|  |  |
| --- | --- |
| Картинки по запросу самгту новый логотип | МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  ВЫСШЕГО ОБРАЗОВАНИЯ  «САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» |

Кафедра «Вычислительная техника»

Лабораторная работа № 3

по дисциплине «Программное обеспечение обработки визуальных данных»

Выполнили:

студенты 3-ИАИТ- 9

*Котляров А.А., Ефимушкин Н.А.*

Проверил:

доцент Мартемьянов Б.В.

Самара 2019

**Цель работы:** разработка программы, позволяющей выводить снимки с аппарата дистанционного зондирования земли, реализовать инструмент лупы, позволяющей увеличивать фрагмент рассматриваемого снимка.

**Выполнение:**

**Исходный код программы:**

package pack;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Graphics;

import java.awt.event.MouseEvent;

import java.awt.event.MouseMotionAdapter;

import java.io.File;

import java.io.IOException;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

import java.nio.file.Files;

import java.util.LinkedList;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JCheckBox;

import javax.swing.JComboBox;

import javax.swing.JComponent;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JList;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.event.ListSelectionEvent;

public class Lab3 extends JComponent {

private static class Line {

final int x1;

final int y1;

final int x2;

final int y2;

final Color color;

public Line(int x1, int y1, int x2, int y2, Color color) {

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

this.color = color;

}

}

private final LinkedList<Line> lines = new LinkedList<Line>();

public void addLine(int x1, int x2, int x3, int x4, Color color) {

lines.add(new Line(x1, x2, x3, x4, color));

repaint();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

for (Line line : lines) {

g.setColor(line.color);

g.drawLine(line.x1, line.y1, line.x2, line.y2);

}

}

static int w, h, N;

static short[][] fileShorts;

static int[][] Picture;

public static void main(String[] args) throws IOException {

JFrame ourFrame = new JFrame();

ourFrame.setPreferredSize(new Dimension(1400, 900));

ourFrame.setResizable(false);

ourFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

ourFrame.setLocation(100, 50);

Lab3 file = new Lab3();

Lab3 lupa = new Lab3();

file.setPreferredSize(new Dimension(500, 3000));

lupa.setPreferredSize(new Dimension(300, 300));

JScrollPane scroll = new JScrollPane();

scroll.setViewportView(file);

scroll.setPreferredSize(new Dimension(500, 520));

JList list = new JList();

list.setDragEnabled(false);

list.setPreferredSize(new Dimension(200, 200));

File[] pictures = new File("Pictures").listFiles();

list.setListData(pictures);

JPanel panel = new JPanel();

JPanel panel2 = new JPanel();

JCheckBox check = new JCheckBox();

check.setText("Интерполировать");

JComboBox selectCoef = new JComboBox();

JLabel lb = new JLabel();

selectCoef.setToolTipText("Коэффициент увеличения");

selectCoef.addItem(3);

selectCoef.addItem(5);

selectCoef.setSelectedIndex(0);

panel2.add(selectCoef);

panel2.add(check);

panel.setPreferredSize(new Dimension(350, 600));

panel.setLayout(new FlowLayout());

panel.add(list);

panel.add(panel2);

panel.add(lupa);

//panel.add(lb);

ourFrame.getContentPane().add(scroll, BorderLayout.EAST);

ourFrame.getContentPane().add(panel);

list.addListSelectionListener((ListSelectionEvent e) -> {

if (list.getSelectedIndex() == -1) {

return;

}

try {

byte[] fileBytes = Files.readAllBytes(pictures[list.getSelectedIndex()].toPath());

short[] fileShorts0 = new short[fileBytes.length / 2];

ByteBuffer.wrap(fileBytes).order(ByteOrder.LITTLE\_ENDIAN).asShortBuffer().get(fileShorts0);

w = fileShorts0[0];

h = fileShorts0[1];

int n = 2;

fileShorts = new short[h][w];

for (int y1 = 0; y1 < h; y1++) {

for (int x1 = 0; x1 < w; x1++) {

fileShorts[y1][x1] = (short) (fileShorts0[n] & 0x3FFF);

n++;

}

}

file.setPreferredSize(new Dimension(w, h));

double pixel;

int curs;

Color c;

Picture = new int[h][w];

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

pixel = fileShorts[i][j];

curs = (int) Math.round((double) pixel \* 255 / 1024);

Picture[i][j] = curs;

c = new Color(curs, curs, curs);

file.addLine(j, i, j, i, c);

}

}

} catch (IOException ex) {

Logger.getLogger(Lab3.class.getName()).log(Level.SEVERE, null, ex);

}

});

//

//Метод лупы

//

file.addMouseMotionListener(new MouseMotionAdapter() {

@Override

public void mouseMoved(MouseEvent e) {

if (Picture == null) {

return;

}

N = (int) selectCoef.getSelectedItem();

int X = e.getX();

int Y = e.getY();

if (X > 0 && Y > 0 && Y < h && X < w) {

lb.setText("Яркость: " + Picture[Y][X]);

}

Color c;

if (!check.isSelected()) { //Без интерполяции

int i = 0, j = 0;

for (int k = Y - (lupa.getHeight() / (N \* 2)); k < Y + (lupa.getHeight() / (N \* 2)); k++) {

for (int l = X - (lupa.getWidth() / (N \* 2)); l < X + (lupa.getWidth() / (N \* 2)); l++) {

if (k > 0 && l > 0 && k < h && l < w) {

int I = Picture[k][l];

c = new Color(I, I, I);

} else {

c = Color.black;

}

for (int z = 0; z < N; z++) {

lupa.addLine(i, j + z, i + N, j + z, c);

}

i += N;

}

j += N;

i = 0;

}

} else { // Интерполяция

int i = 0, j = 0;

int[][] a = new int[lupa.getHeight() / N][lupa.getWidth() / N];

for (int k = Y - (lupa.getHeight() / (N \* 2)); k < Y + (lupa.getHeight() / (N \* 2)); k++) {

for (int l = X - (lupa.getWidth() / (N \* 2)); l < X + (lupa.getWidth() / (N \* 2)); l++) {

if (k > 0 && l > 0 && k < h && l < w) {

a[i][j] = Picture[k][l];

j++;

}

}

i++;

j = 0;

}

int[][] b = new int[lupa.getHeight()][lupa.getWidth()];

interpolate(a, b);

for (int k = 0; k < lupa.getHeight(); k++) {

for (int l = 0; l < lupa.getWidth(); l++) {

lupa.addLine(l, k, l, k, new Color(b[k][l], b[k][l], b[k][l]));

}

}

}

}

});

ourFrame.pack();

ourFrame.setVisible(true);

}

static void interpolate(int a[][], int b[][]) {

int oldWidth = a.length;

int oldHeight = a[0].length;

int newWidth = b.length;

int newHeight = b[0].length;

int i, j;

int h, w;

float t;

float u;

float tmp;

float d1, d2, d3, d4;

int p1, p2, p3, p4;

/\* Окрестные пикселы \*/

for (j = 0; j < newHeight; j++) {

// tmp - точка в пределах локальной области (в одном из пикселей)

// w и h - начало координат локальной системы координат

tmp = (float) (j) / (float) (newHeight - 1) \* (oldHeight - 1);

h = (int) tmp;

if (h < 0) {

h = 0;

} else {

if (h >= oldHeight - 1) {

h = oldHeight - 2;

}

}

u = tmp - h;

for (i = 0; i < newWidth; i++) {

tmp = (float) (i) / (float) (newWidth - 1) \* (oldWidth - 1);

w = (int) tmp;

if (w < 0) {

w = 0;

} else {

if (w >= oldWidth - 1) {

w = oldWidth - 2;

}

}

t = tmp - w;

/\* Коэффициенты \*/

d1 = (1 - t) \* (1 - u);

d2 = t \* (1 - u);

d3 = t \* u;

d4 = (1 - t) \* u;

/\* Окрестные пиксели: a[i][j] \*/

// вершины четырех окружающий пикселей

p1 = a[h][w];

p2 = a[h][w + 1];

p3 = a[h + 1][w + 1];

p4 = a[h + 1][w];

/\* Новый пиксел \*/

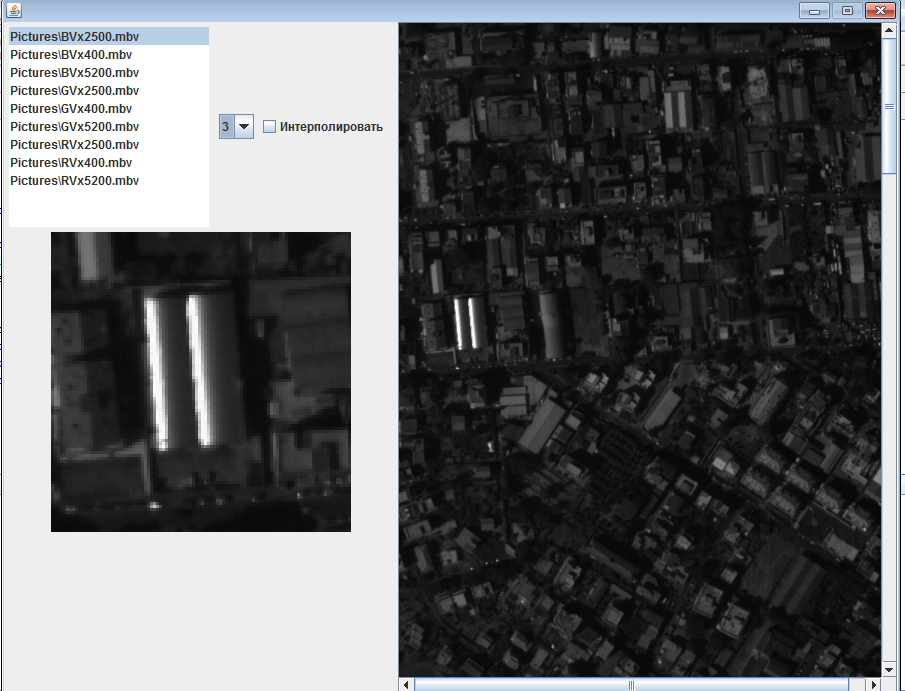
b[j][i] = (int) (p1 \* d1 + p2 \* d2 + p3 \* d3 + p4 \* d4);

}

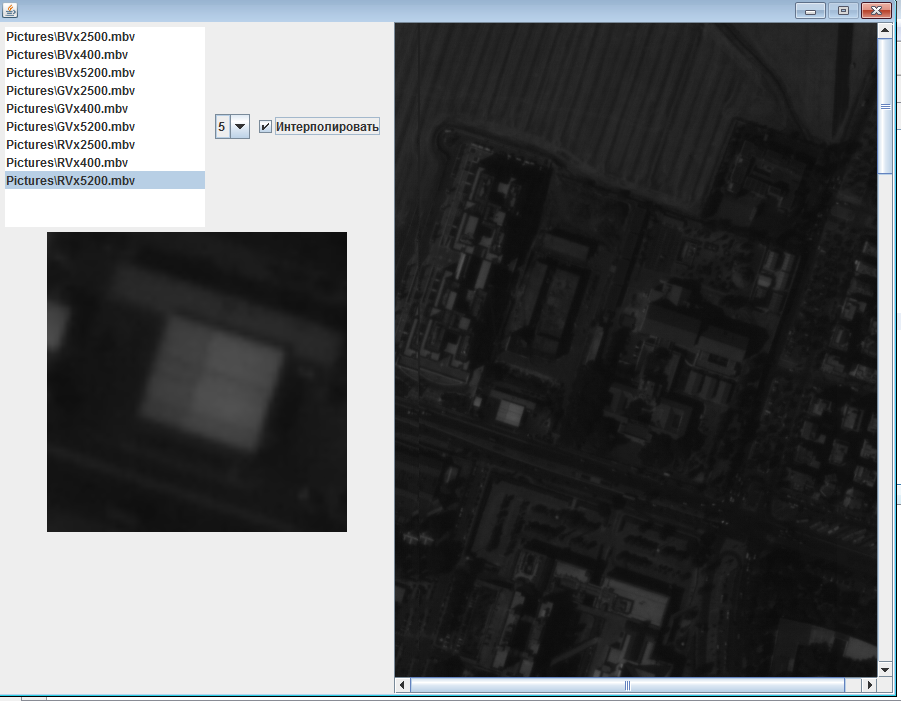
}

}

}



**Результаты:**

****

**Вывод:** на практике изучен метод нахождения промежуточных значений пикселей с помощью метода билинейной интерполяции, реализована лупа, позволяющая увеличивать выбранный фрагмент изображения.