ICP-2036 / Graphics 2 Algorithms

Assignment 2 – 3D Data Viewer (code)

James Jackson
<eeu203@bangor.ac.uk>

December 1, 2014

Contents

1 VDSEX.java 2

2 DataSetLoader.java 2

3 PlaneVisualisation.java 3

4 PlaneVisualisationListener.java 7

5 PlaneVisualisationNonOrthogonalListener.java 10

6 DataSetVisualiser.java 14

7 PlaneMenu.java 18

8 IndexedColor.java 19

9 IndexedColorRenderer.java 20

10 ColorIcon.java 20

11 ColorProfiles.java 21

12 ColorViewer.java 23

13 ColorStepChooser.java 26

14 ChooseFile.java 28

15 Runner.java 29

1 VDSEX.java

```
/*
 * To change this license header, choose License Headers in
     Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;

/**
 *
 * @author eeu203
 */
public class VDSEx extends VolumetricDataSet {
    private int currentDimValue;
    public VDSEx(final File in, final int xSize, final int
        ySize, final int zSize) throws IOException,
        FileNotFoundException{
         this(new FileInputStream(in), xSize, ySize, zSize);
         this.currentDimValue = 0;
    }

    public VDSEx(final InputStream in, final int xSize, final
        int ySize, final int zSize) throws IOException,
        FileNotFoundException{
         super(in, xSize, ySize, zSize);
         this.currentDimValue = 0;
    }




    public void setDimValue(int dimValue){
        this.currentDimValue = dimValue;
    }

    public int getDimValue(){
        return this.currentDimValue;
    }
}
```

2 DataSetLoader.java

```
/*
```

```
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import java.io.File;
import java.io.IOException;

/**
 *
 * @author eeu203
 */
public class DataSetLoader {
    private VDSEx ds;
    private int dimension = 1;
    private int[] dimensionVal = {0,0,0};
    public DataSetLoader(String dataset, int x, int y, int z){
        try{
            ds = new
                VDSEx(getClass().getResourceAsStream(dataset),x,y,z);
        } catch (IOException e){
            System.out.println("Error Loading File");
        }
    }
    public DataSetLoader(File dataset, int x, int y, int z){
        try{
            ds = new VDSEx(dataset,x,y,z);
        } catch (IOException e){
            System.out.println("Error Loading File");
        }
    }
    public VDSEx getSet(){
        return ds;
    }
    public void setDimension(int dim){
        this.dimension = dimension;
    }
    public int getDimension(){
        return this.dimension;
    }
    public void setDimensionVal(int dimension, int dimVal){
        this.dimensionVal[dimension] = dimVal;
    }
    public int[] getDimensionVal(){
        return this.dimensionVal;
    }
}
```

3 PlaneVisualisation.java

```
/*
 * To change this license header, choose License Headers in
     Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import javax.swing.*;
import javax.media.opengl.awt.GLCanvas;
import javax.swing.event.*;
import javax.swing.table.DefaultTableModel;


public class PlaneVisualisation extends JInternalFrame {

    final private VDSEx ds;
    private int dimension;
    private ArrayList<IndexedColor> colorSteps;
    private ArrayList<ArrayList> colorProfiles;
    private GLCanvas glcanvas;
    private JComboBox profileChooser;
    static int openFrameCount = 0;
    static final int xOffset = 280, yOffset = 0;


    final private int[][] combos = {{1,2},{0,2},{0,1}};

    public PlaneVisualisation(VDSEx ds1, int dimension,
        ArrayList<ArrayList> firstColorProfiles, String title){
        super(title,
          true, //resizable
          false, //closable
          false, //maximizable
          false);//iconifiable
        setLayout(new BorderLayout());

        this.ds = ds1;
        this.dimension = dimension;

        this.colorProfiles = firstColorProfiles;

        this.colorSteps = this.colorProfiles.get(0);
        int[] size = ds.getDimensions();
        int largest = 0;
        for(int i: size)
            largest = (i > largest)? i: largest;

        Dimension canvasSize = new Dimension(largest, largest);
        int[] dims = ds.getDimensions();
        glcanvas = new GLCanvas();
        int max = (dimension < 3)? dims[dimension] : 360;
        JSlider ortho = new JSlider(0, max - 1);
        profileChooser = new JComboBox();
        int count = 0;
        for(ArrayList<IndexedColor> single: colorProfiles)
            profileChooser.addItem(String.format("Profile
               %d",++count));

        final ColorViewer colorFrame = new
           ColorViewer(colorSteps, "Colors", glcanvas, this);

        sortColors();

        final GLCanvas _canv = glcanvas;

        ortho.addChangeListener(new ChangeListener(){

            @Override
            public void stateChanged(ChangeEvent e) {
                JSlider slider = (JSlider) e.getSource();
                ds.setDimValue(slider.getValue());
                _canv.repaint();
            }

        });

        final PlaneVisualisation ps = this;

        profileChooser.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                JComboBox cb = (JComboBox) e.getSource();
                colorSteps =
                   colorProfiles.get(cb.getSelectedIndex());
                _canv.repaint();
            }
        });

        if(dimension < 3){
            PlaneVisualisationListener pvl = new
               PlaneVisualisationListener(this, this.ds,
               this.dimension, ds.getMaxValue()/2, largest);
            glcanvas.addGLEventListener(pvl);
        } else {
            PlaneVisualisationNonOrthogonalListener pvl = new
               PlaneVisualisationNonOrthogonalListener(this,
               this.ds, this.dimension, ds.getMaxValue()/2,
                largest);
            glcanvas.addGLEventListener(pvl);
        }


        glcanvas.setPreferredSize(new Dimension(largest,
            largest));
        add(glcanvas, BorderLayout.CENTER);
        JPanel controls = new JPanel(new BorderLayout());
        controls.add(ortho, BorderLayout.NORTH);
        controls.add(profileChooser, BorderLayout.CENTER);
        add(controls, BorderLayout.SOUTH);
        pack();
        int top = (dimension > 2)? 330 : -5;
        setLocation((xOffset*(openFrameCount % 3)) + 150 ,
            yOffset*openFrameCount + top);
        openFrameCount++;
    }



    public double[] getColor(int step){
        double[] color = new double[3];
        int index;
        for(int i = 0; i < colorSteps.size(); i++){
            index = colorSteps.get(i).getIndex();
            if(step > index) continue;
            if(step < index){
                color = makeColor(i ,step);
                break;
            }
            if(step == index) color = getIndexColor(i);
        }
        return color;
    }

    private double[] getIndexColor(int i){
        double color[] = new double[3];
        color[0] = colorSteps.get(i).getRed();
        color[1] = colorSteps.get(i).getGreen();
        color[2] = colorSteps.get(i).getBlue();
        return color;
    }

    private double[] makeColor(int i, int step){
        int diff = colorSteps.get(i).getIndex() -
            colorSteps.get(i-1).getIndex();
        int stepDiff = step - colorSteps.get(i-1).getIndex();
        double rgb[] = new double[3];
        rgb[0] = (colorSteps.get(i-1).getRed() +
            ((colorSteps.get(i).getRed() -
            colorSteps.get(i-1).getRed()) / diff * stepDiff)) /
```

```
                255.0;
            rgb[1] = (colorSteps.get(i-1).getGreen() +
                ((colorSteps.get(i).getGreen() -
                colorSteps.get(i-1).getGreen()) / diff * stepDiff))
                / 255.0;
            rgb[2] = (colorSteps.get(i-1).getBlue() +
                ((colorSteps.get(i).getBlue() -
                colorSteps.get(i-1).getBlue()) / diff * stepDiff))
                / 255.0;
            return rgb;
        }

        private void refactorColors(DefaultTableModel model){
            for(int i = 0; i < model.getRowCount(); i++){
                model.removeRow(i);
            }
            for(IndexedColor color : colorSteps){
                model.addRow(new Object[]{color.getIndex() + "
                    ", color.getRGB() + "    ", "remove"});
            }
        }

        public void sortColors(){
            Collections.sort(colorSteps, new
                Comparator<IndexedColor>() {
                @Override
                public int compare(IndexedColor ic1, IndexedColor
                    ic2){
                    return ic1.getIndex() - ic2.getIndex();
                }
            });
        }

        public void repaintCanvas(){
            glcanvas.repaint();
        }

        public void refactorColorProfiles(){
            if(profileChooser.getItemCount() <
                colorProfiles.size())
                profileChooser.addItem(String.format("Profile
                    %d",colorProfiles.size()));
        }


}
```

4 PlaneVisualisationListener.java

```
package Assignment2;
```

```
import java.awt.*;
import java.awt.geom.Point2D;
import java.util.*;
import javax.media.opengl.*;
import javax.media.opengl.glu.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class PlaneVisualisationListener implements
   GLEventListener {

    private VDSEx ds;
    private GLU glu;
    PlaneVisualisation pv;
    private int dimension;
    private int dimValue;
    private int[][][] volumeData;
    private int canvasSize;

    // x = 0, y = 1, z = 2
    final private int[][] combos = {{1,2},{0,2},{0,1}};

    public PlaneVisualisationListener(PlaneVisualisation pv,
       VDSEx ds, int dimension, int dimValue, int canvasSize){
        this.pv = pv;
        this.ds = ds;
        this.dimension = dimension;
        this.volumeData = ds.getVolumeData();
        this.dimValue = dimValue;
        this.canvasSize = canvasSize;
        ds.setDimValue(dimValue);
    }



    @Override
    public void init(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();
        glu = new GLU();

        gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        gl.glViewport(0, 0, canvasSize, canvasSize);
        gl.glMatrixMode(GL2.GL_PROJECTION);
        gl.glLoadIdentity();
        glu.gluOrtho2D(0.0, canvasSize, 0.0, canvasSize);
    }

    @Override
    public void display(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();
        dimValue = ds.getDimValue();
        int dimensions[] = ds.getDimensions();
        for(int i = 0; i < dimensions[combos[dimension][0]];
            i++){
            for(int j = 0; j <
                dimensions[combos[dimension][1]]; j++){

                int index;
                switch(dimension){
                    case 0:
                        index = volumeData[dimValue][i][j];
                        break;
                    case 1:
                        index = volumeData[i][dimValue][j];
                        break;
                    default:
                        index = volumeData[i][j][dimValue];
                }
                double color[] = pv.getColor(index);
                gl.glColor3d(color[0], color[1], color[2]);
                drawPixel(gl, i, j, -1, 2, new
                    Point2D.Double(0,0));
            }

        }
    }

    @Override
    public void reshape(GLAutoDrawable glad, int i, int i1,
        int i2, int i3) {

    }


    public void drawPixel(GL2 gl, double x, double y, double
        z, int size, Point2D.Double cent){
        gl.glBegin(GL2.GL_POLYGON);
            gl.glVertex3d(x + cent.x, y + cent.y, z);
            gl.glVertex3d(x + size + cent.x, y + cent.y, z);
            gl.glVertex3d(x + size + cent.x, y + size +
                cent.y, z);
            gl.glVertex3d(x + cent.x, y + size + cent.y, z);
        gl.glEnd();
    }

    @Override
    public void dispose(GLAutoDrawable glad) {

    }



}
```

5 PlaneVisualisationNonOrthogonalListener.java

```
package Assignment2;

import java.awt.*;
import java.awt.geom.Point2D;
import java.util.*;
import javax.media.opengl.*;
import javax.media.opengl.glu.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;

public class PlaneVisualisationNonOrthogonalListener
   implements GLEventListener {

    private VDSEx ds;
    private GLU glu;
    PlaneVisualisation pv;
    private int dimension;
    private int dimValue;
    private int[][][] volumeData;
    private int canvasSize;

    // x = 0, y = 1, z = 2
    final private int[][] combos = {{1, 2}, {0, 2}, {0, 1}};

    public
       PlaneVisualisationNonOrthogonalListener(PlaneVisualisation
       pv, VDSEx ds, int dimension, int dimValue, int
       canvasSize) {
        this.pv = pv;
        this.ds = ds;
        this.dimension = dimension;
        this.volumeData = ds.getVolumeData();
        this.dimValue = dimValue;
        this.canvasSize = canvasSize;
        ds.setDimValue(dimValue);
    }

    @Override
    public void init(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();

        glu = new GLU();

        gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        gl.glViewport(0, 0, canvasSize, canvasSize);
        gl.glMatrixMode(GL2.GL_PROJECTION);
        gl.glLoadIdentity();
        glu.gluOrtho2D(0.0, canvasSize, 0.0, canvasSize);
    }
    @Override
    public void display(GLAutoDrawable drawable) {
        GL2 gl = drawable.getGL().getGL2();
        dimValue = ds.getDimValue();
        int dimensions[] = new int[3];
        int angle = dimValue;

        switch(dimension){
            case 3:
                dimensions[0] = ds.getDimensions()[1];
                dimensions[1] = ds.getDimensions()[2];
                dimensions[2] = ds.getDimensions()[0];
                break;
            case 4:
                dimensions[0] = ds.getDimensions()[0];
                dimensions[1] = ds.getDimensions()[2];
                dimensions[2] = ds.getDimensions()[1];
                break;
            default:
                dimensions[0] = ds.getDimensions()[0];
                dimensions[1] = ds.getDimensions()[1];
                dimensions[2] = ds.getDimensions()[2];
                break;
        }

        double theta = angle * (Math.PI / 180);
        double radius = Math.sqrt(2) * (dimensions[0] / 2);

        int x2 = (int) ((dimensions[0] / 2 ) + (radius *
            Math.cos(theta)));
        int y2 = (int) ((dimensions[1] / 2 ) + (radius *
            Math.sin(theta)));

        theta = (angle + 180) * (Math.PI / 180);
        int x1 = (int) ((dimensions[0] / 2 ) + (radius *
            Math.cos(theta)));
        int y1 = (int) ((dimensions[1] / 2 ) + (radius *
            Math.sin(theta)));

        ArrayList<int[]> lines = getLine(gl, x2, y2, x1, y1,
            dimensions);

        int[][] projection = new
            int[lines.size()][dimensions[2]];

        for(int i = 0; i < lines.size(); i++){
            for(int j = 0; j < dimensions[2]; j++){
                switch (dimension){
                    case 3:
                        projection[i][j] =
                            volumeData[j][lines.get(i)[0]][lines.get(i)[1]];
                        break;
                    case 4:
                        projection[i][j] =
                            volumeData[lines.get(i)[0]][j][lines.get(i)[1]];
                        break;
                    default:
                        projection[i][j] =
                            volumeData[lines.get(i)[0]][lines.get(i)[1]][j];
                }
            }
        }
        drawArray(gl, projection, pv);


    }

    public ArrayList<int[]> getLine(GL2 gl, int x1, int y1,
        int x2, int y2, int[] dimensions) {
        ArrayList<int[]> points = new ArrayList<int[]>();

        int dx = x2 - x1;
        int dy = y2 - y1;

        int y = y1;
        int x = x1;

        int incX, incY;

        if (dx >= 0) {
            incX = 1;
        } else {
            incX = -1;
            dx = -dx;
        }

        if (dy >= 0) {
            incY = 1;
        } else {
            incY = -1;
            dy = -dy;
        }
        int error;
        if(dx > 0){
            error = dx >> 1;
        } else {
            error = dy >> 1;
        }

        int length = Math.max(dy, dx) + 1;

        for (int i = 0; i < length; i++) {
            if(error < 0)
                    error *= -1;
            if (dx > dy) {
                error += dy;
                if(error >= dx){
                    error -= dx;

                    y += incY;
                }

                x += incX;

            } else {

                error += dx;
                if(error >= dy){
                    error -= dy;

                    x += incX;
                }

                y += incY;
            }
            if(x > -1 && x < dimensions[0] && y > -1 && y <
               dimensions[1]){
                points.add(new int[]{x, y});
            }

        }

        return points;
    }

    @Override
    public void reshape(GLAutoDrawable glad, int i, int i1,
       int i2, int i3) {

    }

    public void drawPixel(GL2 gl, double x, double y, double
       z, int size, Point2D.Double cent,
            GLAutoDrawable drawable) {
        Double ar = (double)
           Math.min(drawable.getSurfaceWidth(),
           drawable.getSurfaceHeight())
                / Math.max(ds.getDimensions()[0],
                    ds.getDimensions()[2]);
        x *= ar;
        y *= ar;
        z *= ar;
        size *= ar;
        gl.glBegin(GL2.GL_POLYGON);
        gl.glVertex3d(x + cent.x, y + cent.y, z);
        gl.glVertex3d(x + size + cent.x, y + cent.y, z);
        gl.glVertex3d(x + size + cent.x, y + size + cent.y, z);
        gl.glVertex3d(x + cent.x, y + size + cent.y, z);
        gl.glEnd();
    }

    public void drawArray(GL2 gl, int[][] array,
        PlaneVisualisation pv){

        for(int i = 0; i < array.length; i++){
            for(int j = 0; j < array[0].length; j++){
                int index = array[i][j];
                double color[] = pv.getColor(index);
                gl.glColor3d(color[0], color[1], color[2]);
                gl.glBegin(GL2.GL_QUADS);
                    gl.glVertex2i(i, j);
                    gl.glVertex2i(i + 1, j);
                    gl.glVertex2i(i + 1, j + 1);
                    gl.glVertex2i(i, j + 1);
                gl.glEnd();
            }
        }
    }

    @Override
    public void dispose(GLAutoDrawable glad) {

    }

}
```

6 DataSetVisualiser.java

```
package Assignment2;

import java.awt.*;
import java.awt.event.*;
import java.io.File;
import java.util.ArrayList;
import javax.swing.*;
import javax.media.opengl.*;
import javax.swing.event.*;

public class DataSetVisualiser extends JFrame {
    private DataSetLoader dl;
    private VDSEx ds;
    private int[] dimensions;
    private JPanel planesPanel;
    private int panelCount;
    private ArrayList<ArrayList> colorProfiles;
    private ArrayList<PlaneVisualisation> planes;
    private JDesktopPane desktop;
    private ColorProfiles colors;

    public DataSetVisualiser(){
        super("Data Set Visualiser");

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);



        planesPanel = new JPanel(new GridLayout(0, 3, 10, 10));

        colorProfiles = new ArrayList<>();

        ArrayList<IndexedColor> firstProfile = new
            ArrayList<IndexedColor>();

        firstProfile.add(new IndexedColor(0, Color.black));
        firstProfile.add(new IndexedColor(255, Color.white));

        colorProfiles.add(firstProfile);

        ArrayList<IndexedColor> secondProfile = new
            ArrayList<IndexedColor>();

        secondProfile.add(new IndexedColor(0, Color.white));
        secondProfile.add(new IndexedColor(255, Color.black));

        colorProfiles.add(secondProfile);



        planes = new ArrayList<PlaneVisualisation>();



        JPanel layout = new JPanel(new BorderLayout());
        JPanel controlPanel = new JPanel();
        JButton press = new JButton("Add Panel");
        JButton addColorProfile = new JButton("Color
            Profiles");

        press.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                addPanel();
            }
        });

        final DataSetVisualiser ds = this;

        addColorProfile.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                ColorProfiles profiles = new ColorProfiles(ds);
            }
        });

        controlPanel.add(press);
        controlPanel.add(addColorProfile);
        desktop = new JDesktopPane();
        desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
        colors = new ColorProfiles(ds);

        desktop.add(colors);

        layout.add(desktop, BorderLayout.CENTER);
        //layout.add(controlPanel, BorderLayout.CENTER);
        add(layout);
        //setContentPane(desktop);
        setJMenuBar(new PlaneMenu(this));
        //add the GLCanvas just like we would any Component

        setExtendedState(MAXIMIZED_BOTH);


        //center the JFrame on the screen
        centerWindow(this);
    }

    public void loadFile(File file, int[] sizes){

        desktop.removeAll();
        dl = new DataSetLoader(file, sizes[0], sizes[1],
            sizes[2]);

        ds = dl.getSet();
        panelCount = 2;
        dimensions = ds.getDimensions();

        desktop.add(colors);

        planes.removeAll(planes);

        planes.add(new PlaneVisualisation(ds, 0,
            colorProfiles, "Orthogonal X"));
        planes.add(new PlaneVisualisation(ds, 1,
            colorProfiles, "Orthogonal Y"));
        planes.add(new PlaneVisualisation(ds, 2,
            colorProfiles, "Orthogonal Z"));
        planes.add(new PlaneVisualisation(ds, 3,
            colorProfiles, "Non Orthogonal X"));
        planes.add(new PlaneVisualisation(ds, 4,
            colorProfiles, "Non Orthogonal Y"));
        planes.add(new PlaneVisualisation(ds, 5,
            colorProfiles, "Non Orthogonal Z"));
        for(PlaneVisualisation plane: planes){
            desktop.add(plane);
            plane.setVisible(true);
        }
    }

    public void addPanel(){
//        if(panelCount < 6){
//            PlaneVisualisation plane = new
    PlaneVisualisation(ds, 1, colorProfiles);
//            planes.add(plane);
//            planesPanel.add(planes.get(panelCount - 1));
//            panelCount++;
//            pack();
//            centerWindow(this);
//        }
    }

    public void repaintViews(){
        for(PlaneVisualisation plane: planes){
            plane.repaintCanvas();
            plane.refactorColorProfiles();
        }
    }

    public void centerWindow(Component frame) {
        Dimension screenSize =
           Toolkit.getDefaultToolkit().getScreenSize();
        Dimension frameSize  = frame.getSize();

        if (frameSize.width  > screenSize.width )
           frameSize.width  = screenSize.width;
        if (frameSize.height > screenSize.height)
           frameSize.height = screenSize.height;

        frame.setLocation (
            (screenSize.width  - frameSize.width ) >> 1,
            (screenSize.height - frameSize.height) >> 1
        );
    }

    public ArrayList<ArrayList> getColorProfiles(){
        return colorProfiles;
    }

    public void addColorProfile(){
        ArrayList<IndexedColor> newProfile = new
           ArrayList<IndexedColor>();

        newProfile.add(new IndexedColor(0, Color.white));
        newProfile.add(new IndexedColor(255, Color.black));
        colorProfiles.add(newProfile);
    }



}
```

7 PlaneMenu.java

```
/*
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

/**
 *
 * @author jacko
 */
public class PlaneMenu extends JMenuBar {
    JMenu file;
    JMenuItem open;
    JMenuItem exit;
    private DataSetVisualiser ds;
    public PlaneMenu(final DataSetVisualiser ds){
        super();
        file = new JMenu("File");
        open = new JMenuItem("Open");
        exit = new JMenuItem("Exit");
        this.ds = ds;

        file.add(open);

        open.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ev){
                ChooseFile choose = new ChooseFile();
                ds.loadFile(choose.getFile(),
                    choose.getDimensions());
            }
        });
        file.add(exit);

        add(file);
    }
}
```

8 IndexedColor.java

```
package Assignment2;

import java.awt.Color;
/**
 * @title IndexedColor class
 * subclass of awt.Color adds an index instance variable
 * @author James Jackson
 */
public class IndexedColor extends Color{
    private final int index;

    /**
     * Constructor for the IndexedColor
     * @param index the index for the color
     * @param color the color object to set the color
     */
    public IndexedColor(int index, Color color){
        super(color.getRed(), color.getGreen(),
            color.getBlue());
        this.index = index;
    }

    /**
     * Gets the index of the current instance
     * @return the index
     */
    public int getIndex(){
        return index;
    }



    /**
     * Override equals method to compare indexes of the array
     * @param o the object to compare
     * @return are the objects equal
     */
    @Override
    public boolean equals(Object o)
    {
        IndexedColor other;
        if(o instanceof IndexedColor){
            other = (IndexedColor) o;
            return other.getIndex() == this.index;
        } else {
            return false;
        }
    }

    public String toString(){
        return Integer.toString(index);
    }

}
```

9 IndexedColorRenderer.java

```
package Assignment2;

import java.awt.Component;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.ListCellRenderer;
public class IndexedColorRenderer extends JLabel implements
   ListCellRenderer<IndexedColor> {

    public IndexedColorRenderer() {
        setOpaque(true);
    }
    @Override
    public Component getListCellRendererComponent(JList<?
       extends IndexedColor> list, IndexedColor color, int
       index, boolean isSelected, boolean cellHasFocus) {

        String step = color.toString();
        ColorIcon icon = new ColorIcon(color);

        setIcon(icon);
        setText(step);


        if (isSelected) {
            setBackground(list.getSelectionBackground());
            setForeground(list.getSelectionForeground());
        } else {
            setBackground(list.getBackground());
            setForeground(list.getForeground());
        }

        return this;
    }



}
```

10 ColorIcon.java

```
package Assignment2;
```

```
// Class taken from
    http://www.codebeach.com/2007/06/creating-dynamic-icons-in-java.html

import javax.swing.*;
import java.awt.*;

public class ColorIcon implements Icon
{
    private static int HEIGHT = 14;
    private static int WIDTH = 14;

    private Color color;

    public ColorIcon(Color color)
    {
        this.color = color;
    }

    public int getIconHeight()
    {
        return HEIGHT;
    }

    public int getIconWidth()
    {
        return WIDTH;
    }

    public void paintIcon(Component c, Graphics g, int x, int
        y)
    {
        g.setColor(color);
        g.fillRect(x, y, WIDTH - 1, HEIGHT - 1);

        g.setColor(Color.black);
        g.drawRect(x, y, WIDTH - 1, HEIGHT - 1);
    }
}
```

11 ColorProfiles.java

```
/*
 * To change this license header, choose License Headers in
     Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import static Assignment2.PlaneVisualisation.openFrameCount;
import java.awt.BorderLayout;
```

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import javax.media.opengl.awt.GLCanvas;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JInternalFrame;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;


/**
 *
 * @author jacko
 */
public class ColorProfiles extends JInternalFrame {
    private DataSetVisualiser ds;
    private ArrayList<ArrayList> profiles;
    private JList profileList;
    private DefaultListModel<String> profileListModel;
    public ColorProfiles(DataSetVisualiser dsv){
        super("Color Profiles",
          false, //resizable
          false, //closable
          false, //maximizable
          false);//iconifiable
        this.ds = dsv;
        profileListModel = new DefaultListModel<>();
        profileList = new JList(profileListModel);
        loadColorProfiles();
        JButton addProfile = new JButton("Add Profile");
        addProfile.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                int cur = profiles.size();
                ds.addColorProfile();
                ds.repaintViews();
                loadColorProfiles();
                ColorViewer colorFrame = new
                    ColorViewer(profiles.get(cur), "Color
                    Profile", ds);
                colorFrame.showFrame();

            }
        });
        JPanel layout = new JPanel(new BorderLayout());

        layout.add(new JScrollPane(profileList),
            BorderLayout.NORTH);
        layout.add(addProfile, BorderLayout.CENTER);
        add(layout);

        pack();
        setVisible(true);
    }
    public void loadColorProfiles(){
        profileListModel.removeAllElements();
        profiles = ds.getColorProfiles();
        int index = 0;
        for(ArrayList profile: profiles)
            profileListModel.addElement(String.format("Profile
               %d", ++index));
        profileList.addMouseListener(new MouseAdapter(){
            public void mouseClicked(MouseEvent e){
                ColorViewer colorFrame;
                if(e.getClickCount()==2){
                    int selected =
                       profileList.getSelectedIndex();
                    colorFrame = new
                       ColorViewer(profiles.get(selected),
                       "Color Profile", ds);
                    colorFrame.showFrame();
                }
            }

        });
    }
}
```

12 ColorViewer.java

```
package Assignment2;


import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import javax.media.opengl.awt.GLCanvas;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;

/*
 * To change this license header, choose License Headers in
     Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author jacko
 */
public class ColorViewer extends JDialog {

    private ArrayList<IndexedColor> colorSteps;
    private JTable table;
    private JList<IndexedColor> list;
    private GLCanvas glCanvas;
    private JScrollPane scroll;
    private MouseListener listListener;
    private JButton addColorStep;
    private DefaultListModel<IndexedColor> listModel;
    private PlaneVisualisation ps;
    private DataSetVisualiser ds;
    private JPanel container;

    public ColorViewer(ArrayList<IndexedColor> colorSteps,
        String title, GLCanvas glCanvas, PlaneVisualisation ps){
        //super(title);
        this.ps = ps;
        this.colorSteps = colorSteps;
        this.glCanvas = glCanvas;
        buildFrame();
    }

    public ColorViewer(ArrayList<IndexedColor> colorSteps,
        String title, DataSetVisualiser ds){
        //super(title);
        this.colorSteps = colorSteps;
        this.ds = ds;
        buildFrame();
    }

    private void buildFrame(){
        listListener = new MouseAdapter(){
            public void mouseClicked(MouseEvent e){
                if(e.getClickCount()==2){
                    int selected = list.getSelectedIndex();
                    System.out.println("" + selected);
                    ColorStepChooser chooser = new
                       ColorStepChooser(colorSteps.get(selected));
                    int step =
                       colorSteps.get(selected).getIndex();
                    colorSteps.set(selected,
                       chooser.getIndexedColor());
                    listModel.set(selected,
                       colorSteps.get(selected));
                    repaint();
                    ds.repaintViews();
                }
            }

        };
        addColorStep = new JButton("Add Color Step");
        addColorStep.addActionListener(new
           AddColorStepListener());
        container = new JPanel(new BorderLayout());
        buildList(listListener);
        container.add(addColorStep, BorderLayout.CENTER);
        add(container);
        setSize(200,200);
    }

    class AddColorStepListener implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            ColorStepChooser chooser = new ColorStepChooser();
            colorSteps.add(chooser.getIndexedColor());
            Collections.sort(colorSteps, new
               Comparator<IndexedColor>() {
                @Override
                public int compare(IndexedColor ic1,
                   IndexedColor ic2){
                    return ic1.getIndex() - ic2.getIndex();
                }
            });
            listModel.removeAllElements();
            for(IndexedColor color: colorSteps)
                listModel.addElement(color);


            repaint();
            ds.repaintViews();
        }

    }

    public void showFrame(){
        setVisible(true);
    }
```

```
    public void buildList(MouseListener listListener){
        System.out.println("Repaint");
        listModel = new DefaultListModel<>();
        for(IndexedColor color: colorSteps)
            listModel.addElement(color);
        list = new JList<>(listModel);
        list.setCellRenderer(new IndexedColorRenderer());
        list.addMouseListener(listListener);
        scroll = new JScrollPane(list);
        container.add(scroll, BorderLayout.NORTH);
    }

    private void refactorList() {
        this.remove(scroll);
        buildList(listListener);
    }




}
```

13 ColorStepChooser.java

```
package Assignment2;


import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dialog;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSlider;

/*
 * To change this license header, choose License Headers in
     Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author jacko
 */
public class ColorStepChooser extends JDialog {
    private IndexedColor color;
    private JColorChooser colorChooser;
    private JSlider slider;

    public ColorStepChooser(){
        color = new IndexedColor(0, Color.black);
        this.setModal(true);
        this.add(setupPanel());
        this.pack();
        this.setVisible(true);
    }

    public ColorStepChooser(IndexedColor color){
        this.color = color;
        this.setModal(true);
        this.add(setupPanel());
        this.pack();
        this.setVisible(true);
    }

    private JPanel setupPanel(){
        JPanel panel = new JPanel(new BorderLayout());
        colorChooser = new JColorChooser();
        slider = new JSlider(0, 255, color.getIndex());
        JButton ok = new JButton("Ok");
        ok.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e) {
                color = new
                   IndexedColor(slider.getValue(),colorChooser.getColor());
                dispose();
            }
        });
        JPanel indexSlider = new JPanel();
        JLabel sliderLabel = new JLabel("Select Value");

        indexSlider.add(sliderLabel);
        indexSlider.add(slider);
        panel.add(colorChooser, BorderLayout.NORTH);
        panel.add(indexSlider, BorderLayout.CENTER);
        panel.add(ok, BorderLayout.SOUTH);
        return panel;
    }

    public IndexedColor getIndexedColor(){
        return color;
    }

}
```

14 ChooseFile.java

```
/*
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import java.awt.BorderLayout;
import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 *
 * @author jacko
 */
public class ChooseFile extends JDialog {

    private File file;
    private int sizes[] = new int[3];
    private JTextField size[] = new JTextField[3];

    public ChooseFile(){
        this.setModal(true);
        final JFileChooser chooser = new
           JFileChooser("./src/Assignment2");
        JPanel choosePanel = new JPanel(new BorderLayout());
        JPanel textFields = new JPanel();

        String labels[] = {"width","height","depth"};


        JLabel textLabels[] = new JLabel[3];
        for(int i = 0, len = size.length; i < len; i++){
            textLabels[i] = new JLabel(labels[i]);
            size[i] = new JTextField();
            size[i].setPreferredSize(new Dimension(50,30));
            textFields.add(textLabels[i]);
            textFields.add(size[i]);
        }
        //JButton go = new JButton("Load");

        chooser.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ev){
                int i = 0;
                for(JTextField sizeInput: size){
                    System.out.println(sizeInput.getText());
                    sizes[i++] =
                        Integer.parseInt(sizeInput.getText());
                }
                file = chooser.getSelectedFile();

                dispose();
            }
        });
        choosePanel.add(chooser, BorderLayout.CENTER);
        choosePanel.add(textFields, BorderLayout.NORTH);
        //choosePanel.add(go, BorderLayout.SOUTH);
        add(choosePanel);
        pack();
        setVisible(true);
    }

    public File getFile(){
        return file;
    }
    public int[] getDimensions(){
        return sizes;
    }
}
```

15 Runner.java

```
/*
 * To change this license header, choose License Headers in
     Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Assignment2;

import javax.swing.SwingUtilities;

/**
 *
 * @author jacko
 */
public class runner {
    public static void main(String[] args) {
        final DataSetVisualiser app = new DataSetVisualiser();

        SwingUtilities.invokeLater (
            new Runnable() {
                public void run() {
                    app.setVisible(true);
                }
            }
        );
    }
}
```