Московский авиационный институт

Факультет №3. Системы управления, информатика и электроэнергетика.

Кафедра №304.

Пояснительная записка

к курсовому проекту по курсу:

«Объектно-ориентированное программирование».

Тема работы:

«Фрактальные изображения»

Студент

Группы М3О-224Бк-17

Архинчеев С.Б.

Принял:

Чечиков Ю. Б.

Москва 2019 г.

**Содержание**

**1. Теоретическая часть 3**

**1.1 Алгоритмическая база алгоритма 4**

**1.1.1 Фрактал Мандельброта и фрактал Жюлиа 4**

**1.1.2 Фрактал Мандельброта с видоизмененным алгоритмом 4**

**1.1.3 Трикорн фрактал 4**

**2. Структура программы 5**

**2.1 Диаграмма классов 5**

**2.2 Схема алгоритма 7**

**3. Текст программы 8**

**4. Результат работы программы 34**

**1. Теоретическая часть.**

Для разработки программы, генерирующей фрактальные изображения (далее фракталы) необходимо дать определение фракталу.

Фрактал - множество, обладающее свойством самоподобия (объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет ту же форму, что и одна или более частей). В математике под фракталами понимают множества точек в евклидовом пространстве, имеющие дробную метрическую размерность, либо метрическую размерность, отличную от топологической, поэтому их следует отличать от прочих геометрических фигур, ограниченных конечным числом звеньев. Самоподобные фигуры, повторяющиеся конечное число раз, называются предфракталами.

Фракталы естественным образом возникают при изучении нелинейных динамических систем. Наиболее изучен случай, когда динамическая система задаётся итерациями многочлена или голоморфной функции комплексной переменной на плоскости.

В данной работе рассматриваются следующие фракталы:

* Фрактал на основе множества Мандельброта;
* Фрактал на основе множества Жюлиа;
* Фрактал на основе множества Мандельброта с видоизмененным алгоритмом;
* Фрактал на основе множества Мандельброта, более известный как Трикорн;

Множество Жюлиа - множество точек бифуркации для многочлена (или другой похожей функции), то есть тех значений *z0*, для которых поведение последовательности *zn* может резко меняться при сколь угодно малых изменениях *z0*.

Другой вариант получения фрактальных множеств — введение параметра в многочлен *F(z)* и рассмотрение множества тех значений параметра, при которых последовательность *zn* демонстрирует определённое поведение при фиксированном *z0*.

Множество Мандельброта – это множество всех *сС,* при которых *zn* для , и *z0* не стремится к бесконечности (в данной работе *z0=0*).

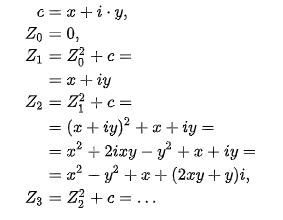
То есть, это множество таких *c*, для которых существует такое действительное *R*, что неравенство |*zn*| < *R* выполняется при всех натуральных *n*.

**1.1 Алгоритмическая база алгоритма.**

**1.1.1 Фрактал Мандельброта и фрактал Жюлиа.**

В алгоритмах построения описанных выше фракталов используется следующее соотношение: .

Распишем начало итераций:



и так далее.

Соответственно это делается до тех пор пока выполняется условие |*zn*| < *R*, либо пока не выполнятся все итерации. В данной работе *R=2*.

**1.1.2 Фрактал Мандельброта с видоизмененным алгоритмом.**

Для разбора построения следующего фрактала необходимо раскрыть условие:

Чтобы получить видоизмененный фрактал Мандельброта необходимо изменить условие следующим образом: .

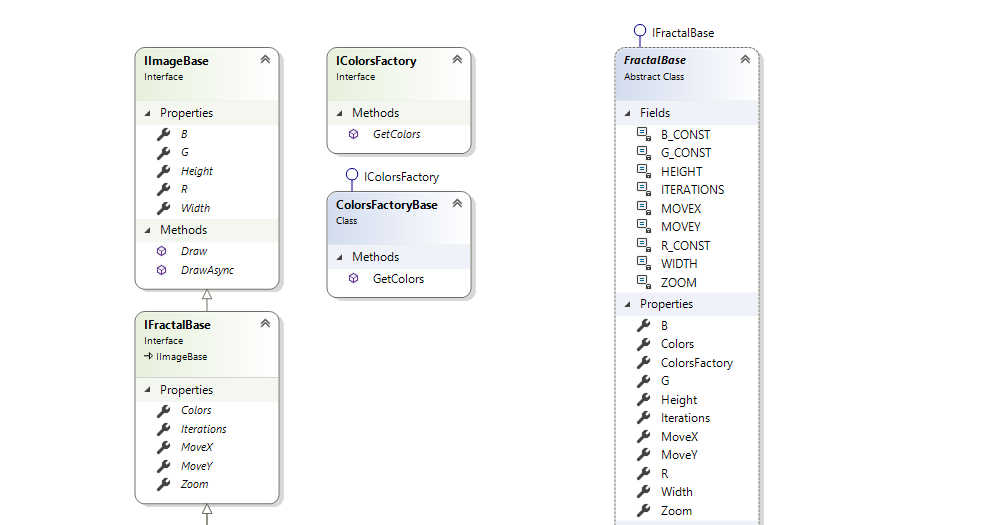
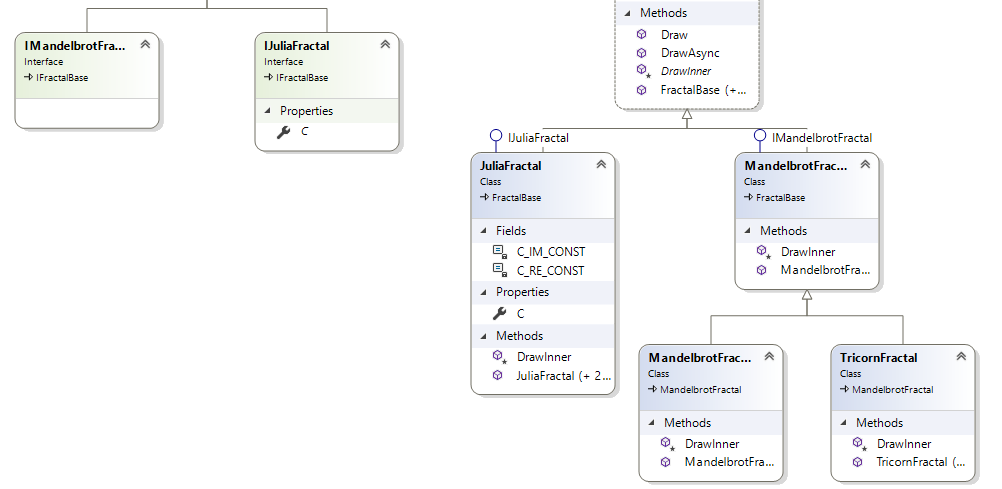
**1.1.3 Трикорн фрактал.**

Для построения Трикорн фрактала требуется соотношение отличное от того, что используется в описанных выше фракталах: , где комплексно сопряженная функция к .

Комплексно сопряженной функцией к функции называется функция вида , что означает что в рассмотренных выше итерациях необходимо поменять знак у мнимой части функции.

**2. Структура программы.**

**2.1 Диаграмма классов**

**Интерфейсы:**

1) IImageBase – интерфейс базового изображения, объект реализующий его может быть нарисован;

2) IFractalBase – интерфейс базового фрактала, наследуется от IImageBase;

3) IMandelbrotFractal – интерфейс фрактала Мандельброта, который описывает необходимые свойтсва и методы для визуализации фрактала Мандельброта, наследуется от IFractalBase;

4) IJuliaFractal – интерфейс фрактала Жюлиа, который описывает необходимые свойтсва и методы для визуализации фрактала Жюлиа, наследуется от IFractalBase;

5) IColorsFactory – интерфейс фабрики цветов, позволяет сформировать цветовую гамму;

**Классы:**

1) ColorsFactoryBase – реализует интерфейс IColorsFactory;

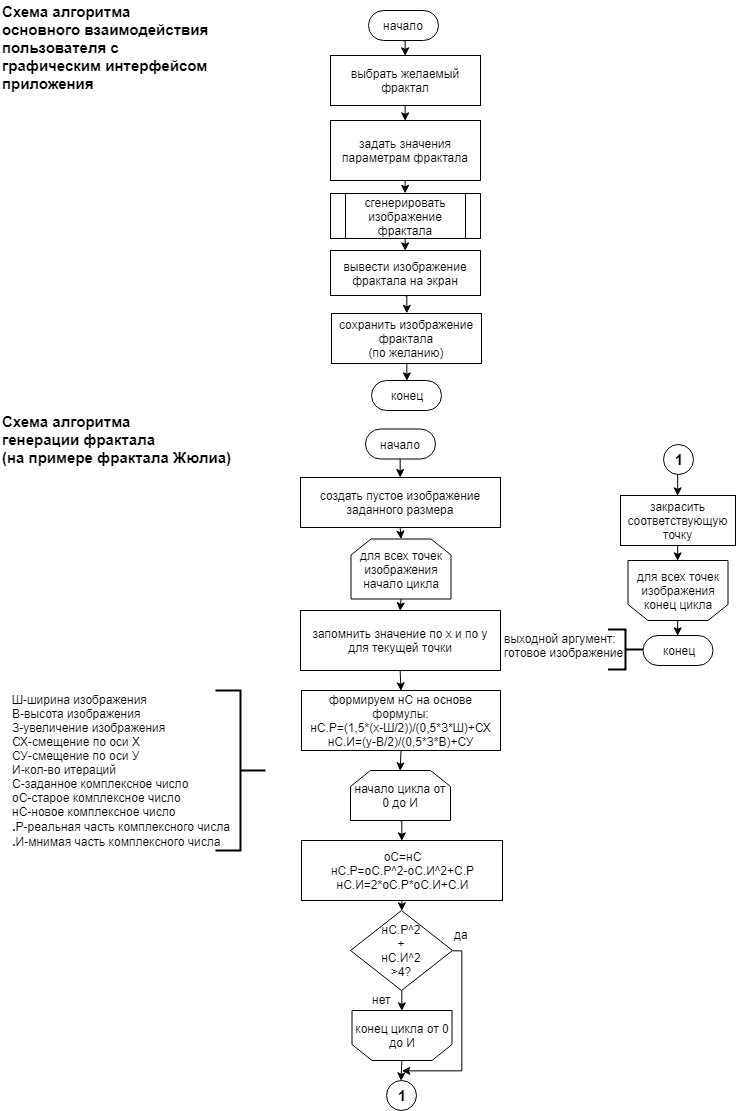
2) FractalBase – абстрактный класс, представляющий базовый фрактал, который может быть нарисован, реализует интерфейс IFractalBase;

3) JuliaFractal – класс фрактала Жюлиа, наследует FractalBase и реализует IJuliaFractal;

4) MandelbrotFractal – класс фрактала Мандельброта, наследует FractalBase и реализует IMandelbrotFractal;

5) MandelbrotFractalModernized – класс детализированного фрактала Мандельброта, наследует MandelbrotFractal;

6) TricornFractal – класс фрактала Трикорн, наследует MandelbrotFractal;

**2.2 Схема алгоритма.**

**3. Текст программы.**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Threading;

using System.Windows.Forms;

using Fractals;

namespace FractalsForm

{

/// <summary>

/// Класс основной формы приложения.

/// </summary>

public partial class Fractals : Form

{

private Dictionary<int, FractalViewModel> \_fractalviewmodels = new Dictionary<int, FractalViewModel>();

private IFractalBase \_selectedFractal;

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Fractals"/> с базовыми значениями.

/// </summary>

public Fractals()

{

InitializeComponent();

}

/// <summary>

/// Обработчик события загрузки формы.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form1\_Load(object sender, EventArgs e)

{

FormInitialize();

}

/// <summary>

/// Обработка события нажатия на кнопку создать на вкладке Мандельброт Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private async void ButtonGenerateMF\_Click(object sender, EventArgs e)

{

ButtonGenerateMF.Enabled = false;

var mf = \_selectedFractal as MandelbrotFractal;

if (mf == null)

return;

int temp1;

double temp2;

mf.R = int.TryParse(TextBoxRMF.Text, out temp1) ? temp1 % 256 : mf.R;

mf.B = int.TryParse(TextBoxBMF.Text, out temp1) ? temp1 % 256 : mf.B;

mf.G = int.TryParse(TextBoxGMF.Text, out temp1) ? temp1 % 256 : mf.G;

mf.Width = int.TryParse(TextBoxWidthMF.Text, out temp1) ? temp1 : mf.Width;

mf.Height = int.TryParse(TextBoxHeightMF.Text, out temp1) ? temp1 : mf.Height;

mf.Zoom = double.TryParse(TextBoxZoomMF.Text, out temp2) ? temp2 : mf.Zoom;

mf.Iterations = int.TryParse(TextBoxIterationsMF.Text, out temp1) ? temp1 : mf.Iterations;

mf.MoveX = double.TryParse(TextBoxMoveXMF.Text, out temp2) ? temp2 : mf.MoveX;

mf.MoveY = double.TryParse(TextBoxMoveYMF.Text, out temp2) ? temp2 : mf.MoveY;

\_fractalviewmodels[0].Image = await mf.DrawAsync();

PictureBoxMandelbrot.Image = \_fractalviewmodels[0].Image;

ButtonGenerateMF.Enabled = true;

}

/// <summary>

/// Обработка события нажатия на кнопку сохранить на вкладке Мандельброт Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

\_fractalviewmodels[0].Image.Save("MF.jpg");

}

/// <summary>

/// Обработка события смены вкладки.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (\_fractalviewmodels.ContainsKey(tabControl1.SelectedIndex))

\_selectedFractal = \_fractalviewmodels[tabControl1.SelectedIndex].Fractal;

}

/// <summary>

/// Обработчики событий нажатия на кнопки клавиатуры для текстбоксов.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

#region KeyPress

private void TextBoxG\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxR\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxB\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxWidth\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxHeight\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxIterations\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxZoom\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == ',') || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxMoveX\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == ',') || (key == '-') || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxMoveY\_1\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == ',') || (key == '-') || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxC\_Re\_2\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == ',') || (key == '-') || (key == 8)))

{

e.Handled = true;

}

}

private void TextBoxC\_Im\_2\_KeyPress(object sender, KeyPressEventArgs e)

{

var tb = sender as TextBox;

var key = e.KeyChar;

if (!(char.IsDigit(key) || (key == ',') || (key == '-') || (key == 8)))

{

e.Handled = true;

}

}

#endregion

/// <summary>

/// Обработка события нажатия на кнопку сохранить на вкладке Жюлиа Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

\_fractalviewmodels[1].Image.Save("JF.jpg");

}

/// <summary>

/// Обработка события нажатия на кнопку создать на вкладке Жюлиа Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private async void ButtonGenerateJF\_Click(object sender, EventArgs e)

{

ButtonGenerateJF.Enabled = false;

var jf = \_selectedFractal as JuliaFractal;

if (jf == null)

return;

int temp1;

double temp2, tempR, tempI;

jf.R = int.TryParse(TextBoxRJF.Text, out temp1) ? temp1 % 256 : jf.R;

jf.B = int.TryParse(TextBoxBJF.Text, out temp1) ? temp1 % 256 : jf.B;

jf.G = int.TryParse(TextBoxGJF.Text, out temp1) ? temp1 % 256 : jf.G;

jf.Width = int.TryParse(TextBoxWidthJF.Text, out temp1) ? temp1 : jf.Width;

jf.Height = int.TryParse(TextBoxHeightJF.Text, out temp1) ? temp1 : jf.Height;

jf.Zoom = double.TryParse(TextBoxZoomJF.Text, out temp2) ? temp2 : jf.Zoom;

jf.Iterations = int.TryParse(TextBoxIterationsJF.Text, out temp1) ? temp1 : jf.Iterations;

jf.MoveX = double.TryParse(TextBoxMoveXJF.Text, out temp2) ? temp2 : jf.MoveX;

jf.MoveY = double.TryParse(TextBoxMoveYJF.Text, out temp2) ? temp2 : jf.MoveY;

tempR = double.TryParse(TextBoxC\_ReJF.Text, out temp2) ? temp2 : jf.C.Real;

tempI = double.TryParse(TextBoxC\_ImJF.Text, out temp2) ? temp2 : jf.C.Imaginary;

jf.C = new System.Numerics.Complex(tempR, tempI);

\_fractalviewmodels[1].Image = await jf.DrawAsync();

PictureBoxJulia.Image = \_fractalviewmodels[1].Image;

ButtonGenerateJF.Enabled = true;

}

/// <summary>

/// Обработка события нажатия на кнопку создать на вкладке Детализированный Мандельброт Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private async void ButtonGenerateMMF\_Click(object sender, EventArgs e)

{

ButtonGenerateMMF.Enabled = false;

var mmf = \_selectedFractal as MandelbrotFractalModernized;

if (mmf == null)

return;

int temp1;

double temp2;

mmf.R = int.TryParse(TextBoxRMMF.Text, out temp1) ? temp1 % 256 : mmf.R;

mmf.B = int.TryParse(TextBoxBMMF.Text, out temp1) ? temp1 % 256 : mmf.B;

mmf.G = int.TryParse(TextBoxGMMF.Text, out temp1) ? temp1 % 256 : mmf.G;

mmf.Width = int.TryParse(TextBoxWidthMMF.Text, out temp1) ? temp1 : mmf.Width;

mmf.Height = int.TryParse(TextBoxHeightMMF.Text, out temp1) ? temp1 : mmf.Height;

mmf.Zoom = double.TryParse(TextBoxZoomMMF.Text, out temp2) ? temp2 : mmf.Zoom;

mmf.Iterations = int.TryParse(TextBoxIterationsMMF.Text, out temp1) ? temp1 : mmf.Iterations;

mmf.MoveX = double.TryParse(TextBoxMoveXMMF.Text, out temp2) ? temp2 : mmf.MoveX;

mmf.MoveY = double.TryParse(TextBoxMoveYMMF.Text, out temp2) ? temp2 : mmf.MoveY;

\_fractalviewmodels[2].Image = await mmf.DrawAsync();

PictureBoxMandelbrotModernized.Image = \_fractalviewmodels[2].Image;

ButtonGenerateMMF.Enabled = true;

}

/// <summary>

/// Обработка события нажатия на кнопку сохранить на вкладке Детализированный Мандельброт Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

\_fractalviewmodels[2].Image.Save("MMF.jpg");

}

/// <summary>

/// Обработка события нажатия на кнопку создать на вкладке Трикорн Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private async void ButtonGenerateTF\_Click(object sender, EventArgs e)

{

ButtonGenerateTF.Enabled = false;

var tf = \_selectedFractal as TricornFractal;

if (tf == null)

return;

int temp1;

double temp2;

tf.R = int.TryParse(TextBoxRTF.Text, out temp1) ? temp1 % 256 : tf.R;

tf.B = int.TryParse(TextBoxBTF.Text, out temp1) ? temp1 % 256 : tf.B;

tf.G = int.TryParse(TextBoxGTF.Text, out temp1) ? temp1 % 256 : tf.G;

tf.Width = int.TryParse(TextBoxWidthTF.Text, out temp1) ? temp1 : tf.Width;

tf.Height = int.TryParse(TextBoxHeightTF.Text, out temp1) ? temp1 : tf.Height;

tf.Zoom = double.TryParse(TextBoxZoomTF.Text, out temp2) ? temp2 : tf.Zoom;

tf.Iterations = int.TryParse(TextBoxIterationsTF.Text, out temp1) ? temp1 : tf.Iterations;

tf.MoveX = double.TryParse(TextBoxMoveXTF.Text, out temp2) ? temp2 : tf.MoveX;

tf.MoveY = double.TryParse(TextBoxMoveYTF.Text, out temp2) ? temp2 : tf.MoveY;

\_fractalviewmodels[3].Image = await tf.DrawAsync();

PictureBoxTricorn.Image = \_fractalviewmodels[3].Image;

ButtonGenerateTF.Enabled = true;

}

/// <summary>

/// Обработка события нажатия на кнопку сохранить на вкладке Трикорн Фрактал.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

\_fractalviewmodels[3].Image.Save("TF.jpg");

}

/// <summary>

/// Установка начальных значений полям ввода.

/// </summary>

private void FormInitialize()

{

// ToDo: загружать словарь фракталов извне

\_fractalviewmodels.Add(0, new FractalViewModel { Fractal = new MandelbrotFractal() });

\_fractalviewmodels.Add(1, new FractalViewModel { Fractal = new JuliaFractal() });

\_fractalviewmodels.Add(2, new FractalViewModel { Fractal = new MandelbrotFractalModernized() });

\_fractalviewmodels.Add(3, new FractalViewModel { Fractal = new TricornFractal() });

\_selectedFractal = \_fractalviewmodels[0].Fractal;

tabControl1.SelectedIndex = 0;

TextBoxRMF.Text = \_fractalviewmodels[0].Fractal.R.ToString();

TextBoxRJF.Text = \_fractalviewmodels[1].Fractal.R.ToString();

TextBoxRMMF.Text = \_fractalviewmodels[2].Fractal.R.ToString();

TextBoxRTF.Text = \_fractalviewmodels[3].Fractal.R.ToString();

TextBoxBMF.Text = \_fractalviewmodels[0].Fractal.B.ToString();

TextBoxBJF.Text = \_fractalviewmodels[1].Fractal.B.ToString();

TextBoxBMMF.Text = \_fractalviewmodels[2].Fractal.B.ToString();

TextBoxBTF.Text = \_fractalviewmodels[3].Fractal.B.ToString();

TextBoxGMF.Text = \_fractalviewmodels[0].Fractal.G.ToString();

TextBoxGJF.Text = \_fractalviewmodels[1].Fractal.G.ToString();

TextBoxGMMF.Text = \_fractalviewmodels[2].Fractal.G.ToString();

TextBoxGTF.Text = \_fractalviewmodels[3].Fractal.G.ToString();

TextBoxWidthMF.Text = \_fractalviewmodels[0].Fractal.Width.ToString();

TextBoxWidthJF.Text = \_fractalviewmodels[1].Fractal.Width.ToString();

TextBoxWidthMMF.Text = \_fractalviewmodels[2].Fractal.Width.ToString();

TextBoxWidthTF.Text = \_fractalviewmodels[3].Fractal.Width.ToString();

TextBoxHeightMF.Text = \_fractalviewmodels[0].Fractal.Height.ToString();

TextBoxHeightJF.Text = \_fractalviewmodels[1].Fractal.Height.ToString();

TextBoxHeightMMF.Text = \_fractalviewmodels[2].Fractal.Height.ToString();

TextBoxHeightTF.Text = \_fractalviewmodels[3].Fractal.Height.ToString();

TextBoxZoomMF.Text = \_fractalviewmodels[0].Fractal.Zoom.ToString();

TextBoxZoomJF.Text = \_fractalviewmodels[1].Fractal.Zoom.ToString();

TextBoxZoomMMF.Text = \_fractalviewmodels[2].Fractal.Zoom.ToString();

TextBoxZoomTF.Text = \_fractalviewmodels[3].Fractal.Zoom.ToString();

TextBoxIterationsMF.Text = \_fractalviewmodels[0].Fractal.Iterations.ToString();

TextBoxIterationsJF.Text = \_fractalviewmodels[1].Fractal.Iterations.ToString();

TextBoxIterationsMMF.Text = \_fractalviewmodels[2].Fractal.Iterations.ToString();

TextBoxIterationsTF.Text = \_fractalviewmodels[3].Fractal.Iterations.ToString();

TextBoxMoveXMF.Text = \_fractalviewmodels[0].Fractal.MoveX.ToString();

TextBoxMoveXJF.Text = \_fractalviewmodels[1].Fractal.MoveX.ToString();

TextBoxMoveXMMF.Text = \_fractalviewmodels[2].Fractal.MoveX.ToString();

TextBoxMoveXTF.Text = \_fractalviewmodels[3].Fractal.MoveX.ToString();

TextBoxMoveYMF.Text = \_fractalviewmodels[0].Fractal.MoveY.ToString();

TextBoxMoveYJF.Text = \_fractalviewmodels[1].Fractal.MoveY.ToString();

TextBoxMoveYMMF.Text = \_fractalviewmodels[2].Fractal.MoveY.ToString();

TextBoxMoveYTF.Text = \_fractalviewmodels[3].Fractal.MoveY.ToString();

var temp = \_fractalviewmodels[1].Fractal as JuliaFractal;

TextBoxC\_ReJF.Text = temp.C.Real.ToString();

TextBoxC\_ImJF.Text = temp.C.Imaginary.ToString();

}

/// <summary>

/// Обратка события выбора смены локализации на русский язык.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Thread.CurrentThread.CurrentUICulture = new CultureInfo("ru-RU");

\_fractalviewmodels.Clear();

Controls.Clear();

InitializeComponent();

FormInitialize();

}

/// <summary>

/// Обратка события выбора смены локализации на английский язык.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Thread.CurrentThread.CurrentUICulture = new CultureInfo("en-US");

\_fractalviewmodels.Clear();

Controls.Clear();

InitializeComponent();

FormInitialize();

}

}

}

**Конец файла;**

using System.Drawing;

using Fractals;

namespace FractalsForm

{

/// <summary>

/// Экземпляр модели данных фрактала.

/// </summary>

class FractalViewModel

{

/// <summary>

/// Изображение фрактала.

/// </summary>

public Bitmap Image { get; set; }

/// <summary>

/// Фрактал.

/// </summary>

public IFractalBase Fractal { get; set; }

}

}

**Конец файла;**

using System.Drawing;

namespace Fractals

{

/// <summary>

/// Интерфейс фабрики цветов.

/// </summary>

public interface IColorsFactory

{

/// <summary>

/// Возвращает массив цветов.

/// </summary>

/// <returns>Массив цветов</returns>

Color[] GetColors(int R, int G, int B);

}

}

**Конец файла;**

using System.Drawing;

using System.Threading.Tasks;

namespace Fractals

{

/// <summary>

/// Интерфейс базового изображения.

/// </summary>

public interface IImageBase

{

/// <summary>

/// Осуществляет создание изображения.

/// </summary>

/// <returns>Готовое изображение</returns>

Bitmap Draw();

/// <summary>

/// Осуществляет ассинхронное создание изображения.

/// </summary>

/// <returns></returns>

Task<Bitmap> DrawAsync();

/// <summary>

/// Возвращает ширину изображения.

/// </summary>

int Width { get; set; }

/// <summary>

/// Возвращает высоту изображения.

/// </summary>

int Height { get; set; }

/// <summary>

/// Возвращает параметр R цветовой гаммы изображения.

/// </summary>

int R { get; set; }

/// <summary>

/// Возвращает параметр G цветовой гаммы изображения.

/// </summary>

int G { get; set; }

/// <summary>

/// Возвращает параметр B цветовой гаммы изображения.

/// </summary>

int B { get; set; }

}

}

**Конец файла;**

using System.Drawing;

namespace Fractals

{

/// <summary>

/// Интерфейс базового фрактала.

/// </summary>

public interface IFractalBase : IImageBase

{

/// <summary>

/// Возвращает количество итераций.

/// </summary>

int Iterations { get; set; }

/// <summary>

/// Возвращает зум.

/// </summary>

double Zoom { get; set; }

/// <summary>

/// Возвращает массив цветов.

/// </summary>

/// <returns>Массив цветов</returns>

Color[] Colors { get; set; }

/// <summary>

/// Возвращает смещение по Х.

/// </summary>

double MoveX { get; set; }

/// <summary>

/// Возвращает смещение по Y.

/// </summary>

double MoveY { get; set; }

}

}

**Конец файла;**

using System.Numerics;

namespace Fractals

{

/// <summary>

/// Интерфейс фрактала Жюлиа.

/// </summary>

public interface IJuliaFractal : IFractalBase

{

/// <summary>

/// Возвращает комплексное число необходимое для вычислений в алгоритме Жюлиа.

/// </summary>

Complex C { get; set; }

}

}

**Конец файла;**

namespace Fractals

{

/// <summary>

/// Интерфейс фрактала Мандельброта.

/// </summary>

public interface IMandelbrotFractal : IFractalBase

{

}

}

**Конец файла;**

using System.Linq;

using System.Drawing;

namespace Fractals

{

/// <inheritdoc/>

public class ColorsFactoryBase : IColorsFactory

{

/// <inheritdoc cref="IColorsFactory.GetColors"/>

public Color[] GetColors(int R, int G, int B)

{

return Enumerable.Range(0, 256)

.Select(c => Color.FromArgb(((c & R) \* 85) % 256, ((c >> G) \* 36) % 256, ((c >> R & B) \* 36) % 256))

.ToArray();

}

}

}

**Конец файла;**

using System.Drawing;

using FastBitmapLib;

using System.Threading.Tasks;

namespace Fractals

{

/// <summary>

/// Представляет абстрактный экземпляр базового фрактала.

/// </summary>

public abstract class FractalBase : IFractalBase

{

private const int ITERATIONS = 300;

private const int R\_CONST = 1;

private const int B\_CONST = 1;

private const int G\_CONST = 1;

private const int WIDTH = 1920;

private const int HEIGHT = 1080;

private const double ZOOM = 1;

private const double MOVEX = 0;

private const double MOVEY = 0;

/// <inheritdoc cref="IFractalBase.Iterations"/>

public int Iterations { get; set; }

/// <inheritdoc cref="IFractalBase.Zoom"/>

public double Zoom { get; set; }

/// <inheritdoc cref="IFractalBase.MoveX"/>

public double MoveX { get; set; }

/// <inheritdoc cref="IFractalBase.MoveY"/>

public double MoveY { get; set; }

/// <inheritdoc cref="IImageBase.Width"/>

public int Width { get; set; }

/// <inheritdoc cref="IImageBase.Height"/>

public int Height { get; set; }

/// <inheritdoc cref="IImageBase.R"/>

public int R { get; set; }

/// <inheritdoc cref="IImageBase.G"/>

public int G { get; set; }

/// <inheritdoc cref="IImageBase.B"/>

public int B { get; set; }

/// <inheritdoc cref="IFractalBase.Colors"/>

public Color[] Colors { get; set; }

/// <inheritdoc cref="IColorsFactory.GetColors(int, int, int)"/>

public IColorsFactory ColorsFactory { get; set; }

/// <inheritdoc cref="IImageBase.Draw"/>

public Bitmap Draw()

{

Colors = ColorsFactory.GetColors(R, G, B);

var fractal = new Bitmap(Width, Height);

FastBitmap fb = new FastBitmap(fractal);

fb.Lock();

DrawInner(fb);

fb.Unlock();

return fractal;

}

/// <inheritdoc cref="IImageBase.Draw"/>

public Task<Bitmap> DrawAsync()

{

return Task.Run(() => Draw());

}

/// <summary>

/// Метод, выполняющий отрисовку фрактала.

/// </summary>

/// <param name="fractal">Изображение фрактала</param>

protected abstract void DrawInner(FastBitmap fractal);

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="FractalBase"/> с заданными значениями поля фабрика цветов.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

public FractalBase(IColorsFactory colorsFactory) : this(colorsFactory, R\_CONST, B\_CONST, G\_CONST, WIDTH, HEIGHT, ZOOM, ITERATIONS, MOVEX, MOVEY)

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="FractalBase"/> с базовыми значениями.

/// </summary>

public FractalBase() : this(new ColorsFactoryBase(), R\_CONST, B\_CONST, G\_CONST, WIDTH, HEIGHT, ZOOM, ITERATIONS, MOVEX, MOVEY)

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="FractalBase"/> с заданными значениями.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

/// <param name="r">Компонент цвета r</param>

/// <param name="b">Компонент цвета b</param>

/// <param name="g">Компонент цвета g</param>

/// <param name="width">Ширина</param>

/// <param name="height">Высота</param>

/// <param name="zoom">Увеличение</param>

/// <param name="iterations">Кол-во итераций</param>

/// <param name="movex">Смещение по Х</param>

/// <param name="movey">Смещение по Y</param>

protected FractalBase(IColorsFactory colorsFactory, int r, int b, int g, int width, int height, double zoom, int iterations, double movex, double movey)

{

ColorsFactory = colorsFactory;

R = r;

G = g;

B = b;

Width = width;

Height = height;

Zoom = zoom;

Iterations = iterations;

MoveX = movex;

MoveY = movey;

}

}

}

**Конец файла;**

using System.Numerics;

using FastBitmapLib;

using System.Linq;

namespace Fractals

{

/// <summary>

/// Представляет экземпляр фрактала Жюлиа.

/// </summary>

public class JuliaFractal : FractalBase, IJuliaFractal

{

private const double C\_RE\_CONST = -0.70176;

private const double C\_IM\_CONST = -0.3842;

/// <inheritdoc cref="IJuliaFractal.C"/>

public Complex C { get; set; }

/// <inheritdoc cref="FractalBase.DrawInner"/>

protected override void DrawInner(FastBitmap fractal)

{

Enumerable.Range(0, Width \* Height).AsParallel().ForAll(xy =>

{

Complex newC, oldC;

int x = xy % Width, y = xy / Width, i = 0;

newC = new Complex(1.5 \* (x - Width / 2) / (0.5 \* Zoom \* Width) + MoveX,

(y - Height / 2) / (0.5 \* Zoom \* Height) + MoveY);

for (; i < Iterations; i++)

{

oldC = newC;

newC = new Complex(oldC.Real \* oldC.Real - oldC.Imaginary \* oldC.Imaginary + C.Real,

2 \* oldC.Real \* oldC.Imaginary + C.Imaginary);

if ((newC.Real \* newC.Real + newC.Imaginary \* newC.Imaginary) > 4)

break;

}

fractal.SetPixel(x, y, Colors[i % 256]);

});

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="JuliaFractal"/> с заданным значением фабрики цветов.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

public JuliaFractal(IColorsFactory colorsFactory) : base(colorsFactory)

{

C = new Complex(C\_RE\_CONST, C\_IM\_CONST);

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="JuliaFractal"/> с базовыми значениями.

/// </summary>

public JuliaFractal() : this(new ColorsFactoryBase())

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="JuliaFractal"/> с заданными значениями.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

/// <param name="r">Компонент цвета r</param>

/// <param name="b">Компонент цвета b</param>

/// <param name="g">Компонент цвета g</param>

/// <param name="width">Ширина</param>

/// <param name="height">Высота</param>

/// <param name="zoom">Увеличение</param>

/// <param name="iterations">Кол-во итераций</param>

/// <param name="movex">Смещение по Х</param>

/// <param name="movey">Смещение по Y</param>

/// <param name="c">Комплексное число для алгоритма Жюлиа</param>

protected JuliaFractal(IColorsFactory colorsFactory, int r, int b, int g, int width, int height, double zoom, int iterations, double movex, double movey, Complex c) : base(colorsFactory, r, b, g, width, height, zoom, iterations, movex, movey)

{

C = c;

}

}

}

**Конец файла;**

using System.Numerics;

using FastBitmapLib;

using System.Linq;

namespace Fractals

{

/// <summary>

/// Представляет экземпляр фрактала Мандельброта.

/// </summary>

public class MandelbrotFractal : FractalBase, IMandelbrotFractal

{

/// <inheritdoc cref="FractalBase.DrawInner"/>

protected override void DrawInner(FastBitmap fractal)

{

Enumerable.Range(0, Width \* Height).AsParallel().ForAll(xy =>

{

Complex C, newC, oldC;

int x = xy % Width, y = xy / Width, i = 0;

C = new Complex((x - Width / 2) / (0.5 \* Zoom \* Width) + MoveX,

(y - Height / 2) / (0.5 \* Zoom \* Height) + MoveY);

newC = new Complex(0, 0);

for (; i < Iterations; i++)

{

oldC = newC;

newC = new Complex(oldC.Real \* oldC.Real - oldC.Imaginary \* oldC.Imaginary + C.Real,

2 \* oldC.Real \* oldC.Imaginary + C.Imaginary);

if ((newC.Real \* newC.Real + newC.Imaginary \* newC.Imaginary) > 4)

break;

}

fractal.SetPixel(x, y, Colors[i % 256]);

});

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="MandelbrotFractal"/> с заданным значением фабрики цветов.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

public MandelbrotFractal(IColorsFactory colorsFactory) : base(colorsFactory)

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="MandelbrotFractal"/> с базовыми значениями.

/// </summary>

public MandelbrotFractal() : base()

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="MandelbrotFractal"/> с заданными значениями.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

/// <param name="r">Компонент цвета r</param>

/// <param name="b">Компонент цвета b</param>

/// <param name="g">Компонент цвета g</param>

/// <param name="width">Ширина</param>

/// <param name="height">Высота</param>

/// <param name="zoom">Увеличение</param>

/// <param name="iterations">Кол-во итераций</param>

/// <param name="movex">Смещение по Х</param>

/// <param name="movey">Смещение по Y</param>

protected MandelbrotFractal(IColorsFactory colorsFactory, int r, int b, int g, int width, int height, double zoom, int iterations, double movex, double movey) : base(colorsFactory, r, b, g, width, height, zoom, iterations, movex, movey)

{

}

}

}

**Конец файла;**

using System.Numerics;

using FastBitmapLib;

using System.Linq;

namespace Fractals

{

/// <summary>

/// Представляет экземпляр модернизированного фрактала Мандельброта.

/// </summary>

public class MandelbrotFractalModernized : MandelbrotFractal

{

/// <inheritdoc cref="FractalBase.DrawInner"/>

protected override void DrawInner(FastBitmap fractal)

{

Enumerable.Range(0, Width \* Height).AsParallel().ForAll(xy =>

{

Complex C, newC, oldC;

int x = xy % Width, y = xy / Width, i = 0;

C = new Complex((x - Width / 2) / (0.5 \* Zoom \* Width) + MoveX,

(y - Height / 2) / (0.5 \* Zoom \* Height) + MoveY);

newC = new Complex(0, 0);

for (; i < Iterations; i++)

{

oldC = newC;

newC = new Complex(oldC.Real \* oldC.Real - oldC.Imaginary \* oldC.Imaginary + C.Real,

2 \* oldC.Real \* oldC.Imaginary + C.Imaginary);

if ((newC.Real \* newC.Real - newC.Imaginary \* newC.Imaginary) > 4)

break;

}

fractal.SetPixel(x, y, Colors[i % 256]);

});

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="MandelbrotFractalModernized"/> с заданным значением фабрики цветов.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

public MandelbrotFractalModernized(IColorsFactory colorsFactory) : base(colorsFactory)

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="MandelbrotFractalModernized"/> с базовыми значениями.

/// </summary>

public MandelbrotFractalModernized() : base()

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="MandelbrotFractalModernized"/> с заданными значениями.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

/// <param name="r">Компонент цвета r</param>

/// <param name="b">Компонент цвета b</param>

/// <param name="g">Компонент цвета g</param>

/// <param name="width">Ширина</param>

/// <param name="height">Высота</param>

/// <param name="zoom">Увеличение</param>

/// <param name="iterations">Кол-во итераций</param>

/// <param name="movex">Смещение по Х</param>

/// <param name="movey">Смещение по Y</param>

protected MandelbrotFractalModernized(IColorsFactory colorsFactory, int r, int b, int g, int width, int height, double zoom, int iterations, double movex, double movey) : base(colorsFactory, r, b, g, width, height, zoom, iterations, movex, movey)

{

}

}

}

**Конец файла;**

using System.Numerics;

using FastBitmapLib;

using System.Linq;

namespace Fractals

{

/// <summary>

/// Представляет экземпляр фрактала Трикорн.

/// </summary>

public class TricornFractal : MandelbrotFractal

{

/// <inheritdoc cref="FractalBase.DrawInner"/>

protected override void DrawInner(FastBitmap fractal)

{

Enumerable.Range(0, Width \* Height).AsParallel().ForAll(xy =>

{

Complex C, newC, oldC;

int x = xy % Width, y = xy / Width, i = 0;

C = new Complex((x - Width / 2) / (0.5 \* Zoom \* Width) + MoveX,

(y - Height / 2) / (0.5 \* Zoom \* Height) + MoveY);

newC = new Complex(0, 0);

for (; i < Iterations; i++)

{

oldC = newC;

newC = new Complex(oldC.Real \* oldC.Real - oldC.Imaginary \* oldC.Imaginary + C.Real,

-2 \* oldC.Real \* oldC.Imaginary + C.Imaginary);

if ((newC.Real \* newC.Real + newC.Imaginary \* newC.Imaginary) > 4)

break;

}

fractal.SetPixel(x, y, Colors[i % 256]);

});

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="TricornFractal"/> с заданным значением фабрики цветов.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

public TricornFractal(IColorsFactory colorsFactory) : base(colorsFactory)

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="TricornFractal"/> с базовыми значениями.

/// </summary>

public TricornFractal() : base()

{

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="TricornFractal"/> с заданными значениями.

/// </summary>

/// <param name="colorsFactory">Фабрика цветов</param>

/// <param name="r">Компонент цвета r</param>

/// <param name="b">Компонент цвета b</param>

/// <param name="g">Компонент цвета g</param>

/// <param name="width">Ширина</param>

/// <param name="height">Высота</param>

/// <param name="zoom">Увеличение</param>

/// <param name="iterations">Кол-во итераций</param>

/// <param name="movex">Смещение по Х</param>

/// <param name="movey">Смещение по Y</param>

protected TricornFractal(IColorsFactory colorsFactory, int r, int b, int g, int width, int height, double zoom, int iterations, double movex, double movey) : base(colorsFactory, r, b, g, width, height, zoom, iterations, movex, movey)

{

}

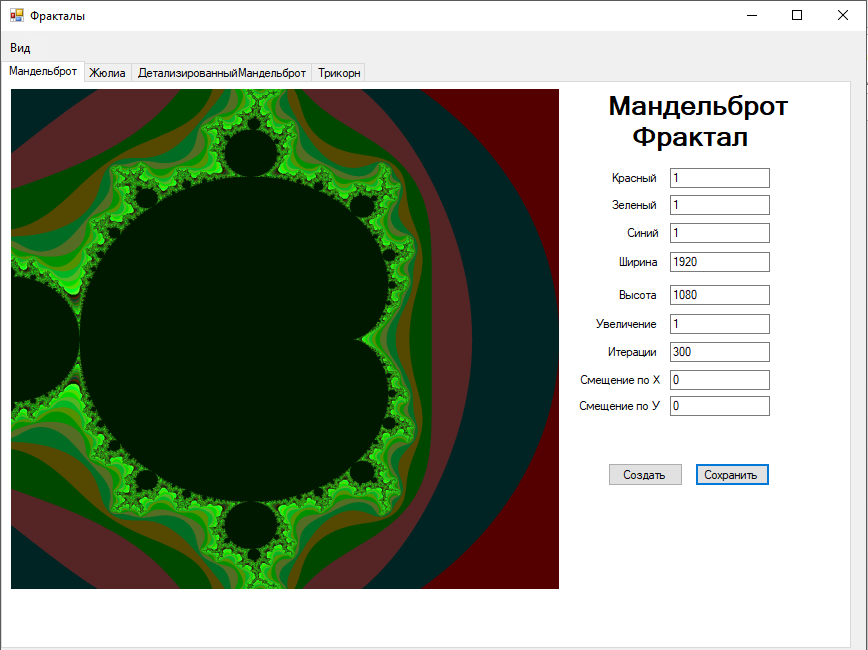
}

}

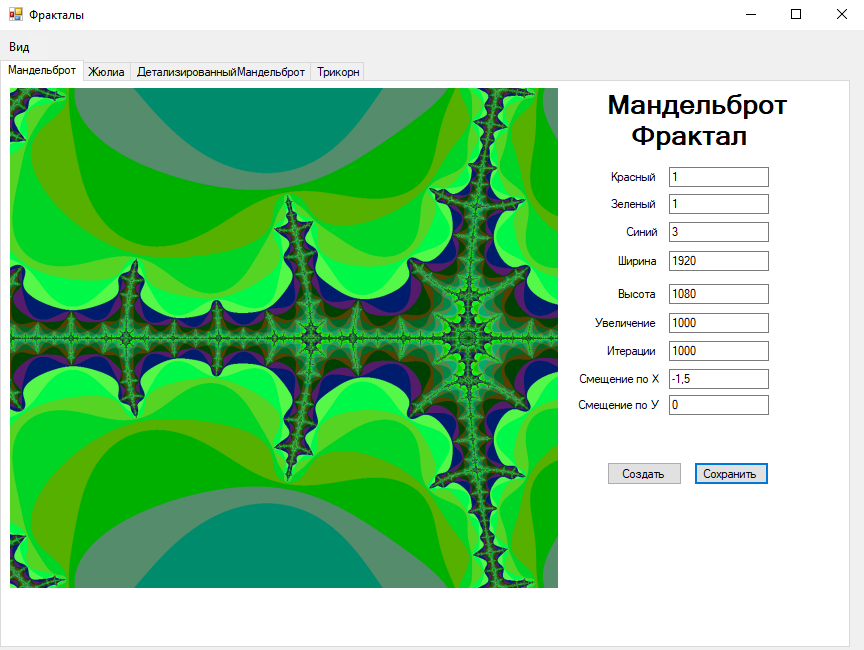
**Конец файла;**

**4. Результат работы программы.**

1) Мандельброт фрактал:

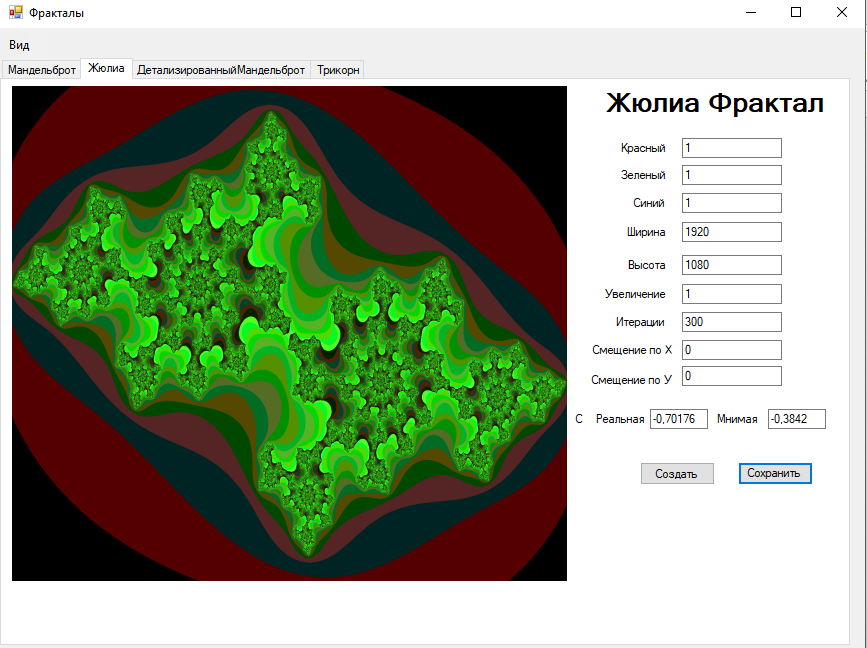
А) Базовые значения

Б) Произвольные значения

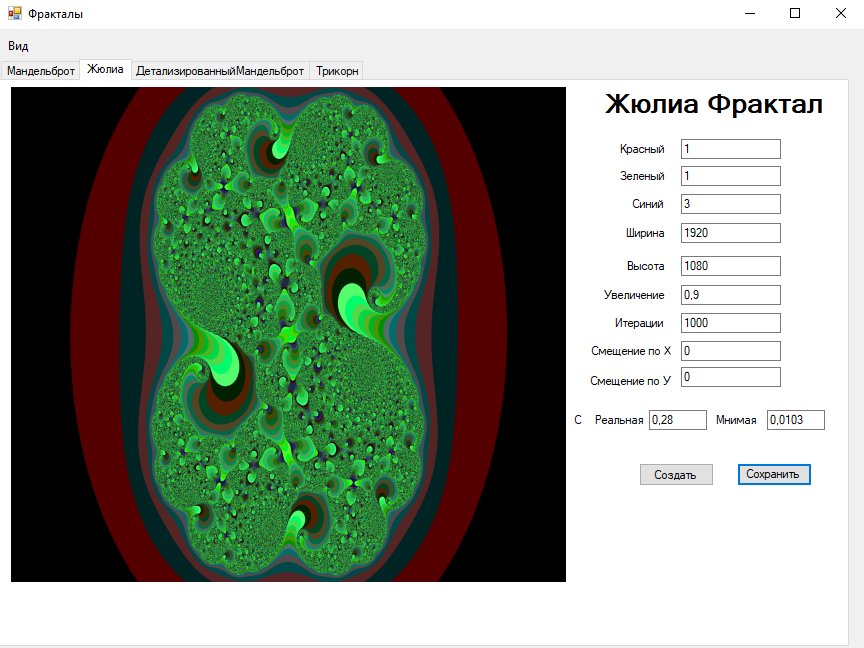


2) Жюлиа фрактал

А) Базовые значения

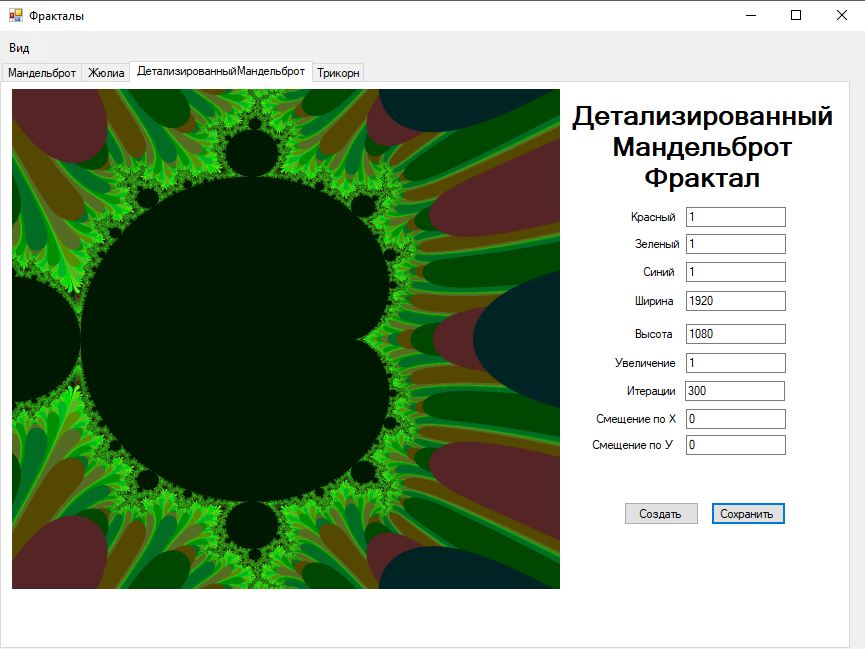


Б) Произвольные значения

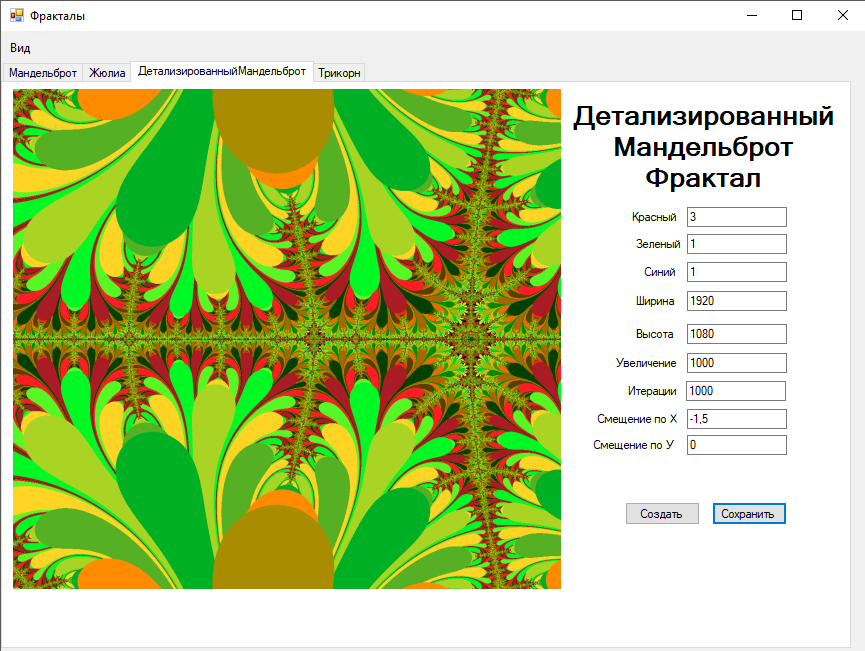


3) Детализированный Мандельброт фрактал

А) Базовые значения

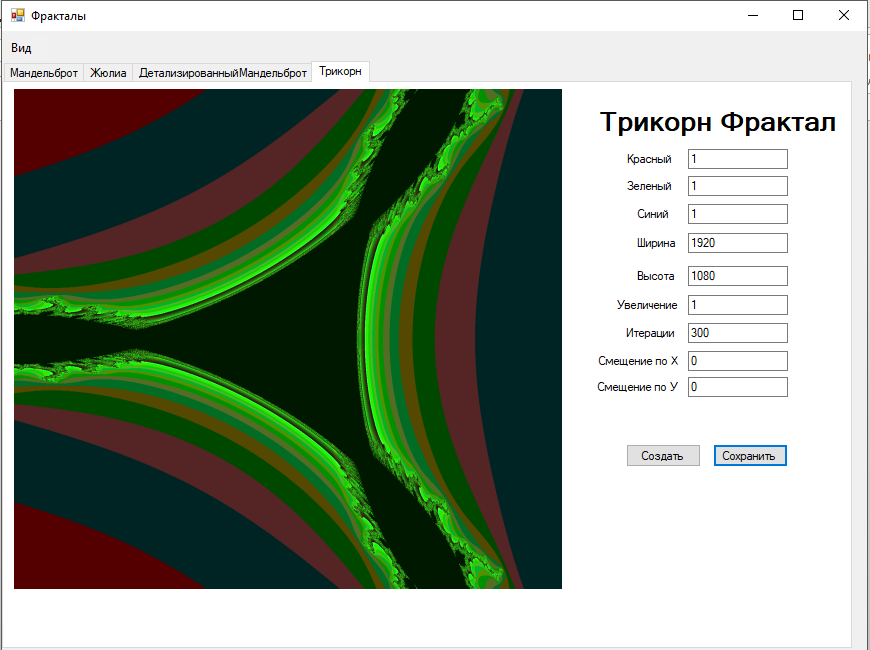


Б) Произвольные значения



4) Трикорн фрактал

А) Базовые значения



Б) Произвольные знаечния

