1. About the contribution

Wireless photo capturing rover is definitely a good device to capture object with the angle that human might not be able to capture, thus adding convenience within photo capturing process before using paranomic software, industrial commercial software to turn the object in 2D photos into the any wide-angle view or representation of a physical space, whether in painting, drawing, photography, or a three-dimensional model.

Regardless the future paranomic software will be free or not, relying on industrial software away from the group is not a good spirit for a university senior design. In addition, the photo that is taken from the rover might not be in a good condition to be data sample. For example, if the object is hanged, we must need to be flipped in the photo; if the environment does not have enough light source, we must need to increase the brightness or contrast of the photo; if paranomic software loads both JPG and PNG file type, the software must need to able handle our photo file type.

As most of my teammates were busy on hardware coding, paranomic software had gone through several updates, for my teammates worked for a conference or a publication was not the goal. To ensure the paranomic rendering is reach to the best level as possible, I assisted the team to create a photo processing software, to give helpful supports to create the best paranomic rendering as possible.

My developed software was written in JAVA in which it handles all of the image post-processes. Even though with all the project difficulties, some basic functions are implemented into the software, which include most of the photo processing methods, for example, horizontal flip, vertical flip, grey scale, contrast, brightness calibration, and a lot of functions to assist the team to calibrate the photo. The software was able to load multiple photo, and proceed photo calibration for all of them within a single click. Due to the scope of my works, some normal file management functionalities are also implemented to the software, for example, loading, saving.etc. All functionalities listed were displayed in a GUI fashion. In general, members can simply click on the button of my software to process the team picture, since all the methods were incorporated with the intuitive designs which add another level of convenience to the team to handle photo that was taken under bad conditions or was not in its optimized state.

At the end, all of the code that I engineered helped the team to create the best paranomic rendering as possible with the main focus of handling all of the image calibrations, and image file managements within all the personal computer operating system. Beside adding the photo processing functionalities, and heavy duty coding in GUI, I also expanded the amount functionalities besides the fundamental in my software package.

In regarding the senior design paper, I have written this software contribution report section. With the permission from Undergraduate Director, Prof. Kamoua and the meetings with Prof. Mikhail Gouzman, and all the members, till the end of the semester, all of the developed codes could be used for the class ESE441, report section were sent to Project Director, Prof. Mikhail Gouzman, ESE440,441 Lecturer, Prof. Wendy Tang.

2. Overview of the Post-Photo Processing software

Java API Usage

Both the framework and the mini-game application will be developed using the Java programming languages. As such, this design will make use of the classes specified in Figure 2.2.

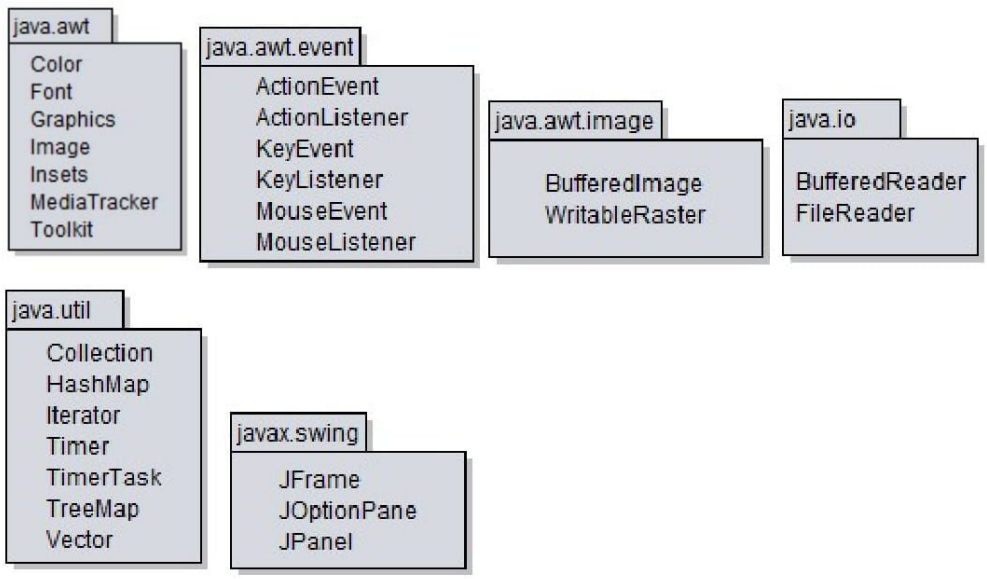


Figure 2.1: Java API Classes and Packages To Be Used

Class/Interface	Use
ActionEvent	For getting information about an action event like which button was pressed.
ActionListener	For responding to an action event, like a button press. We will provide our own custom implementation of this interface.

Table 2.2: Uses for classes in the Java API's java.awt.event package

Class/Interface	Use
BufferedReader	For reading text files, we'll use this for loading some game data at startup.
FileReader	For reading files.

Table 2.3: Uses for classes in the Java API's java.io package

Class/Interface	Use
JFrame	Provides the window for our GUI.
JOptionPane	Provides a popup dialog for error feedback.
JPanel	Provides a canvas for the game to be rendered onto.

Table 2.4: Uses for classes in the Java API's javax.swing package

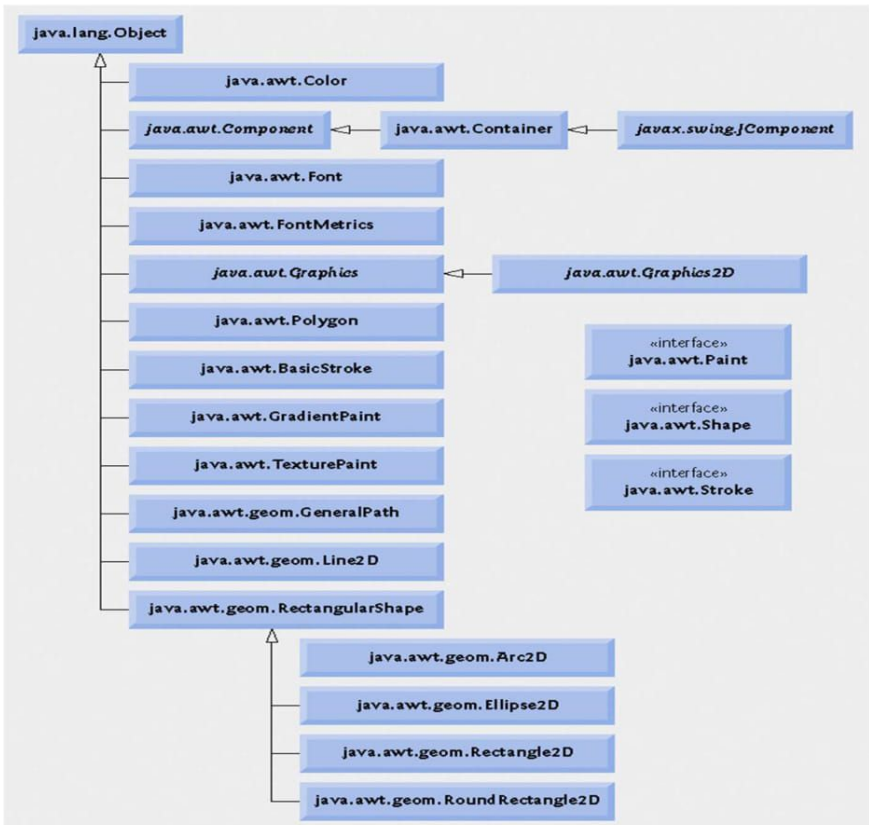
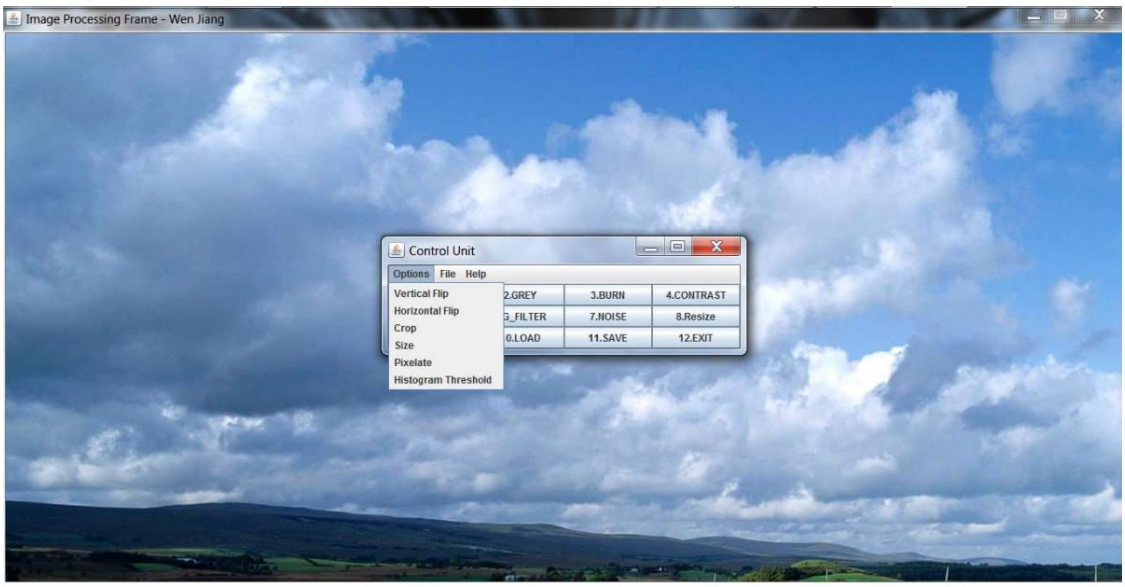


Fig. 2.5 Classes and interfaces used in this chapter from Java's original graphics capabilities and from the Java 2D API. [*Note:* Class Object appears here because it is the superclass of the Java class hierarchy.]

000 blanc White	255	255	255	FFFFFF
208 Lavender-VY DK	148	91	128	945B80
209 Lavender-DK	206	148	186	CE94BA
210 Lavender-MD	236	207	225	ECCFE1
211 Lavender-LT	243	218	228	F3DAE4
221 Shell Pink-VY DK	156	41	74	9C294A
223 Shell Pink-LT	219	128	115	DB8073
224 Shell Pink-VY LT	255	199	176	FFC7B0
225 Shell Pink-ULT VY L	255	240	228	FFF0E4
300 Mahogany-VY DK	143	57	38	8F3926
301 Mahogany-MD	209	102	84	D16654
304 Christmas Red-MD	188	0	97	BC0061
. . .				
975 Golden Brown-DK	158	67	18	9E4312
976 Golden Brown-MD	246	141	57	F68D39
977 Golden Brown-LT	255	164	73	FFA449
986 Forest Green-VY DK	58	82	65	3A5241
987 Forest Green-DK	83	97	73	536149
988 Forest Green-MD	134	145	110	86916E
989 Forest Green	134	153	110	86996E
991 Aquamarine-DK	47	91	73	2F5B49
3740 Antique Violet-DK	156	125	133	9C7D85
3743 Antique Violet-VY L	235	235	231	EBEBE7
3746 Blue Violet-DK	149	102	162	9566A2
3747 Blue Violet-VY LT	230	236	232	E6ECE8
3750 Antique Blue-VY DK	12	91	108	0C5B6C
3752 Antique Blue-VY LT	194	209	206	C2D1CE
3753 Ant. Blue-ULT VY LT	237	247	247	EDF7F7
3755 Baby Blue	158	176	206	9EB0CE
3756 Baby Blue-ULT VY LT	248	248	252	F8F8FC
3760 Wedgewood	102	142	152	668E98
3761 Sky Blue-LT	227	234	230	E3EAE6
3765 Peacock Blue-VY DK	24	128	134	188086
. . .				

Fig. 3 | Some Color constants and their RGB values.

4. Screen Capture of the Final Software



5. Background of Photo Processing

Vertical Flip: The Flip Vertically command reverses the active layer vertically, that is, from top to bottom. It leaves the dimensions of the layer and the pixel information unchanged.

Horizontal Flip: The Flip Horizontally command reverses the active layer horizontally, that is, from left to right. It leaves the dimensions of the layer and the pixel information unchanged.

Brighten: The brightness of an image can be adjusted by performing a linear traversal of the one-dimensional array of pixel intensities adding the same value to the intensity of every pixel. Adding a positive value will make the image brighter and adding a negative value will make the image darker. In the graphic below, the example image was brightened by adding 100 to the intensity of every pixel. Notice that there is still very little variation in the pixel intensities.

Pixelate: In computer graphics, pixelation is caused by displaying a bitmap or a section of a bitmap at such a large size that individual pixels, small single-colored square display elements that comprise the bitmap, are visible. Such an image is said to be pixelated.

Noise: Image noise is random (not present in the object imaged) variation of brightness or color information in images, and is usually an aspect of electronic noise. It can be produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of image capture that adds spurious and extraneous information.

Invert: Sometimes refer to revert. A positive image is a normal image. A negative image is a total inversion, in which light areas appear dark and vice versa. A negative color image is additionally color-reversed, with red areas appearing cyan, greens appearing magenta and blues appearing yellow. Film negatives usually have less contrast, but a wider dynamic range, than the final printed positive images. The contrast typically increases when they are printed onto photographic paper. When negative film images are brought into the digital realm, their contrast may be adjusted at the time of scanning or, more usually, during subsequent post-processing.

Histogram thresholding: Histogram thresholding is a technique used to separate objects from the background, it is not always possible to do this, especially if the background has similar colors or grey scale as the objects. Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary image.

Cropping: Cropping refers to the removal of the outer parts of an image to improve framing, accentuate subject matter or change aspect ratio. Depending on the application, this may be performed on a physical photograph, artwork or film footage, or achieved digitally using image editing software. The term is common to the film, broadcasting, photographic, graphic design and printing industries.

Burned: An image is said to be burned when its original gamut considerably exceeds the target gamut, or when the result of processing considerably exceeds the image's gamut, resulting in clipping. Colloquially, an image is burned when it contains uniform blobs of color, black, or white where there should actually be detail. While burned images in color are typically not pleasing and need to be avoided, black and white photographs can sometimes be enhanced artistically by burning them; the decision to burn, along with the degree of burning is a subjective matter.

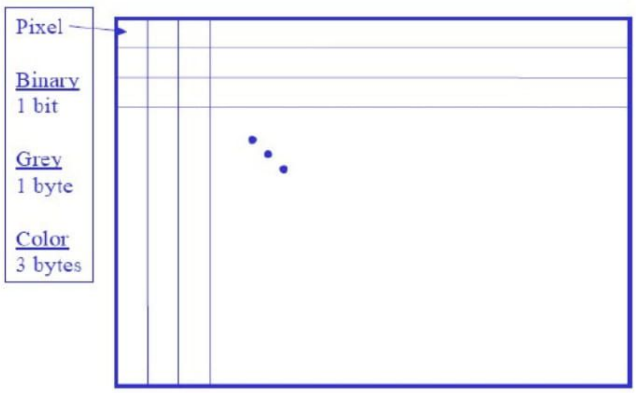
Gaussian Filter: Sometime refer to A Gaussian blur (also known as Gaussian smoothing) which is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales—see scale space representation and scale space implementation.

Grayscale: In photography and computing, a grayscale or greyscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bilevel or binary images). Grayscale images have many shades of gray

in between. Grayscale images are also called monochromatic, denoting the presence of only one (mono) color (chrome).

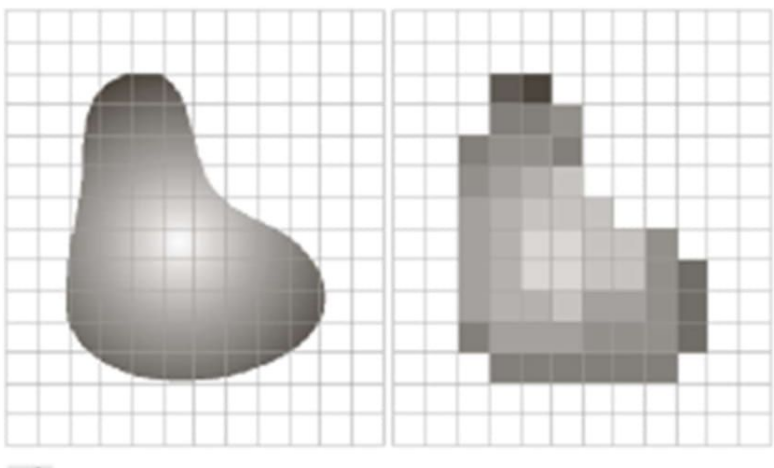
6. Analysis and Discussion

Image Representation - Digital Images are 2D arrays (matrices) of numbers. An image consists of a rectangular array of dots called pixels. The size of the image is usually specified as width X height (in numbers of pixels).



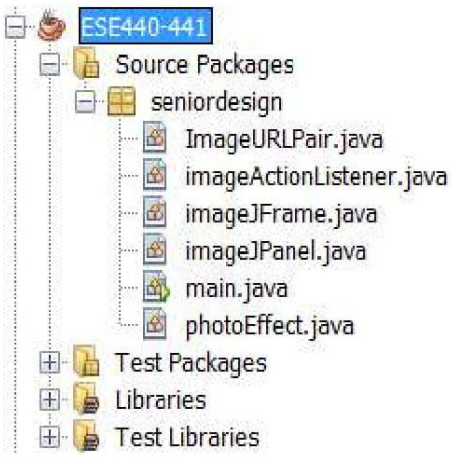
The physical size of the image, in inches or centimeters or whatever, depends on the resolution of the device on which the image is displayed. Resolution is usually measured in terms of DPI, which stands for dots per inch.

We can then have digitalization of images, i.e. sampling corresponds to a discretization of the space. Quantization corresponds to a discretization of the intensity values. In Java 2D coordinate system scheme for identifying every point on screen, the upper left corner of GUI component has coordinates (0,0)

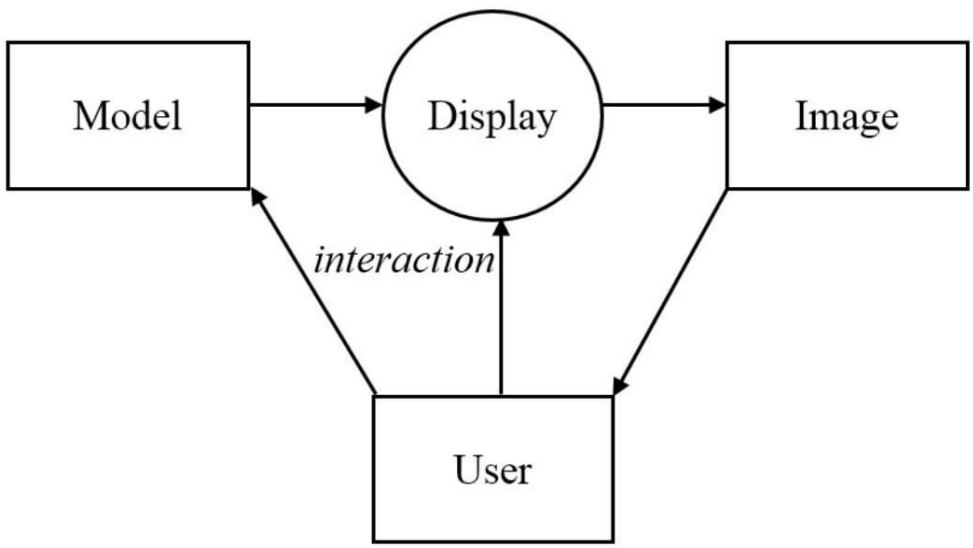


Java 2D API provides advanced 2D graphics capabilities part of the Java 2 Platform, Standard Edition includes features for processing line art, text and images. The class

java.awt.Graphics2D enables drawing with Java 2D API offers three graphics primitives: images, text and shapes java.awt.Graphics subclass contains seven state attributes clipping, compositing, font, paint, rendering hints, stroke and transforms.



Above is the file structure and formats. Note that all necessary data and art files must accompany this program. Above figure specifies the necessary file structure the launched application should use. Note that all necessary images can be loaded from any image directory.



For the software, I deployed the above schematic. The interaction will be determined by user's preference, which include all of the photo processing techniques and management of the files. For more detail of the methodology on the photo processing, please see the comments I provided along with the JAVA code.

During this senior design project, we had exposed to project development and technical difficulties that were in the different aspect of the project. As a head of software development, photo-post processing methodologies of various imaging technique were studied and utilized. The calibration on processed photo were further understood. I could expose these techniques more in the detail with my team members, however I could not achieve that objectives due to the time constraints on the others.

On the software development, I added the normal data management and the viewing effect and have gathered up the GUI functionalities to let users i.e. team members to have a centralized photo processing center. We could then easily to manage our photo all in a single click. During sensitive photo calibration, for example to increase the brightness and contrast of the photo, all team members also could input adjustable amount to my software. For more information on the coding technicality, please see the codes and comment provided in the coding section.

For the demand within the menu, only a single click is needed. The exploration on different photo processing techniques, reveled the optimum aspect of the photo. Given one or multiple photo files, I understood the implementation of software image processing technique could provide on output photo. The developed software could handle JPG, JPEG, and PNG. GIF must be a single frame photo for this software to respond.

I had sacrificed a lot of time and effort on this project. At the end, I am still thankful for the permissions and acknowledges from the people that I received. Apart from having skills to solve the technicality within my contribution, since there a lot of management issue with the this group, thus, this senior design is still quite useful for me to get involved in the project development and to prepare myself to handle the situations that arisen in the future.

7. Supporting Information

Note that this document should serve as a reference for those implementing the code, so I provided a table of contents and javadoc comment to help quickly find important sections.
SeniorDesign\dist\javadoc\index.html

7.1. Table of Content

7.2. Reference

K.M.M, Rao. "INTRODUCTION_TO_IMAGE_PROCESSING_29aug06.pdf." Readings in Image Processing. Web. 25 Apr 2013.

<http://www.drkmm.com/resources/INTRODUCTION_TO_IMAGE_PROCESSING_29aug06.pdf>.

Wikipedia. Web. 20 May 2013.

<http://en.wikipedia.org/wiki/Category:Image_processing>.

GIMP. Web. 20 May 2013. <http://docs.gimp.org/>.

Smbt – "A smart ambient utilities framework ." . Eclipse. Web. 25 Apr 2013.

<http://ubqt.eclipselabs.org.codespot.com/svn-history/r6900/trunk/net.sf.smbt.ui.widgets/src-widgets/net/sf/smbt/ui/widgets/common/ImageUtils.java>.

An-Najah National University. Web. 22 May 2013.

<http://elearning.najah.edu/OldData/pdfs/Image%20Processing%20basics%20and%20Java.ppt>

7.3. Appendixes

==========
==========

```
package seniordesign;

/*
 * Contribution: Image Post-Processes Software
 *
 * Course: ESE440,441
 * Author: Wen Jiang
 * Email:wenjiang@live.com
 *
 * As a head of OS software of senior design, I help the team to assist the best paranomic
rendering as possible
 * with the main focus of handling all of the image calibrations, and image file
managements in our common OS.
 * All of the codes and reports can be used for the ESE440-ESE441 Senior Designs of Stony
Brook University.
 */

import java.awt.image.BufferedImage;

/**
 *
 * @author WenJ
 */
public class ImageURLPair {

    /**
     *
     */
    public BufferedImage image;
    /**
     *
     */
    public String url;
    /**
     *
     */
    public BufferedImage backstack;

    /**
     *
     * @param image
     * @param url
     */
    public ImageURLPair(BufferedImage image, String url) {
```

```
        this.image = image;
        this.url = url;
    }
}
```

==========

==========

```
/*
 * To implemented the function in detail to handler to handles different photoeffect on
our photo.
 */

package seniordesign;

/*
 * Contribution: Image Post-Processes Software
 *
 * Course: ESE440,441
 * Author: Wen Jiang
 * Email:wenjiang@live.com
 *
 * As a head of OS software of senior design, I help the team to assist the best paranomic
rendering as possible
 * with the main focus of handling all of the image calibrations, and image file
managements in our common OS.
 * All of the codes and reports can be used for the ESE440-ESE441 Senior Designs of Stony
Brook University.
 */

import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import javax.imageio.ImageIO;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

/**
 *
 * @author WenJ
 */
public class imageActionListener implements ActionListener {

    JFrame frame;
    imageJPanel imagePanel;

    /**
```

```
     *
     * @param f
     * @param ijp
     */
    public imageActionListener(JFrame f, imageJPanel ijp) {
        frame = f;
        imagePanel = ijp;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (!e.getActionCommand().equals("undo")) {
            imagePanel.createBackstack();
        }
        switch (e.getActionCommand()) {
            case "EXIT":
                System.out.println("Thank you for using the post-photoprocessing
software");
                System.exit(0);
                break;
            case "ABOUT":
                JOptionPane.showMessageDialog(frame,
                        "Contribution: Image Post-Processes Software\n" + "Course:
ESE440,441\n"
                        + "Author: Wen Jiang\n"
                        + "Email:wenjiang@live.com\n\n"
                        + "As a head of OS software of senior design, I help the team to
assist the best paranomic rendering as possible\n"
                        + "with the main focus of handling all of the image calibrations,
and image file managements in our common OS.\n\n"
                        + "All of the codes and reports can be used for the ESE440-ESE441
Senior Designs of Stony Brook University.");
                break;
            case "vFlip":
                this.imagePanel.flipVertical();
                break;
            case "size":
                this.imagePanel.getPictureSize();
                break;
            case "LOADMOVIE":
                this.imagePanel.loadMovie();
                break;
            case "SAVE":
                this.imagePanel.saveImages();
                break;
            case "LOAD":
                //String dir = JOptionPane.showInputDialog("Input a directory
ex.C:\\Users\\WenJ\\Documents\\NetBeansProjects\\SeniorDesign\\sample.jpg");
                //File imageFile = new File(dir);
                JFileChooser fc = new JFileChooser();
                fc.setMultiSelectionEnabled(true);
                if (fc.showOpenDialog(this.imagePanel) == JFileChooser.APPROVE_OPTION) {
```

```
                    for (File f : fc.getSelectedFiles()) {
                        this.imagePanel.loadImageFromBar(f);
                    }
                }
                imagePanel.activeButtons();
                break;
            case "BRIGHTEN":
                this.imagePanel.brightenImage();
                break;
            case "hFlip":
                this.imagePanel.flipHorizontal();
                break;
            case "pixelate":
                this.imagePanel.pixelate();
                break;
            case "noise":
                this.imagePanel.noise();
                break;
            case "invert":
                this.imagePanel.invert();
                break;
            case "histogramThreshold":
                this.imagePanel.histogramThreshold();
                break;
            case "gaussian":
                this.imagePanel.gaussianFilter();
                break;
            case "burn":
                this.imagePanel.burn();
                break;
            case "greyscale":
                this.imagePanel.greyScale();
                break;
            case "crop":
                this.imagePanel.crop();
                break;
            case "undo":
                this.imagePanel.undo();
                break;
            case "contrast":
                this.imagePanel.contrast();
                break;
            case "resize":
                this.imagePanel.resize();
                break;
        }
    }
}
```

==========
==========

```
/*
 * Basic JFrame implementation that set up the frame of the windows.
 */
package seniordesign;

/*
 * Contribution: Image Post-Processes Software
 *
 * Course: ESE440,441
 * Author: Wen Jiang
 * Email:wenjiang@live.com
 *
 * As a head of OS software of senior design, I help the team to assist the best paranomic
rendering as possible
 * with the main focus of handling all of the image calibrations, and image file
managements in our common OS.
 * All of the codes and reports can be used for the ESE440-ESE441 Senior Designs of Stony
Brook University.
 */

import java.awt.Graphics;
import java.awt.Image;
import javax.swing.JFrame;
import java.io.File;
import javax.swing.*;

/**
 *
 * @author WenJ
 */
public class imageJFrame extends JFrame {

    /**
     *
     */
    public imageJPanel panel;
    private Image image;

    /**
     *
     * @param image
     */
    public void setImage(Image image) {
        this.image = image;
    }

    @Override
    public void paintComponents(Graphics g) {
        super.paintComponents(g);
```

```
        g.drawImage(image, 0, 0, null);
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        g.drawImage(image, 0, 0, null);
    }

    /**
     * Basic Constructor Sets up a JFrame and load the image
     * @param imageFile
     */
    public imageJFrame(File imageFile) {
        super("Image Processing Frame - Wen Jiang");
        panel = new imageJPanel(imageFile, this);

        getContentPane().add(panel);
        setSize(1366, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setResizable(false);
        setVisible(true);

        panel.launchFrame();
        this.setImage(panel.image);
        this.paintComponents(this.getGraphics());
    }
}
```

==========

==========

```
/*
 * To add action to our listener from mouse click, in which the switch function
 * is implemented to handles different photoeffect on our photo inquery.
 */
package seniordesign;

/*
 * Contribution: Image Post-Processes Software
 *
 * Course: ESE440,441
 * Author: Wen Jiang
 * Email:wenjiang@live.com
 *
 * As a head of OS software of senior design, I help the team to assist the best paranomic
rendering as possible
 * with the main focus of handling all of the image calibrations, and image file
managements in our common OS.
```

```
 * All of the codes and reports can be used for the ESE440-ESE441 Senior Designs of Stony
Brook University.
 */

import javax.media.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import javax.swing.JPanel;
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import javax.imageio.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.RenderedImage;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.media.control.FrameGrabbingControl;
import seniordesign.imageActionListener;
import sun.awt.image.ToolkitImage;

/**
 *
 * @author WenJ
 */
public class imageJPanel extends JPanel {

    private imageJFrame parent;
    private JFrame f = new JFrame("Basic GUI");
    private java.util.List<ImageURLPair> images = new ArrayList<ImageURLPair>();
    //Panel
    private JPanel pnlNorth = new JPanel();
    private JPanel pnlSouth = new JPanel();
    private JPanel pnlEast = new JPanel();
    private JPanel pnlWest = new JPanel();
    private JPanel pnlCenter = new JPanel();
    private JPanel pnlExtra = new JPanel();
    //Button
    private JButton btnNorth = new JButton("North");
    private JButton btnSouth = new JButton("South");
    private JButton btnEast = new JButton("East");
    private JButton btnWest = new JButton("West");
    private JButton btnCenter = new JButton("Center");
    private JButton btnExtra = new JButton("extra");
    //Menu
    private JMenuBar mb = new JMenuBar();
    private JMenu mnuFile = new JMenu("File");
    private JMenu options = new JMenu("Options");
```

```
private JMenu mnuHelp = new JMenu("Help");
private JMenuItem mnuItemQuit = new JMenuItem("Quit");
private JMenuItem mnuItemSave = new JMenuItem("SAVE");
private JMenuItem mnuItemLoad = new JMenuItem("Load");
private JMenuItem size = new JMenuItem("Size");
private JMenuItem mnuItemLoadMovie = new JMenuItem("Load Movie");
private JMenuItem mnuItemAbout = new JMenuItem("About");
private JMenuItem verticalFlip = new JMenuItem("Vertical Flip");
private JMenuItem horizontalFlip = new JMenuItem("Horizontal Flip");
private FrameGrabbingControl frameGrabber;
private Player player;
private JMenuItem pixelate = new JMenuItem("Pixelate");
private JMenuItem crop = new JMenuItem("Crop");
private JMenuItem histogramThreshold = new JMenuItem("Histogram Threshold");
// Create two images, backimage is the one that has not go throught the new effect,
// but image has new effect.
/**
 *
 */
public BufferedImage image;
private BufferedImage backimage;
private Component frame;

/**
 *
 * @param imageFile
 * @param parent
 */
public imageJPanel(File imageFile, imageJFrame parent) {
    //create Frame
    f = new JFrame("Control Unit");
    this.parent = parent;
    // North quadrant
    pnlNorth = new JPanel();
    pnlNorth.setLayout(new GridLayout(1, 3));
    // South quadrant
    pnlSouth = new JPanel();
    pnlSouth.setLayout(new GridLayout(1, 3));
    // East quadrant
    pnlEast = new JPanel();
    pnlEast.setLayout(new GridLayout(1, 3));
    // West quadrant
    pnlWest = new JPanel();
    pnlWest.setLayout(new GridLayout(1, 3));
    // Center quadrant
    pnlCenter = new JPanel();
    pnlCenter.setLayout(new GridLayout(1, 3));

    pnlExtra = new JPanel();
    pnlExtra.setLayout(new BoxLayout(pnlExtra, BoxLayout.X_AXIS));
    // menu
    mb = new JMenuBar();
```

```
mnuFile = new JMenu("File");
imageActionListener al = new imageActionListener(this.f, this);
mnuItemSave = new JMenuItem("Save File");
mnuItemSave.setActionCommand("SAVE");
mnuItemSave.addActionListener(al);
mnuItemLoad = new JMenuItem("Load");
mnuItemLoad.setActionCommand("LOAD");
mnuItemLoad.addActionListener(al);
mnuItemLoadMovie.setActionCommand("LOADMOVIE");
mnuItemLoadMovie.addActionListener(al);
mnuItemQuit = new JMenuItem("Quit");
mnuItemQuit.setActionCommand("EXIT");
mnuItemQuit.addActionListener(al);
mnuHelp = new JMenu("Help");
mnuItemAbout = new JMenuItem("About");
mnuItemAbout.setActionCommand("ABOUT");
mnuItemAbout.addActionListener(al);
options = new JMenu("Options");

// method
verticalFlip = new JMenuItem("Vertical Flip");
verticalFlip.setActionCommand("vFlip");
verticalFlip.addActionListener(al);
horizontalFlip = new JMenuItem("Horizontal Flip");
horizontalFlip.setActionCommand("hFlip");
horizontalFlip.addActionListener(al);
size = new JMenuItem("Size");
size.setActionCommand("size");
size.addActionListener(al);

crop = new JMenuItem("Crop");
crop.setActionCommand("crop");
crop.addActionListener(al);
pixelate = new JMenuItem("Pixelate");
pixelate.setActionCommand("pixelate");
pixelate.addActionListener(al);
histogramThreshold = new JMenuItem("Histogram Threshold");
histogramThreshold.setActionCommand("histogramThreshold");
histogramThreshold.addActionListener(al);

// add methods to our options
options.add(verticalFlip);
options.add(horizontalFlip);
options.add(crop);
options.add(size);
options.add(pixelate);
options.add(histogramThreshold);

mb.add(options);
mnuFile.add(mnuItemLoadMovie);
mnuFile.add(mnuItemLoad);
mnuFile.add(mnuItemSave);
```

```
mnuFile.add(mnuItemQuit);
mnuHelp.add(mnuItemAbout);
mb.add(mnuFile);
mb.add(mnuHelp);
options.setEnabled(false);

// add button function to our button in our quadrants
btnNorth = new JButton("1.BRIGHTEN");
btnNorth.setActionCommand("BRIGHTEN");
btnNorth.addActionListener(al);
btnNorth.setEnabled(false);
pnlNorth.add(btnNorth);

btnNorth = new JButton("2.GREY");
btnNorth.setActionCommand("greyscale");
btnNorth.addActionListener(al);
btnNorth.setEnabled(false);
pnlNorth.add(btnNorth);

btnNorth = new JButton("3.BURN");
btnNorth.setActionCommand("burn");
btnNorth.addActionListener(al);
btnNorth.setEnabled(false);
pnlNorth.add(btnNorth);

btnNorth = new JButton("4.CONTRAST");
btnNorth.setActionCommand("contrast");
btnNorth.addActionListener(al);
btnNorth.setEnabled(false);
pnlNorth.add(btnNorth);

btnCenter = new JButton("5.INVERT");
btnCenter.setActionCommand("invert");
btnCenter.addActionListener(al);
btnCenter.setEnabled(false);

pnlCenter.add(btnCenter);

btnCenter = new JButton("6.G_FILTER");
btnCenter.setActionCommand("gaussian");
btnCenter.addActionListener(al);
btnCenter.setEnabled(false);
pnlCenter.add(btnCenter);
btnCenter = new JButton("7.NOISE");
btnCenter.addActionListener(al);
btnCenter.setActionCommand("noise");
btnCenter.setEnabled(false);
pnlCenter.add(btnCenter);
btnSouth = new JButton("9.UNDO PRE");
btnSouth.setActionCommand("undo");
btnSouth.addActionListener(al);
btnSouth.setEnabled(false);
```