Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Департамент

Программной инженерии

Контрольное домашнее задание
по дисциплине
«Программирование»

Тема работы: Программа для построения фракталов на платформе Windows Forms

Выполнил: студент группы 183 (2)

_____ Верко Р.А.

тел: 89196160900

e-mail адрес: raverko@edu.hse.ru

Преподаватель: Чуйкин Николай Константинович

Москва, 2018 год. Модуль 2

Содержание

1. Условие задачи

Разработать оконное приложение Windows Forms Application, позволяющее:

1. Отрисовывать три вида фракталов, которые определены в индивидуальном варианте.

(Вариант 33: Квазиклевер, H-фрактал, Фрактал "Центр масс треугольника").

2. Предоставлять пользователю выбор текущего фрактала для отрисовки.

3. Предоставлять пользователю возможность устанавливать количество шагов рекурсии (её глубину - количество рекурсивных вызовов). При изменении глубины рекурсии фрактал должен быть автоматически перерисован.

4. Автоматически перерисовывать фрактал при изменении размеров окна. Окно обязательно должно быть масштабируемым

5. Предоставлять пользователю возможность выбора двух цветов startColor и endColor. Цвет startColor используется для отрисовки элементов первой итерации, цвет endColor - для отрисовки элементов последней итерации. Цвета для промежуточных итераций должны вычисляться с использованием линейного градиента.

6. Сообщать о некорректном вводе данных, противоречивых или недопустимых значениях данных и других нештатных ситуациях во всплывающих окнах типа MessageBox.

7. Должна быть предусмотрена возможность сохранения фрактала в виде картинки

8. Предусмотреть возможность изменения масштаба фрактала для его детального просмотра. Увеличение должно быть 2, 3 и 5-кратным

9. Предусмотреть возможность перемещения изображения, в т.ч. при увеличенном изображении (пункт 8).

2. Функции разрабатываемого приложения

2.1 Варианты использования

Данная программа может использоваться главным образом в целях обучения студентов, для наглядного примера работы компьютерной графики и для практического применения при изучении самоподобных фигур с помощью их поэтапного построения. Также возможно использование в развлекательных целях.

2.2 Описание интерфейса пользователя

Интерфейс пользователя представляет собой окно (рис 1), большую часть которого занимает объект PictureBox (1.1), имеющий белый фон. На нём будет изображаться построенный нами фрактал. Слева от него колонка с различными настройками и вариациями построения фигур. ListBox (1.2) обеспечивает простой и понятный выбор типа фрактала. Поле TextBox (1.3) принимает с клавиатуры глубину рекурсии, с которой программа будет отрисовывать фигуру. Объект ComboBox (1.4) позволяет выбрать увеличение картинки для её подробного изучения. Две кнопки типа Button (1.5) вызывают диалоговые окна, с помощью которых можно выбрать начальное и конечное значения цветов, на основе которых автоматически будет простроен линейный градиент длиной в глубину рекурсии. После выбора фон кнопки принимает выбранный пользователем цвет. Для выбранного в ListBox (1.2) фрактала «Квазиклевер» появляется дополнительный ComboBox (1.6) с выбором незадействованной стороны. Ниже находятся две кнопки, отвечающие непосредственно за отрисовку с учетом полученных значений (1.7) , и за сохранение фрактала в виде картинки в память компьютера (1.8).

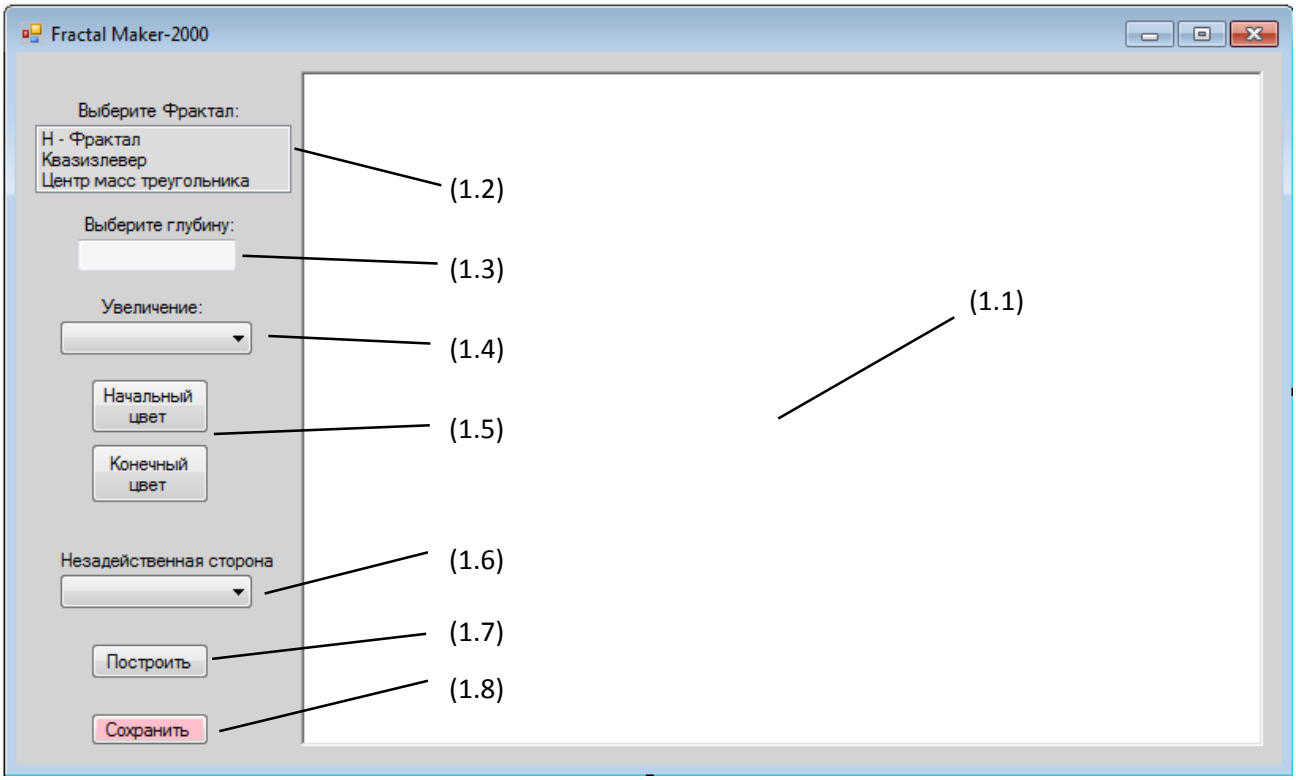

Рис. 1. Основное окно программы

3. Структура приложения

3.1. Диаграмма классов

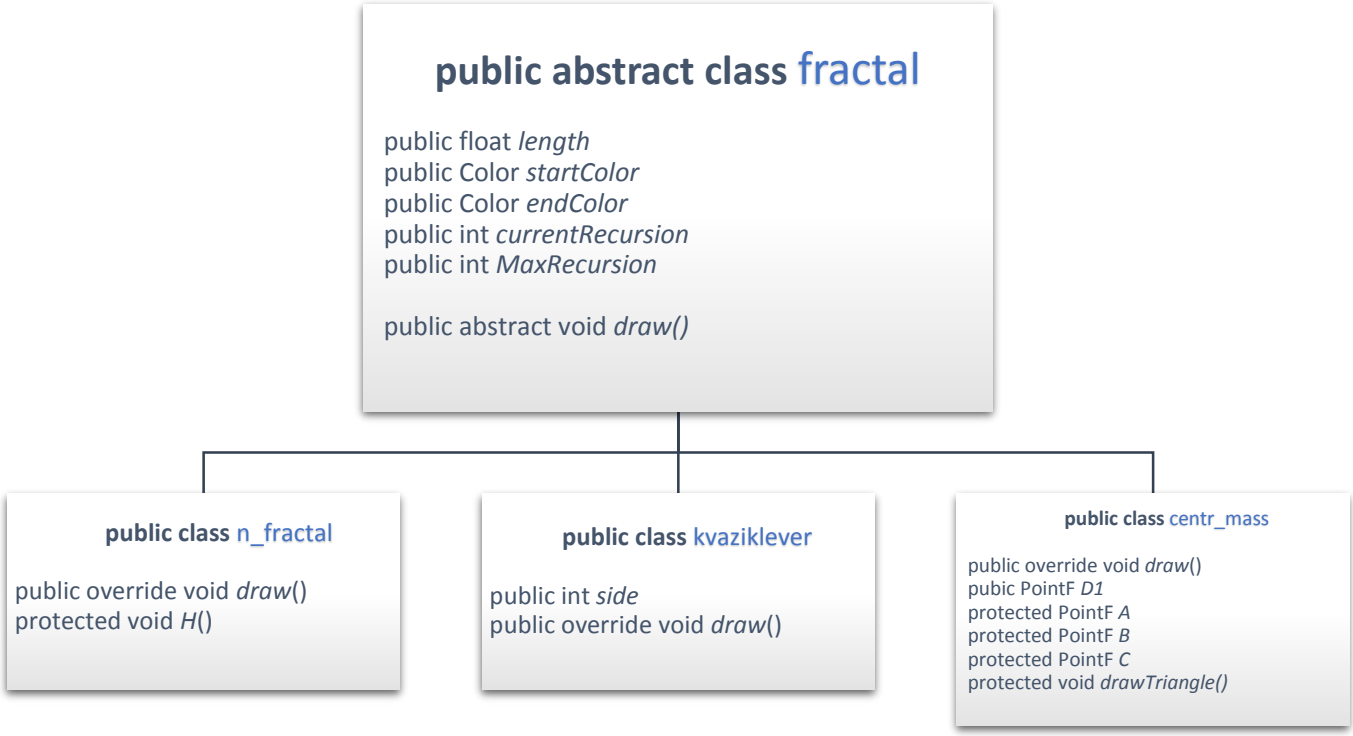

рис. 2. Диаграмма классов

3.2 Описание классов, их полей и методов

- `static class Program - класс, содержащий метод Main()`
- `public partial class Form1: Form - форма, содержащая основные методы для работы с фракталами`
  - `int wid = SystemInformation.PrimaryMonitorSize.Width;`
  - `int heig = SystemInformation.PrimaryMonitorSize.Height;`
  - `public static Bitmap mainbit;// графика для сохранения картинки`
  - `public static Graphics maingraphics;`
  - `int whichfractal; // номер текущего фрактала`
  - `int depth; // текущая максимальная глубина рекурсии`
  - `static int depth2; // дополнительная глубина для постройки фракталов`
  - `n_fractal nf = new n_fractal();`
  - `kvaziklever kvazi = new kvaziklever();`

- centr_mass centr_m = new centr_mass();
- public static int GetDepth { get => depth2; }
- float deltaX = 0;
- float deltaY = 0;
- float oldX = 0;
- float oldY = 0;
- int rMax;
- int rMin;
- int gMax;
- int gMin;
- int bMax;
- int bMin;
- public float height_;
- public float width_;
- public static List<Color> colorList = new List<Color>();
- bool exist = false;
- public static Color colorforPen(int n)
- public bool Depth() - Устанавливает глубину для отрисовки массива и количества цветов в гамме
- public void Colors(int size)- устанавливает цветовой градиент длиной в глубину рекурсии
- private void button1_Click(object sender, EventArgs e)- кнопка "Построить" - строит график в центре видимой площади
- public void pictureBox1_Paint(object sender, PaintEventArgs e) - метод отрисовки picturebox
- private void button2_Click(object sender, EventArgs e) – выбор начального цвета из палитры
- private void button3_Click(object sender, EventArgs e) – выбор конечного цвета из палитры
- private void listBox1_SelectedIndexChanged(object sender, EventArgs e) - изменение выбранного фрактала устанавливает видимость индивидуальных настроек -
- /// для квазифрактала появляется возможность выбора пустой стороны
- private void pictureBox1_MouseDown(object sender, MouseEventArgs e) - запись начальных значений координат мыши при нажатии на picturebox
- private void pictureBox1_MouseUp(object sender, MouseEventArgs e)- изменение положения центра фрактала с последующей его отрисовкой в этой точке
- private void ClearView() - метод очистки видимой части picturebox для новой отрисовки
- private void SelectSize() - установка увеличения фрактала - увеличение значимой части отрисовки
- private void Form1_ResizeEnd(object sender, EventArgs e) - Перерисовка фрактала после изменения размера
- private void Form1_SizeChanged(object sender, EventArgs e)- перерисовка при максимизации
- private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)- изменение незадействованной стороны квазиклевера

- private void button4_Click(object sender, EventArgs e) - Сохранение картинки. Возможны 2 варианта : Jpeg и Png. Сохраняется увеличенная область, нежели область видимости,
- для большего захвата отрисованного фрактала. Цвет фона - белый.

- public abstract class fractal – абстрактный класс, содержащий основные поля и метод отрисовки будущих фракталов
  - public float length; // длина значимой части
  - public Color startColor; // начальный цвет
  - public Color endColor; // конечный цвет
  - public int currentRecursion; // текущая глубина рекурсии
  - public int MaxRecursion; // глубина рекурсии, задаваемая полем depth
  - public abstract void draw(float x1, float y1, float razm_f, int numrec, int maxrec, PictureBox pb);//основной метод отрисовки
- public class n_fractal : fractal - класс "H-Фрактала"
  - public override void draw(float x1, float y1, float razm_f, int numrec, int maxrec, PictureBox pb)
    - x1 значение Width
    - y1 значение Height
    - razm_f размер одной линии
    - numrec текущая глубина рекурсии
    - maxrec максимальная глубина рекурсии
    - pb picturebox для отрисовки
  - protected void H(float x, float y, float razmer, PictureBox pb, int numrec) - отрисовка одной буквы H
- public class kvaziklever : fractal - класс фрактала «Квазиклевер»
  - public int side; // неиспользуемая сторона
  - public override void draw(float x1, float y1, float razm_f, int numrec, int maxrec, PictureBox pb)
    - x1 значение Width
    - y1 значение Height
    - razm_f размер одной линии
    - numrec текущая глубина рекурсии
    - maxrec максимальная глубина рекурсии
    - pb picturebox для отрисовки
- public class centr_mass : fractal - класс фрактала «центр масс»
  - public PointF D1;
  - protected PointF A;
  - protected PointF B;
  - protected PointF C;
  - public override void draw(float x1, float y1, float razm_f, int numrec, int maxrec, PictureBox pb) - запуск рекурсивной функции отрисовки фрактала
  - protected void drawTriangle(PointF A, PointF B, PointF C, int maxrec, PictureBox pb) - Отрисовка векторов к центру масс у 1 треугольника

4. Распределение исходного кода по файлам проекта

1. Form1.cs – файл содержит главную форму программы и методы для обработки изменений параметров построения фракталов. Так же присутствуют методы класса fractal и наследуемых классов n_fractal, kvaziclever, centr_mass.
    a. Form1.Designer.cs - файл конструктора, в котором элементы формы инициализируются. Если какой-либо элемент перетаскивается и отбрасывается в окне формы, тогда этот элемент будет автоматически инициализирован в этом классе.
    b. Form1.resx – системный файл ресурсов, чаще всего используется для хранения строк.

2. Program.cs – файл содержит метод Main() – стандартную точку входа в реализуемое приложение.

5. Контрольный пример и описание результатов

1. Запустим выполняемый файл .exe , чтобы запустить наше приложение.
2. Перед нами откроется пустое окно программы (рис. 1).
3. Для начала, выберем тип фрактала, который мы собираемся простроить (рис. 2)

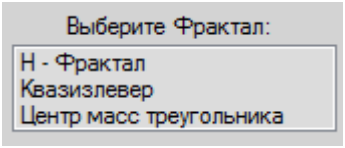

рис. 2

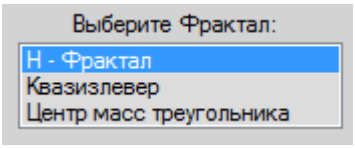

рис. 3

4. Выбрав «H – Фрактал», кликнув на него указателем (рис. 3), перейдем к выбору его глубины. В пустое поле впишем цифру 6 с клавиатуры. (рис. 4) , (рис. 5)

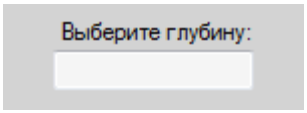

рис. 4

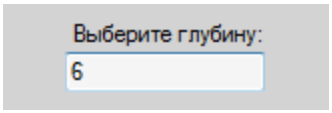

рис. 5

5. Из выпадающего списка выберем увеличение 1x (рис. 6)

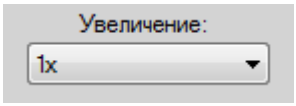

рис. 6

6. С помощью кнопок назначим начальный и конечный цвета. (рис. 7) – (рис.11)

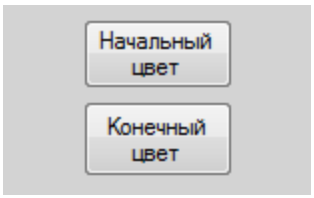

рис. 7

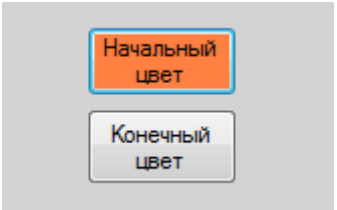

рис. 9

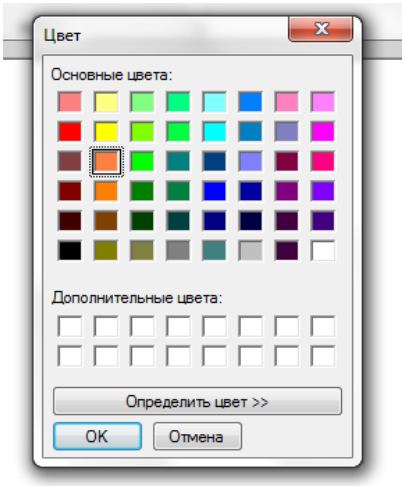

рис. 8

рис. 10

рис. 11

7. Нажмем клавишу «Построить». Перед нами отобразится требуемый фрактал. (рис. 12)

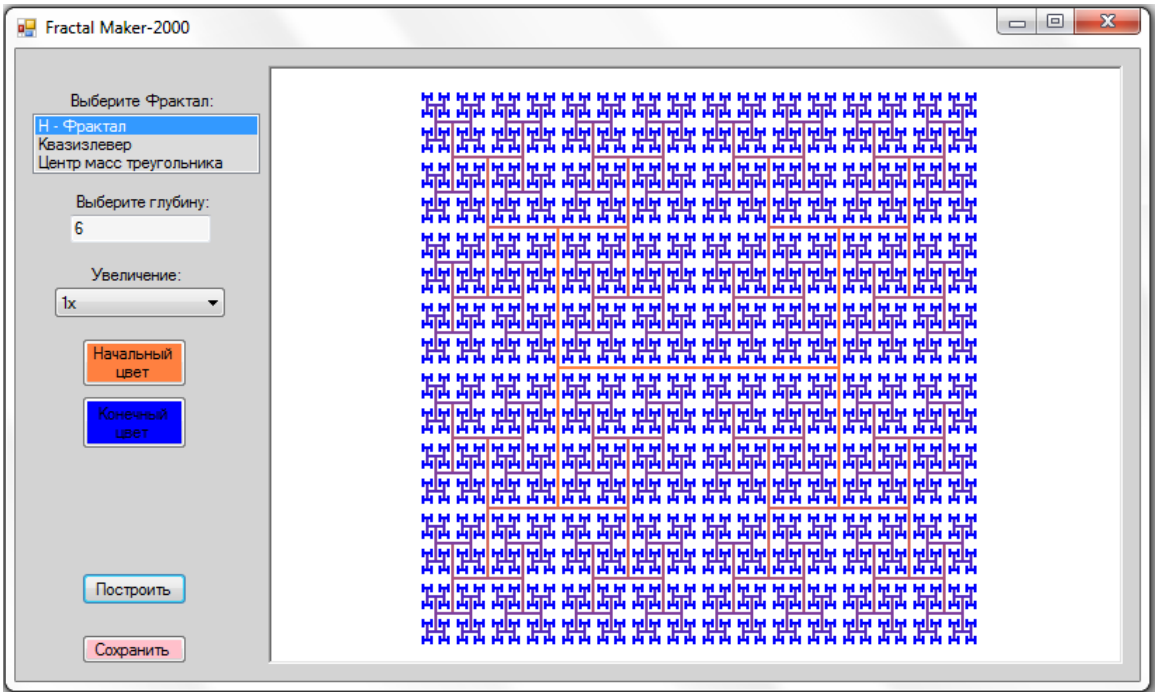

рис. 12

8. Нажав на клавишу «Сохранить» откроется диалоговое окно, позволяющее выбрать путь и тип сохраняемого файла. (рис. 13).

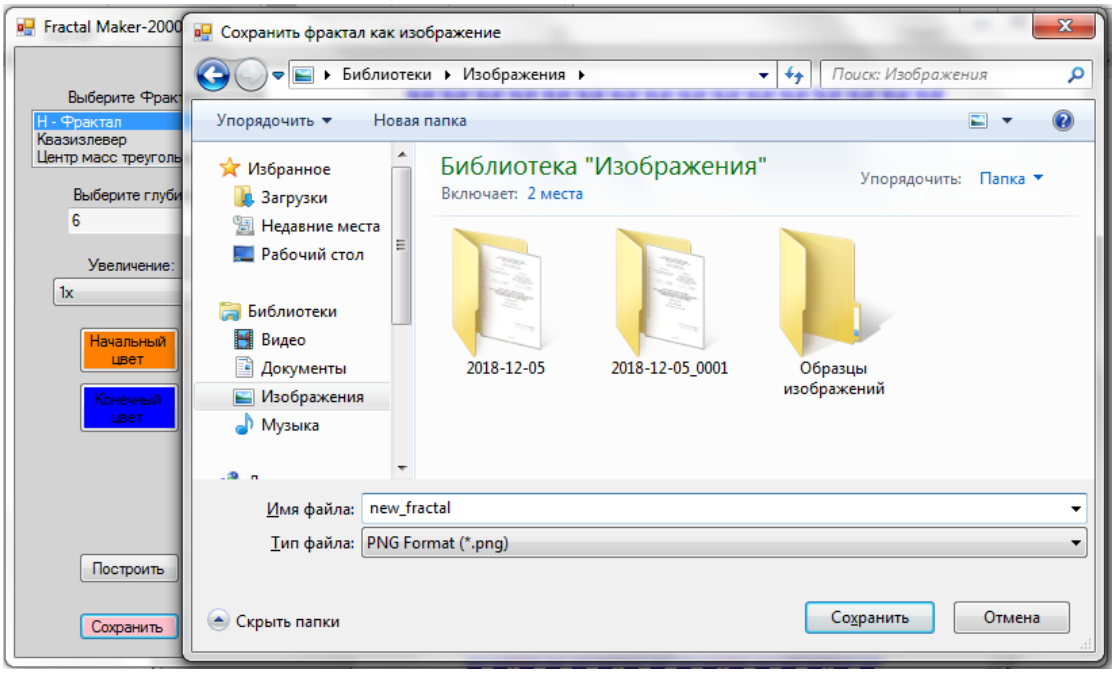

рис. 13

9. Построенную фигуру можно передвигать, приближать, используя выбор увеличения и нажатие кнопки «Построить». Так же возможно изменение размера окна (рис. 14).

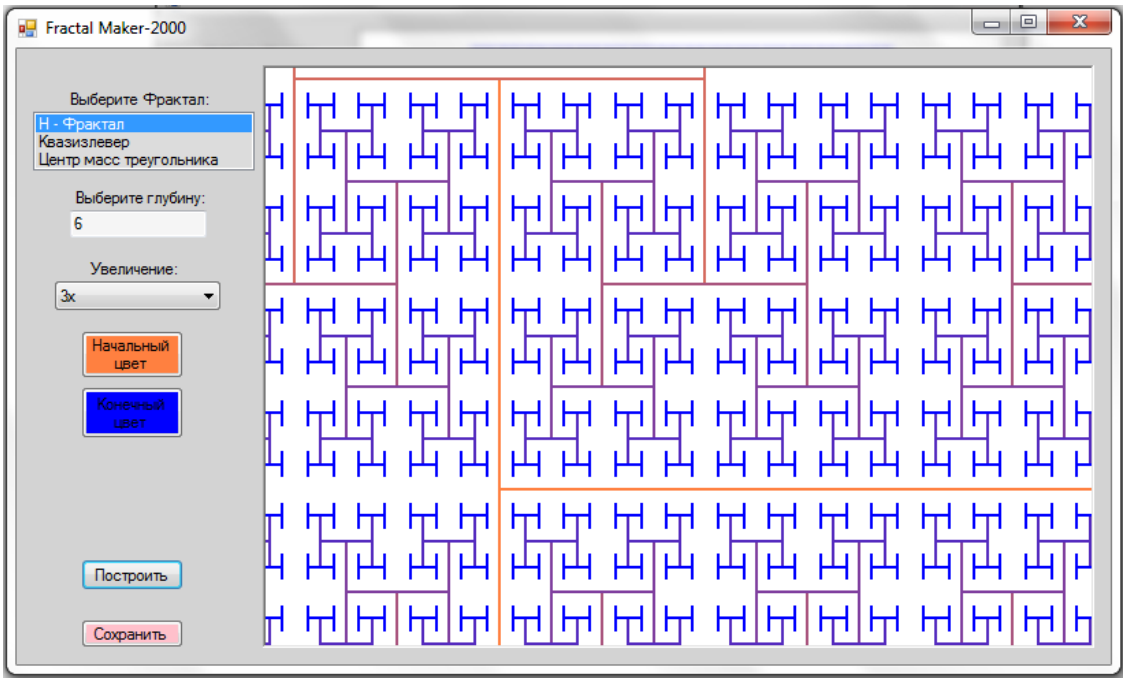

рис. 14 – увеличенное смещённое изображение

10. Аналогичный функционал доступен и для других двух фракталов (рис. 15) , (рис.16).

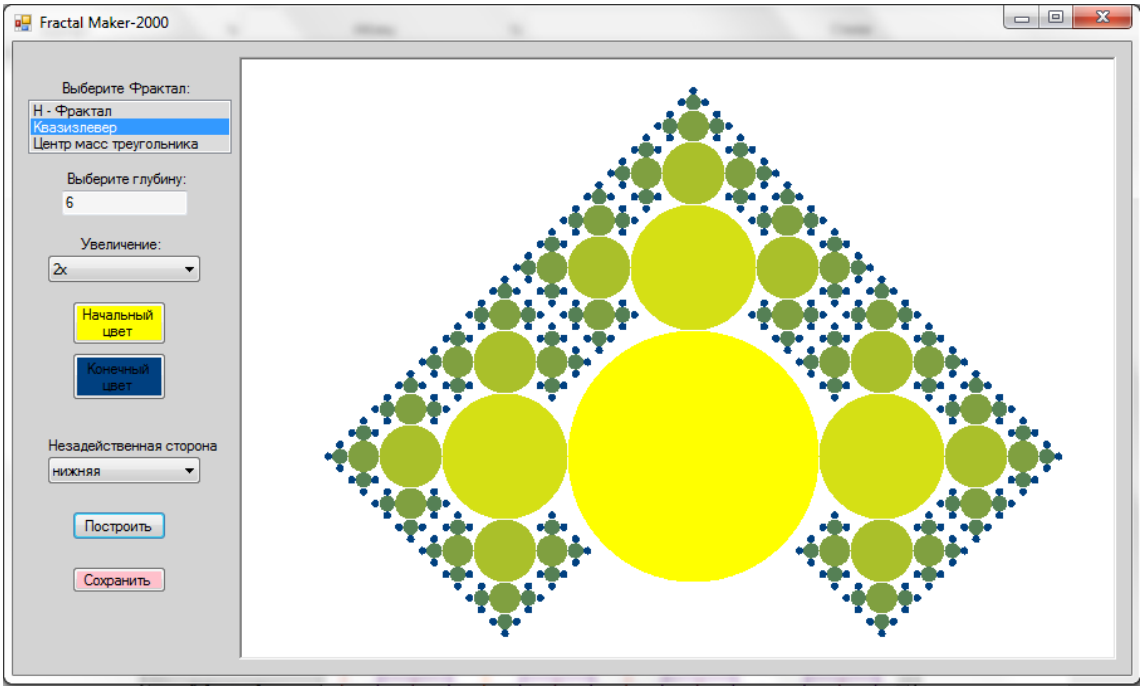

рис. 15 – фрактал «Квазиклевер»

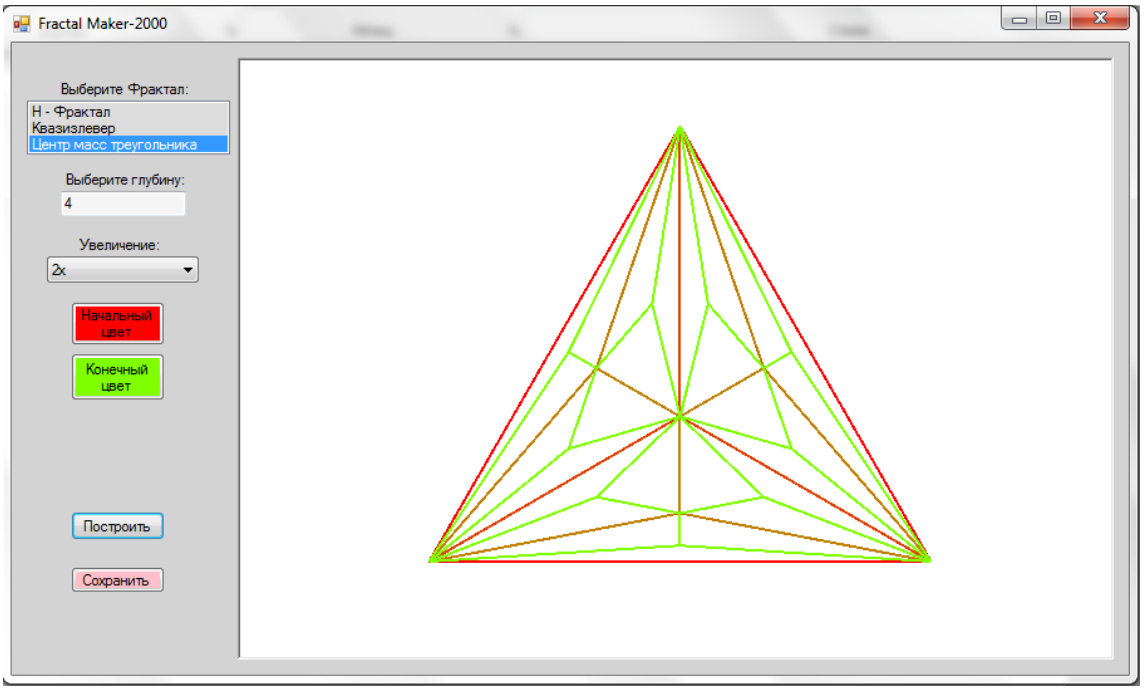

рис. 16 – фрактал «Центр масс треугольника»

11. В памяти компьютера успешно сохранился рисунок фрактала (рис. 17)

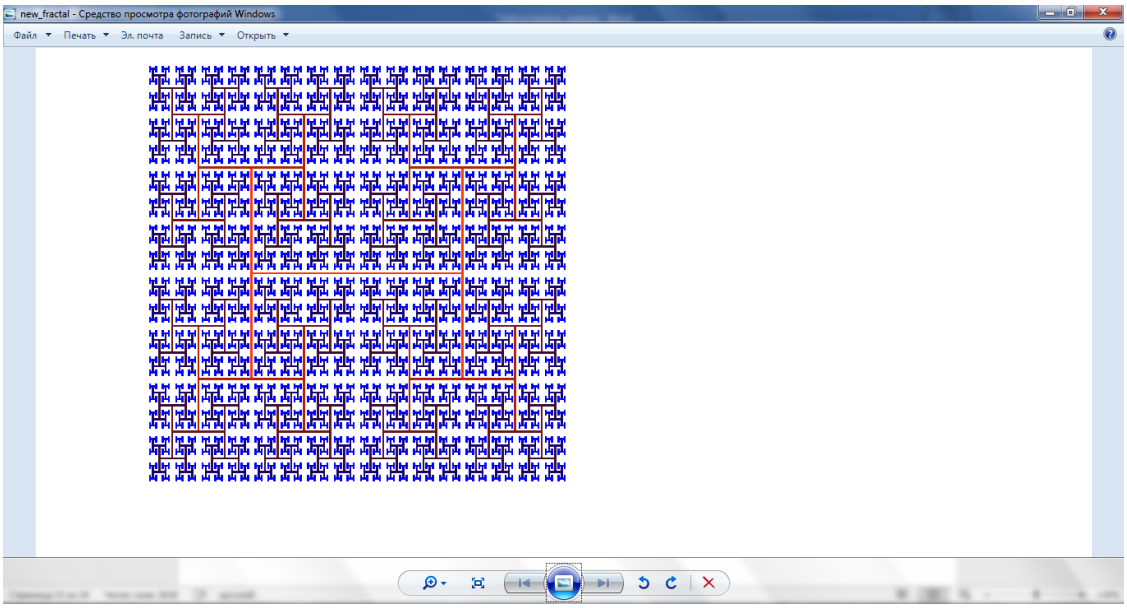

рис. 17 – открытый файл сохранённой нами фигуры

6. Сообщения пользователю

1. При вводе отрицательного значения в поле «Глубина рекурсии» пользователю выводится сообщение типа MessageBox с текстом «Ошибка задания глубины рекурсии!».
2. При отсутствии выбранного значения в пункте «Увеличение» пользователю выводится сообщение типа MessageBox с текстом «Выберите увеличение!».
3. При отсутствии выбранного значения в пункте «Незадействованная сторона» пользователю выводится сообщение типа MessageBox с текстом «По умолчанию незадействованной будет выбрана правая сторона ».
4. При попытке передвинуть фигуру, или сохранить её до первого создания фрактала пользователю выводится сообщение типа MessageBox с текстом «Создайте фрактал».
5. В случае успешного сохранения картики в память устройства пользователю выводится сообщение типа MessageBox с текстом «Файл успешно сохранён!».

7. Код программы

```
namespace WinForm_Fractals_KDZ_MF
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
int wid = SystemInformation.PrimaryMonitorSize.Width;
int heig = SystemInformation.PrimaryMonitorSize.Height;
centr_m.D1.X = pictureBox1.Width / 2;
centr_m.D1.Y = pictureBox1.Height / 2;
MinimumSize = new Size(wid / 2, heig / 2); // установка минимальных значений окна
height_ = pictureBox1.Height / 2; // начальные значения высоты и ширины, равные центру отображаемого окна
width_ = pictureBox1.Width / 2;
comboBox4.Visible = false; // настройка
label6.Visible = false;
mainbit = new Bitmap(1000, 1000);
maingraphics = Graphics.FromImage(mainbit); // графика и битмап для сохранения картинки
maingraphics.Clear(Color.White);
}
public static Bitmap mainbit;// графика для сохранения картинки
public static Graphics maingraphics;
int whichfractal; // номер текущего фрактала
int depth; // текущая максимальная глубина рекурсии
static int depth2; // дополнительная глубина для постройки фракталов
n_fractal nf = new n_fractal();
kvaziklever kvazi = new kvaziklever(); // строим первичные объекты классов
centr_mass centr_m = new centr_mass();
```

```
public static int GetDepth { get => depth2; }
float deltaX = 0;
float deltaY = 0;
float oldX = 0;
float oldY = 0;
int rMax;
int rMin;
int gMax;
int gMin;
int bMax;
int bMin;
public float height_;
public float width_;
public static List<Color> colorList = new List<Color>();
bool exist = false;

public static Color colorforPen(int n) => colorList[n];

/// <summary>
/// Устанавливает глубину для отрисовки массива и количества цветов в гамме
/// </summary>
/// <returns>возвращает true, если введено корректное значение, false в обратном</returns>
public bool Depth()
{
if (!int.TryParse(textBox1.Text, out depth) || depth < 1)
{
MessageBox.Show("Ошибка задания глубины рекурсии!");
return false;
}

else
```

```
{
if (depth > 10||depth<1) depth = 10; // максимальная отрисовка - 10. Далее не видна разница.
nf.MaxRecursion = depth;
kvazi.MaxRecursion = depth;
centr_m.MaxRecursion = depth;
depth2 = depth;
return true;
}

}
/// <summary>
/// устанавливает цветовой градиент длиной в глубину рекурсии
/// </summary>
/// <param name="size"> глубина рекурсии</param>
public void Colors(int size)
{
rMax = button3.BackColor.R;
rMin = button2.BackColor.R;
gMax = button3.BackColor.G;
gMin = button2.BackColor.G;
bMax = button3.BackColor.B;
bMin = button2.BackColor.B;
colorList.Clear();
colorList.Add(Color.FromArgb(rMin, gMin, bMin)); // первый цвет - исходный
for (int i = 1; i < size-1; i++)
{
var rAverage = rMin + ((rMax - rMin) * i / size);
var gAverage = gMin + ((gMax - gMin) * i / size); // size - 2 генерируемые
var bAverage = bMin + ((bMax - bMin) * i / size);
colorList.Add(Color.FromArgb(rAverage, gAverage, bAverage)); // последний цвет - исходный
}
```

```
colorList.Add(Color.FromArgb(rMax, gMax, bMax));
}

/// <summary>
/// кнопка "Построить" - строит график в центре видимой площади
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button1_Click(object sender, EventArgs e)
{

height_ = pictureBox1.Height/2; //высота элемента pictureBox1
width_ = pictureBox1.Width/2; // ширина

Graphics pic = pictureBox1.CreateGraphics();
pic.Clear(Color.White); // очистка поля

if (comboBox4.SelectedIndex != 0 && comboBox4.SelectedIndex != 1 &&
comboBox4.SelectedIndex != 2 && comboBox4.SelectedIndex != 3 &&
listBox1.SelectedIndex==1)
{
MessageBox.Show("По умолчанию незадействованной будет выбрана правая
сторона");
} // Для Квазиклевера - если пользователь не выбрал свободную сторону

if (Depth() && (comboBox1.SelectedIndex == 0 || comboBox1.SelectedIndex ==
1|| comboBox1.SelectedIndex == 2 || comboBox1.SelectedIndex==3))
{
exist = true;
Colors(depth); // определение цветов
SelectSize();
switch (listBox1.SelectedIndex)
{
```

```
case 0:
{

whichfractal = 0;
ClearView();
nf.draw(width_, height_, nf.length, 1, nf.MaxRecursion, pictureBox1);

break;
}
case 1:
{
whichfractal = 1;
ClearView();
kvazi.draw(width_, height_, kvazi.length, kvazi.side, kvazi.MaxRecursion,
pictureBox1);

}
break;
case 2:
{
whichfractal = 2;
ClearView();
centr_m.draw(width_, height_, 1, 1, centr_m.MaxRecursion, pictureBox1);
}
break;

}

}

if (comboBox1.SelectedIndex != 0 && comboBox1.SelectedIndex != 1 &&
comboBox1.SelectedIndex != 2 && comboBox1.SelectedIndex!=3)
{
```

```
MessageBox.Show("Выберите увеличение!");
} // проверка на выбор увеличения


}


/// <summary>
/// метод отрисовки picturebox
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
public void pictureBox1_Paint(object sender, PaintEventArgs e)
{
}
/// <summary>
/// выбор начального цвета из палитры
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button2_Click(object sender, EventArgs e)
{
if (colorDialog1.ShowDialog() == DialogResult.OK)
{
button2.BackColor = colorDialog1.Color; // устанавливаем цвет клавиши в
выбранный нами
}
}
/// <summary>
/// выбор конечного цвета из палитры
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button3_Click(object sender, EventArgs e)
```

```
{
if (colorDialog2.ShowDialog() == DialogResult.OK)
{
button3.BackColor = colorDialog2.Color;// устанавливаем цвет клавиши в выбранный нами
}
}

/// <summary>
/// изменение выбранного фрактала устанавливает видимость индивидуальных настроек -
/// для квазифрактала появляется возможность выбора пустой стороны
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
if (listBox1.SelectedIndex == 0)
{
label6.Visible = false;
comboBox4.Visible = false;
}
else if (listBox1.SelectedIndex == 1)
{
label6.Visible = true;
comboBox4.Visible = true;
}
else if (listBox1.SelectedIndex == 2)
{
label6.Visible = false;
comboBox4.Visible = false;
}
}
```

```
/// <summary>
/// запись начальных значений координат мыши при нажатии на picturebox
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
if (exist)
{
oldX = Cursor.Position.X;
oldY = Cursor.Position.Y;
}
else { MessageBox.Show("Создайте фрактал!"); }
}
/// <summary>
/// изменение положения центра фрактала с последующей его отрисовкой в этой точке
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
deltaX = oldX - Cursor.Position.X;
deltaY = oldY - Cursor.Position.Y;
width_ -= deltaX;
height_ -= deltaY;
if (exist)
{
ClearView();
switch (whichfractal)
{
case 0: nf.draw(width_, height_, nf.length, 1, nf.MaxRecursion, pictureBox1); break;
```

```
case 1: kvazi.draw(width_, height_, kvazi.length, kvazi.side,
kvazi.MaxRecursion, pictureBox1); break;

case 2: centr_m.draw(width_, height_, 1, 1, centr_m.MaxRecursion,
pictureBox1); break;

}

}

else { MessageBox.Show("Создайте фрактал!"); }

}

/// <summary>

/// метод очистки видимой части picturebox для новой отрисовки

/// </summary>

private void ClearView()

{

Graphics pic = pictureBox1.CreateGraphics();

pic.Clear(Color.White);

//Colors(depth);

}

/// <summary>

/// установка увеличения фрактала - увеличение значимой части отрисовки

/// </summary>

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

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

private void SelectSize()

{

switch (comboBox1.SelectedIndex)

{

case 0:

{

if (Depth())

{

nf.length = 100;

kvazi.length = 50;

centr_m.length = 2;
```

```
}
}
break;
case 1:
{
if (Depth())
{
nf.length = 200;
kvazi.length = 100;
centr_m.length = 4;
}
}
break;
case 2:
{
if (Depth())
{
nf.length = 300;
kvazi.length = 150;
centr_m.length = 6;
}
}
break;
case 3:
{
if (Depth())
{
nf.length = 500;
kvazi.length = 250;
centr_m.length = 10;
}
}
```

```
break;
}

}
/// <summary>
/// Перерисовка фрактала после изменения размера
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_ResizeEnd(object sender, EventArgs e)
{
if (exist)
{
ClearView();
switch (whichfractal)
{
case 0: nf.draw(width_, height_, nf.length, 1, nf.MaxRecursion,
pictureBox1); break;
case 1: kvazi.draw(width_, height_, kvazi.length, kvazi.side,
kvazi.MaxRecursion, pictureBox1); break;
case 2: centr_m.draw(width_, height_, 1, 1, centr_m.MaxRecursion,
pictureBox1); break;
}
}

}
/// <summary>
/// перерисовка при максимизации
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_SizeChanged(object sender, EventArgs e)
{
```

```
if (WindowState == FormWindowState.Minimized)
{
ClearView();
if (exist)
{
switch (whichfractal)
{
case 0: nf.draw(width_, height_, nf.length, 1, nf.MaxRecursion, pictureBox1); break;
case 1: kvazi.draw(width_, height_, kvazi.length, kvazi.side, kvazi.MaxRecursion, pictureBox1); break;
case 2: centr_m.draw(width_, height_, 1, 1, centr_m.MaxRecursion, pictureBox1); break;
}
}
}
}
/// <summary>
/// изменение незадействованной стороны квазиклевера
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
{
kvazi.side = comboBox4.SelectedIndex;
}
/// <summary>
/// Сохранение картинки. Возможны 2 варианта : Jpeg и Png. Сохраняется увеличенная область, нежели область видимости,
/// для большего захвата отрисованного фрактала. Цвет фона - белый.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

```
private void button4_Click(object sender, EventArgs e)
{
if (exist)
{
Bitmap bitmapImageOfFractal = new Bitmap(pictureBox1.Width,
pictureBox1.Height);
SaveFileDialog saveImage = new SaveFileDialog();
saveImage.Filter = "JPeg Format (*.jpeg)|*.jpeg|PNG Format (*.png)|*.png";
saveImage.Title = "Сохранить фрактал как изображение";
pictureBox1.DrawToBitmap(bitmapImageOfFractal, new Rectangle(0, 0,
pictureBox1.Width, pictureBox1.Height));
maingraphics.Clear(Color.White);
ClearView();
switch (whichfractal)
{
case 0: nf.draw(width_, height_, nf.length, 1, nf.MaxRecursion,
pictureBox1); break;
case 1: kvazi.draw(width_, height_, kvazi.length, kvazi.side,
kvazi.MaxRecursion, pictureBox1); break;
case 2: centr_m.draw(width_, height_, 1, 1, centr_m.MaxRecursion,
pictureBox1); break;
}

if (saveImage.ShowDialog() == DialogResult.OK) //для избавления от
исключения в случае нажатия на кнопку "Отмена"
{
if (saveImage.FileName != "")
{
FileStream fs = (FileStream)saveImage.OpenFile();
switch (saveImage.FilterIndex)
{
case 1:
mainbit.Save(fs, ImageFormat.Jpeg);
break;
```

```
case 2:
mainbit.Save(fs, ImageFormat.Png);
break;
}
fs.Close();
MessageBox.Show("Файл успешно сохранен! ", "Сохранение",
MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}
}
else { MessageBox.Show("Создайте фрактал!"); } // Проверка!
}
}
public abstract class fractal
{
public float length; // длина значимой части
public Color startColor; // начальный цвет
public Color endColor; // конечный цвет
public int currentRecursion; // текущая глубина рекурсии
public int MaxRecursion; // глубина рекурсии, задаваемая полем depth
public abstract void draw(float x1, float y1, float razm_f, int numrec, int
maxrec, PictureBox pb);//основной метод отрисовки
}
/// <summary>
/// класс "H-Фрактала"
/// </summary>
public class n_fractal : fractal
{/// <summary>
/// </summary>
/// <param name="x1">значение Width</param>
/// <param name="y1">значение Height</param>
/// <param name="razm_f">размер одной линии</param>
/// <param name="numrec">текущая глубина рекурсии</param>
```

```
/// <param name="maxrec">максимальная глубина рекурсии </param>
/// <param name="pb">picturebox для отрисовки</param>
public override void draw(float x1, float y1, float razm_f, int numrec, int maxrec, PictureBox pb)
{
//вершины фигуры H
float x11; float y11;
float x01; float y01;
float x00; float y00;
float x10; float y10;
x11 = x1 + razm_f;
y11 = y1 + razm_f;
x01 = x1 - razm_f;
y01 = y1 + razm_f;//задаем 4 новых значения (x,y), и создаем рекурсию,
x10 = x1 + razm_f;//вызывая эту же функцию в этих 4 точках.
y10 = y1 - razm_f;
x00 = x1 - razm_f;
y00 = y1 - razm_f;
H(x1, y1, razm_f, pb, numrec - 1);//рисуем одну фигуру H
numrec++;
razm_f = razm_f / 2;//уменьшаем размер в 2 раза
// если размер не меньше минимального, то рисуем в 4-х вершинах
if (maxrec >= numrec)
{
draw(x11, y11, razm_f, numrec, maxrec, pb);
draw(x01, y01, razm_f, numrec, maxrec, pb);
draw(x10, y10, razm_f, numrec, maxrec, pb);
draw(x00, y00, razm_f, numrec, maxrec, pb);
}
}
/// <summary>
/// отрисовка одной буквы H
```

```
/// </summary>
/// <param name="x">значение Width центра будущей фигуры </param>
/// <param name="y">значение Height центра будущей фигуры</param>
/// <param name="razmer">уменьшенный размер одной линии</param>
/// <param name="pb">объект для отрисовки</param>
/// <param name="numrec">текущая глубина рекурсии</param>
protected void H(float x, float y, float razmer, PictureBox pb, int
numrec)//функция отрисовки одной H
{
Pen myPen = new Pen(Form1.colorforPen(numrec), 2);
Graphics g = Graphics.FromHwnd(pb.Handle);
// графика для сохранения
Form1.maingraphics.DrawLine(myPen, x - razmer, y - razmer, x - razmer, y +
razmer);
Form1.maingraphics.DrawLine(myPen, x - razmer, y, x + razmer, y);
Form1.maingraphics.DrawLine(myPen, x + razmer, y - razmer, x + razmer, y +
razmer);
// Рисуем в нужной нам графике
g.DrawLine(myPen, x - razmer, y - razmer, x - razmer, y + razmer);
g.DrawLine(myPen, x - razmer, y, x + razmer, y);
g.DrawLine(myPen, x + razmer, y - razmer, x + razmer, y + razmer);
}
}
/// <summary>
/// класс фрактала "Квазиклевер"
/// </summary>
public class kvaziklever : fractal
{
public int side; // неиспользуемая сторона
/// <summary>
/// рекурсивная отрисовка фигуры квазиклевера
/// </summary>
/// <param name="x1">значение Width центра будущей фигуры </param>
```

```
/// <param name="y1">значение Height центра будущей фигуры</param>
/// <param name="razm_f">радиус круга</param>
/// <param name="numrec">текущая глубина рекурсии</param>
/// <param name="maxrec">максимальная глубина рекурсии</param>
/// <param name="pb">объект для отрисовки</param>
public override void draw(float x1, float y1, float razm_f, int numrec, int maxrec, PictureBox pb)
{
//создадим графику из передаваемого picturebox
Graphics g = pb.CreateGraphics();
SolidBrush s = new SolidBrush(Form1.colorforPen(Form1.GetDepth-maxrec));

g.FillEllipse(s, x1 - razm_f, y1 - razm_f, 2 * razm_f, 2 * razm_f);
Form1.maingraphics.FillEllipse(s, x1 - razm_f, y1 - razm_f, 2 * razm_f, 2 * razm_f);
if (maxrec != 1)
{
float[] x = new float[4];
float[] y = new float[4];
float d = 3 * razm_f / 2;
x[0] = x1 - d;
y[0] = y1;
x[1] = x1;
y[1] = y1 - d;
x[2] = x1 + d;
y[2] = y1;
x[3] = x1;
y[3] = y1 + d;
for (int i = 0; i < 4; i++)
{ // рисуем на всех сторонах, кроме родительского
if (i - numrec == 2 || i - numrec == -2)
continue;
draw(x[i], y[i], razm_f / 2, i, maxrec - 1,pb);
```

```
}
}
}
}
/// <summary>
/// класс фрактала "Центр масс"
/// </summary>
public class centr_mass : fractal
{
public PointF D1;
protected PointF A;
protected PointF B;
protected PointF C;
/// <summary>
/// запуск рекурсивной функции отрисовки фрактала
/// </summary>
/// <param name="x1">значение Width центра будущей фигуры </param>
/// <param name="y1">значение Height центра будущей фигуры</param>
/// <param name="razm_f">радиус круга</param>
/// <param name="numrec">текущая глубина рекурсии</param>
/// <param name="maxrec">максимальная глубина рекурсии</param>
/// <param name="pb">объект для отрисовки</param>
public override void draw(float x1, float y1, float razm_f, int numrec, int
maxrec, PictureBox pb)
{
D1.X = x1;
D1.Y = y1;
// для удобной смены координат и перерисовки треугольника, мы подсчитываем
численные зависимости координат
//точек (A,B,C) от фактического центра масс D(x1,y1). Таким образом мы во
многом упрощаем код и структуру метода.
A = new PointF(D1.X + 50f*length, D1.Y + 28.8675f*length);
B = new PointF(D1.X - 50f * length, D1.Y + 28.8675f * length);
```

```
C = new PointF(D1.X, D1.Y - 57.735f * length);
Graphics g = pb.CreateGraphics();
Pen myPen = new Pen(Form1.colorforPen(numrec-1), 2);
g.DrawLine(myPen, A.X, A.Y, B.X, B.Y);
g.DrawLine(myPen, B.X, B.Y, C.X, C.Y);
g.DrawLine(myPen, A.X, A.Y, C.X, C.Y);
Form1.maingraphics.DrawLine(myPen, A.X, A.Y, B.X, B.Y);
Form1.maingraphics.DrawLine(myPen, B.X, B.Y, C.X, C.Y);
Form1.maingraphics.DrawLine(myPen, A.X, A.Y, C.X, C.Y);
drawTriangle(A, B, C, maxrec-1,pb);
}
/// <summary>
/// Отрисовка векторов к центру масс у 1 треугольника
/// </summary>
/// <param name="A">координаты точки A</param>
/// <param name="B">координаты точки B</param>
/// <param name="C">координаты точки C</param>
/// <param name="maxrec">глубина текущей рекурсии</param>
/// <param name="pb">объект для отрисовки</param>
protected void drawTriangle(PointF A, PointF B, PointF C, int maxrec,
PictureBox pb)
{
if (maxrec != 0)
{
PointF D = new PointF();    //Точка центра масс
PointF v1 = new PointF();   //Вектор AB
PointF v2 = new PointF();   //Вектор AC
//Вектор AB
v1.X = B.X - A.X;
v1.Y = B.Y - A.Y;
//Вектор AC
v2.X = C.X - A.X;
```

```
v2.Y = C.Y - A.Y;

D.X = A.X + (v1.X + v2.X) / 3;      //К точке A прибавим сумму векторов AB и
AC, деленную на 3

D.Y = A.Y + (v1.Y + v2.Y) / 3;      //и получим координаты центра масс

Pen myPen = new Pen(Form1.colorforPen(Form1.GetDepth - maxrec), 2);

Graphics g = Graphics.FromHwnd(pb.Handle);

g.DrawLine(myPen, A.X, A.Y, D.X, D.Y);    //Рисуем отрезки от вершин к
центру масс

g.DrawLine(myPen, B.X, B.Y, D.X, D.Y);

g.DrawLine(myPen, C.X, C.Y, D.X, D.Y);

Form1.maingraphics.DrawLine(myPen, A.X, A.Y, D.X, D.Y);    //Рисуем отрезки
от вершин к центру масс

Form1.maingraphics.DrawLine(myPen, B.X, B.Y, D.X, D.Y);

Form1.maingraphics.DrawLine(myPen, C.X, C.Y, D.X, D.Y);

drawTriangle(A, B, D, maxrec - 1,pb);    //Вызываем рекурсивно процендуру
для полученных

drawTriangle(B, C, D, maxrec - 1,pb);    //треугольников, с итерацией,
меньшей на 1

drawTriangle(A, C, D, maxrec - 1,pb);

}

}

}

}
```

8. Список литературы

1. Герберт Шилдт: C# 4.0. Полное руководство // C# 4.0: The Complete Reference, «Вильямс» 2015.
2. Руководство по программированию на C# - MSDN – Microsoft [Электронный ресурс]: русский ресурс, содержащий информацию по документации .NET Framework 4.7.2 , System.Drawing Namespace: [сайт]. URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing?view=netframework-4.7.2 (дата обращения: 01.12.2018).
3. Компьютерная графика // уроки, алгоритмы, программы, примеры. [Электронный ресурс] URL: http://grafika.me/ (дата обращения: 03.12.2018).
4. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ Р 7.0.5—2008. ЕСПД. БИБЛИОГРАФИЧЕСКАЯ ССЫЛКА. Общие требования и правила составления / Издание официальное, Москва, Стандартинформ, 2008.