Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Департамент

Программной инженерии

Контрольное домашнее задание
по дисциплине
«Программирование»

Тема работы: Фракталы

Выполнил(а): студент группы 183 (2)
_____ Имамов Р.М.
тел.89877528126
e-mail адрес: rmimamov@edu.hse.ru

Преподаватель: Чуйкин.Н.К.

Москва, 2018 год. Модуль 2

Оглавление

1. Условие задачи

Вариант №74:
Разработать оконное приложение (шаблон Windows Forms Application) позволяющее:

- Отрисовывать три вида фракталов, которые определены в индивидуальном варианте. Описание фракталов и список вариантов представлены ниже.
- Предоставлять пользователю выбор текущего фрактала для отрисовки.
- Предоставлять пользователю возможность устанавливать количество шагов рекурсии (её глубину - количество рекурсивных вызовов). При изменении глубины рекурсии фрактал должен быть автоматически перерисован. Следите за переполнением стека.
- Автоматически перерисовывать фрактал при изменении размеров окна. Окно обязательно должно быть масштабируемым. Вы можете задать минимальный и максимальный размер окна. Максимальным считается размер окна, соответствующий размеру экрана, а минимальным размером окна считается половинный размер экрана (как по длине, так и по ширине).
- 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.
- Сообщать о некорректном вводе данных, противоречивых или недопустимых значениях данных и других нештатных ситуациях во всплывающих окнах типа MessageBox
- Должна быть предусмотрена возможность сохранения фрактала в виде картинки (формат выбрать самостоятельно).
- Предусмотреть возможность изменения масштаба фрактала для его детального просмотра. Увеличение должно быть 2, 3 и 5-кратным
- Предусмотреть возможность перемещения изображения, в т.ч. при увеличенном изображении.
- На интерфейсе может быть предусмотрена дополнительная функциональность на Ваше усмотрение

Все исходные данные вводит пользователь с помощью экранных форм, содержащих поля для текстового ввода или списки значений для выбора пользователем.

2. Функции разрабатываемого приложения

2.1. Варианты использования

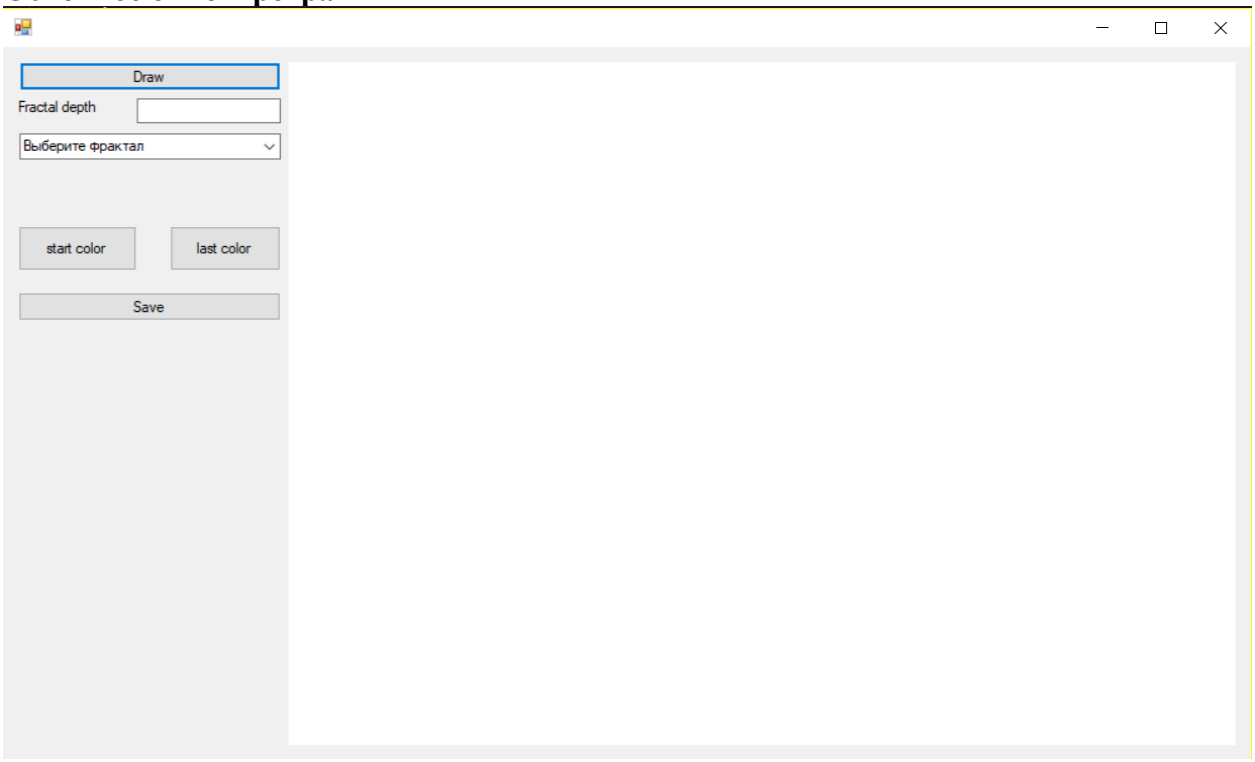
Данное приложение не имеет никакого прикладного применения. Она может служить в виде красивой игрушки.

2.2. Описание интерфейса пользователя

При открытии программы пользователь попадает в основное окно, имеющее несколько функциональный клавиш. Чтобы отрисовать фрактал пользователю следует написать желаемую глубину фрактала, если число слишком большое появится всплывающее окно, которое обозначит границы вводимых значений, также пользователю ему нужно выбрать фрактал , который нужно нарисовать, в противном случае будут появится всплывающее окно, показывающее, что то или иное действие не было выполнено. Также можно выбрать начальный и конечный цвет фрактала, что приведет к рисованию фрактала с цветовым градиентом. Также имеется Возможность сохранения полученного в pictureBox изображения, при нажатии на кнопку SAVE. Также при выборе в ComboBox фрактала «Дерево Пифагора» появится 3 дополнительные функциональные клавиши. Две из которых являются TrackBar`ами. При изменении положения ползунка, будет перерисовываться фрактал. Кроме того повится TextBox, при вводе в которое число от 0,5 до 1,5 измениться соотношение линий фрактала.

1. Button Start вызывает рисование фрактала.
2. Label1 Fractal depth.
3. textBox1 принимает значение глубины фрактала.
4. ComboBox1 дает на выбор 3 фрактала, один из которых нужно выбрать пользователю.
5. Button3 StartColor- начальный цвет рекурсии.
6. Button4 LastColor- конечный цвет рекурсии.
7. Button2 Save – сохраняет изображение на pictureBox.
8. Label3 first angle.
9. Label4 second angle.
10. Label5 coefficient.
11. trackBar1 отвечает за первый угол наклона дерева Пифагора.
12. trackBar2 отвечает за второй угол наклона дерева Пифагора.
13. textBox2 принимает коэффициент отношения линий в дереве Пифагора.
14. PictureBox собственно элемент, на которм рисуются фракталы.

Основное окно Программы



3. Структура приложения

3.1. Описание классов, их полей и методов

Классы:

- **static class Program** класс, содержащий метод Main()
- **public partial class Form1: Form** форма, содержащая основные методы для рисования фракталов.
- **class Fractal** промежуточный класс между From1 и классами самих фракталов, имеющий virtual Draw(), переопределенный в классах наследниках.
- **class Piftree : Fractal** класс наследник переопределяющий Draw() и реализующий рисование Дерева Пифагора.
- **class H-Fractal : Fractal** класс наследник переопределяющий Draw() и реализующий рисование Н-фрактала.
- **class Triangular : Fractal** класс наследник переопределяющий Draw() и реализующий рисование Треугольника центра масс.

Методы:

- **static class Program :**
  - **static void Main()** метод являющий началом программы. В данном методе имеется MassegeBox, которые вызывает приветсвенное окно.
- **public partial class Form1: Form**
  - **private void PictureBox_MouseDown()** метод нажатия левой кнопки мыши.
  - **private void PictureBox_MouseMove()** движение мыши.
  - **void Clear()** очищает PictureBox.
  - **void This_MouseWheel** метод увеличения изображение(вызывает перерисовку фрактала) скролом мыши.
  - **Private void Choise()** метод, созданный для корректной отрисовки фрактала, выбирает один из 3 фракталов.
  - **Private void InputInspect()** проверка глубины ввода
  - **Private void InputInspect2()** проверка выбора фрактала и коэффициента для дерева Пифагора
  - **Private void PictureBoxOne()** метод выбирающий к какому фракталу обратиться чтобы там его отрисовать
  - **Private void Triangular()** метод вызывающий рисование Треугольника центра масс
  - **Private void H_Fractal()** метод вызывающий рисование Н-флактала
  - **Private void Piftree()** метод вызывающий рисование Триугольника центра масс
  - **private List<double> Koef1()** заполняет лист коэффициентов и возвращает его
  - **private void Vision()** скрывает или показывает два трекбола и окошко для коэффициента
  - **private List<Color> Gradient()** заполняет и возвращает лсит цветов для градиента
  - **private void button2_Click()** кнопка Save
  - **private void draw_Click()** кнопка Start
  - **private void Form1_SizeChanged()** изменяет размеры bitmap and grafics также перерисовывает фрактал
  - **private void button3_Click()** startcolor

    - **private void button4_Click()** lastcolor
    - **private void trackBar1_Scroll()** бегунок изменяющий первый угол дерева Пифагора
    - **private void trackBar2_Scroll()** бегунок изменяющий второй угол дерева Пифагора
    - **private void comboBox1_SelectedIndexChanged_1()** выбирает фрактал из комбобокса
- **public class Fractal**
    - **public virtual void Draw(double x, double y,ref Graphics gr, List<Color> colorList, Pen pn)** виртуальный метод который будет переопределен позже
- **class Piftree : Fractal**
    - **public void CustomDraw(double x, double y, ref Graphics gr, List<Color> colorList, Pen pn, double ang = Math.PI /2)** метод реализующий рекурсивное рисование
    - **public override void Draw(double x, double y, ref Graphics gr, List<Color> colorList, Pen pn)** переопределенный метод , который вызывать CustomDrawing
- **class H_fractal : Fractal**
    - **public override void Draw(double x, double y, ref Graphics gr, List<Color> colorList, Pen pn)** переопределенный метод реализующий рекурсивное рисование
- **class Triangular:Fractal**
    - **public override void Draw(double x, double y, ref Graphics gr, List<Color> colorList, Pen pn)** переопределнный дроу вызывающий TreeDots
    - **private void Treedots(Pen pn, double x, double y, List<Color> colorList, ref Graphics gr)** Рисует три точки и сам треуглоьник
    - **private void Drawing(Pen pn, PointF A, PointF B, PointF C, ref Graphics gr, List<Color> colorList)** вызывает рекурсивное рисование фрактала

Поля:

- **public partial class Form1 : Form**
    - **PointF mousePos** точка мышки
    - **PointF center** точка центра питурбокса
    - **PointF nowcenter** переменная центра для движения фрактала
    - **public Graphics gr** графика
    - **public Bitmap mp** битмап
    - **public Pen pn** ручка
    - **int depth** глубина рекурсии
    - **float defaultleng** длина линий
    - **Color defaultPen** стандартный цвет ручки
    - **string selectFractal** выбор фрактала
    - **bool flaginp** флажок
    - **bool flaginpkoef** флажок
    - **int type** флажок
    - **protected double ang1** угол один Дерева Пифагора
    - **protected double ang2** угол два Дерева Пифагора
    - **protected double koef** коэффициент Дерева Пифагора
    - **List<Color> colorList** лист цветов в для градиента
    - **List<double> Koeflist** лист коэффициентов для Дерева Пифагора
- **public class Fractal**
    - **public float leng** длина линий
    - **protected Color startColor** начальный цвет

    - **protected Color lastColor** коннечный цвет
    - **protected int depthzero** счетчик глубины
    - **public int depthmax** глубина рекурсии
- **class Piftree : Fractal**
    - **protected double ang1** первый угол
    - **protected double ang2** второй угол
    - **protected List<double> koef** лсит коэффициентов соотношения сторон
    - **bool flag = true** костыл, чтобы был
- **class H_fractal : Fractal**
    - **bool flag**
- **class Triangular:Fractal**
    - **bool flag** флажок

4. Распределение исходного кода по файлам проекта

1. Form1.cs – файл конструктор
2. Fractal.cs – абстрактный класс фрактала
3. H-fractal.cs – рекурсивное рисование H-фрактала
4. PifTree.cs - рекурсивное рисование дерева Пифагора
5. Triangular.cs - рекурсивное рисование Треугольника центра масс
6. Program.cs - начало программы

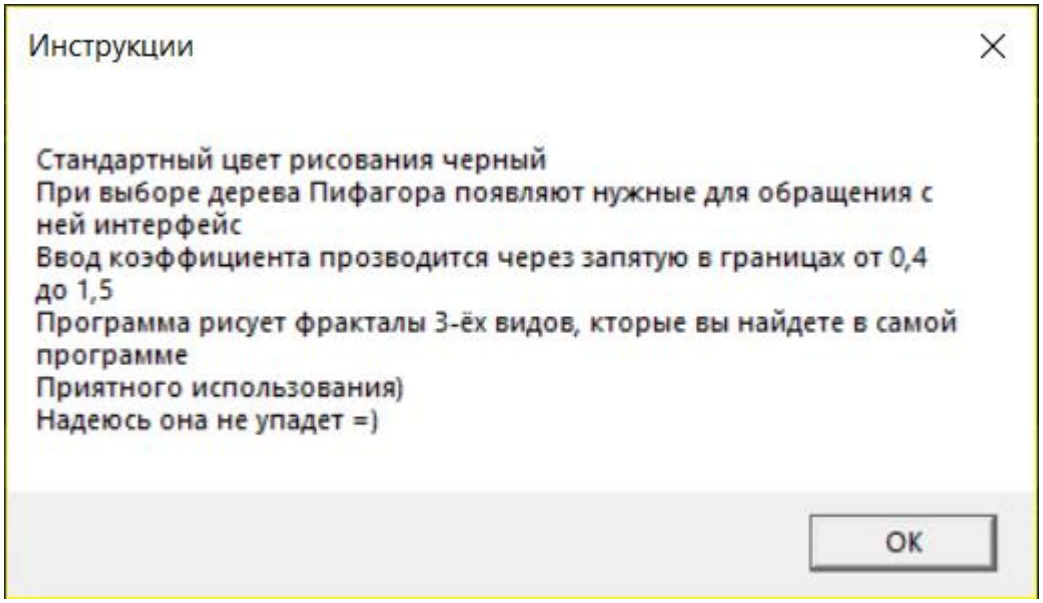
5. Контрольный пример и описание результатов

Для проверки работоспособности программы были прочитаны txt файлы кодировки UTF-8.

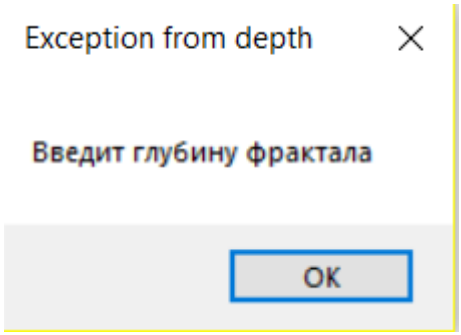
Результаты тестирования:

1. При запуске программы появляется приветсвенное окно с инструкциями
2. Дальше появляется основное окно программы
3. Чтобы она запустилась нужно выбрать один из 3-ёх фракталов и глубину рекурсии. После чего нажать на кнопку START.
4. Также имеется опция выбора начального и конечного цвета для градиента в рисовании фрактала
5. Кроме того, при выборе в комбобоксе Дерева Пифагора появляется новый интерфейс, в котором можно будет изменять наклон ветвей дерева и изменить коэффициент размеров линий фрактала
6. Можно сохранить фрактал нажав на кнопку Save.

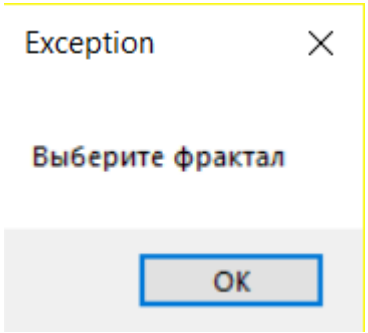
6. Сообщения пользователю



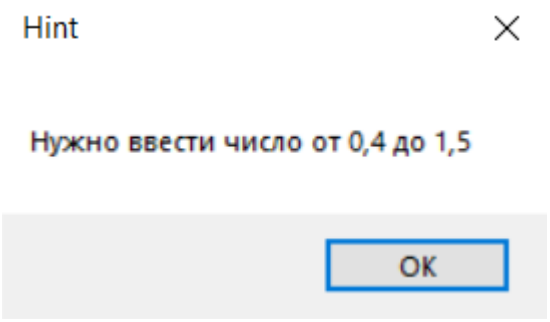
Окно с инструкциями появляется при запуске программы



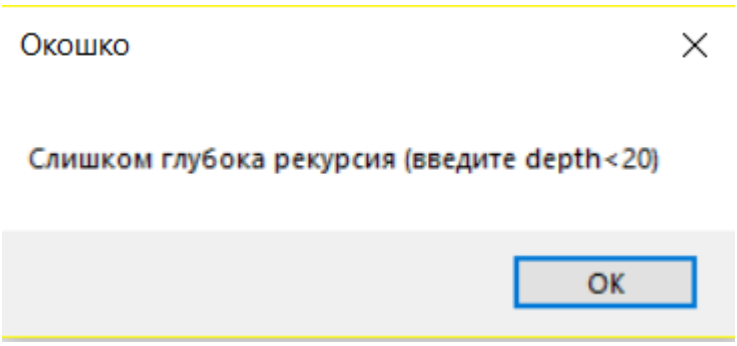
Сообщение появляется при при пустом окне ввода глубины фрактала или некореектном вводе(буквы отрицательное занчение)

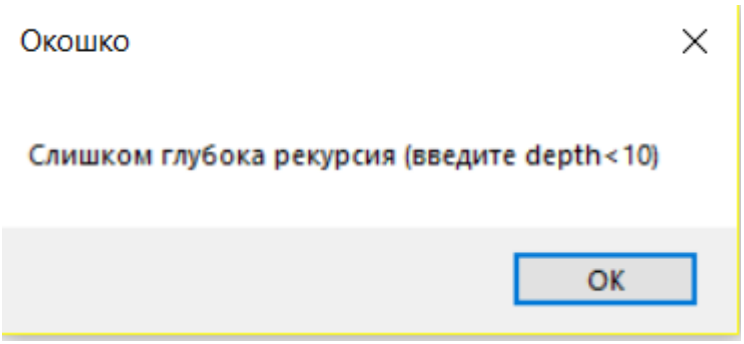


Сообщение появляется если не выбрать фрактал

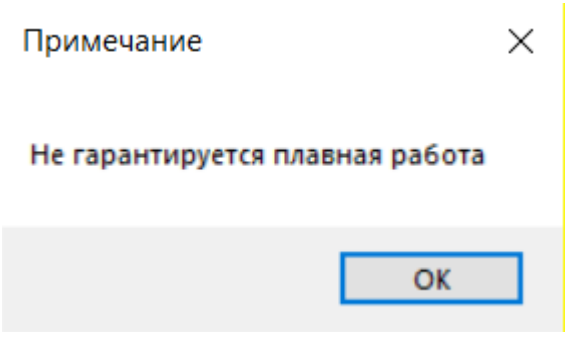


Сообщение появляет если ввести некорректный коэффициент

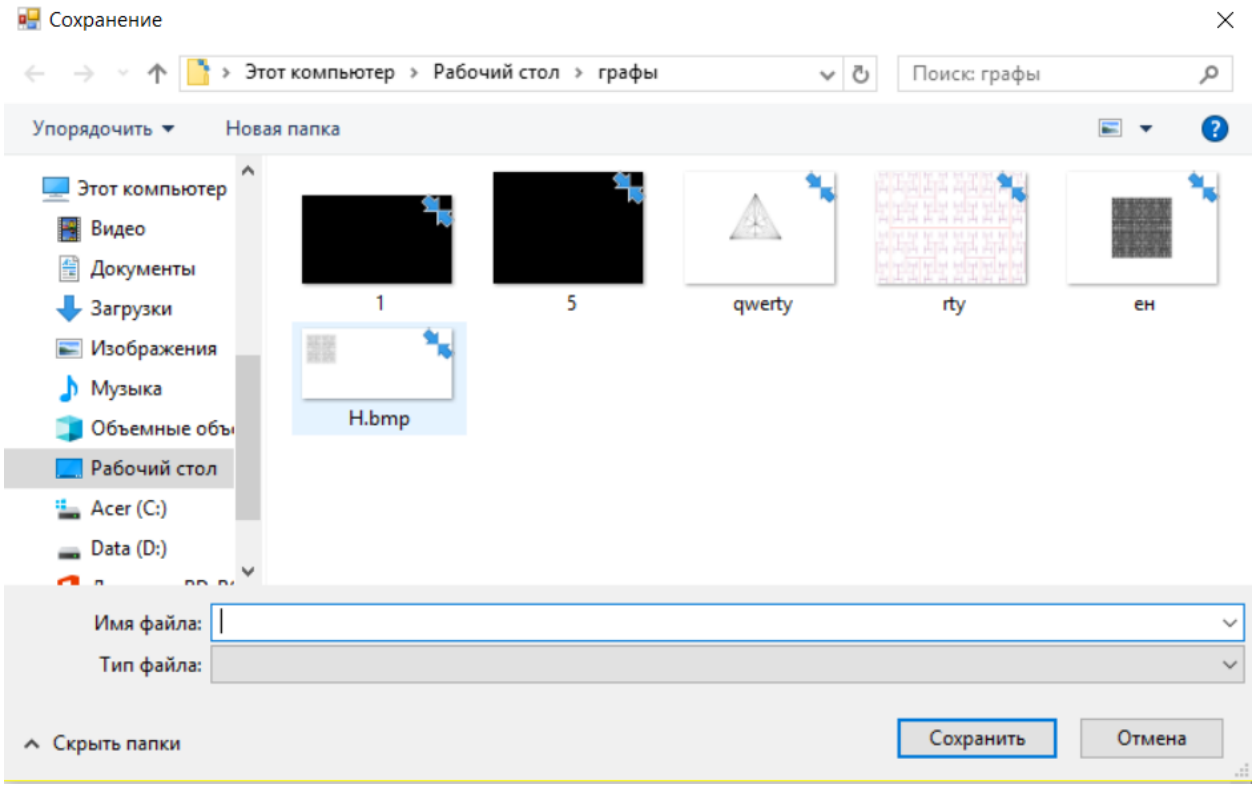




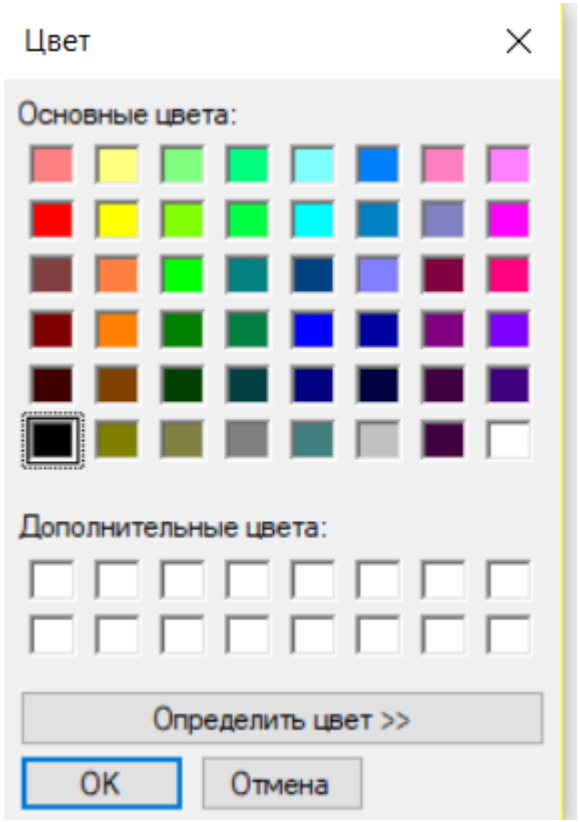
Сообщение появляется при вводе очень глубокой рекурсии



Сообщение появляется при большем числе рекурсии, но при котором прорамма еще может работать



Сохранение



Выбор цвета для градиента

Остальные сообщения не появляются при корректной работе программы, но при непредвиденных ситуациях появляется окно с текстом «Что-то Пошло не так»!

Список литературы

1. Герберт Шилдт: C# 4.0. Полное руководство, 2011.
2. https://stackoverflow.com
3. http://www.cyberforum.ru/programming/
4. https://metanit.com/sharp/tutorial/
5. http://qaru.site/
6. https://docs.microsoft.com/ru-ru/dotnet/csharp/
7. https://tproger.ru/tag/c-sharp/
8. http://grafika.me/node/297

Тест код Программы

```
static class Program
   {
      /// <summary>
      /// Главная точка входа для приложения.
      /// </summary>
      [STAThread]
      static void Main()
      {
         //это сделанно чтобы это сообщение вызывалось всего лишь раз
            MessageBox.Show("Стандартный цвет рисования черный\n" +
               "При выборе дерева Пифагора появляют нужные для обращения с ней
интерфейс\n" +
               "Ввод коэффициента прозводится через запятую в границах от 0,4 до 1,5\n" +
               "Программа рисует фракталы 3-ёх видов, кторые вы найдете в самой
программе\n" +
               "Приятного использования)\n" +
               "Надеюсь она не упадет =)", "Инструкции");

         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }


public partial class Form1 : Form
   {
      PointF mousePos;// точка, в которой щапонимается знчени мыши
      PointF center;//месро откуда начинает рисоваться фрактал
      PointF nowcenter;
      public Graphics gr; //Графика
      public Bitmap mp; //Битмап
      public Pen pn; //Ручка
      int depth;//глубина фрактала
      float defaultleng;//минимальная длина елемента фрактала
      Color defaultPen;// цвет ручки при ее не инициализации
      string selectFractal;
      bool flaginp;
      bool flaginpkoef;
      int type;
      protected double ang1;
      protected double ang2;
      protected double koef;
      List<Color> colorList;
      List<double> Koeflist;
```

```
        /// <summary>
        /// Конструктор
        /// </summary>
        public Form1()
        {
            MinimumSize = new Size(Screen.PrimaryScreen.Bounds.Size.Width/2,
Screen.PrimaryScreen.Bounds.Size.Height/2);
            InitializeComponent();
            center = new PointF(pictureBox.Width / 2, pictureBox.Height / 2);
            pictureBox.SizeMode = PictureBoxSizeMode.Zoom;
            MouseWheel += new MouseEventHandler(This_MouseWheel);
            mp = new Bitmap(pictureBox.Width, pictureBox.Height);
            gr = Graphics.FromImage(mp);
            pn = new Pen(defaultPen);
            defaultPen = Color.Black;
            ang1 = Math.PI / 4;
            ang2 = Math.PI / 6;
            koef = 1;

        }


        /// <summary>
        /// even click on mouse
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pictureBox_MouseDown(object sender, MouseEventArgs e)
        {

            mousePos = MousePosition;
            nowcenter = center;
        }

        /// <summary>
        /// movind picture
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pictureBox_MouseMove(object sender, MouseEventArgs e)
        {
            if ((selectFractal != null) || (depth != 0))
            {

                if (e.Button == MouseButtons.Left)
                {

                    DoubleBuffered = true;

                    center.X = nowcenter.X - mousePos.X + MousePosition.X;
                    center.Y = nowcenter.Y - mousePos.Y + MousePosition.Y;
                    Choise();
```

```
        }
    }

}

/// <summary>
/// Clear pictureBox
/// </summary>
void Clear()
{
    gr.Clear(Color.White);
    pictureBox.Image = mp;
}

/// <summary>
/// ZOOM
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void This_MouseWheel(object sender, MouseEventArgs e)
{
    if((selectFractal!=null)||(depth!=0))
    {
        if (e.Delta != 0)
        {
            if (e.Delta <= 0)
            {

                defaultleng /= e.Delta / 60 * (-1);//уменьшает изображение в 2 раза
                if (defaultleng < 1 || double.IsInfinity(defaultleng))
                    defaultleng = 1;

                Choise();

                //pictureBoxOne(sender, e);

                return;
            }
            else
            {

                defaultleng += e.Delta / 24;//увеличивает в на 5 т.е в 2 3 4 5 6 7 8 и далее раз
                if (defaultleng > 100 || double.IsInfinity(defaultleng))
                    defaultleng = 100;

                Choise();
                //pictureBoxOne(sender, e);

                return;
            }
        }
```

```
            }
        }

        /// <summary>
        /// выбор отрисовка фрактала(необходимый метод для правильной работы
приувеличении и перемещении картинки1
        /// </summary>
        private void Choise()
        {
            switch (type)
            {
                case (3):
                    Triangular();
                    break;
                case (2):
                    H_Fractal();
                    break;
                case (1):
                    Piftree();
                    break;


            }

        }

        /// <summary>
        /// depth input
        /// </summary>
        private void InputInspect()
        {
            try
            {
                depth = int.Parse(textBox1.Text);
                if (int.Parse(textBox1.Text) <= 0)
                {
                    throw new ArgumentException();
                }
            }
            catch (ArgumentException ex)
            {
                MessageBox.Show("Некоррректный ввод глубины фрактала", "Exception from
depth");
                flaginp = true;
                return;
            }
            catch (FormatException ext)
            {
                MessageBox.Show("Некоррректный ввод глубины фрактала", "Exception from
depth");
                flaginp = true;
                return;
```

```
            }
            catch (Exception)
            {
                MessageBox.Show("Что-то пошло не так!?", "Exception");
                flaginp = true;
                return;
            }

             if (selectFractal == null) MessageBox.Show("Выберите фрактал",
"Exception");//просто чтобы выкинуть окошко
        }

        /// <summary>
        /// проверка ввода коэффициенциента
        /// </summary>
        private void InputInspect2()
        {
            try
            {
                if (textBox2.Text == "") return;
                 koef = double.Parse(textBox2.Text);
                if (double.Parse(textBox2.Text) <= 0)
                {
                    throw new ArgumentException();
                }
            }
            catch (ArgumentException ex)
            {
                MessageBox.Show(ex.Message, "Exception");
                flaginpkoef = true;
                return;
            }

            catch (FormatException ext)
            {
                MessageBox.Show(ext.Message, "Exception");
                flaginpkoef = true;
                return;
            }
            try
            {
                if ((koef < 0.4) || (koef > 1.5))
                {
                    throw new ArgumentException();
                }
            }
            catch(ArgumentException)
            {
                MessageBox.Show("Нужно ввести число от 0,4 до 1,5", "Hint");
                flaginpkoef = true;
                koef = 1;
                return;
```

```
        }

    }

    /// <summary>
    /// Main function
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void pictureBoxOne(object sender, EventArgs e)
    {

        switch (selectFractal)
        {
            case "Дерево Пифагора":
                type = 1;
                if ((depth > 11) && (depth <=20)) MessageBox.Show("Не гарантируется плавная
работа", "Примечание");
                if (depth > 20) { MessageBox.Show("Слишком глубока рекурсия (введите
depth<20)", "Окошко"); return; }
                Piftree();
                return;

            case "H-фрактал":
                type = 2;
                if ((depth > 5) && (depth <= 10)) MessageBox.Show("Не гарантируется плавная
работа", "Примечание");
                if (depth > 10) { MessageBox.Show("Слишком глубока рекурсия (введите
depth<10)", "Окошко"); return; }
                H_Fractal();
                return;

            case "Центр масс треугольника":
                type = 3;
                if ((depth > 7) && (depth <= 10)) MessageBox.Show("Не гарантируется плавная
работа", "Примечание");
                if (depth > 10) { MessageBox.Show("Слишком глубока рекурсия (введите
depth<10)", "Окошко"); return; }
                Triangular();
                return;

            case "":
                MessageBox.Show("Выберите фрактал", "Input Exception");
                return;

        }

        pictureBox.Image = mp;

    }

    /// <summary>
```

```
        /// вызов треугольника центра масс
        /// </summary>
        /// <param name="colorList"></param>
        private void Triangular()
        {
            Clear();
            Fractal frac1 = new Triangular(Color.Black, Color.Black, depth, defaultleng * 30);
            frac1.Draw(center.X, center.Y, ref gr, colorList, pn);
        }

        /// <summary>
        /// вызов H-фрактала
        /// </summary>
        /// <param name="colorList"></param>
        private void H_Fractal()
        {
            Clear();
            Fractal frac1 = new H_fractal(Color.Black, Color.Black, depth, defaultleng*30);
            frac1.Draw(center.X, center.Y, ref gr, colorList, pn);
        }

        /// <summary>
        /// вызов дерева пифагора
        /// </summary>
        /// <param name="colorList"></param>
        private void Piftree()
        {
            Clear();
            Fractal frac = new Piftree(Color.Black, Color.Black, depth, defaultleng, Koeflist, ang1,
ang2);
            frac.Draw(center.X, center.Y, ref gr, colorList, pn);
        }

        /// <summary>
        /// изменение коэффицента
        /// </summary>
        /// <param name="Koef"></param>
        private List<double> Koef1()
        {
            List<double> Koefls = new List<double>();

            if (textBox2.Text != "")
            {
                double k=1;
                //if (koef > 1) k = koef / depth;
                //else k = koef / -depth;

                for (int i = 0; i < depth; i++)
                {
                    k *= koef;
                    Koefls.Add(k);
                }
```

```
    }
    else
    {
        for (int i = 0; i < depth; i++)
        {
            Koefls.Add(1);
        }
    }

    return Koefls;
}

/// <summary>
/// видимость тракбаров для измения угла дерева пифагора
/// </summary>
private void Vision()
{
    if(selectFractal== "Дерево Пифагора")
    {
        label3.Visible = true;
        label4.Visible = true;
        label5.Visible = true;
        trackBar1.Visible = true;
        trackBar2.Visible = true;
        textBox2.Visible = true;
    }
    else
    {
        label3.Visible = false;
        label4.Visible = false;
        label5.Visible = false;
        trackBar1.Visible = false;
        trackBar2.Visible = false;
        textBox2.Visible = false;
    }
}

/// <summary>
/// Color gradient
/// </summary>
/// <param name="colorList"></param>
private List<Color> Gradient()
{
    List<Color> CL= new List<Color>();
    int rMin = colorDialog1.Color.R;
    int gMin = colorDialog1.Color.G;
    int bMin = colorDialog1.Color.B;
    int argMin = colorDialog1.Color.ToArgb();

    if (argMin == 0)
    {
        argMin = defaultPen.ToArgb();
```

```
            }

            int rMax = colorDialog2.Color.R;
            int gMax = colorDialog2.Color.G;
            int bMax = colorDialog2.Color.B;
            int argMax = colorDialog2.Color.ToArgb();

            if (argMax == 0)
            {
                argMax = defaultPen.ToArgb();
            }

            for (int i = 0; i <= depth; i++)
            {
                var rAverage = rMin + ((rMax - rMin) * i / depth);
                var gAverage = gMin + ((gMax - gMin) * i / depth);
                var bAverage = bMin + ((bMax - bMin) * i / depth);
                CL.Add(Color.FromArgb(rAverage, gAverage, bAverage));
            }
            return CL;
        }

        /// <summary>
        /// SAVE
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button2_Click(object sender, EventArgs e)
        {
            SaveFileDialog save = new SaveFileDialog();
            save.ShowDialog();
            string filename = save.FileName + ".jpg";
            mp.Save(filename, System.Drawing.Imaging.ImageFormat.Jpeg);
        }

        /// <summary>
        /// button start
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void draw_Click(object sender, EventArgs e)
        {
            center = new PointF(pictureBox.Width / 2, pictureBox.Height / 2);
            defaultleng = 5;//каждый раз при нажатии кнопки старт размер фрактала становится
стандартным
            InputInspect();
            if (flaginp) { flaginp = false; return; }//флажок для того чтобы не вылетаало окно
ошибки по миллион раз
            colorList = Gradient();
            if (selectFractal == "Дерево Пифагора")
            {
                InputInspect2();
```

```
            if (flaginpkoef) { flaginpkoef = false; return; }//флажок для того чтобы не
вылетаало окно ошибки по миллион раз
                Koeflist = Koef1();
            }
            pictureBoxOne(sender, e);
        }

        /// <summary>
        /// Измение размера pictureBox и перерисовка фрактала
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Form1_SizeChanged(object sender, EventArgs e)
        {

            try
            {
                mp = new Bitmap(pictureBox.Width, pictureBox.Height);
                gr = Graphics.FromImage(mp);
            }
            catch(ArgumentException)//ловит ошибку но не выдает сообщения т.к. это не
зависит от пользователя
            {
            }
            catch (Exception)
            {
                MessageBox.Show("Что-то пошло не так!?", "Exception");
            }

            if ((selectFractal != null) || (depth != 0))
                Choise();




        }

        /// <summary>
        /// ColorDialog(startcolor)
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button3_Click(object sender, EventArgs e)
        {
            colorDialog1.ShowDialog();
            button3.BackColor = colorDialog1.Color;//красим кнопку
            if (button3.BackColor == Color.Black) button3.ForeColor = Color.White;//красим
буквы
            if (button3.BackColor != Color.Black) button3.ForeColor = Color.Black;//красим буквы


        }
```

```
        /// <summary>
        /// ColorDialog2(lastcolor)
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button4_Click(object sender, EventArgs e)
        {
            colorDialog2.ShowDialog();
            button4.BackColor = colorDialog2.Color;//красим кнопку
            if (button4.BackColor == Color.Black) button4.ForeColor = Color.White;//красим
буквы
            if (button4.BackColor != Color.Black) button4.ForeColor = Color.Black;//красим буквы



        }

        /// <summary>
        /// делает красивыую анимацию обдувания
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            if (type == 1)
            {
                ang1 = Math.PI / trackBar1.Value;
                Piftree();
            }
        }

        /// <summary>
        /// делает красивую аимацию обдувания
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void trackBar2_Scroll(object sender, EventArgs e)
        {
            if (type == 1)
            {
                ang2 = Math.PI / trackBar2.Value;
                Piftree();
            }
        }

        /// <summary>
        /// метод для выбора фрактала из комбобокса
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void comboBox1_SelectedIndexChanged_1(object sender, EventArgs e)
```

```
        {
            try
            {
                selectFractal = comboBox1.SelectedItem.ToString();
                Vision();
                if (selectFractal == null) throw new NullReferenceException();
            }
            catch (NullReferenceException)
            {
                MessageBox.Show("Выберите фрактал", "Exception");
            }
            catch (Exception)
            {
                MessageBox.Show("Что-то пошло не так!?", "Exception");
            }
        }
    }



public class Fractal
    {
        public float leng;                  //по тз
        protected Color startColor;         //по тз
        protected Color lastColor;          //по тз
        protected int depthzero;            //по тз
        public int depthmax;                //по тз

        public Fractal( Color startColor, Color lastColor, int depthmax, float leng)
        {
            this.leng = leng;                       //ну банально инициализируем
            this.startColor = startColor;           //ну банально инициализируем
            this.lastColor = lastColor;             //ну банально инициализируем
            this.depthmax = depthmax;               //ну банально инициализируем
            this.depthzero = 0;
        }

        /// <summary>
        /// виртуальный метод котрый переопределяем в наследниках
        /// </summary>
        /// <param name="x"></param>
        /// <param name="y"></param>
        /// <param name="gr"></param>
        /// <param name="colorList"></param>
        /// <param name="pn"></param>
        public virtual void Draw(double x, double y,ref Graphics gr, List<Color> colorList, Pen pn)
        {

        }

    }
```

```
class Triangular:Fractal
   {
      bool flag = true;//флажок чтобы был
      public Triangular(Color startColor, Color lastColor, int depthmax, float leng) :
base(startColor, lastColor, depthmax, leng)
      {
      }

      public override void Draw(double x, double y, ref Graphics gr, List<Color> colorList, Pen
pn)
      {
         pn.Color = colorList[depthzero];//сразу инициализируем ручку

         Treedots(pn, x, y, colorList,ref gr);//тк передаю точко центр надо сначала найти
вершины треугольника

      }

      /// <summary>
      /// вершины треугольника
      /// </summary>
      /// <param name="pn"></param>
      /// <param name="x"></param>
      /// <param name="y"></param>
      /// <param name="colorList"></param>
      /// <param name="gr"></param>
      private void Treedots(Pen pn, double x, double y, List<Color> colorList, ref Graphics gr)
      {
         PointF A = new PointF((float)x,(float)(y-leng));
         PointF B = new PointF((float)(x + leng * Math.Cos(Math.PI / 6)), (float)(y + leng /2));
         PointF C = new PointF((float)(x - leng * Math.Cos(Math.PI / 6)), (float)(y + leng /2));

         gr.DrawLine(pn, A.X, A.Y, B.X, B.Y);//рисуем первый треугольник
         gr.DrawLine(pn, B.X, B.Y, C.X, C.Y);//рисуем первый треугольник
         gr.DrawLine(pn, A.X, A.Y, C.X, C.Y);//рисуем первый треугольник

         Drawing(pn, A , B , C , ref gr, colorList);//переходим к рисованию самого фрактала

      }

      /// <summary>
      /// собсвеннно сам фрактал
      /// </summary>
      /// <param name="pn"></param>
      /// <param name="A"></param>
      /// <param name="B"></param>
      /// <param name="C"></param>
      /// <param name="gr"></param>
      /// <param name="colorList"></param>
      private void Drawing(Pen pn, PointF A, PointF B, PointF C, ref Graphics gr, List<Color>
colorList)
      {
```

```
            if (depthzero == depthmax) return;

            PointF D = new PointF();   //точка флоат)

            depthzero++;

            pn.Color = colorList[depthzero];

            D.X = (A.X + B.X + C.X)/3; // находим координаты центра
            D.Y = (A.Y + B.Y + C.Y)/3; // находим координаты центра

            try
            {
               gr.DrawLine(pn, A.X, A.Y, D.X, D.Y); //рисуем линии
               gr.DrawLine(pn, B.X, B.Y, D.X, D.Y); //рисуем линии
               gr.DrawLine(pn, C.X, C.Y, D.X, D.Y); //рисуем линии
            }
            catch (OverflowException e)//вроде не должно выпадать но на всякий случай))
            {
               if (flag)
               {
                  MessageBox.Show(e.Message, "Exception");
                  flag = false;
               }
               return;
            }
            catch (Exception)
            {
               MessageBox.Show("Что-то пошло не так!?", "Exception");
            }

            //depthzero++;

            Drawing(pn, A, B, D, ref gr, colorList);  //заходим в рекурсию
            Drawing(pn, B, C, D, ref gr, colorList);  //заходим в рекурсию
            Drawing(pn, A, C, D, ref gr, colorList);  //заходим в рекурсию

            depthzero--;
         }

      }


class H_fractal : Fractal
   {
      bool flag = false;//флажок чтобы был
      public H_fractal(Color startColor, Color lastColor, int depthmax, float leng) :
base(startColor, lastColor, depthmax, leng)
      {

      }
```

```
        public override void Draw(double x, double y, ref Graphics gr, List<Color> colorList, Pen
pn)
        {
            depthzero++;
            leng /= 2;//уменьшаем длину тк так в тз)

            if(depthzero<depthmax)
            {
                try
                {
                    Draw(x - leng, y - leng, ref gr, colorList, pn);       //тут без магии все рисуем и не
напрягаемся
                    leng *= 2;                                  //тут без магии все рисуем и не напрягаемся
                    Draw(x - leng, y + leng, ref gr, colorList, pn);       //тут без магии все рисуем и не
напрягаемся
                    leng *= 2;                                  //тут без магии все рисуем и не напрягаемся
                    Draw(x + leng, y - leng, ref gr, colorList, pn);       //тут без магии все рисуем и не
напрягаемся
                    leng *= 2;                                  //тут без магии все рисуем и не напрягаемся
                    Draw(x + leng, y + leng, ref gr, colorList, pn);       //тут без магии все рисуем и не
напрягаемся
                    leng *= 2;
                }
                catch (OverflowException e)//вроде не должно выпадать но на всякий случай))
                {
                    if (flag)
                    {
                        MessageBox.Show(e.Message, "Exception");
                        flag = false;
                    }
                    return;
                }
                catch (Exception)
                {
                    MessageBox.Show("Что-то пошло не так!?", "Exception");
                }
            }

            depthzero--;
            pn.Color = colorList[depthzero];//очевидно цвет линий

            gr.DrawLine(pn, (float)(x - leng), (float)y,(float)( x + leng), (float)y);
//уходим в рекурсию))
            gr.DrawLine(pn, (float)(x - leng),(float)( y - leng), (float)(x - leng), (float)(y + leng));
//уходим в рекурсию))
            gr.DrawLine(pn, (float)(x + leng), (float)(y - leng), (float)(x + leng), (float)(y + leng));
//уходим в рекурсию))
        }
    }

class Piftree : Fractal
    {
```

```
        protected double ang1;
        protected double ang2;
        protected List<double> koef;
        bool flag = true;//флажок чтобы был


        public Piftree(Color startColor, Color lastColor, int depthmax, float leng, List<double>
koef, double ang1, double ang2):
            base(startColor,lastColor, depthmax, leng)
        {
            this.koef = koef;
            this.ang1 = ang1;
            this.ang2 = ang2;
        }

        public void CustomDraw(double x, double y, ref Graphics gr, List<Color> colorList, Pen
pn, double ang = Math.PI / 2)
        {
            if (depthmax != 0)
            {
                double x1 = Math.Round(x + (depthmax * Math.Cos(ang)) * leng * koef[depthzero]),
                       y1 = Math.Round(y - (depthmax * Math.Sin(ang)) * leng *  koef[depthzero]);

                pn.Color = colorList[depthzero];

                try
                {
                    gr.DrawLine(pn, (float)x, (float)y, (float)x1, (float)y1);//рисуем линию ловим
переполнение
                }
                catch (OverflowException e1)//вроде не должно выпадать но на всякий случай))
                {
                    if (flag)
                    {
                        MessageBox.Show(e1.Message, "Exception");
                        flag = false;
                    }
                    return;
                }
                catch (Exception)
                {
                    MessageBox.Show("Что-то пошло не так!?", "Exception");
                }

                x = x1;
                y = y1;


                depthzero++;

                depthmax--;
```

```
            pn.Color = colorList[depthzero];

            ang += ang1;
            CustomDraw(x, y, ref gr, colorList, pn, ang);

            pn.Color = colorList[depthzero];

            ang -= ang1;
            ang -= ang2;


            CustomDraw(x, y, ref gr, colorList, pn, ang);

            depthzero--;

            ang += ang2;

            depthmax++;

        }
    }
    /// <summary>
    /// пришлось запариться, чтобы было красиво передать углы
    /// </summary>
    /// <param name="x"></param>
    /// <param name="y"></param>
    /// <param name="gr"></param>
    /// <param name="colorList"></param>
    /// <param name="pn"></param>
    public override void Draw(double x, double y, ref Graphics gr, List<Color> colorList, Pen
pn)
    {
        CustomDraw(x, y, ref gr, colorList, pn);
    }

}
```