ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №1

За курсом «Обробка та розпізнавання зображень»

Студента групи ПА-19-2

Ільяшенко Єгора Віталійовича

Кафедра комп’ютерних технологій, ДНУ

**Множина Жуліа**

A picture containing background pattern

Description automatically generated

Graphical user interface

Description automatically generated

A picture containing tree, stop, light, background

Description automatically generated

Graphical user interface

Description automatically generated

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Numerics;

namespace Lab\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

comboBox1.SelectedIndex = 0;

}

const float escape\_radius = 2;

int max\_iterations\_count;

const int default\_iteration\_count = 300;

const int canvas\_width = 500;

const int canvas\_height = 500;

Bitmap bmp = null;

private void button1\_Click(object sender, EventArgs e)

{

bmp = new Bitmap(canvas\_width, canvas\_height);

max\_iterations\_count = (int)numericUpDown1.Value;

if(max\_iterations\_count <= 0)

max\_iterations\_count = default\_iteration\_count;

CanvasRefresh();

}

void CanvasRefresh()

{

int iteration = 0;

double zx;

double zy;

string[] c = comboBox1.SelectedItem.ToString().Split('|');

double cx = double.Parse(c[0]);

double cy = double.Parse(c[1]);

for (int x\_pixel = 0; x\_pixel < canvas\_width; x\_pixel++)

for (int y\_pixel = 0; y\_pixel < canvas\_height; y\_pixel++)

{

double x\_pos = (double)(x\_pixel - canvas\_width / 2 + x\_shift) / canvas\_width \* escape\_radius \* scale;

double y\_pos = (double)(y\_pixel - canvas\_height / 2 + y\_shift) / canvas\_height \* escape\_radius \* scale;

zx = x\_pos;

zy = y\_pos;

iteration = 0;

while (zx \* zx + zy \* zy < escape\_radius && iteration < max\_iterations\_count)

{

double new\_zx = zx \* zx - zy \* zy + cx;

double new\_zy = 2 \* zx \* zy + cy;

zx = new\_zx;

zy = new\_zy;

iteration++;

}

if (iteration == max\_iterations\_count)

bmp.SetPixel(x\_pixel, y\_pixel, Color.Black);

else

bmp.SetPixel(x\_pixel, y\_pixel, Color.FromArgb((int)Math.Min((256f \* trackBar1.Value \* iteration / max\_iterations\_count), 255), 0, 0));

}

pictureBox1.Image = bmp;

Refresh();

}

int x\_shift = 0;

int y\_shift = 0;

double scale = 1;

protected override bool ProcessCmdKey(ref Message msg, Keys keyData)

{

if (keyData == Keys.Q)

{

scale /= 1.5;

CanvasRefresh();

}

if (keyData == Keys.E)

{

scale \*= 1.5;

CanvasRefresh();

}

if (keyData == Keys.W)

{

y\_shift -= 50;

CanvasRefresh();

}

if (keyData == Keys.A)

{

x\_shift -= 50;

CanvasRefresh();

}

if (keyData == Keys.S)

{

y\_shift += 50;

CanvasRefresh();

}

if (keyData == Keys.D)

{

x\_shift += 50;

CanvasRefresh();

}

return base.ProcessCmdKey(ref msg, keyData);

}

}

}