected gives new size. Then we render whole image in graphics window with new size.

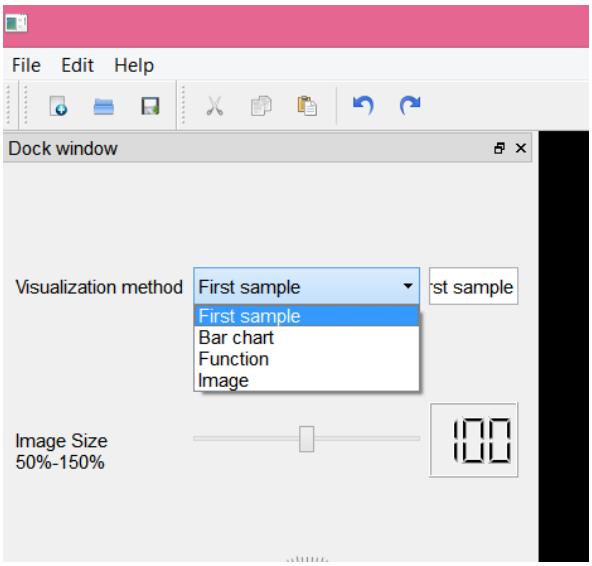
Image Brightness

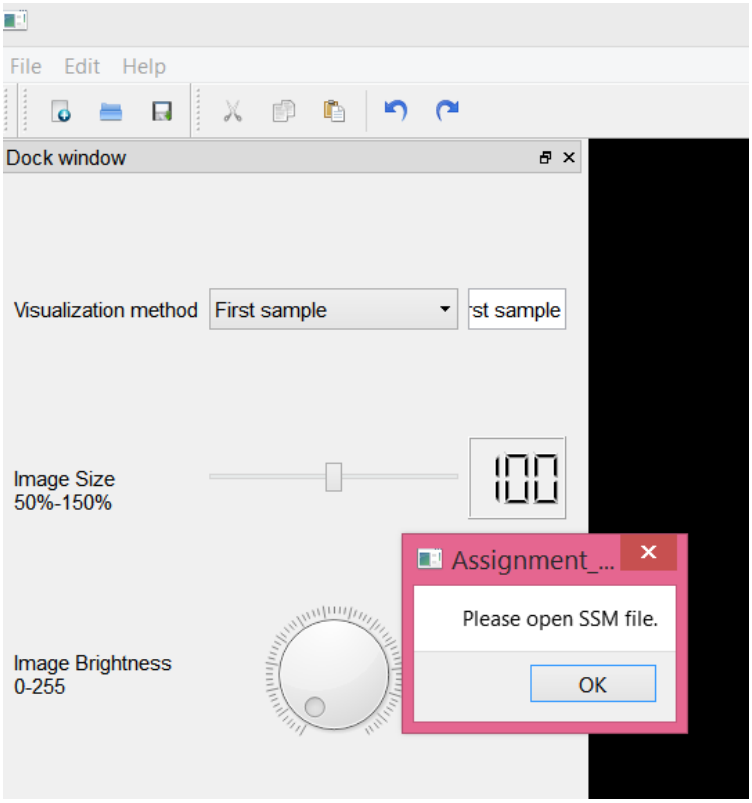
Pixels are first read from frame buffer to array in memory. Pixel transfer modes for R, G, B are modified to new value calculated as follows:

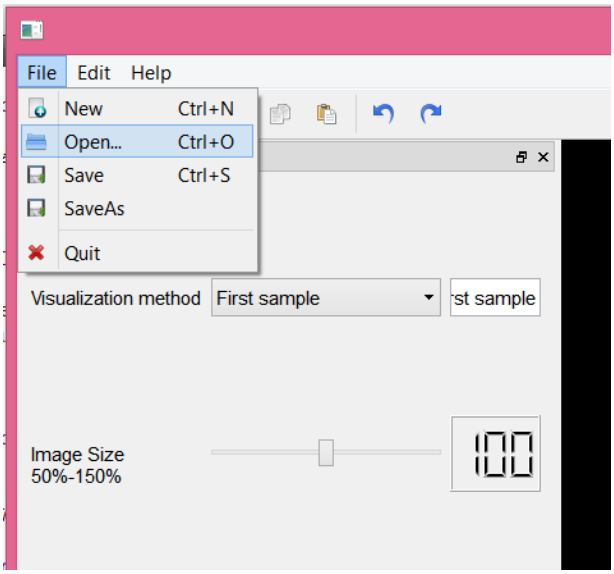
1 + newBrightnessValue/255

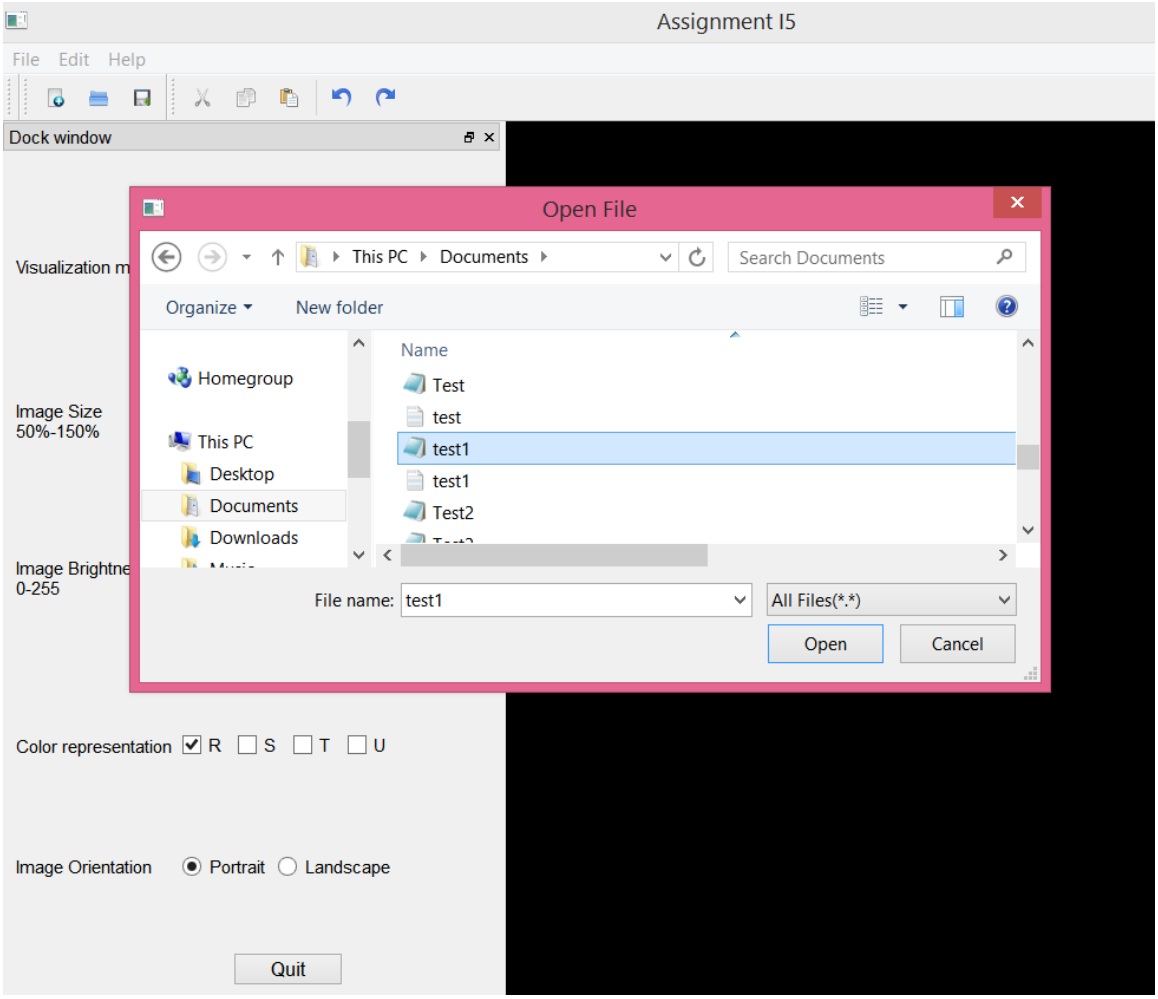
Then we change pixel store mode to unpack with bytes swapped since R, G, B bytes are stored in reverse order in memory. Then pixels are drawn using glDrawPixels in graphics window from that array.

Screenshots

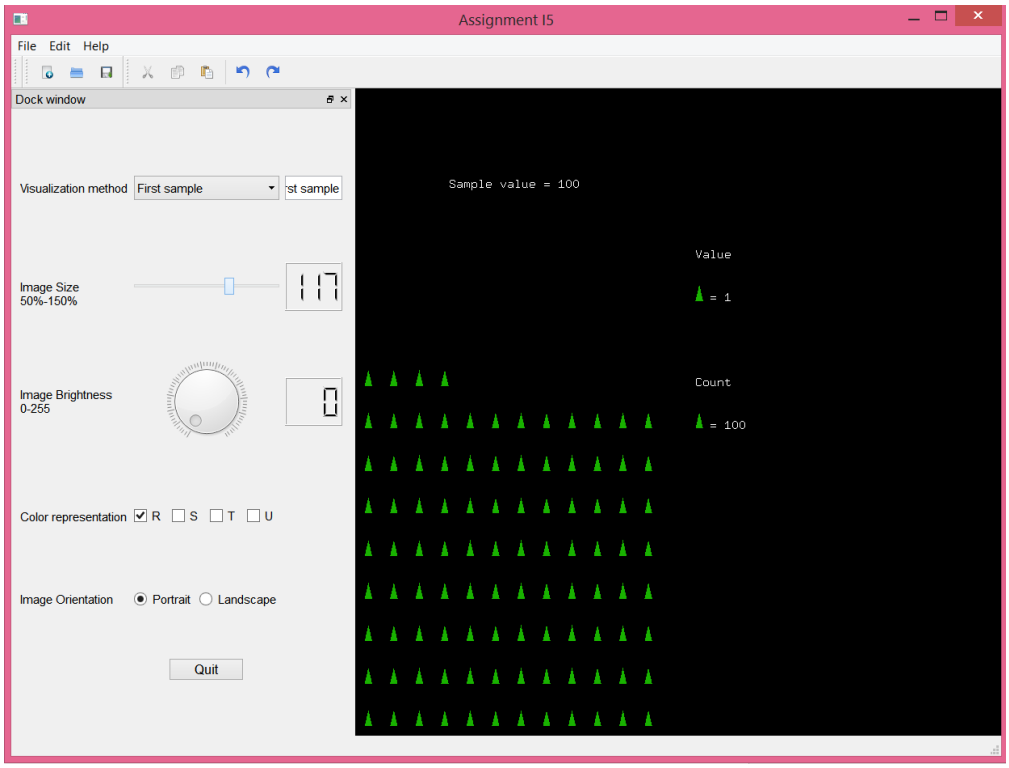




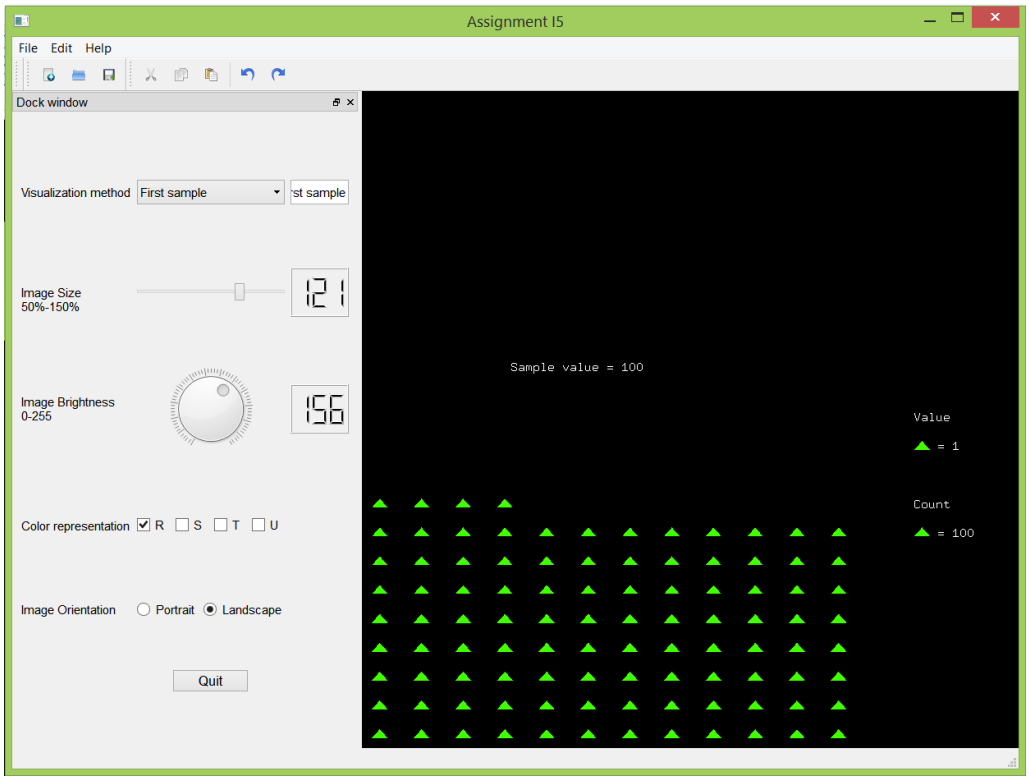

File Edit Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
SaveAs
Quit
Visualization method
First sample
Image Size
50%-150%
100


Assignment I5
File Edit Help
Dock window
Visualization m
Image Size
50%-150%
Image Brightne
0-255
Color representation R S T U
Image Orientation Portrait Landscape
Quit
Open File
This PC Documents
Search Documents
Organize
New folder
Homegroup
This PC
Desktop
Documents
Downloads
Name
Test
test
test1
test1
Test2
File name: test1
All Files(*.*)
Open
Cancel

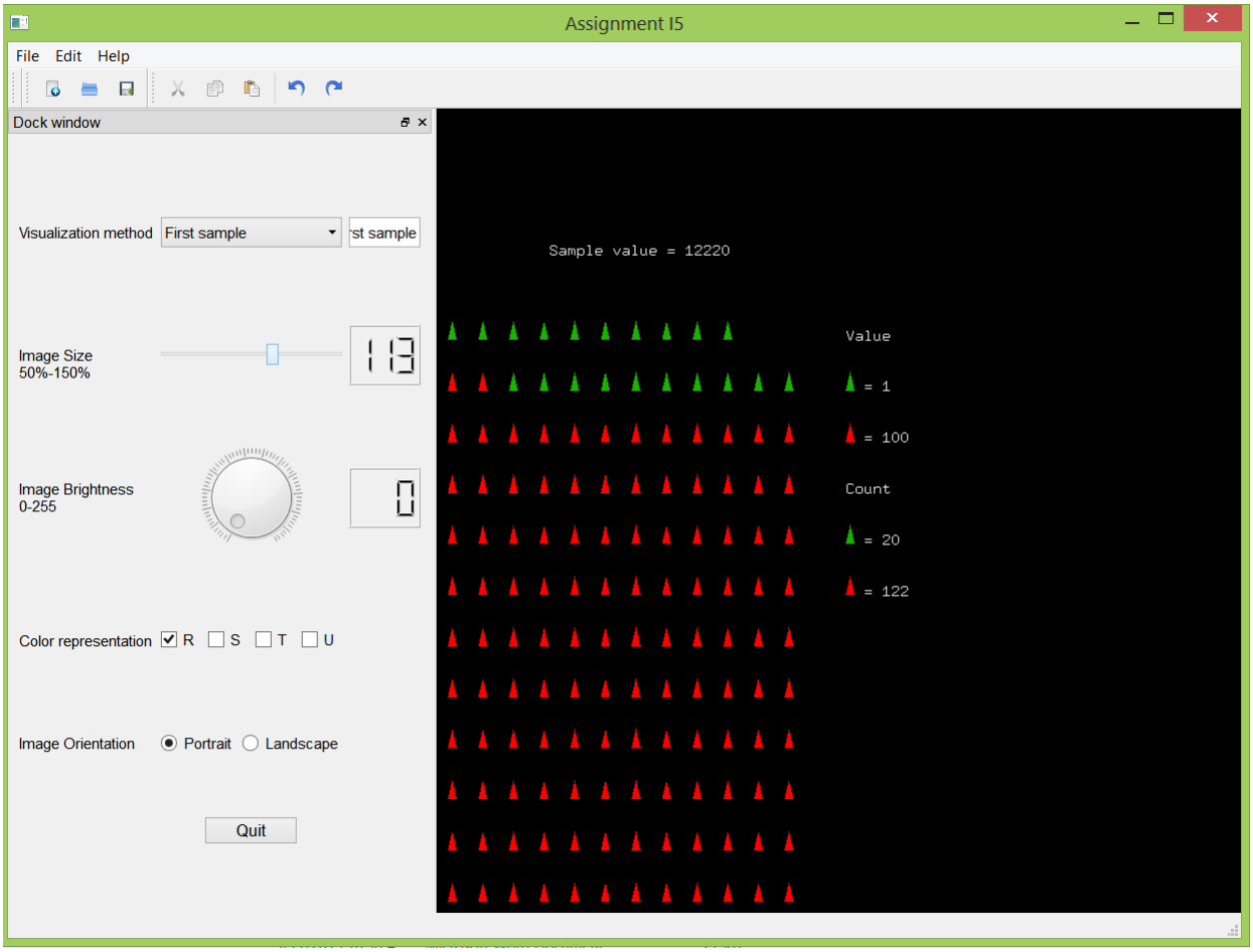
Visualization method – triangles – sample value = 100, Portrait orientation



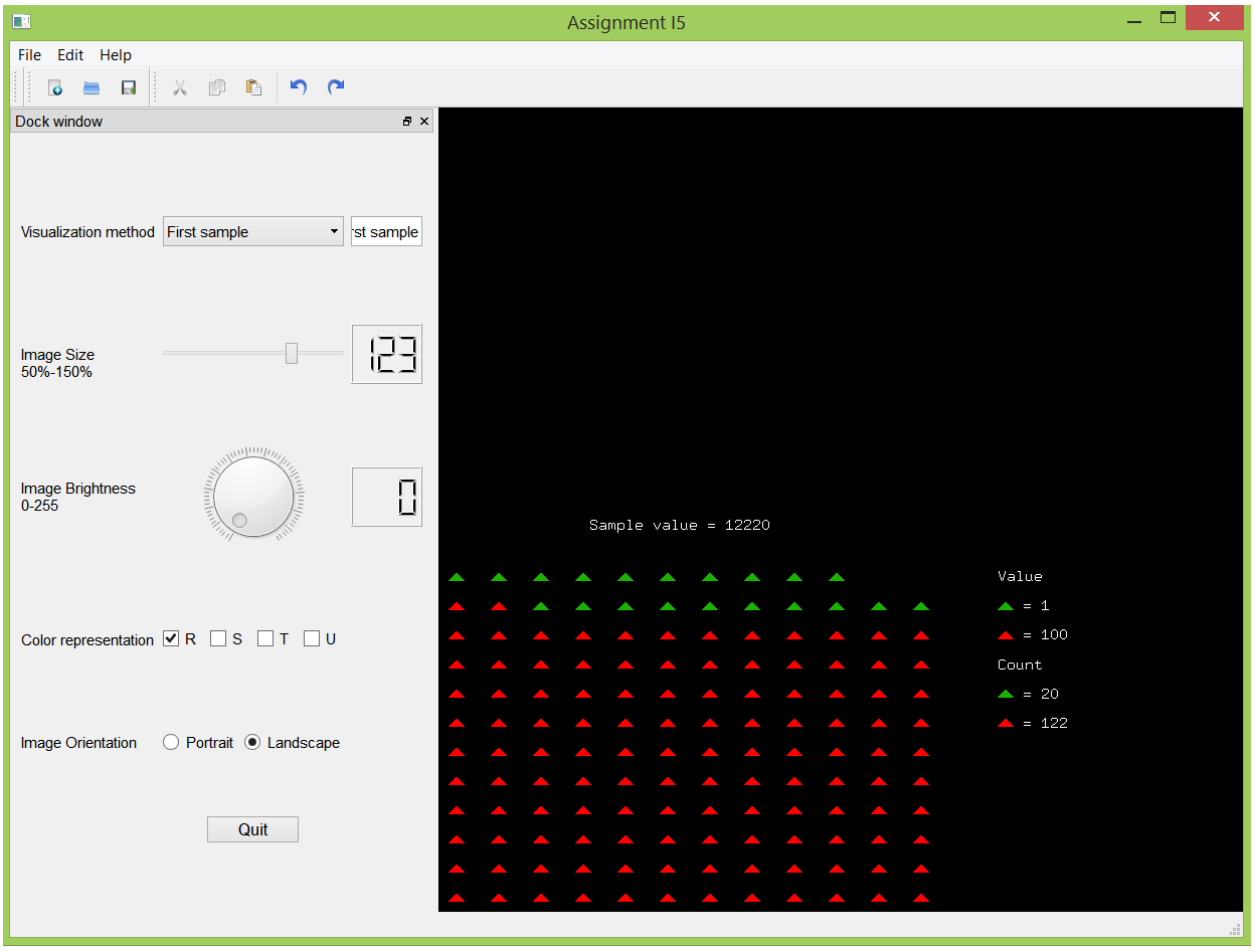
Change orientation to Landscape



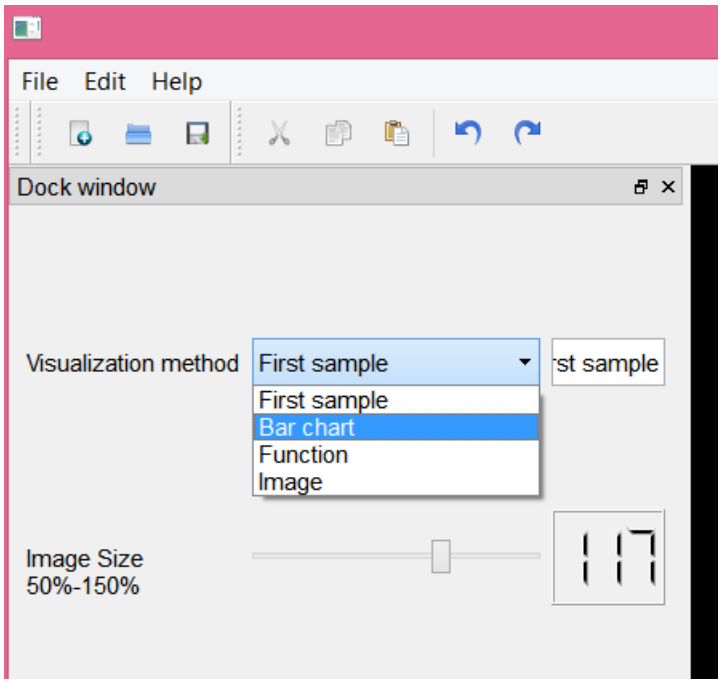
If the first sample in ssm file is bigger than 144, then red triangles are displayed with each triangle equal to 100 value.

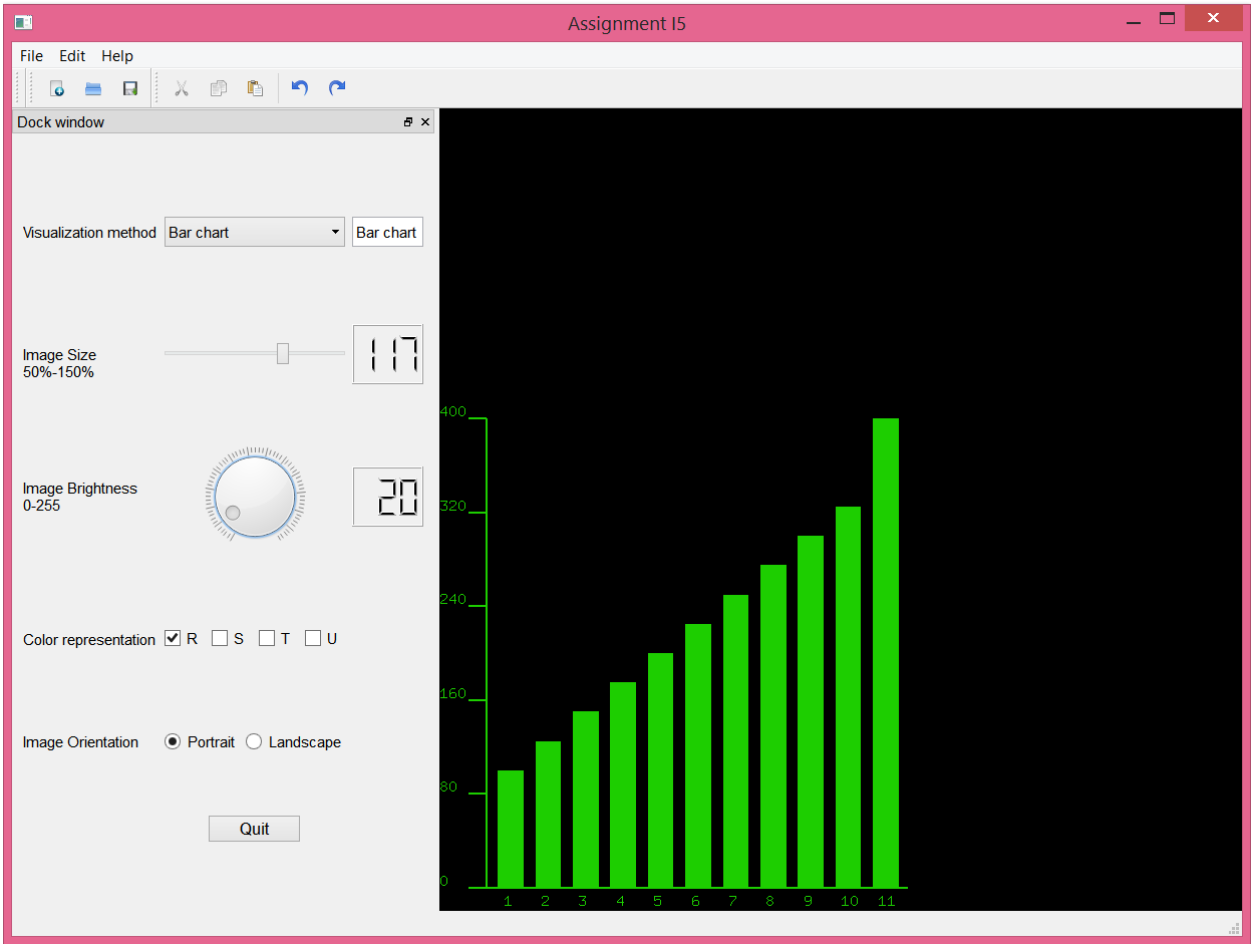


Change orientation to landscape

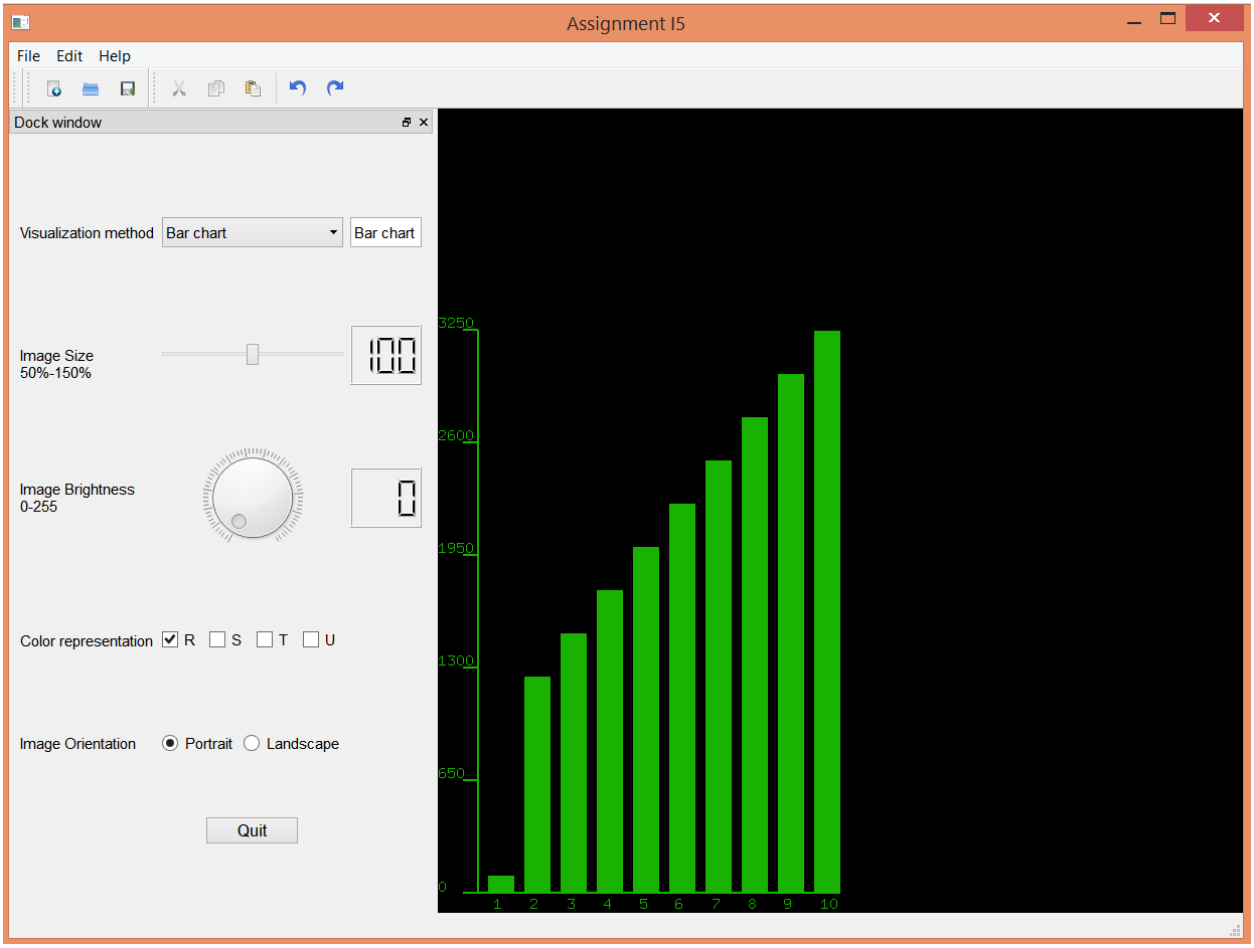


Select bar chart from Visualization methods

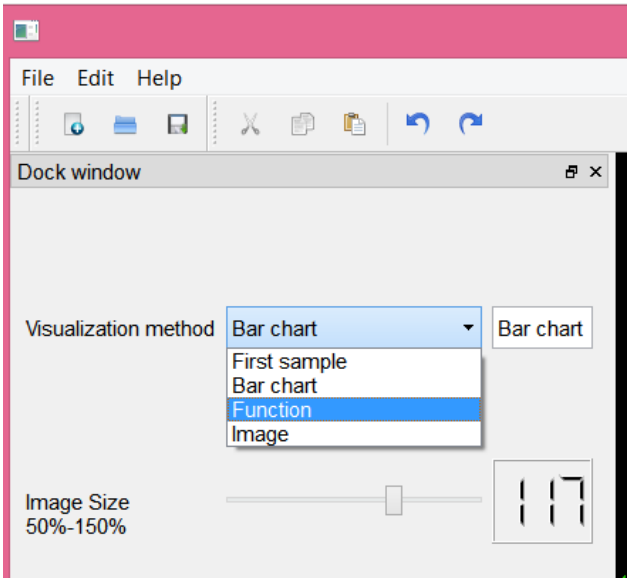




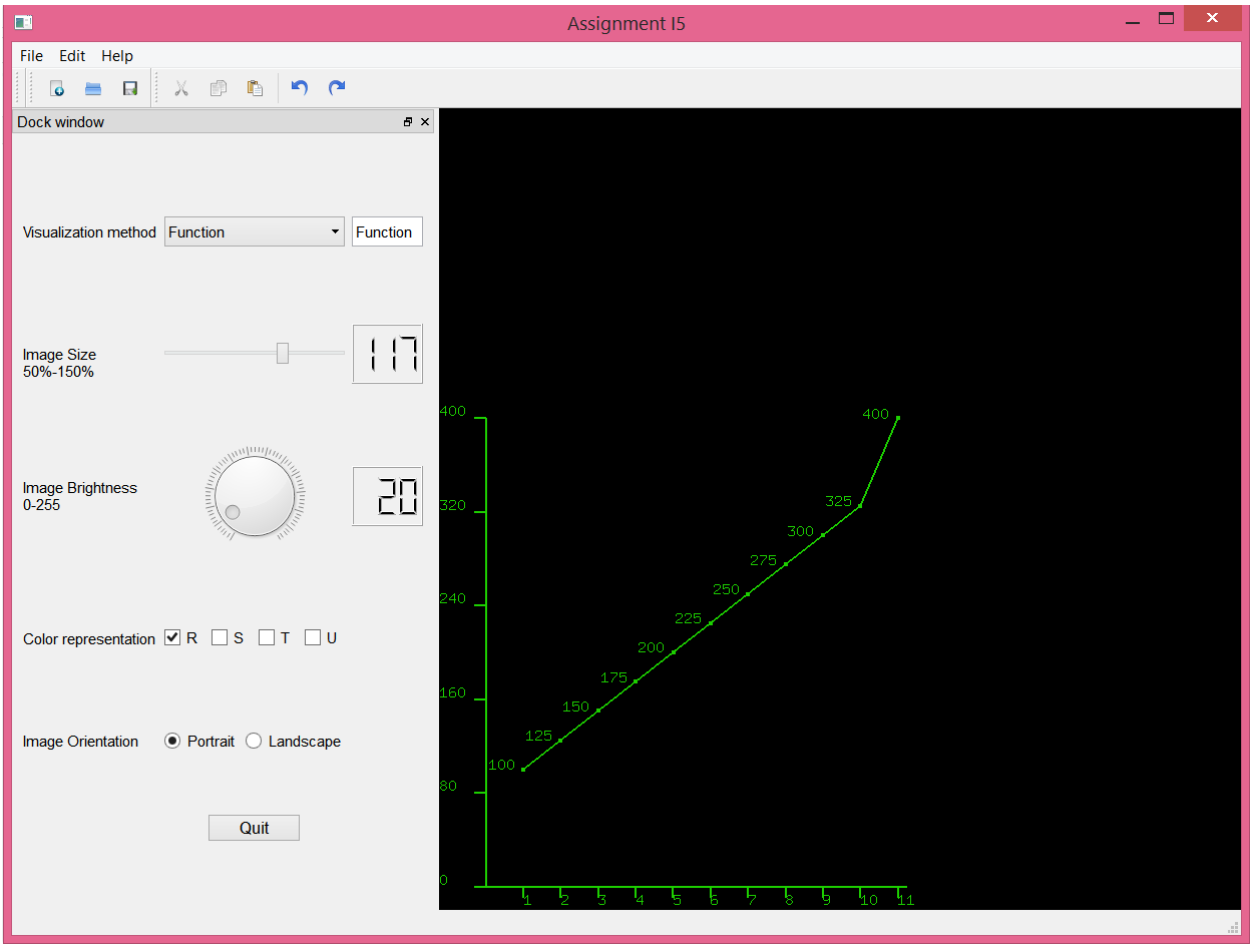
When sample file values are above viewport size values, bar height is scaled down to fit viewport.



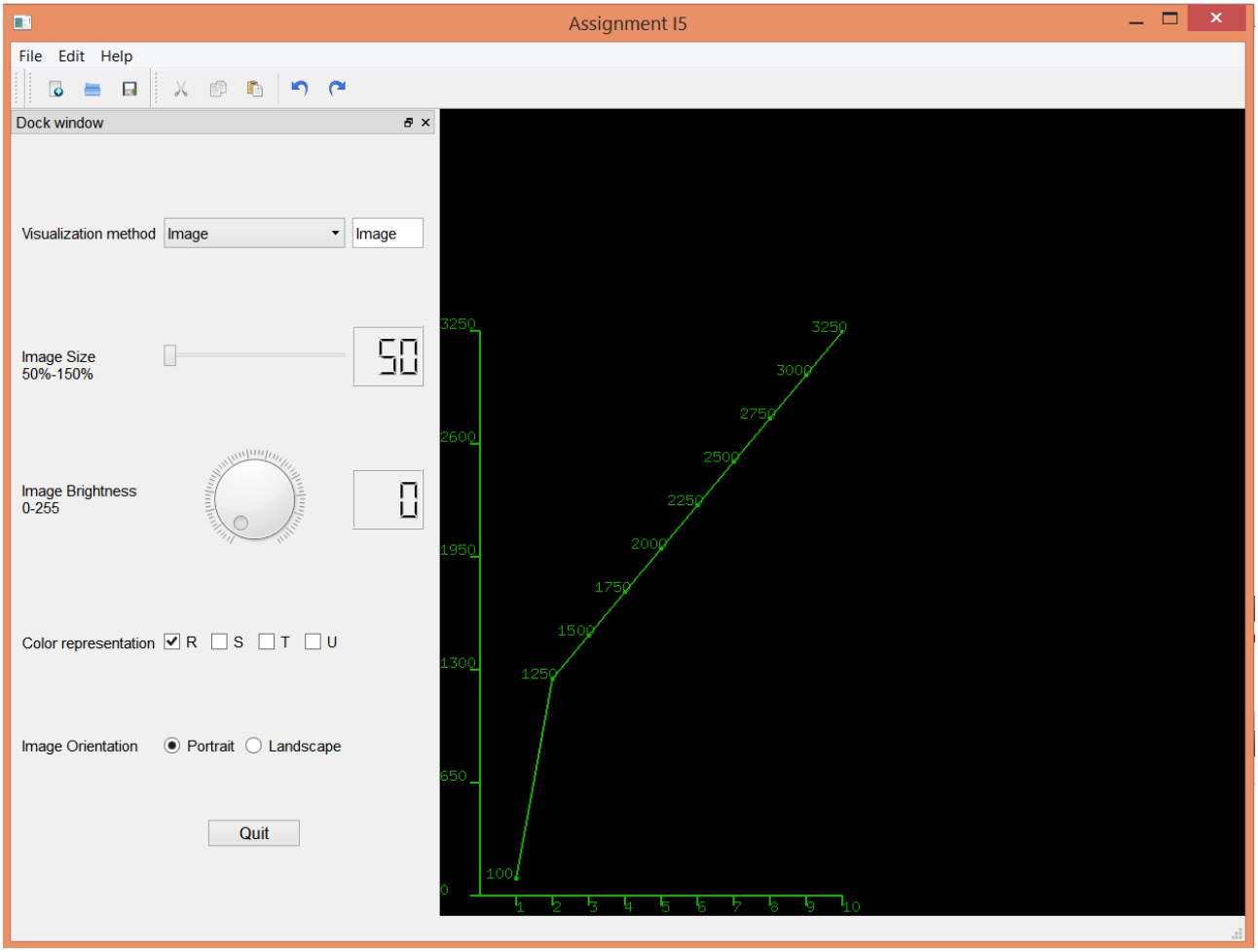
Select Function from Visualization methods.

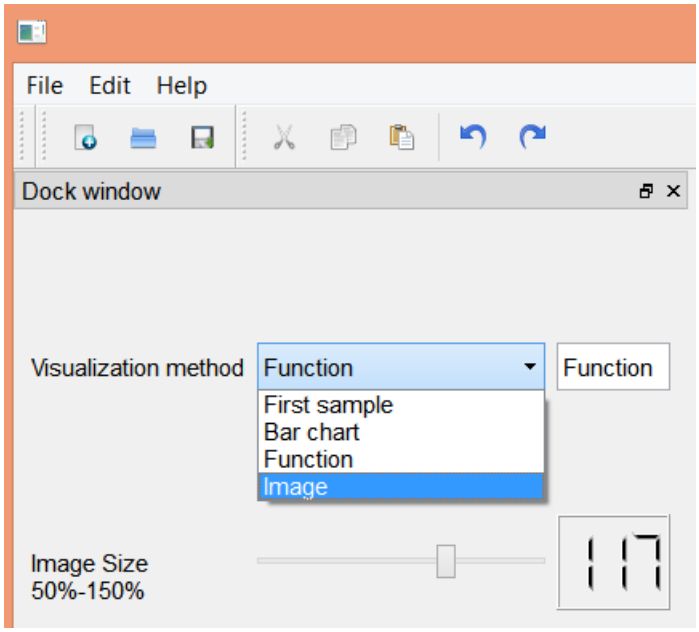


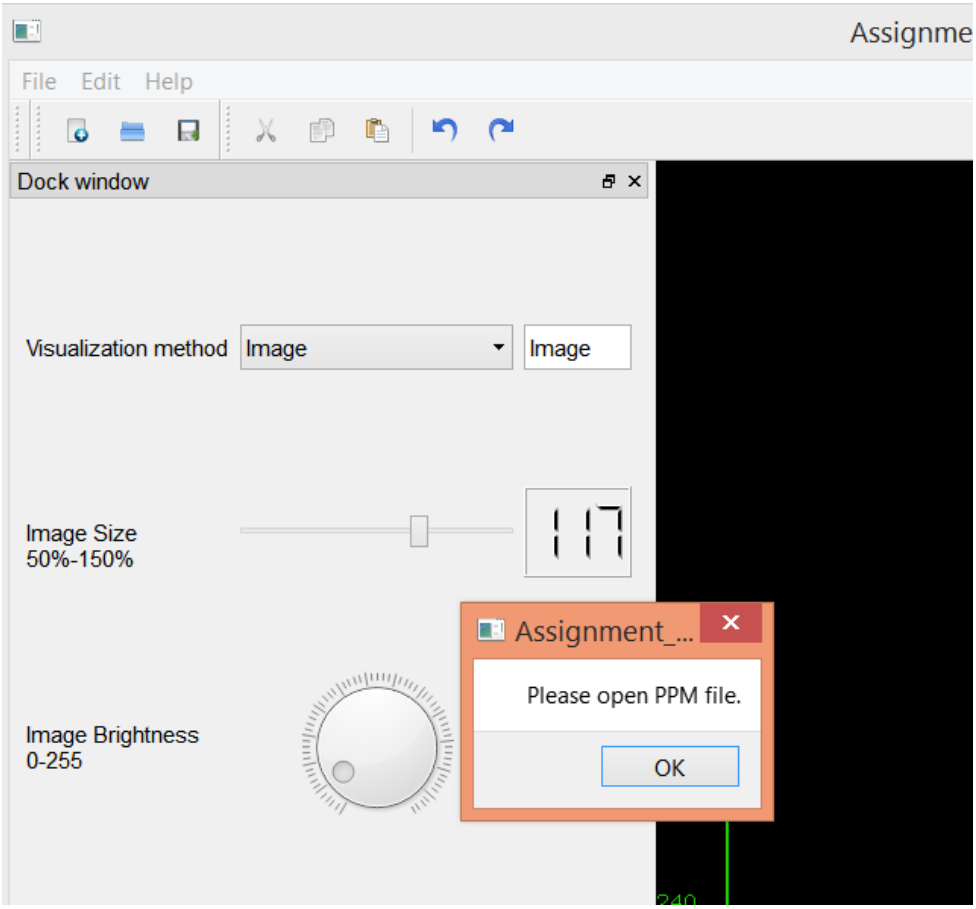
Graph – Values less than viewport size

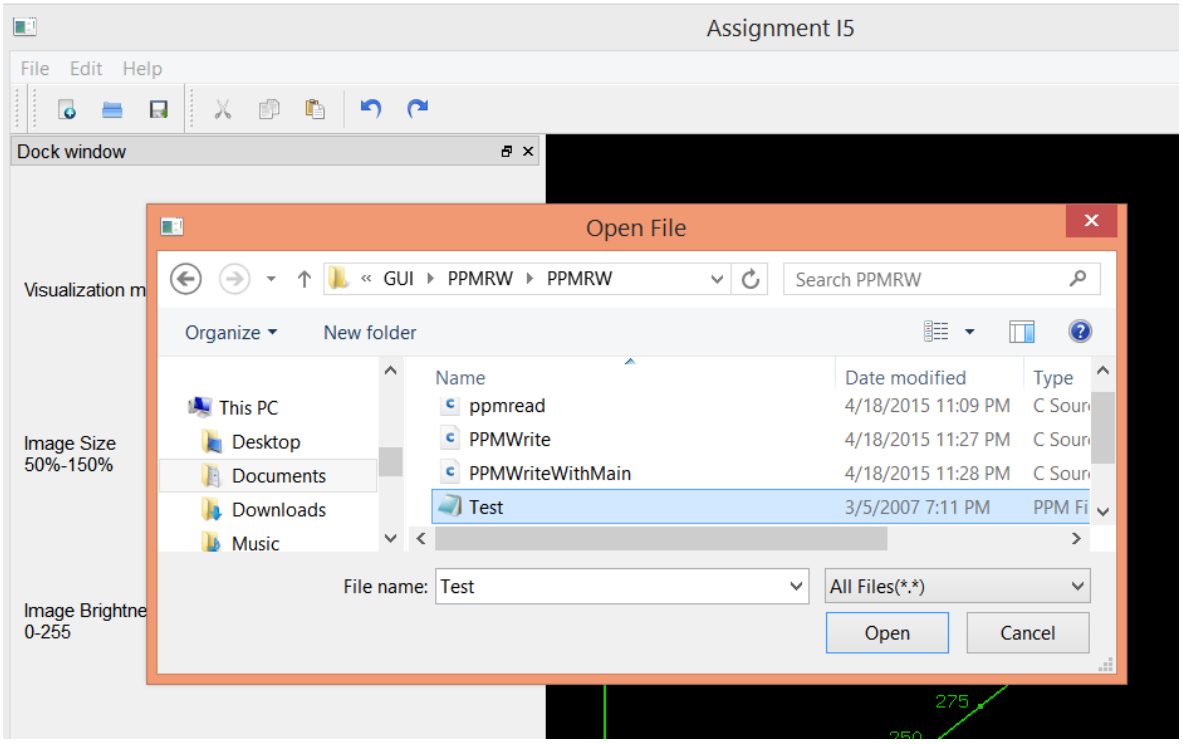


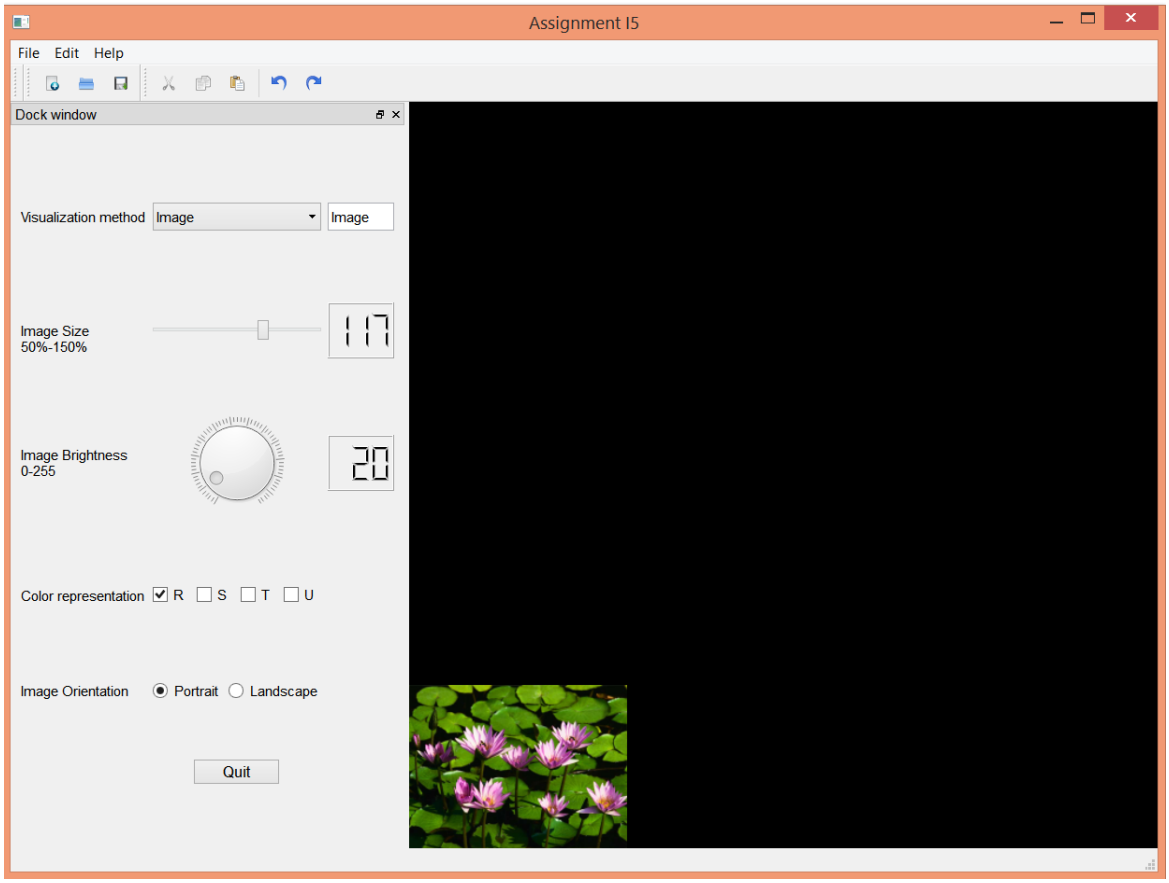
Graph – Values more than viewport size.



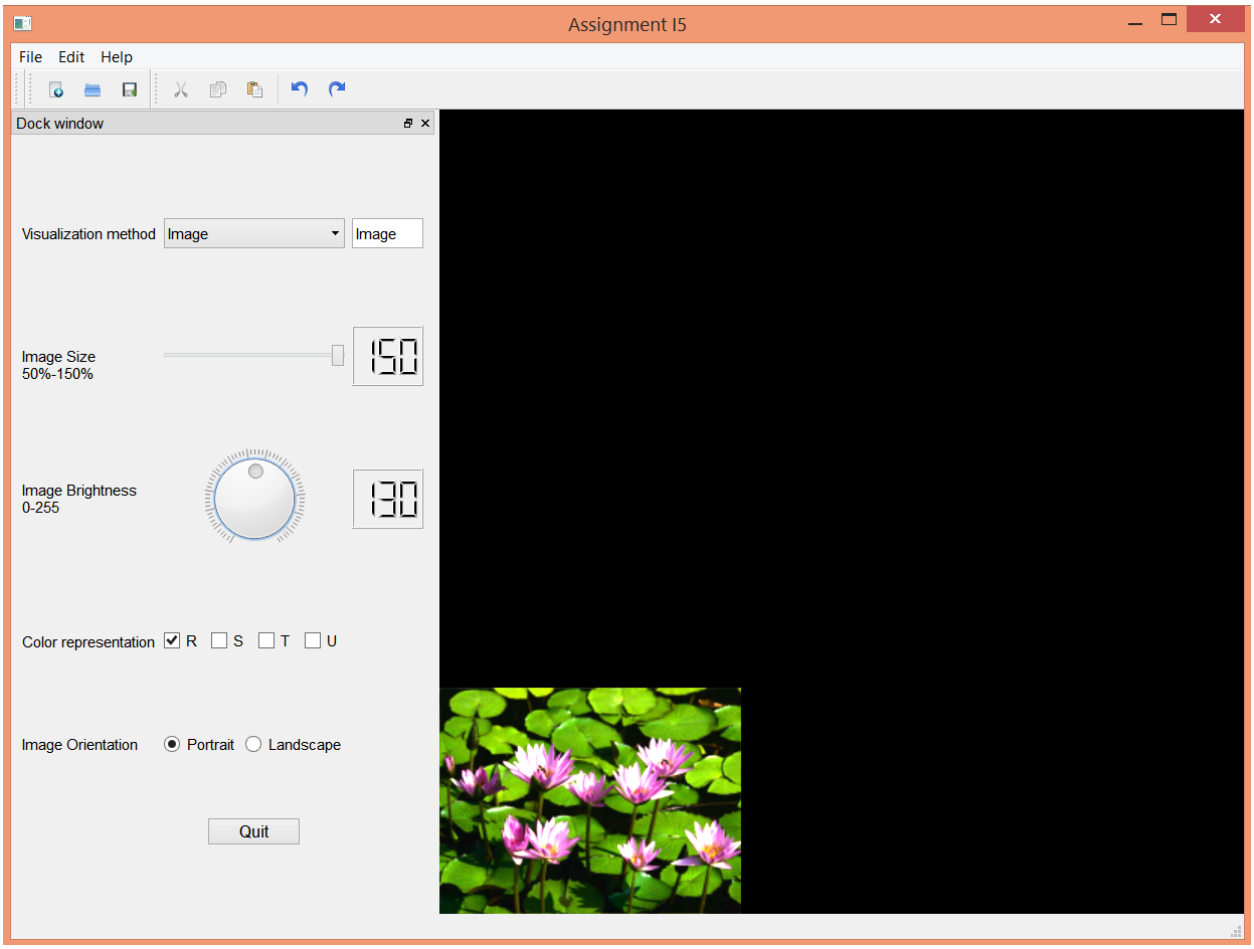

File Edit Help
Dock window
Visualization method
Function
Function
First sample
Bar chart
Function
Image
Image Size
50%-150%
117


Assignme
File Edit Help
Dock window
Visualization method
Image
Image
Image Size
50%-150%
117
Image Brightness
0-255
Assignment_...
Please open PPM file.
OK

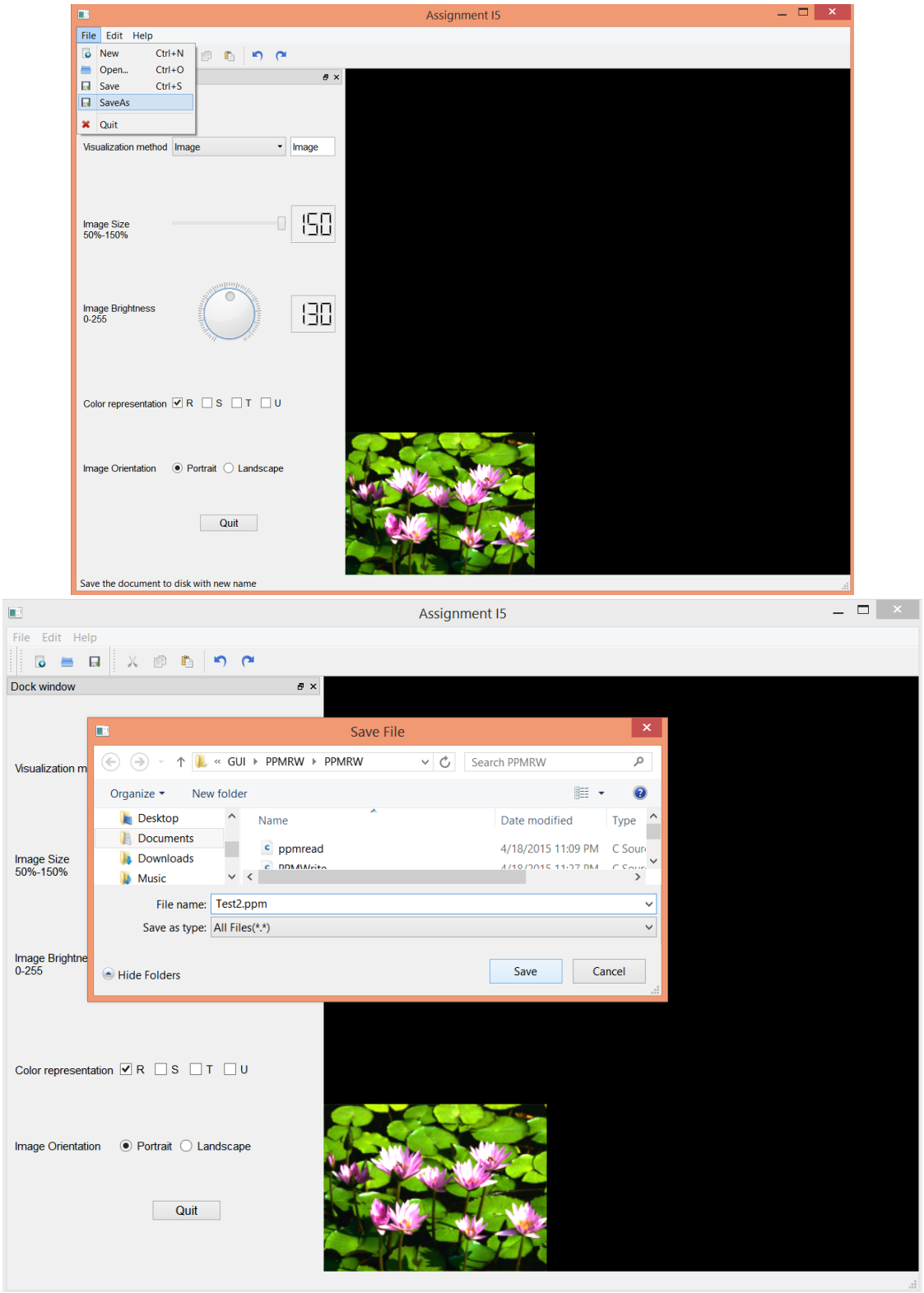

Assignment I5
File Edit Help
Dock window
Visualization m
Image Size
50%-150%
Image Brightne
0-255
Open File
« GUI ▸ PPMRW ▸ PPMRW
Search PPMRW
Organize
New folder
This PC
Desktop
Documents
Downloads
Music
Name
Date modified
Type
ppmread
4/18/2015 11:09 PM
PPMWrite
4/18/2015 11:27 PM
PPMWriteWithMain
4/18/2015 11:28 PM
Test
3/5/2007 7:11 PM
File name: Test
All Files(*.*)
Open
Cancel
275


Assignment I5
File Edit Help
Dock window
Visualization method
Image
Image
Image Size
50%-150%
117
Image Brightness
0-255
20
Color representation
R S T U
Image Orientation
Portrait
Landscape
Quit

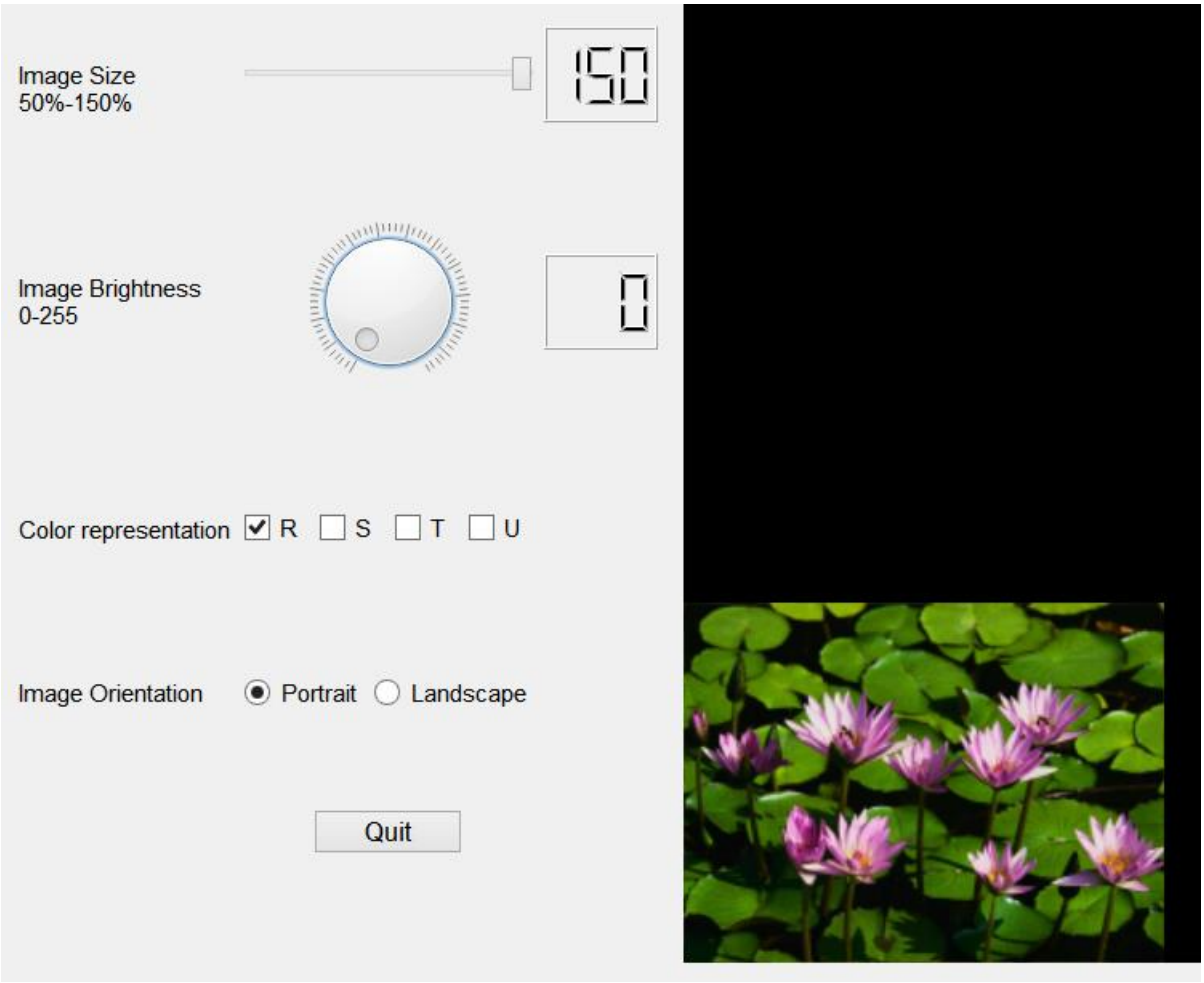
Increase brightness



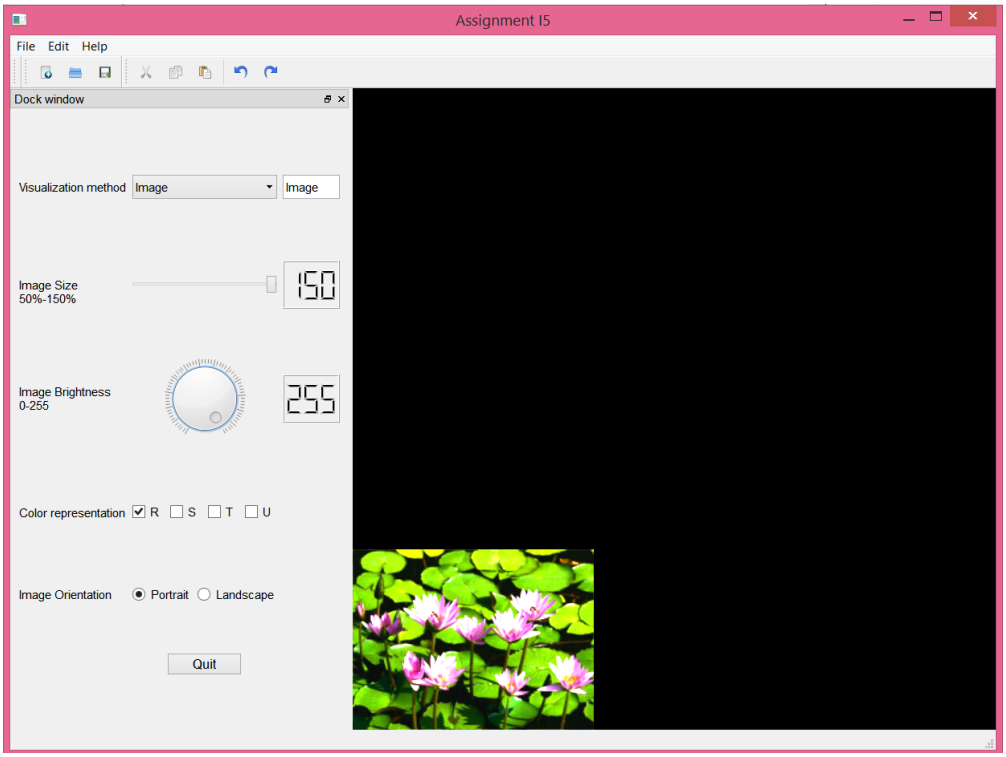
Save PPM image in a file

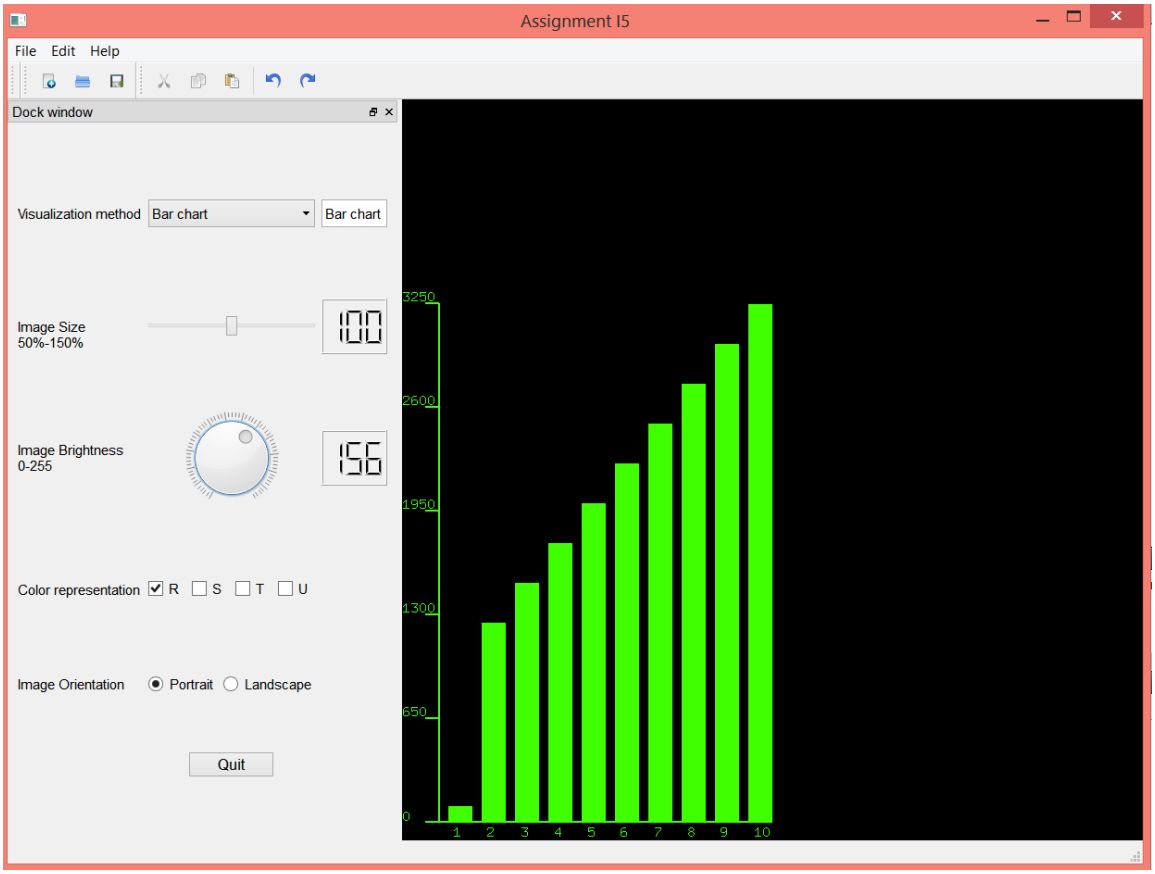


Increase image size to maximum (150)



Increase brightness to maximum





Source Code

TextEditOperations.qrc

```
<!DOCTYPE RCC><RCC version="1.0">
<qresource>
<file>images/copy.png</file>
    <file>images/cut.png</file>
    <file>images/newFile.png</file>
    <file>images/openFile.png</file>
    <file>images/paste.png</file>
    <file>images/saveFile.png</file>
    <file>images/quit.png</file>
    <file>images/about.png</file>
    <file>images/undo.png</file>
    <file>images/redo.png</file>
    <file>images/qtabout.png</file>
</qresource>
</RCC>
```

Header Files (*.h)

Globj.h

```
#ifndef GLOBJ_H
#define GLOBJ_H

#include <QtOpenGL/QGLWidget>

class globj: public QGLWidget
{
    Q_OBJECT
public:
    globj(QWidget *parent = 0);
    ~globj();
    char scene;
    unsigned int noOfTri;
    int width, height, maxVal;
    GLuint* image;
    int noOfSamples;
    unsigned int ssmMaxVal;
    QStringList samplesg;
    int sizeVal, brightVal;
    void changeOrientation();
    void drawTriangles();
    char visualizationMethod;
    void drawVS();
    void changeSize();
    void changeBrightness();
    void drawImage(int width, int height, int maxVal, GLuint* image);
```

```
    void PPMWrite(QString fname);
protected:
    void initializeGL();
    void resizeGL(int w, int h);
    void paintGL();
private:
    GLint x, y;
    int rows , cols;
    int windW, windH;
    GLint barW, barH, baseX;
    QColor faceColors[6];

    void drawTriangle();
    void drawBarChart();
    void calculateXY(GLint row, GLint column);
    void drawGraph();
    GLint calcBaseX(int barNo, int pushY);
    float calculateSizef();
    void displayLegendSymbol(int row, int column, float R, float G, float B,
int symbolVal);
    void displayLegendTitle(int row, int column, char *title);
};
#endif // GLOBJ_H
```

Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTextEdit>
#include "globj.h"
#include <QRadioButton>
#include <QTextStream>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    globj *globj1;
    unsigned int firstSample;
private slots:
    void newFile();
    bool saveAs();
    bool save();
    bool saveFile(const QString&);
    void open();
    void undo();
    void redo();
    void cut();
    void copy();
    void paste();
    void quit();
    void about();
    void aboutQt();
    void toggleView();
    void getIndex(int);
    void getVal(int);
    void getDialVal(int);
private:
    Ui::MainWindow *ui;

    void createActions();
    void createMenus();
    void createToolBars();
    void createStatusBar();
    void createDockWindow();
    bool okToContinue();
    void setCurrentFile(const QString&);
    QString strippedName(const QString&);
    int lineNo;
    void parseSSMFile(QTextStream &stream);
```

```
    void parsePPMFile(QTextStream &stream);
    bool ssmFileOpened;
    bool ppmFileOpened;
    int imageWidth, imageHeight , imageMaxVal;
    GLuint *image;
    QString curFile;
    QTextEdit *textEdit;
    QRadioButton *rb1;
    QRadioButton *rb2;
    QMenu *fileMenu;
    QMenu *editMenu;
    QMenu *helpMenu;
    QMenu *quitMenu;

    QToolBar *fileToolBar;
    QToolBar *editToolBar;

    QAction *newFileAct;
    QAction *openAct;
    QAction *saveAct;
    QAction *saveAsAct;
    QAction *quitAct;

    QAction *cutAct;
    QAction *copyAct;
    QAction *pasteAct;
    QAction *undoAct;
    QAction *redoAct;

    QAction *aboutAct;
    QAction *aboutQtAct;

};

#endif // MAINWINDOW_H
```

Source Files (*.cpp)

Globj.cpp

```
#include "globj.h"
#include <QtGui/QtGui>
#include <QtOpenGL/QtOpenGL>
#include <math.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <string.h>
#include <glut.h>
#include <GL/glext.h>

#define GAP 10
#define COLS 14
#define ROWS 12
GLint boxW, boxH;
//GLint baseY = 20;
GLint baseY = 40;
globj::globj(QWidget *parent):QGLWidget(parent)
{
    rows = 12;
    cols = 12;
    windW = 400, windH = 600;
}

globj::~globj()
{

}

//Initialize the GL settings
void globj::initializeGL()
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    glClearDepth(1.0f);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
}

void globj::changeOrientation()
{
    switch(scene)
    {
    case 'a': // Portrait
    {
        windW = 400;
        windH = 600;
        resizeGL(windW,windH);
        updateGL();
        break;
    }
    case 'b': // Landscape
    {
        windW = 600;
```

```
        windH = 400;
        resizeGL(windW,windH);
        updateGL();
        break;
    }

    }
}
//Set up the viewport based on the screen dimentions
//Function is called implicitly by initializeGL and when screen is resized
void globj::resizeGL( int w, int h )
{
    w = windW;
    h = windH;

    glViewport(0,0,(GLint)w,(GLint)h);

    //setup the projection and switch to model view for transformations
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    // glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 2.0);
    glOrtho(0.0,(GLdouble)w,0.0,(GLdouble)h,-15,15);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

void globj::calculateXY(GLint row, GLint column)
{
    boxW = (windW - (COLS + 1) * GAP) / COLS;
    boxH = (windH - (ROWS + 1) * GAP - 40) / ROWS;

    x = GAP + column * (boxW + GAP);
    y = GAP + row * (boxH + GAP);
}


void globj::drawTriangle()
{
    glBegin(GL_TRIANGLES);
    glVertex2i(x,y);
    glVertex2i(x+boxW/2, y);
    glVertex2i(x+boxW/4, y+boxH/2);
    glEnd();
}

void doRasterString( float x, float y, float z, char *s)
{
    char c;
    glRasterPos3f(x,y,z);
    glColor3f( 0.1,  0.7,  0.0 );
    for ( ; (c = *s) != '\0'; s++ )
        glutBitmapCharacter(GLUT_BITMAP_9_BY_15, c);
}

void globj::drawTriangles()
{
```

```
cols=13; rows = 12;
int noOfTriangles;
int exactROWS = 0, extra = 0;
int noOfOneTriangles = 0;
int noOfSpecialTriangles = 0;
char value[100];
sprintf(value, "Sample value = %d", noOfTri);
glColor3f( 1.0,  1.0,  1.0 );
doRasterString(windW/4,(windH-20),10,(char*)(value));
if (noOfTri <= 144)
{
    noOfTriangles = noOfTri;
    noOfOneTriangles = noOfTri;

    exactROWS = noOfTriangles/12;
    extra = noOfTriangles % 12;
    if (extra > 0)
    {
        rows = exactROWS + 1;
    }
    else
        rows = exactROWS;
    cols = 12;
    for(int row = 0; row < rows; row++)
    {
        for(int column = 0; column < cols; column++)
        {
            if(row > (exactROWS-1 ))
            {cols = extra;}
            calculateXY(row, column);
            drawTriangle();
        }
    }
}
else
{
    int multiple = noOfTri/100;
    noOfOneTriangles = noOfTri % 100;
    noOfSpecialTriangles = multiple;
    noOfTriangles = multiple + noOfOneTriangles;

    exactROWS = noOfTriangles/12;
    extra = noOfTriangles % 12;
    if (extra > 0)
    {
        rows = exactROWS + 1;
    }
    else
        rows = exactROWS;
    cols = 12;
    int trianglesDrawn = 0;

    for(int row = 0; row < rows; row++)
    {
        for(int column = 0; column < cols; column++)
        {
            if(row > (exactROWS-1 ))
```

```
                {cols = extra;}
                calculateXY(row, column);
                if(trianglesDrawn < multiple)
                {
                    glColor3f( 1.0,  0.0,  0.0 );
                }
                else
                    glColor3f( 0.1,  0.7,  0.0 );
                drawTriangle();
                trianglesDrawn += 1;
            }
        }
        glFlush();

    }

    displayLegendTitle(11, 13, "Value");
    displayLegendSymbol(10, 13, 0.1, 0.7, 0.0, 1);

    if(noOfTri > 144)
    {
        displayLegendSymbol(9, 13, 1.0, 0.0, 0.0, 100);
    }

    displayLegendTitle(8, 13, "Count");
    displayLegendSymbol(7,13,0.1,0.7,0.0,noOfOneTriangles);
    if(noOfTri > 144)
    {
        displayLegendSymbol(6,13,1.0,0.0,0.0,noOfSpecialTriangles);
    }
}

void globj::displayLegendSymbol(int row, int column, float R, float G, float
B, int symbolVal)
{
    char value1[20];
    calculateXY(row,column);
    glColor3f( R,  G,  B );
    drawTriangle();
    sprintf(value1, " = %d", symbolVal);
    glColor3f( 1.0,  1.0,  1.0 );
    doRasterString(x+boxW/2, y, 0.0, (char*)(value1));
}

void globj::displayLegendTitle(int row, int column, char* title)
{
    calculateXY(row,column);
    glColor3f( 1.0,  1.0,  1.0 );
    doRasterString(x, y , 0.0, title);
}

void globj::drawVS()
{
    updateGL();
}
```

```
GLint globj::calcBaseX(int barNo, int pushY)
{
    baseX = GAP + barNo * (barW + GAP)  + pushY;
    return baseX;
}


void globj::drawBarChart()
{
    //int pushY = 40;
    int pushY = 60;
    barW = (windW - GAP * noOfSamples - pushY)/noOfSamples ;

    GLint baseX;
    GLint baseX2;
    char value[60];
    unsigned int ns[noOfSamples];
    unsigned int maxValue = ssmMaxVal;
    for(int i = 0; i < noOfSamples; ++i)
    {
        ns[i] = samplesg.at(i).toInt();
    }
    if (ssmMaxVal > windH - baseY - 20)
    {
        unsigned int diff = ssmMaxVal - (windH - baseY - 20);
        float diffPercent = (diff*100/(float)ssmMaxVal);
        unsigned int sampleVal;
        for(int i = 0; i < noOfSamples; i++)
        {
            sampleVal = samplesg.at(i).toInt();
            ns[i] = sampleVal - ((diffPercent*sampleVal)/100) ;
        }
        maxValue = ssmMaxVal - diff;
        /* sprintf(value, "Values scaled down by %d percent",
(int)diffPercent);
        glColor3f( 0.1,  0.7,  0.0 );
        doRasterString(windW/4, windH-20 ,10, (char*)(value));*/
    }


    baseX = calcBaseX(0, pushY);

    // Y axis
    glLineWidth(2.0f);
    glBegin(GL_LINES);
    glColor3f( 0.1,  0.7,  0.0 );
    glVertex2i(baseX - GAP , baseY);
    glVertex2i(baseX - GAP, maxValue + baseY);
    glEnd();

    // Tick marks
    GLint lenY = maxValue ;
    GLint slot = lenY/5;
    for(int i = 0; i < 6; ++i)
    {
        glBegin(GL_LINES);
        glVertex2i(baseX - GAP - 15, baseY + slot*i);
```

```
        glVertex2i(baseX - GAP, baseY + slot*i);
        glEnd();
    }

    // Value on Y axis
    float slotVal = ssmMaxVal/5;
    int newSlotVal = 0;
    for(int i = 0; i < 6; i++)
    {
        sprintf(value, "%d", newSlotVal);
        doRasterString(baseX  - GAP - 30, baseY + slot*i +2 ,10,
(char*)(value));
        newSlotVal = newSlotVal + slotVal;
    }

    // X axis
    glBegin(GL_LINES);
    glColor3f( 0.1,  0.7,  0.0 );
    glVertex2i(baseX - barW/2, baseY);
    glVertex2i(windW, baseY);
    glEnd();

    int xValPos;
    int xVal;

    for(int j = 0; j < noOfSamples; j++)
    {
        baseX = calcBaseX(j, pushY);
        baseX2 = baseX + barW;

        // Values on X axis
        xValPos =  baseX + barW/4;
        xVal = j+1;
        sprintf(value, "%d", xVal);
        doRasterString(xValPos, baseY -15 ,10, (char*)(value));

        // drawing the bars
        glBegin(GL_QUADS);
        glColor3f( 0.1,  0.7,  0.0 );// light green
        glVertex2i(baseX , baseY);
        glVertex2i(baseX2, baseY);
        glVertex2i(baseX2, baseY + ns[j]);
        glVertex2i(baseX, baseY + ns[j]);
        glEnd();

    }
}
void setWindow(double left, double right, double bottom, double top)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(left, right, bottom, top);
}
void globj::drawGraph()
{
    unsigned int ns[noOfSamples];
    unsigned int maxValue = ssmMaxVal;
```

```
    for(int i = 0; i < noOfSamples; ++i)
    {
        ns[i] = samplesg.at(i).toInt();
    }
    glColor3f( 0.1,  0.7,  0.0 );
    //setWindow(0.0, (double)noOfSamples + 1, 0.0, (double)ssmMaxVal);
    char value[10];
    int y, pushY = 40;
    int  xUnit = (windW-pushY)/noOfSamples;
    int xVal[noOfSamples];
    xVal[0] = xUnit + pushY;

    //if (ssmMaxVal + baseY > windH)
    if (ssmMaxVal > windH - baseY - 20)
    {
        int diff = ssmMaxVal - (windH - baseY - 20);
        float diffPercent = (diff*100/(float)ssmMaxVal);
        unsigned int sampleVal;
        for(int i = 0; i < noOfSamples; i++)
        {
            sampleVal = samplesg.at(i).toInt();
            ns[i] = sampleVal - ((diffPercent*sampleVal)/100);
        }
        maxValue = ssmMaxVal - diff;
        /* sprintf(value, "Values scaled down by %d percent",
(int)diffPercent);
        glColor3f( 0.1,  0.7,  0.0 );
        doRasterString(windW/4, windH-20 ,10, (char*)(value));*/
    }
    // draw graph
    glBegin(GL_LINE_STRIP);
    for(int i = 0; i < noOfSamples; ++i)
    {
        y = ns[i] + baseY;
        glVertex2i(xVal[i], y);
        xVal[i+1] = xVal[i] + xUnit;
    }
    glEnd();

    // draw points on graph
    glPointSize(4.0);
    glBegin(GL_POINTS);
    for(int i = 0; i < noOfSamples; ++i)
    {
        y = ns[i] + baseY;
        glVertex2i(xVal[i], y);


    }
    glEnd();

    for(int i = 0; i < noOfSamples; ++i)
    {
        // write values on points
        y = ns[i] + baseY;
        sprintf(value, "%d", samplesg.at(i).toInt());
        doRasterString( xVal[i] - 30 , y ,10, (char*)(value));
```

```
        // Values on X axis
        sprintf(value, "%d", i+1);
        doRasterString(xVal[i], baseY -15 ,10, (char*)(value));

        //Tick marks on X axis
        glBegin(GL_LINES);
        glVertex2i(xVal[i], baseY );
        glVertex2i(xVal[i], baseY - 10);
        glEnd();
    }
    GLint params[4];
    glGetIntegerv(GL_VIEWPORT, params);

    // Y axis
    glLineWidth(2.0f);
    glBegin(GL_LINES);
    glColor3f( 0.1,  0.7,  0.0 );
    glVertex2i(params[0] + pushY , baseY);
    glVertex2i(params[0] + pushY, maxValue + baseY);
    glEnd();

    // X axis
    glBegin(GL_LINES);
    glColor3f( 0.1,  0.7,  0.0 );
    glVertex2i(params[0] + pushY, baseY);
    glVertex2i(windW, baseY);
    glEnd();

    // Tick marks on Y axis
    GLint lenY = maxValue;
    GLint slot = lenY/5;
    for(int i = 0; i < 6; ++i)
    {
        glBegin(GL_LINES);
        glVertex2i(params[0] + pushY-10, baseY + slot*i);
        glVertex2i(params[0] + pushY, baseY + slot*i);
        glEnd();
    }

    // Value on Y axis
    float slotVal = ssmMaxVal/5;
    int newSlotVal = 0;
    for(int i = 0; i < 6; i++)
    {
        sprintf(value, "%d", newSlotVal);
        doRasterString( params[0] , baseY + slot*i +2 ,10, (char*)(value));
        newSlotVal = newSlotVal + slotVal;
    }

}

void globj::changeSize()
{
    float size = calculateSizef();
    glViewport(0, 0,  windW * size , windH * size);
```

```
    updateGL();
}

void globj::changeBrightness()
{
    unsigned char *data;
    data = (unsigned char*)malloc(3*800*800);
    // if(out==0) {printf("error Allocating Memory \n");return ;}
    if (data != 0)
    {
        glDisable(GL_DEPTH_TEST);
        // glReadBuffer(GL_COLOR_ATTACHMENT0);
        glReadPixels(1, 1, 800, 800, GL_RGB, GL_UNSIGNED_BYTE, &data[0]);
        //glReadBuffer(GL_BACK);
    }
    float brightness = 1 + (brightVal/255.0);
    glPixelTransferf(GL_RED_SCALE, brightness);
    glPixelTransferf(GL_GREEN_SCALE, brightness);
    glPixelTransferf(GL_BLUE_SCALE, brightness);
    glPixelStorei(GL_UNPACK_SWAP_BYTES,GL_TRUE);

    glRasterPos2i(0,0);
    glDrawPixels(800,800,GL_RGB, GL_UNSIGNED_BYTE, data);
}
float globj:: calculateSizef()
{
    float s =  ((float)sizeVal)/100.0;
    return s;
}

void globj::PPMWrite(QString fname)
{
    /*
    w-Weight, w-Hieght, and d-Depth - Depth is the max value of pixels
    in the frame buffer
    */
    int w, h, d;
    int *y;

    int start_x=0, start_y=0 ; /* starting position in Frame Buffer - default
= (0,0) */

    unsigned char      * out;

    y = new int;
    float s = calculateSizef();
    w = width *s ;
    h = height * s;
    //glReadBuffer(GL_LEFT);
    glGetIntegerv(GL_RED_BITS, y);
    d = (1 << *y) - 1;


    QFile file(fname);

    out = (unsigned char *) malloc(3*w*h);
    if(out==0) {printf("error Allocating Memory \n");return ;}
```

```
    glPixelStorei(GL_PACK_ALIGNMENT,1); /* byte aligned output */
    glReadPixels(start_x,start_y,w,h, GL_RGB,GL_UNSIGNED_BYTE,&out[0]);

    int arrSize = 3*w*h;

    if (!file.open(QIODevice::WriteOnly | QIODevice::Text))
    {
        QMessageBox::warning(this, tr("Caution!"), tr("Cannot write to file
%1:\n%2.")
                             .arg(fname) .arg(file.errorString()));

    }else
    {
        QTextStream outF(&file);
        int val;

        outF << "P3\n";
        outF << w;
        outF << " ";
        outF << h;
        outF << "\n";
        outF << d;
        outF << "\n";
        for(int i = arrSize-1; i >=0; i--)
        {

            val = (out[i]);
            outF <<  val ;
            outF << ' ';
        }

    }
    delete y;
}

//Paints the GL scene
void globj::paintGL()
{
    glClear (GL_COLOR_BUFFER_BIT);
    glClear(GL_DEPTH_BUFFER_BIT);

    if (visualizationMethod == 'A')
    {
        glColor3f (1.0, 0.5, 1.0);
        drawTriangles();

    }
    if (visualizationMethod == 'B')
    {
        drawBarChart();

    }
    if (visualizationMethod == 'C')
    {
        drawGraph();
```

```
    }
    if (visualizationMethod == 'D')
    {
        glClear (GL_COLOR_BUFFER_BIT);
        float size = calculateSizef();
        float s;
        s=255./maxVal;
        unsigned char* image1;
        // for brightnening image by s.
        glPixelTransferf(GL_RED_SCALE, s);
        glPixelTransferf(GL_GREEN_SCALE, s);
        glPixelTransferf(GL_BLUE_SCALE, s);
        glPixelStorei(GL_UNPACK_SWAP_BYTES,GL_TRUE);
        //
        //glRasterPos2i(0,0);

        // glPixelZoom(size,size);
        // glDrawPixels(width,height,GL_RGB, GL_UNSIGNED_INT, image);
        image1 = (unsigned char*)malloc(3*windW*windH);
        if(sizeVal >= 50)
        {
            int success =
gluScaleImage(GL_RGB,width,height,GL_UNSIGNED_INT,image,width * size, height
* size,GL_UNSIGNED_BYTE,image1 );
            if(success != 0)
            {
                QMessageBox msgBox;
                msgBox.setText((QString::number(success)));

                msgBox.exec();
            }
            glRasterPos2i(0,0);
            glDrawPixels(width * size, height * size,GL_RGB,
GL_UNSIGNED_BYTE, image1);}
        glFlush();

    }

    if (brightVal > 0)
    {
        unsigned char *data;
        data = (unsigned char*)malloc(3*800*800);
        // if(out==0) {printf("error Allocating Memory \n");return ;}
        if (data != 0)
        {
            glDisable(GL_DEPTH_TEST);
            glReadBuffer(GL_COLOR_ATTACHMENT0);
            glReadPixels(0, 0, 800, 800, GL_RGB, GL_UNSIGNED_BYTE, &data[0]);
            glReadBuffer(GL_BACK);
        }
        float brightness = 1 + (brightVal/255.0);
        glPixelTransferf(GL_RED_SCALE, brightness);
        glPixelTransferf(GL_GREEN_SCALE, brightness);
        glPixelTransferf(GL_BLUE_SCALE, brightness);
        glPixelStorei(GL_UNPACK_SWAP_BYTES,GL_TRUE);
```

```
        glRasterPos2i(0,0);
        glDrawPixels(800,800,GL_RGB, GL_UNSIGNED_BYTE, data);
    }
    /* don't wait!
   * start processing buffered OpenGL routines
   */
    glFlush ();
}
```

Main.cpp

```
#include "mainwindow.h"
#include <QApplication>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.resize( 800, 800 );
    w.show();

    return a.exec();
}
```

Mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QtGui"
#include "QMessageBox"
#include "QFileDialog"
#include "QFile"
#include <QDockWidget>
#include <QLCDNumber>
#include <QComboBox>
#include <QLabel>
#include <QWidget>
#include <QVBoxLayout>
#include <QString>
#include <QLineEdit>
#include <QSlider>
#include <QDial>
#include <QHBoxLayout>
#include <QRadioButton>
#include <QCheckBox>
#include <QPushButton>
#include <QtGui/QIcon>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    textEdit = new QTextEdit;
    // setCentralWidget(textEdit);

    createActions();
    createMenus();
    createToolBars();
    createStatusBar();

    // create a graphics window of 800 by 800 pixels.
    globj1 = new globj(this);
    setCentralWidget(globj1);
    globj1->setFixedSize(800,800);
    this->adjustSize();

    createDockWindow();

    setWindowTitle(tr("Assignment I5"));
    curFile = "";
    ssmFileOpened = false;
    ppmFileOpened = false;
}

MainWindow::~MainWindow()
{
    delete globj1;
    delete ui;
```

```
}
void MainWindow::createDockWindow()
{
    QLabel *label = new QLabel("Visualization method");
    QFont f( "Arial", 9);
    label->setFont(f);
    label->setFixedHeight(60);

    QLabel *size = new QLabel("\nImage Size\n50%-150%");
    size->setFont(f);
    QLabel *bright = new QLabel("Image Brightness\n0-255");
    bright->setFont(f);
    QLabel *color = new QLabel("Color representation");
    color->setFont(f);
    color->setFixedHeight(60);
    QLabel *orient = new QLabel("Image Orientation");
    orient->setFont(f);

    QComboBox *combo = new QComboBox();
    combo->addItem("First sample");
    combo->addItem("Bar chart");
    combo->addItem("Function");
    combo->addItem("Image");

    combo->setFixedHeight(30);
    combo->setFont(f);

    QSlider *Slider = new QSlider(Qt::Horizontal);
    Slider->setRange(50,150);
    Slider->setValue(100);
    globj1->sizeVal = Slider->value();

    QDial *dial = new QDial();
    dial->setNotchesVisible(true);
    dial->setRange(0,255);
    dial->setValue(0);

    QCheckBox *pb1 = new QCheckBox("R");
    QCheckBox *pb2 = new QCheckBox("S");
    QCheckBox *pb3 = new QCheckBox("T");
    QCheckBox *pb4 = new QCheckBox("U");
    pb1->setChecked(true);
    pb1->setFont(f);
    pb2->setFont(f);
    pb3->setFont(f);
    pb4->setFont(f);

    rb1 = new QRadioButton("Portrait");
    rb2 = new QRadioButton("Landscape");
    rb1->setChecked(true);
    rb1->setFont(f);
    rb2->setFont(f);

    QLineEdit *le = new QLineEdit();
    le->setReadOnly(true);
    le->setText("First sample");
    le->setFixedHeight(30);
```

```
    le->setFont(f);

    QLCDNumber *LCD1 = new QLCDNumber(3);
    LCD1->display(Slider->value());
    LCD1->setFixedHeight(60);
    QLCDNumber *LCD2 = new QLCDNumber(3);
    LCD2->setFixedHeight(60);
    QPushButton *quit  = new QPushButton();
    quit->setText("Quit");
    QFont q( "Arial", 10);
    quit->setFont(q);


    /* Connecting combo box to line edit by passing a string */
    connect(combo, SIGNAL(currentIndexChanged(QString)), le,
SLOT(setText(QString)));
    connect(combo, SIGNAL(activated(int)), this, SLOT(getIndex(int)));

    /* Connecting Slider to an LCD by passing an integer value*/
    connect(Slider,SIGNAL(valueChanged(int)), LCD1, SLOT(display(int)));
    connect(Slider,SIGNAL(valueChanged(int)), this, SLOT(getVal(int)));


    /* Connecting Dial to an LCD by passing an integer value */
    connect(dial,SIGNAL(valueChanged(int)), LCD2, SLOT(display(int)));
    connect(dial,SIGNAL(valueChanged(int)), this, SLOT(getDialVal(int)));

    // Connecting clicked signal of quit button to quit function
    connect(quit,SIGNAL(clicked()),qApp, SLOT(quit()));

    //Set Signals for clicking buttons
    connect(rb1, SIGNAL(clicked()), this, SLOT(toggleView()));
    connect(rb2, SIGNAL(clicked()), this, SLOT(toggleView()));

    QDockWidget *dock =  new QDockWidget(tr("Dock window"), this);
    dock->setFont(f);
    dock->setAllowedAreas(Qt::LeftDockWidgetArea);
    QHBoxLayout *hlayout = new QHBoxLayout(dock);
    QHBoxLayout *hlayout1 = new QHBoxLayout(dock);
    QGridLayout *glayout = new QGridLayout(dock);
    glayout->addWidget(label,1,1,1,1,0);
    glayout->addWidget(combo,1,2,1,1,0);
    glayout->addWidget(le,1,3,1,1,0);
    glayout->addWidget(size,2,1,1,1,0);
    glayout->addWidget(Slider,2,2,1,1,0);
    glayout->addWidget(LCD1,2,3,1,1,0);
    glayout->addWidget(bright,3,1,1,1,0);
    glayout->addWidget(dial,3,2,1,1,0);
    glayout->addWidget(LCD2,3,3,1,1,0);
    glayout->addWidget(color,4,1,1,1,0);
    glayout->addLayout(hlayout,4,2,1,1,0);
    hlayout->addWidget(pb1);
    hlayout->addWidget(pb2);
    hlayout->addWidget(pb3);
    hlayout->addWidget(pb4);
    glayout->addWidget(orient,5,1,1,1,0);
    glayout->addLayout(hlayout1,5,2,1,1,0);
```

```
    hlayout1->addWidget(rb1);
    hlayout1->addWidget(rb2);

    glayout->addWidget(quit,6,2,1,1,Qt::AlignAbsolute);
    QWidget *newWidget = new QWidget(this);
    newWidget->setLayout(glayout);

    dock->setWidget(newWidget);

    addDockWidget(Qt::LeftDockWidgetArea,dock);
}


void MainWindow::getIndex(int ind)
{
    if(ind == 0 || ind == 1 || ind == 2)
    {
        if (ssmFileOpened == false)
        {
            QMessageBox msgBox;
            msgBox.setText("Please open SSM file.");
            msgBox.exec();
        }
        else
        {
            if (ind == 0)
            {
                globj1->visualizationMethod = 'A';
                globj1->noOfTri = firstSample;
                globj1->drawVS();
            }
            if(ind == 1)
            {
                globj1->visualizationMethod = 'B';
                globj1->drawVS();
            }
            if(ind == 2)
            {
                globj1->visualizationMethod = 'C';
                globj1->drawVS();
            }
        }
    }

    if (ind == 3)
    {
        if (ppmFileOpened == true)
        {
            globj1->visualizationMethod = 'D';
            globj1->width = imageWidth;
            globj1->height = imageHeight;
            globj1->maxVal = imageMaxVal;
            globj1->image = image;
            globj1->drawVS();
        }
        else
        {
```

```
            QMessageBox msgBox;
            msgBox.setText("Please open PPM file.");
            msgBox.exec();
        }
    }
}

void MainWindow::getVal(int val)
{
    globj1->sizeVal = val;
    globj1->changeSize();
}

void MainWindow::getDialVal(int val)
{
    globj1->brightVal = val;
    globj1->drawVS();
}

void MainWindow::toggleView()
{
    //change the scene variable in the gl object to the clicked button
    if(rb1->isChecked())
        globj1->scene = 'a';

    else if(rb2->isChecked())
        globj1->scene = 'b';

    globj1->changeOrientation();

}


void MainWindow::createActions()
{
    newFileAct = new QAction( QIcon(":/images/newFile.png"), "&New", this );

    /* By using the appropriate QKeySequence::StandardKey enum value, we
ensure that Qt
          will provide the correct shortcuts for the platform on which the
application is running.*/
    newFileAct->setShortcuts(QKeySequence::New);
    newFileAct->setStatusTip("Create a new file");
    newFileAct->setToolTip("Create a new file");
    connect( newFileAct, SIGNAL(triggered())  , this,SLOT( newFile() ) );

    openAct = new QAction(QIcon(":/images/openFile.png"),tr("&Open..."),
this);
    openAct->setShortcuts(QKeySequence::Open);
    openAct->setStatusTip(tr("Open an existing file"));
    connect(openAct, SIGNAL(triggered()), this, SLOT(open()));

    saveAct = new QAction(QIcon(":/images/saveFile.png"), tr("&Save"), this);
    saveAct->setShortcuts(QKeySequence::Save);
    saveAct->setStatusTip(tr("Save the document to disk"));
    connect(saveAct, SIGNAL(triggered()), this, SLOT(save()));
```

```
    saveAsAct = new QAction(QIcon(":/images/saveFile.png"), tr("&SaveAs"),
this);
    saveAsAct->setShortcuts(QKeySequence::SaveAs);
    saveAsAct->setStatusTip(tr("Save the document to disk with new name"));
    connect(saveAsAct, SIGNAL(triggered()), this, SLOT(saveAs()));

    cutAct = new QAction(QIcon(":/images/cut.png"),tr("&Cut"), this);
    cutAct->setShortcuts(QKeySequence::Cut);
    cutAct->setStatusTip(tr("Cut some text from file"));
    connect(cutAct, SIGNAL(triggered()), this, SLOT(cut()));

    copyAct = new QAction(QIcon(":/images/copy.png"),tr("&Copy"), this);
    copyAct->setShortcuts(QKeySequence::Copy);
    copyAct->setStatusTip(tr("Copy selected text to clipboard"));
    connect(copyAct, SIGNAL(triggered()), this, SLOT(copy()));

    pasteAct = new QAction(QIcon(":/images/paste.png"),tr("&Paste"), this);
    pasteAct->setShortcuts(QKeySequence::Paste);
    pasteAct->setStatusTip(tr("Paste copied text to file"));
    connect(pasteAct, SIGNAL(triggered()), this, SLOT(paste()));

    undoAct = new QAction(QIcon(":/images/undo.png"),tr("&Undo"), this);
    undoAct->setShortcuts(QKeySequence::Undo);
    undoAct->setStatusTip(tr("Reverse the changes done"));
    connect(undoAct, SIGNAL(triggered()), this, SLOT(undo()));

    redoAct = new QAction(QIcon(":/images/redo.png"),tr("&Redo"), this);
    redoAct->setShortcuts(QKeySequence::Redo);
    redoAct->setStatusTip(tr("Redo the last edit action"));
    connect(redoAct, SIGNAL(triggered()), this, SLOT(redo()));

    quitAct = new QAction(QIcon(":/images/quit.png"),tr("&Quit"), this);
    quitAct->setShortcuts(QKeySequence::Quit);
    quitAct->setStatusTip("Quit the application");
    connect(quitAct, SIGNAL(triggered()),this,SLOT(quit()));

    aboutAct = new QAction(QIcon(":/images/about.png"),tr("&About"), this);
    aboutAct->setStatusTip(tr("Show the application's About box"));
    connect(aboutAct, SIGNAL(triggered()), this, SLOT(about()));

    aboutQtAct = new QAction(QIcon(":/images/qtabout.png"),tr("About &Qt"),
this);
    aboutQtAct->setStatusTip(tr("Show the Qt library's About box"));
    connect(aboutQtAct, SIGNAL(triggered()), qApp, SLOT(aboutQt()));

    // disabled cut, copy buttons in menu. will be enabled when there is
selected text to cut or copy.
    cutAct->setEnabled(false);
    copyAct->setEnabled(false);
    connect(textEdit, SIGNAL(copyAvailable(bool)), cutAct,
SLOT(setEnabled(bool)));
    connect(textEdit, SIGNAL(copyAvailable(bool)), copyAct,
SLOT(setEnabled(bool)));

}

void MainWindow::createMenus()
```

```
{
    fileMenu = menuBar()->addMenu(tr("&File"));
    fileMenu->addAction(newFileAct);
    fileMenu->addAction(openAct);
    fileMenu->addAction(saveAct);
    fileMenu->addAction(saveAsAct);
    fileMenu->addSeparator();
    fileMenu->addAction(quitAct);

    editMenu = menuBar()->addMenu(tr("&Edit"));
    editMenu->addAction(cutAct);
    editMenu->addAction(copyAct);
    editMenu->addAction(pasteAct);
    editMenu->addSeparator();
    editMenu->addAction(undoAct);
    editMenu->addAction(redoAct);

    helpMenu = menuBar()->addMenu(tr("&Help"));
    helpMenu->addAction(aboutAct);
    helpMenu->addAction(aboutQtAct);

}


void MainWindow::createToolBars()
{
    //create a toolBar
    fileToolBar = addToolBar(tr("File"));
    //add the same action in the toolbar.
    fileToolBar->addAction(newFileAct);
    fileToolBar->addAction(openAct);
    fileToolBar->addAction(saveAct);

    editToolBar = addToolBar("Edit");
    editToolBar->addAction(cutAct);
    editToolBar->addAction(copyAct);
    editToolBar->addAction(pasteAct);
    editToolBar->addSeparator();
    editToolBar->addAction(undoAct);
    editToolBar->addAction(redoAct);

}


void MainWindow::createStatusBar()
{
    statusBar()->showMessage(tr(""));
}

// Function to check if file is modified and inform user.

bool MainWindow::okToContinue()
{
    /* if (textEdit->document()->isModified())
  {
   int r = QMessageBox::warning(this, tr("Text File"),
           tr("The document has been modified.\n"
```

```
            "Do you want to save your changes?"),
            QMessageBox::Yes | QMessageBox::Default,
            QMessageBox::No,
            QMessageBox::Cancel | QMessageBox::Escape);
    if (r == QMessageBox::Yes)
    {
       return save();
    }
    else if (r == QMessageBox::Cancel)
    {
       return false;
    }
   }*/
     return true;
}


// Gets the filename to be displayed in status bar from full file path .

QString MainWindow::strippedName(const QString &fullFileName)
{
    return QFileInfo(fullFileName).fileName();
}


void MainWindow::setCurrentFile(const QString &fileName)
{
    curFile = fileName;
    //textEdit->document()->setModified(false);
    QString shownName = "Untitled";
    if (!curFile.isEmpty())
    {
        shownName = strippedName(curFile);
    }

    setWindowFilePath(shownName);
}

void MainWindow::newFile()
{
    if (okToContinue())
    {
        textEdit->clear();
        setCurrentFile("");
    }
}


void MainWindow::parseSSMFile(QTextStream& stream)
{
    int noOfSamples = 0;
    unsigned int maxValue = 0;
    QString str;
    QStringList ls1;
    str = stream.readLine();
    lineNo = 1;
    if (str.size() > 0)
```

```
    {
        if (!(str.at(0)=='S' && str.at(1)== '2'))
        {
            QMessageBox msgbox;
            msgbox.setText("This is not SSM file");
            msgbox.exec();

        }
        else
        {
            while(!stream.atEnd())
            {
                str = stream.readLine();
                lineNo += 1;
                if (!(str.startsWith('#') ))
                {
                    ls1  =  str.split(' ', QString::SkipEmptyParts);
                    if ((noOfSamples == 0) || (maxValue == 0))
                    {
                        noOfSamples =  ls1.at(0).toInt();
                        globj1->noOfSamples = noOfSamples;
                        maxValue = ls1.at(1).toInt();
                        globj1->ssmMaxVal = maxValue;
                    }
                    else
                    {
                        firstSample = ls1.at(0).toInt();
                        globj1->samplesg = ls1;
                    }
                }
            }
        }
        ssmFileOpened = true;
    }
    else
    {
        QMessageBox msgBox;
        msgBox.setText("SSM file is empty!");
        msgBox.exec();
    }
}


void MainWindow::parsePPMFile(QTextStream& stream)
{
    imageWidth = 0; imageHeight = 0; imageMaxVal = -1;
    QString str;
    QStringList ls;
    str = stream.readLine();
    lineNo = 1;
    if (str.size() > 0)
    {
        if (!(str.at(0)=='P' && str.at(1)== '3'))
        {
            QMessageBox msgbox;
            msgbox.setText("This is not PPM file");
            msgbox.exec();
```

```
        }
        else
        {
            str = stream.readLine();
            int counter = 0, wh = 0;
            while(str.startsWith('#'))
            {
                str = stream.readLine();
                lineNo += 1;
            }

            if (!(str.startsWith('#') ))
            {
                ls  =  str.split(' ');
                if ((imageWidth == 0) || imageHeight == 0)
                {
                    imageWidth =  ls.at(0).toInt();
                    imageHeight = ls.at(1).toInt();
                }
            }
            str = stream.readLine();
            ls = str.split(' ');
            if(imageMaxVal == -1)
            {
                imageMaxVal = ls.at(0).toInt();
            }
            wh = imageWidth * imageHeight;
            image= (GLuint*) malloc(3*sizeof(GLuint)*wh);
            int i = 0;
            while (!stream.atEnd())
            {
                str = stream.readLine();
                ls = str.split(' ',QString::SkipEmptyParts);
                counter = 0;
                while(counter < ls.count())
                {
                    image[3*wh-3*i-3] = ls.at(counter).toInt();
                    image[3*wh-3*i-2] = ls.at(counter+1).toInt();
                    image[3*wh-3*i-1] = ls.at(counter+2).toInt();
                    counter = counter + 3;
                    i = i + 1;
                }

            }
        }
        ppmFileOpened = true;
    }
    else
    {
        QMessageBox msgBox;
        msgBox.setText("PPM file is empty!");
        msgBox.exec();
    }
}
```

```
void MainWindow::open()
{
    if (okToContinue())
    { // displays open file dialog with only text files and gets filename.
        QString fileName = QFileDialog::getOpenFileName(this,tr("Open
File"),"",tr("All Files(*.*)"));
        QFile file( fileName ); // Read the text from a file
        if ( !fileName.isEmpty() && file.open(
QIODevice::ReadOnly|QIODevice::Text ) )
        {

            QTextStream stream(&file );
            QFileInfo fi(file);
            QString ext = fi.suffix();
            if (ext == "ssm")
                parseSSMFile(stream);
            if(ext == "ppm")
                parsePPMFile(stream);

            //textEdit->setPlainText( stream.readAll() );

        }
        // keeps track of current file opened so that during saving we have
filename.
        setCurrentFile(fileName);
        statusBar()->showMessage(tr("File Opened"),2000);

    }
}

// Function to save the file. if currently opened file in text editor is
modified then saveFile is called.
// if it is new file then saveas is called to get filename.
bool MainWindow::save()
{
    if (curFile.isEmpty()) {
        return saveAs();
    } else {
        return saveFile(curFile);
    }
}

//  Function to save file when filename is known.
bool MainWindow::saveFile(const QString &fileName)
{
    QFile file(fileName);
    QFileInfo fi(file);
    QString ext = fi.suffix();

    if (ext == "ppm")
        globj1->PPMWrite(fileName);

    setCurrentFile(fileName);

    // update status bar
    statusBar()->showMessage(tr("File saved"), 2000);
```

```
    return true;
}

// called when file is saved with new name or a new file is saved. Displays a
dialog box to
// get filename and location of file
bool MainWindow::saveAs()
{
    QString fileName = QFileDialog::getSaveFileName(this,tr("Save
File"),"",tr("All Files(*.*)"));

    // if user pressed cancel so filename is empty
    if (fileName.isEmpty())
        return false;

    return saveFile(fileName);
}

// cuts selected text in text editor
void MainWindow::cut()
{
    textEdit->cut();
}

// copies selected text from text editor to clipboard
void MainWindow::copy()
{
    textEdit->copy();
}

// pastes text from clipboard to text editor.
void MainWindow::paste()
{
    if (textEdit->canPaste())
        textEdit->paste();
}

// undo the recent changes in text editor
void MainWindow::undo()
{
    QTextDocument *document = textEdit->document();
    document->undo();
}

// redo the recent changes in text editor
void MainWindow::redo()
{
    QTextDocument *document = textEdit->document();
    document->redo();
}
void MainWindow::quit()
{
    QApplication::quit();
}

void MainWindow::about()
{
```

```
    QMessageBox::about(this, tr("About this Assignment"),
                       tr("This assignment is about\n connecting all widgets
in Qt to graphics window."));
}

void MainWindow::aboutQt()
{

}
```