Sahar K Hussain
CAP3027
Section 1925
December 4th, 2015
Term Project - Iteration 4

The work submitted is my own and the honor code was neither bent nor broken.

_____ Sahar K Hussain_____

Summary of my learning experience:

The easiest part of the project task was to again to iteratively add things to what I wanted — almost like the painter's algorithm, adding in layers and building off my previous work. The most difficult parts of the task was to bring all the elements of the project together. I think adding new things like the affine transforms to add the effects to my program was something that really spiced up the program. I made the appropriate changes to the program in terms of feedback from peers and in office hours. I was glad I was able to touch all the points I wanted in my proposal — due to time constraint, illness, as previously discussed was factored in, (I think I would like to keep adding more to this project on my own personal time for things beyond stated intially) but I did the main points for my iteration which is what matters most, I think and I am quite satisfied with the functionality of it all.

Complete GUI Mockup: Added in the folder

Description of what the user can do and the visible effect that the action will have:

From discussed in my last iteration, I was successfully able to implement the basic painters and the fun painters on the top. As an added thing to my project, I made a Slider functional which allows the width/stroke to be changed when the user paints. I feel like this overall made the program a lot more successful. I made sure to add custom functionality and the images to the GUI for a nicer feel. The user is now also able to load and display an image into the program which is different from before as it was only loading. The save method also works too. I had a program where when the picture loaded, the entire GUI would shift but I solved it now by create a separate buffered image for the user to draw on top of which solves this program. I also added the fun filter effects. The user can sharpen, blur, or invoke edge detection on the image. (Zoom was cut out as already discussed) but I would like to implement it at a later time, which was the only thing I cut out (again due to above & documented reasons but I was able to do everything else) though I stated this in the proposal too if time permitted for me I would like to have done the zoom so it was a conscious decision on my part. I was able to get my fun custom fractal painters on the top working which moves when the user paints with it as well. The user can also draw in a randomized color that is interpolated. As another fun effect, I was able to add music to the program as well! Overall, I am quite happy with the program that I was able to flush out and hope to keep adding on to it even after the course!

Sahar K Hussain
CAP3027
Section 1925
November 20, 2015
Term Project - Iteration 4

Standard Program Deliverables:

1) Yes, the program compiles without errors.
2) Yes, the program compiles without warnings.
3) Yes, the program runs without crashing.
4) I tested the program on my personal laptop and utilized emacs for inputting the code and the terminal for compiling and running the program.
5) The program does meet assignment's specifications - N/A
6) No known and suspected bugs.
7) Yes, the program runs correctly.

Source Code:

```
// CAP 3027 Term Project by Sahar K Hussain - Fall 2015
// ArtBytes
// Not your standard Java Swing Paint program.
// ArtBytes is a custom GUI paint program that allows the user to paint with really cool effects
instead of the regular MS paint programs out there
// and has other cool features as well

// Sources: Calling my letsDrawCanvas class


// ============================== Imports the Packages
================================= //
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.*;
import javax.swing.*;
import java.awt.Color;
import java.awt.geom.*;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.*;
import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.imageio.*;
import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
```

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JComponent;
import java.awt.RenderingHints;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.awt.event.MouseAdapter;
import java.awt.BorderLayout;
import javax.swing.border.Border;
import javax.swing.border.TitledBorder;
import javax.swing.BorderFactory;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JFileChooser;
import java.io.File;
import java.io.IOException;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import java.awt.geom.*;
import java.lang.*;
import java.io.InputStream;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;

//
============================================================================
============== //

public class ArtBytes {

    private final int CANVAS_WIDTH = 1280;
    private final int CANVAS_HEIGHT = 720;
    private BufferedImage image = null;
    private ImageIcon displayImage;
    private JLabel newLabel;
    private Clip clip;
    public JSlider slider1 = new JSlider(JSlider.VERTICAL,0,50,25);
    LetsDrawCanvas letsDrawCanvas;

    // Upper Panel Buttons

    //Adding the images to the button icons
```

```
    JButton rainbowButton = new JButton(new ImageIcon("ArtBytesFiles/rainbow.png"));
    JButton prettyCirclesButton = new JButton(new ImageIcon("ArtBytesFiles/
prettyCircles.png"));
    JButton bubblesButton = new JButton(new ImageIcon("ArtBytesFiles/drawBubbles.png"));
    JButton starButton = new JButton(new ImageIcon("ArtBytesFiles/drawStar.png"));

    // Lower Panel Buttons


    //Adding the images to the button icons
    JButton blueButton = new JButton(new ImageIcon("ArtBytesFiles/blue.jpg"));
    JButton cyanButton = new JButton(new ImageIcon("ArtBytesFiles/cyan.jpg"));
    JButton greenButton = new JButton(new ImageIcon("ArtBytesFiles/green.png"));
    JButton lightGrayButton = new JButton(new ImageIcon("ArtBytesFiles/lightgray.png"));
    JButton magentaButton = new JButton(new ImageIcon("ArtBytesFiles/magenta.png"));
    JButton orangeButton = new JButton(new ImageIcon("ArtBytesFiles/orange.png"));
    JButton pinkButton = new JButton(new ImageIcon("ArtBytesFiles/pink.png"));
    JButton redButton = new JButton(new ImageIcon("ArtBytesFiles/red.png"));
    JButton whiteButton = new JButton(new ImageIcon("ArtBytesFiles/white.png"));
    JButton yellowButton = new JButton(new ImageIcon("ArtBytesFiles/yellow.png"));
    JButton randomColorButton = new JButton(new ImageIcon("ArtBytesFiles/random.png"));

    JButton eraserButton = new JButton(new ImageIcon("ArtBytesFiles/eraser.png"));
    JButton EmptyFrameButton = new JButton(new ImageIcon("ArtBytesFiles/leTrash.png"));

    JButton musicOnButton = new JButton(new ImageIcon("ArtBytesFiles/musicOnNote.png"));
    JButton musicOffButton = new JButton(new ImageIcon("ArtBytesFiles/musicOffNote.png"));

    // ============================== The Painters
================================= //

    // Rainbow Painter
    ActionListener rainbowListener = new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            if(e.getSource() == rainbowButton)
            {
            //System.out.println("pressed button");
            letsDrawCanvas.brushState = 'r';
                //letsDrawCanvas.drawRainbow();
            }
        }
    };

    // Prettycircles Painter
    ActionListener prettyListener = new ActionListener()
```

```
{
   public void actionPerformed(ActionEvent e)
   {
      if(e.getSource() == prettyCirclesButton)
      {
      //System.out.println("pressed button");
      letsDrawCanvas.brushState = 'p';
      }
   }
};

// drawBubbles Painter
ActionListener bubbleListener = new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      if(e.getSource() == bubblesButton)
      {
      //System.out.println("pressed button");
      letsDrawCanvas.brushState = 'b';
      }
   }
};

// pencilShave Painter
ActionListener starListener = new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      if(e.getSource() == starButton)
      {
      letsDrawCanvas.brushState = 's';
      }
   }
};

// Music on Player
ActionListener musicPlayListener = new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      if(e.getSource() == musicOnButton)
         {
         playSound();
         }
   }
};

// Turns off the music
```

```
ActionListener musicOffListener = new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      if(e.getSource() == musicOffButton)
      {
      stopMusic();
      }
   }
};

// Basic Painters
ActionListener actionListener = new ActionListener()
{

   public void actionPerformed(ActionEvent e)
   {
      if (e.getSource() == EmptyFrameButton)
      {
         letsDrawCanvas.brushState = 'd';
         letsDrawCanvas.emptyTheFrame();
      }
      else if (e.getSource() == blueButton)
      {
         letsDrawCanvas.brushState = 'd';
         letsDrawCanvas.drawBlue();
      }
      else if (e.getSource() == cyanButton)
      {
         letsDrawCanvas.brushState = 'd';
         letsDrawCanvas.drawCyan();
      }
      else if(e.getSource() == greenButton)
      {
         letsDrawCanvas.brushState = 'd';
         letsDrawCanvas.drawGreen();
      }
      else if(e.getSource() == lightGrayButton)
      {
         letsDrawCanvas.brushState = 'd';
         letsDrawCanvas.drawLight_gray();
      }
      else if (e.getSource() == magentaButton)
      {
         letsDrawCanvas.brushState = 'd';
         letsDrawCanvas.drawMagenta();
      }
      else if(e.getSource() == orangeButton)
      {
```

```
            letsDrawCanvas.brushState = 'd';
            letsDrawCanvas.drawOrange();
        }
        else if(e.getSource() == pinkButton)
        {
            letsDrawCanvas.brushState = 'd';
            letsDrawCanvas.drawPink();
        }
        else if(e.getSource() == redButton)
        {
            letsDrawCanvas.brushState = 'd';
            letsDrawCanvas.drawRed();
        }
        else if(e.getSource() == whiteButton)
        {
            letsDrawCanvas.brushState = 'd';
            letsDrawCanvas.drawWhite();
        }
        else if(e.getSource() == yellowButton)
        {
            letsDrawCanvas.brushState = 'd';
            letsDrawCanvas.drawYellow();
        }
        else if(e.getSource() == randomColorButton)
        {
        letsDrawCanvas.brushState = 'd';
        letsDrawCanvas.drawRandomColors();
        }

        else if(e.getSource() == eraserButton)
        {
            letsDrawCanvas.brushState = 'd';
            letsDrawCanvas.eraser();
        }

    }
};

//
==========================================================================
======== //

    public static void main(String[] args)
    {
        new ArtBytes().userInterface();

    }
```

```
// ============================== Method that implements the user interface
================================== //


        public void userInterface()
        {

        JFrame frame = new JFrame("Art Bytes");
        Container content = frame.getContentPane();
        content.setLayout(new BorderLayout());
        letsDrawCanvas = new LetsDrawCanvas(slider1);
        //playSound();

        content.add(letsDrawCanvas, BorderLayout.CENTER);

//------------------------------The Top Menu--------------------------------------//

        // Implement the Menu

        // Creates the menubar
        JMenuBar menubar = new JMenuBar();
        frame.setJMenuBar(menubar);

        // Attaches the File Option to the main menu bar
        JMenu file = new JMenu("File");
        menubar.add(file);

            JMenuItem open = new JMenuItem("Open");
            file.add(open);

            JMenuItem save = new JMenuItem("Save");
            file.add(save);

            JMenuItem exit = new JMenuItem("Exit");
            file.add(exit);

        // Attaches the Edit Option to the main menu bar
        JMenu edit = new JMenu("Edit");
        menubar.add(edit);

            //JMenuItem zoom = new JMenuItem("Zoom");
            //edit.add(zoom);

            JMenu filter = new JMenu("Filter");
            edit.add(filter);

                JMenuItem sharpen = new JMenuItem("Sharpen");
                filter.add(sharpen);
```

```
            JMenuItem blur = new JMenuItem("Blur");
            filter.add(blur);

            JMenuItem edgeDect = new JMenuItem("Edge Detection");
            filter.add(edgeDect);


// Lets add the actionListeners to the Menu Items!

//File > Open
class openAction implements ActionListener{
   public void actionPerformed (ActionEvent e)
   {
      openAnImage();
   }
}
open.addActionListener(new openAction());

//File > Save
class saveaction implements ActionListener{
   public void actionPerformed (ActionEvent e)
   {
      saveTheImage();
   }
}

save.addActionListener(new saveaction());


//File > Exit
class exitaction implements ActionListener{
   public void actionPerformed (ActionEvent e)
      {
         System.exit(0);
      }
}

exit.addActionListener(new exitaction());

//Edit > Filter > Blur
class bluraction implements ActionListener{
   public void actionPerformed (ActionEvent e)
   {
      float[] matrix = {
         0.111f, 0.111f, 0.111f,
         0.111f, 0.111f, 0.111f,
         0.111f, 0.111f, 0.111f,
      };
```

```
        BufferedImageOp op = new ConvolveOp( new Kernel(3, 3, matrix) );
        for (int i = 0; i != 50; ++i) {
            letsDrawCanvas.g2.drawImage(letsDrawCanvas.image, op, 0,0);
        }

        letsDrawCanvas.repaintCanvas();
    }
}

blur.addActionListener(new bluraction());

//Edit > Filter > Sharpen
class sharpenaction implements ActionListener{
    public void actionPerformed (ActionEvent e)
    {
        float[] matrix = {
            -1.0f, -1.0f, -1.0f,
            -1.0f, 9.0f, -1.0f,
            -1.0f, -1.0f, -1.0f
        };

        BufferedImageOp op = new ConvolveOp( new Kernel(3, 3, matrix) );
        for (int i = 0; i != 45; ++i) {
            letsDrawCanvas.g2.drawImage(letsDrawCanvas.image, op, 0,0);
        }

        letsDrawCanvas.repaintCanvas();
    }
}

sharpen.addActionListener(new sharpenaction());

//Edit > Filter > Edge Detection
class edgeDectAction implements ActionListener{
    public void actionPerformed (ActionEvent e)
    {
        float[] matrix = {
            1.0f, 0.0f, -1.0f,
            1.0f, 0.0f, -1.0f,
            1.0f, 0.0f, -1.0f
        };

    BufferedImageOp op = new ConvolveOp( new Kernel(3, 3, matrix) );
    for (int i = 0; i != 45; ++i) {
        letsDrawCanvas.g2.drawImage(letsDrawCanvas.image, op, 0,0);
    }

    letsDrawCanvas.repaintCanvas();
    }
```

```
}

edgeDect.addActionListener(new edgeDectAction());


// Changes the Slider implementations
class SlideChanger implements ChangeListener{
   public void stateChanged (ChangeEvent e)
   {
      JSlider slider = (JSlider)e.getSource();
      int value = slider.getValue();

      letsDrawCanvas.g2.setStroke(new BasicStroke(value));
   }
}


//------------------------------Upper Panel--------------------------------------//

      // JPanel implementation for the upper Panel Buttons (The "cool" effects) // to be added
      JPanel upperPanelButtons = new JPanel();

      //rainbowButton = new JButton("Rainbow");
      rainbowButton.addActionListener(rainbowListener);

      //prettyCirclesButton = new JButton("Pretty Circles");
      prettyCirclesButton.addActionListener(prettyListener);

      //bubblesButton = new JButton("Bubbles");
      bubblesButton.addActionListener(bubbleListener);

      //starButton = new JButton("Star");
      starButton.addActionListener(starListener);

      //Adds buttons to the upper JPanel
      upperPanelButtons.add(rainbowButton);
      upperPanelButtons.add(prettyCirclesButton);
      upperPanelButtons.add(bubblesButton);
      upperPanelButtons.add(starButton);


      // Adds the lower panel to the top of the screen
      content.add(upperPanelButtons, BorderLayout.NORTH);


//------------------------------Lower Panel--------------------------------------//

      // JPanel implementation for the lower Panel Buttons (The basic Colors)
      JPanel lowerPanelButtons = new JPanel();
```

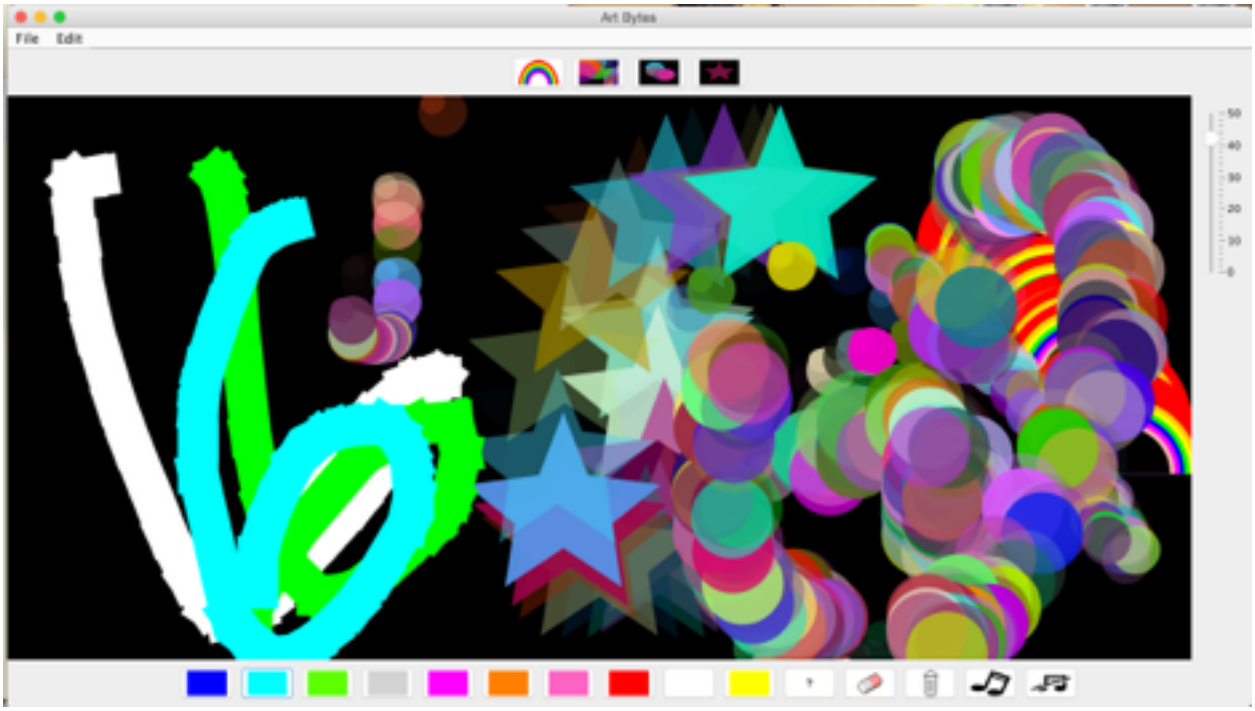
```
//blueButton = new JButton("Blue");
blueButton.addActionListener(actionListener);

//cyanButton = new JButton("Cyan");
cyanButton.addActionListener(actionListener);

//greenButton = new JButton("Green");
greenButton.addActionListener(actionListener);

//lightGrayButton = new JButton("Light Gray");
lightGrayButton.addActionListener(actionListener);

//magentaButton = new JButton("Magenta");
magentaButton.addActionListener(actionListener);

//orangeButton = new JButton("Orange");
orangeButton.addActionListener(actionListener);

//pinkButton = new JButton("Pink");
pinkButton.addActionListener(actionListener);

//redButton = new JButton("Red");
redButton.addActionListener(actionListener);

//whiteButton = new JButton("White");
whiteButton.addActionListener(actionListener);

//yellowButton = new JButton("Yellow");
yellowButton.addActionListener(actionListener);

//randomColorButton = new JButton("Random");
randomColorButton.addActionListener(actionListener);

//eraserButton = new JButton("Eraser");
eraserButton.addActionListener(actionListener);

//EmptyFrameButton = new JButton("Clear");
EmptyFrameButton.addActionListener(actionListener);

//musicOnButton = new JButton("Music On");
musicOnButton.addActionListener(musicPlayListener);

//musicOffButton = new JButton("Music Off");
musicOffButton.addActionListener(musicOffListener);

// Adds the Buttons to the lower JPanel
lowerPanelButtons.add(blueButton);
lowerPanelButtons.add(cyanButton);
```

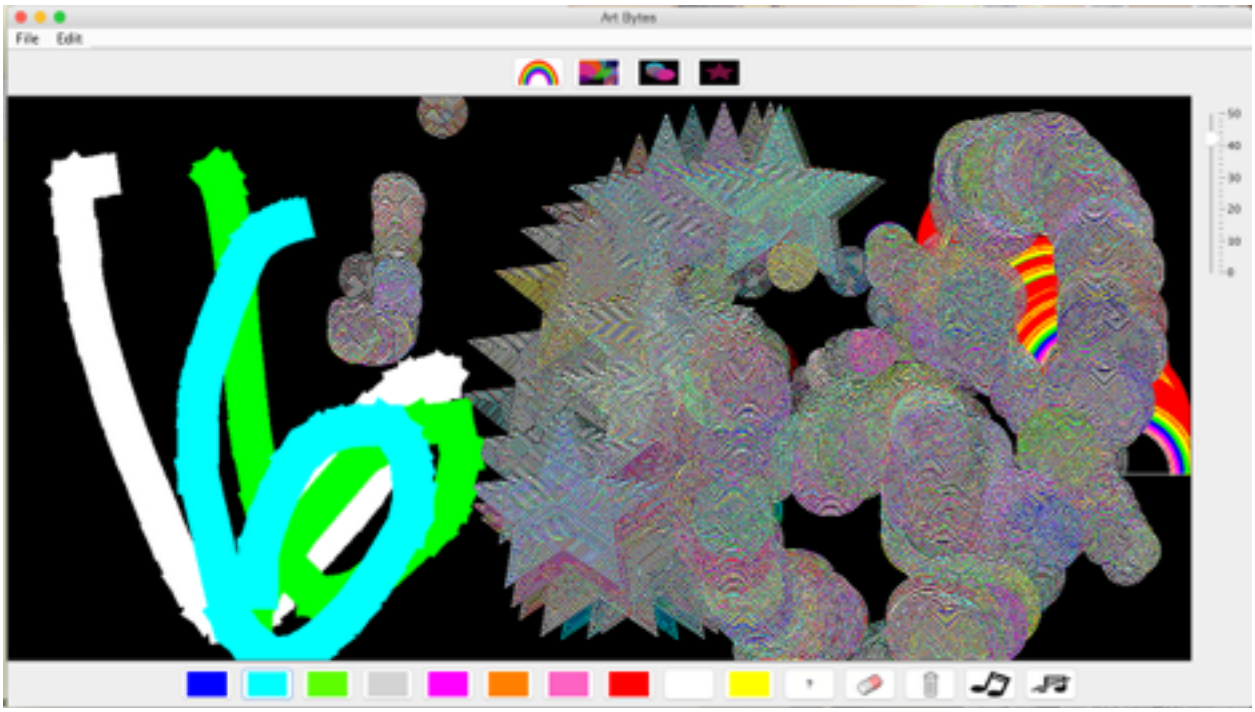
```
        lowerPanelButtons.add(greenButton);
        lowerPanelButtons.add(lightGrayButton);
        lowerPanelButtons.add(magentaButton);
        lowerPanelButtons.add(orangeButton);
        lowerPanelButtons.add(pinkButton);
        lowerPanelButtons.add(redButton);
        lowerPanelButtons.add(whiteButton);
        lowerPanelButtons.add(yellowButton);
        lowerPanelButtons.add(randomColorButton);
        lowerPanelButtons.add(eraserButton);
        lowerPanelButtons.add(EmptyFrameButton);
        lowerPanelButtons.add(musicOnButton);
        lowerPanelButtons.add(musicOffButton);

        // Adds the lower panel to the bottom of the screen
        content.add(lowerPanelButtons, BorderLayout.SOUTH);

//------------------------------Sliders-------------------------------------//

        // JPanel implementation for the side Panel Buttons (The basic Colors)
        JPanel panelEast = new JPanel();

        slider1.setMinorTickSpacing(2);
        slider1.setMajorTickSpacing(10);
        slider1.setPaintTicks(true);
        slider1.setPaintLabels(true);
        slider1.setLabelTable(slider1.createStandardLabels(10));


        // Adds the content to the JPanel
        panelEast.add(slider1);


        //sliderListener
        slider1.addChangeListener(new SlideChanger());

        //Adds the slider to the east side of the screen
        content.add(panelEast, BorderLayout.EAST);


//------------------------------Border-------------------------------------//
//JPanel borderPaneDesign = new JPanel();
//borderPaneDesign.setBorder(BorderFactory.createLineBorder(Color.black));

//TitledBorder title;
//title = BorderFactory.createTitledBorder("title");
//borderPaneDesign.setBorder(title);
//content.add(panelEast, BorderLayout.EAST);
```

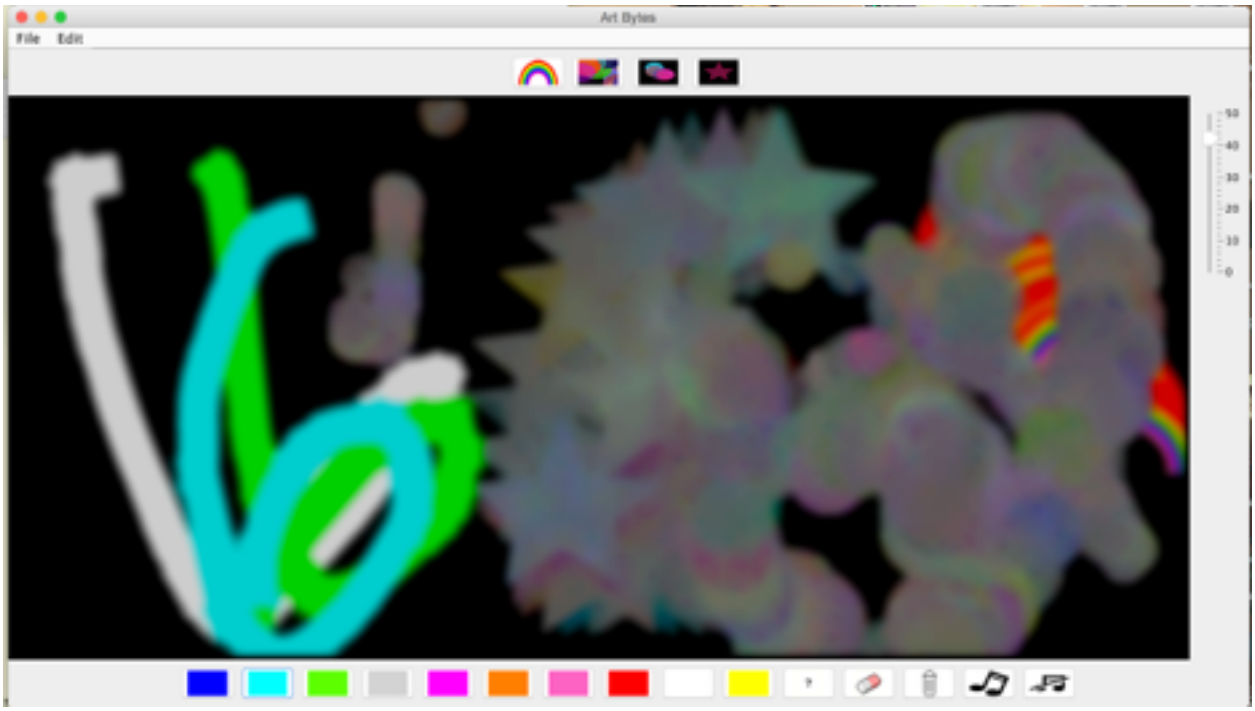
```
//-----------------------------Frame-------------------------------------//

        frame.setSize(CANVAS_WIDTH, CANVAS_HEIGHT);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }

//
================================================================================
================================== //

public void openAnImage(){

JFileChooser chooser =  new JFileChooser();
chooser.setCurrentDirectory(new File(System.getProperty("user.home")));
int display = chooser.showOpenDialog(null);
if (display == JFileChooser.APPROVE_OPTION)
    {
    // Loads a file selected by the user into the canvas
    File selectedFile = chooser.getSelectedFile();
    System.out.println("Selected file: " + selectedFile.getAbsolutePath() );

    try {
        // Displays the imported bufferedimage
        BufferedImage sourceImageBuffered = ImageIO.read(selectedFile);


        // Scales the selected file selected by the user
        //int imgscWidth = sourceImageBuffered.getWidth(this); // gets the width from the source
image
        //int imgscHeight = sourceImageBuffered.getHeight(this); // gets the height from the source
image

        letsDrawCanvas.g2.drawImage(sourceImageBuffered, 0,0, null); // changes the
letsDrawCancas image to the new source image


        // Allows us to draw on top of the imported buffered image
        letsDrawCanvas.repaintCanvas();

    }
    catch (IOException exception) {
        //JOptionPane.showMessageDialog( this, exception);
    }

    // Displays the file selected by the user into the canvas
    //newLabel = new JLabel(selectedFile);
    //add(newLabel);
```

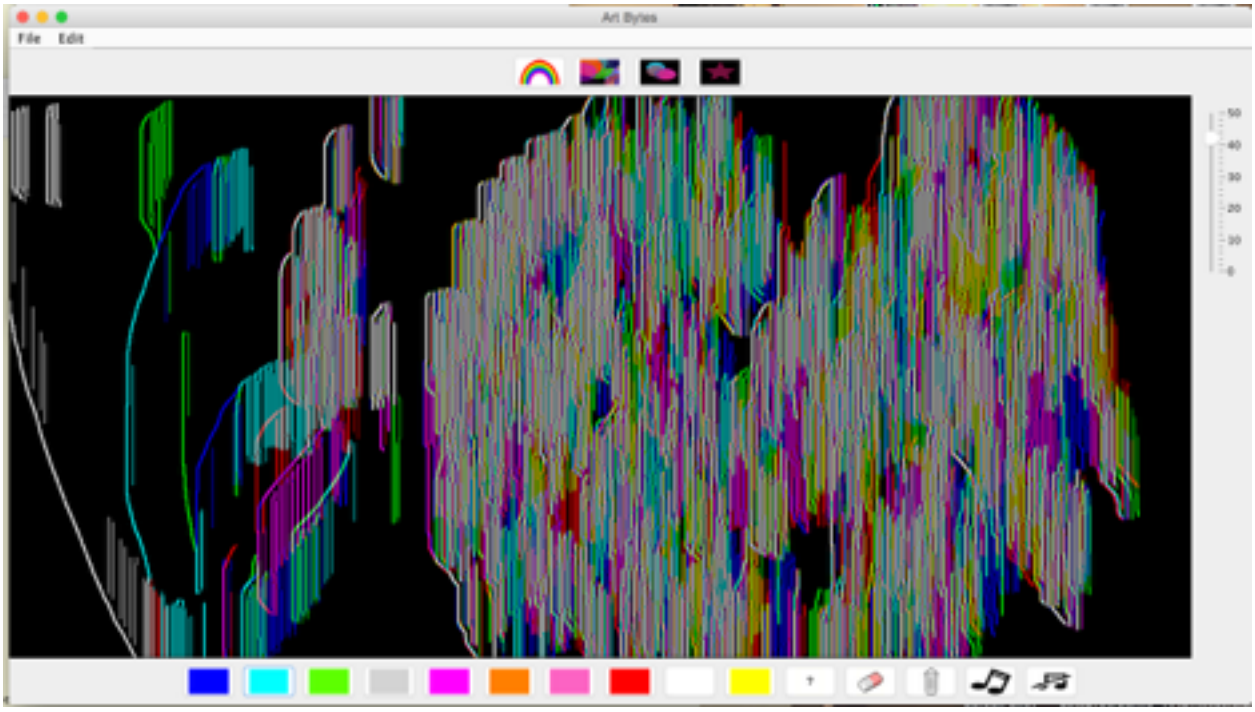
```
    }

}

public void saveTheImage(){
    String savingFileName = (String)JOptionPane.showInputDialog("Enter the desired name for
the PNG file you'd like to save");

    // Saves the file as a png (portable network graphics)
    savingFileName += ".png";

    //Get image from canvas
    image = letsDrawCanvas.getImage();

    File outputFile = new File(savingFileName);
    try
    {
    javax.imageio.ImageIO.write( image, "png", outputFile );
    }
    catch ( IOException e )
    {
        System.out.println(e);
    }
}

public void playSound() {
    try {
        AudioInputStream audioInputStream = AudioSystem.getAudioInputStream(new
File("ArtBytesFiles/relaxingMusic.wav").getAbsoluteFile());
        clip = AudioSystem.getClip();
        clip.open(audioInputStream);
        clip.start();
        } catch(Exception ex) {
        System.out.println("Oops! Wrong music file. Please load the correct file in .wav format");
        ex.printStackTrace();
    }
}

public void stopMusic() {
    clip.stop();
}

}
```

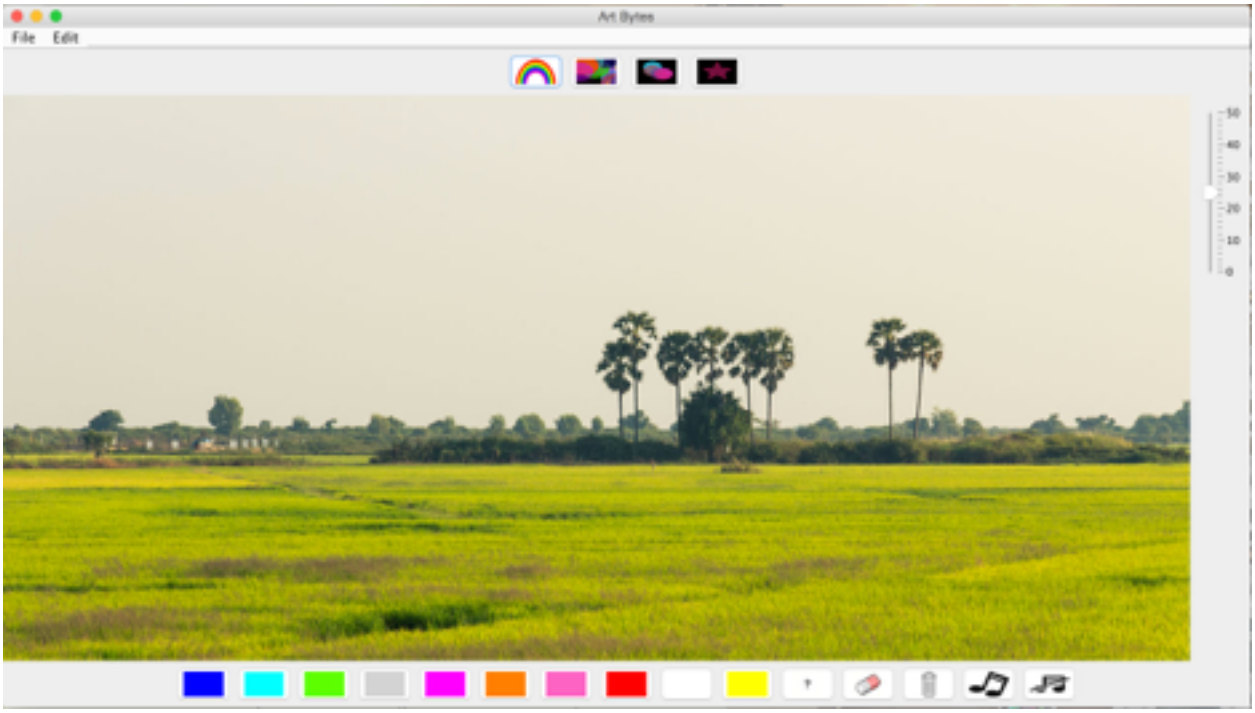
Program Outputs (screenshots):






Art Bytes
File Edit


Art Bytes
File Edit


Art Bytes
File
Edit