НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Дисциплина : «Программирование»

Контрольное Домашнее задание 1

Фракталы в Windows Forms

Вариант 21

Выполнил: Бенуа Кристиан,

студент гр.БПИ182

Преподаватель: Горденко М.К

Москва 2018

Содержание

Бенуа. Вариант 21

1. Условие задачи

Разработать оконное приложение Windows Forms, позволяющее:

1. Отрисовывать три вида фракталов(Вариант 21):
   (a) Кривая Коха
   (b) H-Фрактал
   (c) Круговой Фрактал
2. Предоставлять пользователю выбор текущего фрактала для отрисовки.
3. Предоставлять пользователю возможность устанавливать количество шагов рекурсии. 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. Промежуточные цвета вычисляются исходя из начального и конечного значений цвета и номера итерации (не генерируются случайным образом).
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
10. На интерфейсе может быть предусмотрена дополнительная функциональность на Ваше усмотрение

2. Функции разрабатываемого приложения

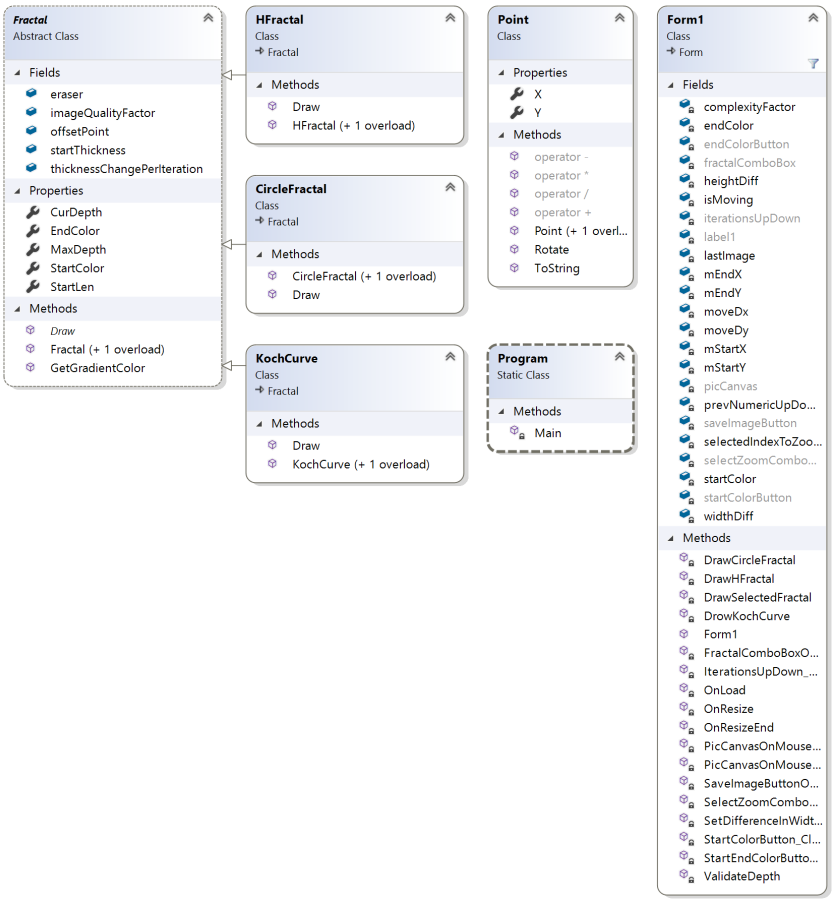
2.1 Варианты Использования

Программа может использоваться для рисования, анализа и сохранения нарисованного фрактала. Пользователь может выбирать фрактал для отрисовки, увеличение, начальный и конечный цвет линейного градиента. После окончания работы пользователь может сохранить фрактал в формате *.png*.

3. Структура Приложения

3.1 Диаграмма классов

Рис. 1: Диаграмма классов



3.2 Описание классов, полей и методов

3.2.1 Form1

Класс основной формы этого приложения. Все взаимодействия с пользователем описано в нем.

3.2.1.1 Поля

1. picCanvas - PictureBox, в котором будет рисоваться фрактал.
2. prevNumericUpDownValue - предыдущее. значение numericUpDown элемента iterationsUpDown
3. selectedIndexToZoom - массив, в котором по i-ому индексу можно получить численное значения увеличения по выбранному индексу selectZoomComboBox.
4. complexityFactor - массив, обозначающий количество новых объектов для отрисовки следующей соотвествующего фрактала.

5. lastImage - Bitmap, в которой хранится последний нарисованный фрактал.
6. mEndX - x-координата позиции на экране, где пользователь отпустил левую кнопку мыши.
7. mEndY - y-координата позиции на экране, где пользователь отпустил левую кнопку мыши.
8. mStartX - x-координата позиции на экране, где пользователь нажал левую кнопку мыши.
9. mStartY - y-координата позиции на экране, где пользователь нажал левую кнопку мыши.
10. moveDx - x-координата последнего смещения картинки.
11. moveDy - double, y-координата последнего смещения картинки.
12. heightDiff - double, разница в высоте формы Form1 и picCanvas.
13. widthDiff - double, разница в ширине формы Form1 и picCanvas.
14. startColor - Color, начальный цвет линейного градиента.
15. endColor - Color, конечный цвет линейного градиента.
16. isMoving - bool, начал ли пользователь движение картинки.
17. selectZoomComboBox - ComboBox для выбора коэффициента увелечения картинки.
18. fractalComboBox - ComboBox для выбора фрактала для отрисовки.
19. saveImageButton - Button для открытия SaveFileDialog для последующего сохранения картинки.
20. startColorButton - Button, открывающий диалог для выбора начального цвета линейного градиента.
21. endColorButton - Button, открывающий диалог для выбора конечного цвета линейного градиента.
22. label1 - Label, для элемента iterationsUpDown.
23. iterationsUpDown - NumericUpDown, в котором можно изменять текущую глубину построения фракталов.

3.2.1.2 Методы

1. Form1 - конструктор формы, задает начальное положение элементам, рисует Кривую Коха, присваивает обработчики событий многим элементам, расположенным на форме.
2. OnResize - обработчик события изменения размера формы Form1, проверяет, развернуто ли окно на весь экран, если да, то вызывает перерисовку фрактала.
3. SaveImageButtonOnClick - обработчик события нажатия на кнопку saveImageButton, открывает SaveFileDialog для выбора места сохранения картинки, если место было выбрано, сохраняет картинку.
4. PicCanvasOnMouseDown - обработчик события нажатия на PictureBox picCanvas, запоминает координаты курсора в момент нажатия.

5. PicCanvasOnMouseUp - обработчик события окончания нажатия PictureBox picCanvas, запоминает координаты курсора в этот момент, считает насколько сместился курсор, вызывает перерисовку фрактала с новой координатой левого верхнего угла картинки на PictureBox picCanvas.
6. ValidateDepth - устанавливает различные максимальные значения глубины для разных фракталов.
7. SelectZoomComboBoxOnSelectedIndexChanged - обработчик события изменения выбранного увеличения в selectZoomComboBox, вызывает перерисовку фрактала.
8. FractalComboBoxOnSelectedIndexChanged -обработчик события изменения выбранного фрактала в fractalComboBox, вызывает перерисовку выбранного фрактала.
9. OnLoad - обработчик события загрузки формы, вызывает SetDifferenceInWidthAndHeight, и рисует Кривую Коха.
10. DrawSelectedFractal - в зависимости от выбранного элемента в fractalComboBox, рисует соответствующий фрактал.
11. OnResizeEnd - обработчик события окончания изменения размера формы Form1, изменяет размер picCanvas и перерисовывает фрактал
12. SetDifferenceInWidthAndHeight - считает разницу между высотой и широтой picCanvas и формы Form1.
13. DrowKochCurve - рисует Кривую Коха.
14. DrawHFractal - рисует H-Фрактал.
15. DrawCircleFractal - рисует Круговой Фрактал.
16. IterationsUpDown_ValueChanged - обработчик события изменения значения в iterationsUpDown, вызывает функцию перерисовки фрактала.
17. StartColorButton_Click - обработчик нажатия на startColorButton, открывает ColorDialog для выбора цвета startColor.
18. StartEndColorButton_Click - обработчик нажатия на endColorButton, открывает ColorDialog для выбора цвета endColor.

3.2.2 Fractal

Абстрактный класс, объявляющий или описывающий методы, которые будут нужны как вспомогательные или будут в дочерних классах.

3.2.2.1 Поля

1. eraser - Pen, отвечающий за стирание уже нарисованных частей фрактала
2. imageQualityFactor - float, отвечает за увеличение картинки и её сохранение в достойном качестве
3. offsetPoint - Point, отвечает за левый верхний угол рисующегося фрактала на Bitmap

4. startThickness - float, отвечает за толщину нарисованных линий на первой итерации
5. thicknessChangePerIteration - float, отвечает за изменении толщины нарисованных линий на соседних уровнях глубины фракталов

3.2.2.2 Свойства

1. CurDepth - int, обозначает текущую глубину рекурсии фрактала.
2. MaxDepth - int, обозначает количество шагов рекурсии, которое задал пользователь.
3. StartColor - Color, начальный цвет линейного градиента.
4. EndColor - Color, конечный цвет линейного градиента.
5. StartLen - double, длина линий или радиус окружности для первого шага рекурсии.

3.2.2.3 Методы

1. Draw - абстрактный метод для рисования фрактала, будет перегружен в наследниках класса
2. Fractal - конструктор:
   (a) Пустой конструктор(без параметров) задает всем методам(которым нужно) значение по умолчанию.
   (b) Полный Конструктор задает методам StartLen, StartColor, EndColor, MaxDepth, CurDepth значения, переданные конструктору.
3. GetGradientColor - генерирует цвет для номера итерации с помощью линейного градиента.

3.2.3 HFractal

Класс, наследник Fractal, реализует методы, необходимые для рисования H-Фрактала

3.2.3.1 Методы

1. Draw - получает на вход картинку(Bitmap), на которой будет рисовать и середину буквы ‘H‘, которую после рисовать и, если нужно, запускает процесс рисования на следующем шаге.
2. HFractal - конструктор:
   (a) Пустой конструктор задает всем методам(которым нужно) значение по умолчанию.
   (b) Полный Конструктор задает методам StartLen, StartColor, EndColor, MaxDepth, CurDepth значения, переданные конструктору.

3.2.4 CircleFractal

Класс, наследник Fractal, реализует методы, необходимые для рисования Кругового Фрактала

3.2.4.1 Методы

1. Draw - получает на вход картинку (Bitmap), на которой будет рисовать и центр окружности, которую он рисует и, если нужно, запускает процесс рисования на следующем шаге.
2. CircleFractal - конструктор:
   (a) Пустой конструктор задает всем методам(которым нужно) значение по умолчанию.
   (b) Полный Конструктор задает методам StartLen, StartColor, EndColor, MaxDepth, CurDepth значения, переданные конструктору.

3.2.5 KochCurve

Класс, наследник Fractal, реализует методы, необходимые для рисования Кривой Коха

3.2.5.1 Методы

1. Draw - получает на вход картинку (Bitmap), на которой будет рисовать и две точки: самую левую и самую правую точку текущей части фрактала. Трансформирует отрезок прямой, рисует его, и, если нужно, запускает процесс рисования на следующем шаге.
2. KochCurve - конструктор:
   (a) Пустой конструктор задает всем методам(которым нужно) значение по умолчанию.
   (b) Полный Конструктор задает методам StartLen, StartColor, EndColor, MaxDepth, CurDepth значения, переданные конструктору.

3.2.6 Point

Класс, в котором определены базовые операции с точками(Есть те, которых нет в встроенном классе Point или PointF).

3.2.6.1 Свойства

1. X - автореализуемое свойство, x-координата точки.
2. Y - автореализуемое свойство, y-координата точки.

3.2.6.2 Методы

1. Point - конструктор
   (a) Пустой конструктор, задает свойствам X, Y значения по умолчанию.
   (b) Полный конструктор, задает свойствам X, Y значения, переданные конструктору.
2. Rotate - принимает на вход точку и угол в радианах, на который ее нужно повернуть. Возвращает повернутую точку.
3. operator- - перегруженный оператор вычитания для двух экземпляров Point, возвращает точку, координаты которой являются разностью левого и правого операнда.

4. operator+ - перегруженный оператор сложения для двух экземпляров Point, возвращает точку, координаты которой являются суммой левого и правого операнда.
5. operator* - перегруженный оператор для умножения экземпляра точки на скаляр(вещественное число), возвращает точку, координаты которой являются произведение координат левого операнда(Point) и правого операнда(вещественное число).
6. operator/ - перегруженный оператор для деления экземпляра точки на скаляр(вещественное число), возвращает точку, координаты которой являются частным координат левого операнда(Point) и правого операнд(вещественное число).
7. ToString - переопределенное преобразование к строке, возвращает строчку в формате (X, Y).

3.2.7 Program

Файл был сгенерирован средой автоматически.

3.2.7.1 Методы

1. Main - точка входа в программу.

4. Распределение исходного кода по файлам проекта

4.1 Form1.cs

Файл, который описывает все взаимодействие пользователя с интерфейсом:

1. Выбор фрактала для отрисовки
2. Выбор увеличения
3. Выбор начального и конечного цвета линейного градиента
4. Сохранение картинки с нарисованным фракталом

4.2 Fractals.cs

Файл описывает класс Fractal, определяющий интерфейс наследников и вспомогательные методы для них.

4.3 Point.cs

Файл описывает класс Point, в котором переопределены операторы сложения, вычитания точек, умножения и деления на скаляр, поворот точки на угол.

4.4 KochCurve.cs

Файл описывает класс KochCurve, в котором определен метод рисования Кривой Коха

4.5 HFractal.cs

Файл описывает класс HFractal, в котором определен метод рисования H-Фрактала

4.6 CircleFractal.cs

Файл описывает класс CircleFractal, в котором определен метод рисования Кругового Фрактала

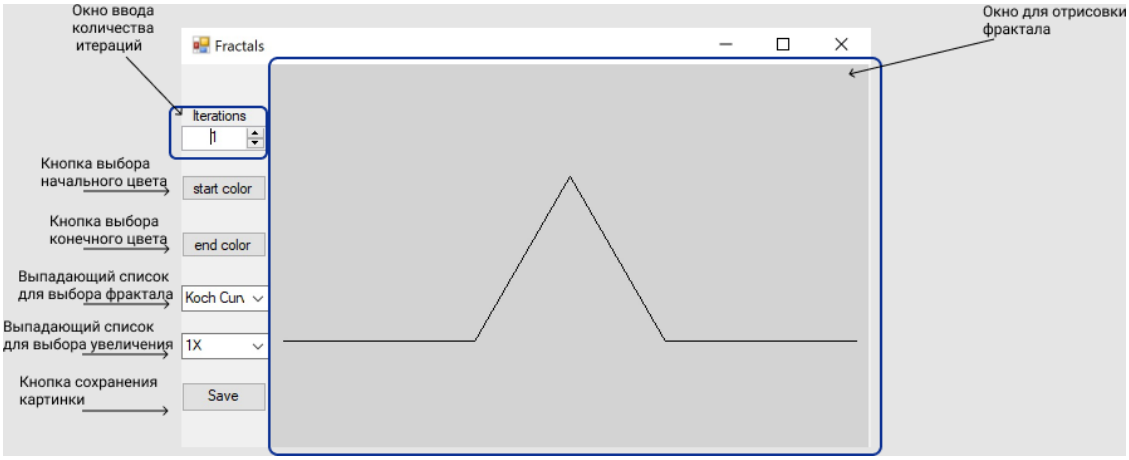
4.7 Program.cs

Автоматически сгенерированный файл, содержит точку входа в программу

5. Описание Интерфейса

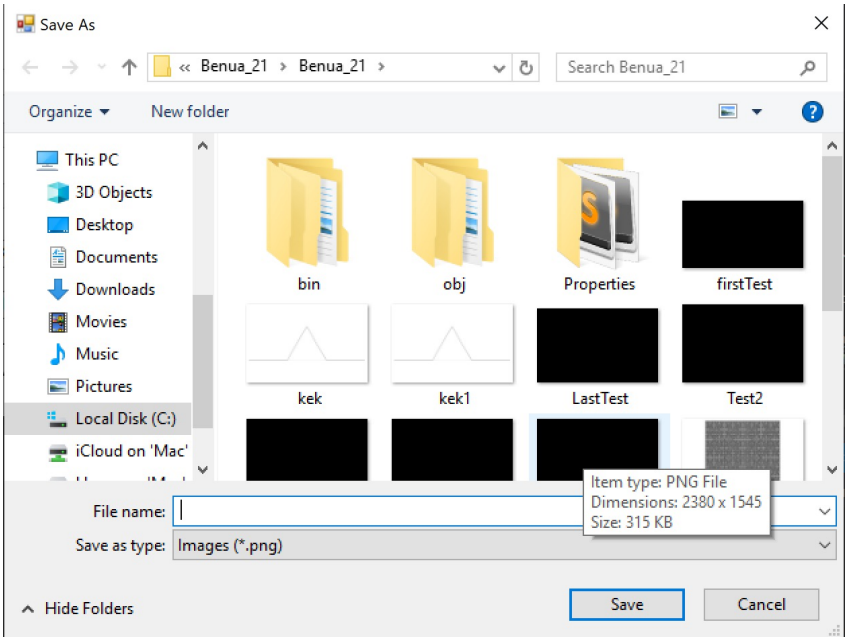
5.1 Главное окно программы

Рис. 2: Главное окно программы



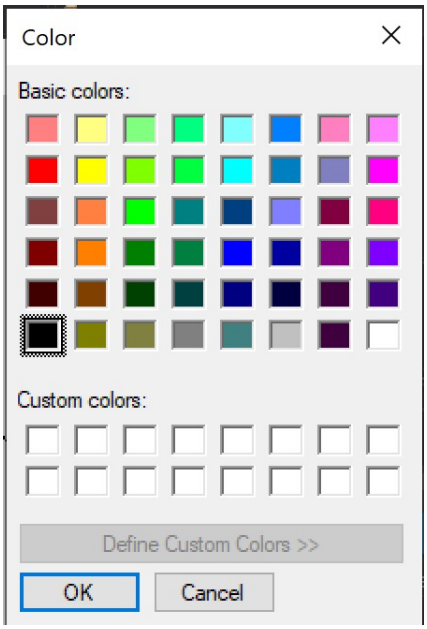
5.2 Диалог сохранения файла

Рис. 3: Диалог сохранения файла



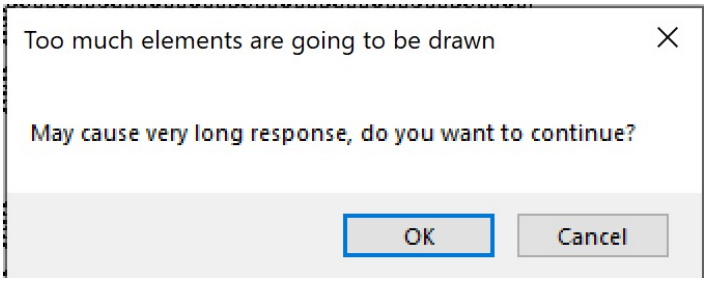
5.3 Диалог выбора цветов

Рис. 4: Диалог выбора цвета



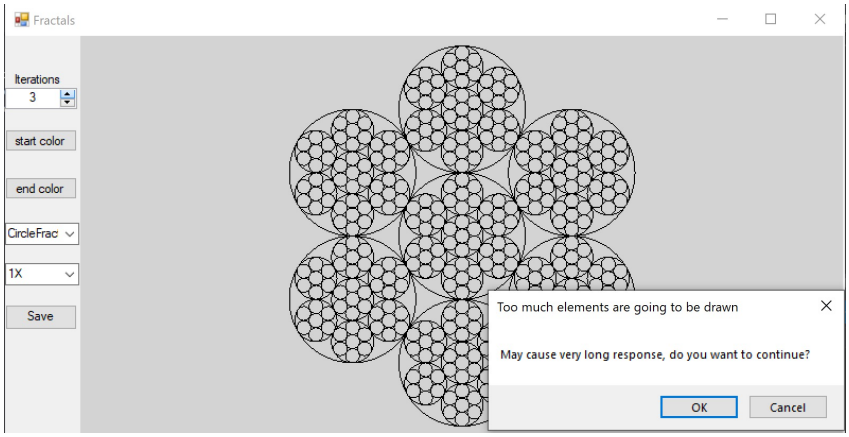
5.4 Предупреждение о возможных проблемах с производительностью

Рис. 5: Предупреждение



6. Контрольный пример

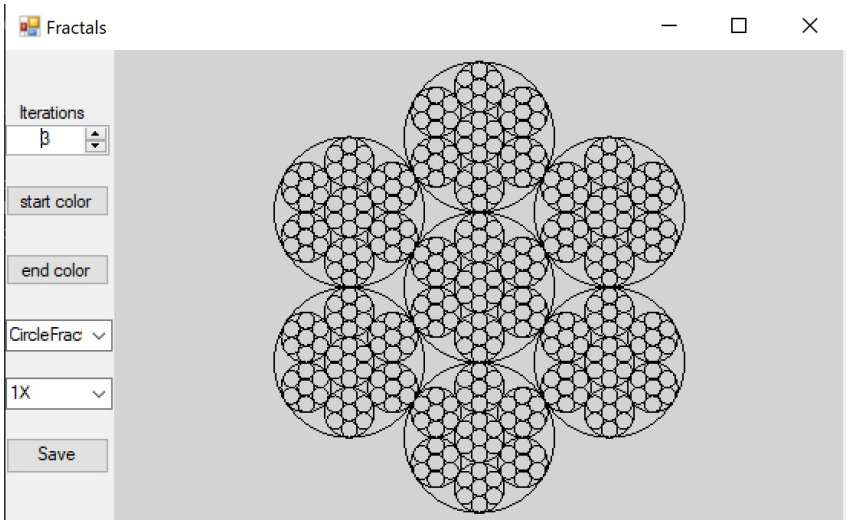
Рис. 6: Контрольный пример : предупреждение



На изображении показана работа программы. Пользователь выбрал Круговой фрактал для отрисовки, отмасштабировал окно, фрактал перерисовался, потом пользователь поставил количество шагов рекурсии равным 3, фрактал опять же перерисовался, теперь пытается поставить количество шагов рекурсии равным 4 и получает предупреждение о возможных проблемах с производительностью.

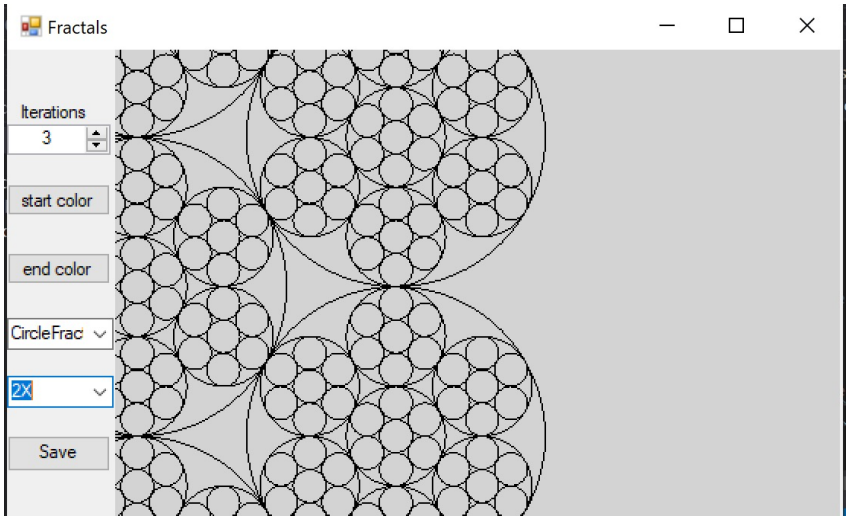
Бенуа. Вариант 21

Рис. 7: Контрольный пример : отмена действия



Пользователь нажал Cancel, то есть отменил свое действие, количество итераций осталось тем же.

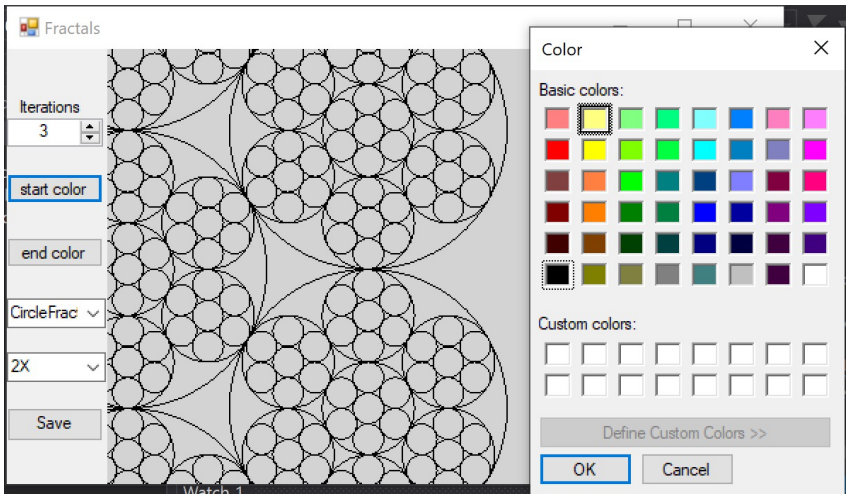
Рис. 8: Контрольный пример : увеличение и перемещение



После пользователь переместил фрактал у выбрал двукратное увеличение.

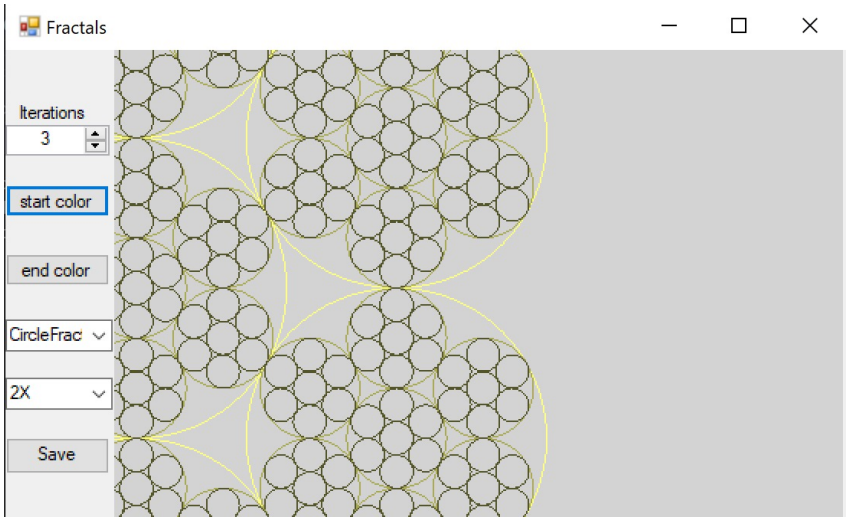
Бенуа. Вариант 21

Рис. 9: Контрольный пример : диалог выбора цвета



Пользователь нажал на кнопку изменения начального цвета, и ему открылся диалог выбора цвета.

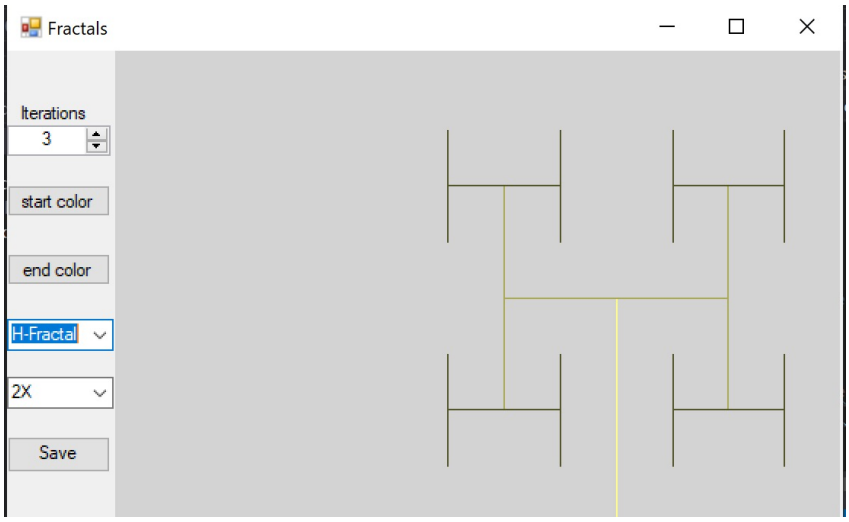
Рис. 10: Контрольный пример : линейный градиент



Пользователь выбрал желтый цвет, нажал на кнопку OK в диалоге выбора цвета. Фрактал перерисовался, используя линейный градиент для каждой итерации.

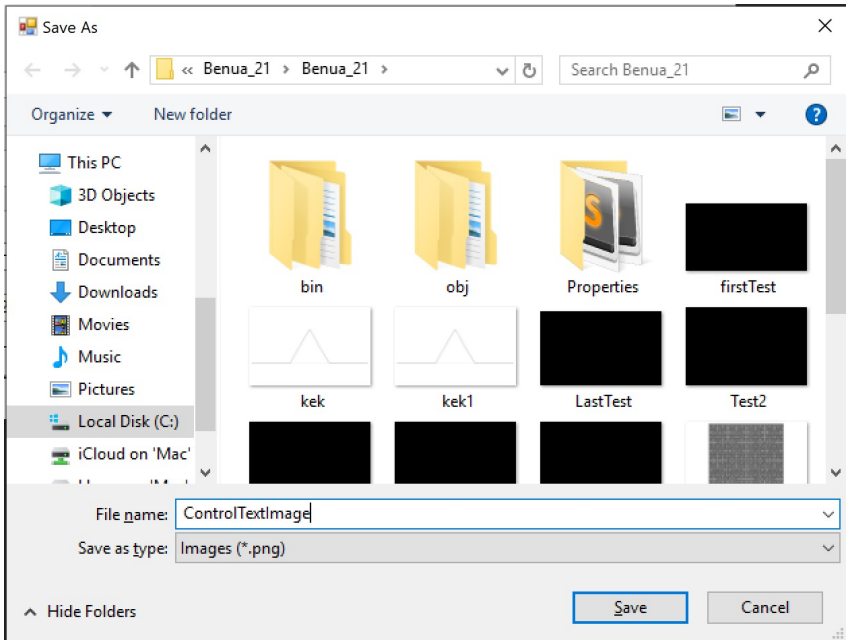
Бенуа. Вариант 21

Рис. 11: Контрольный пример : изменение фрактала



Пользователь поменял фрактал на H-Фрактал, этот фрактал нарисовался с той же цветовой гаммой. Пользователь наблюдает левый верхний угол фрактала.

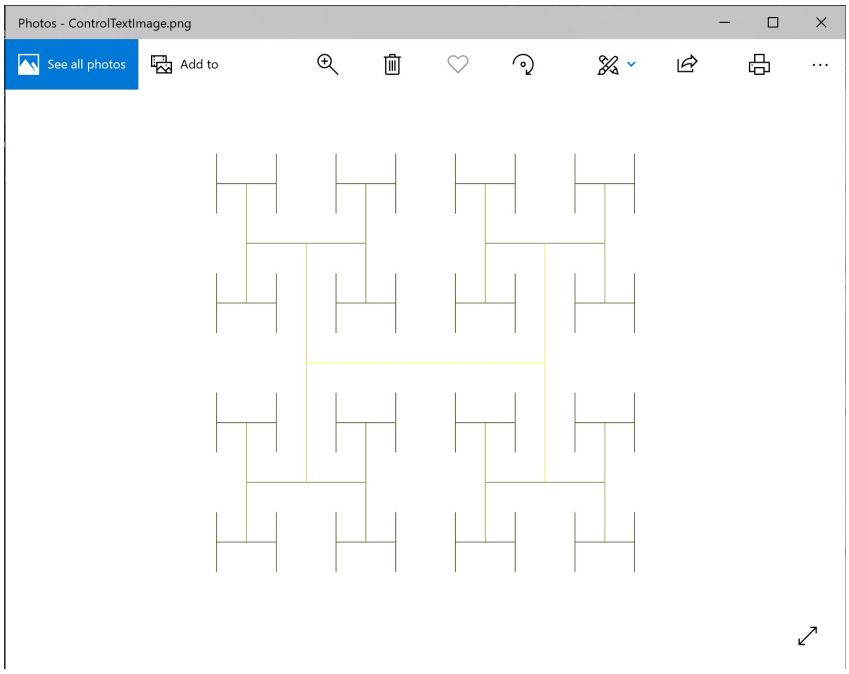
Рис. 12: Контрольный пример: сохранение файла



В открывшемся диалоге сохранения файла пользователь сохраняет файл под именем ControlTextImage.png

Бенуа. Вариант 21

Рис. 13: Контрольный пример: просмотр картинки

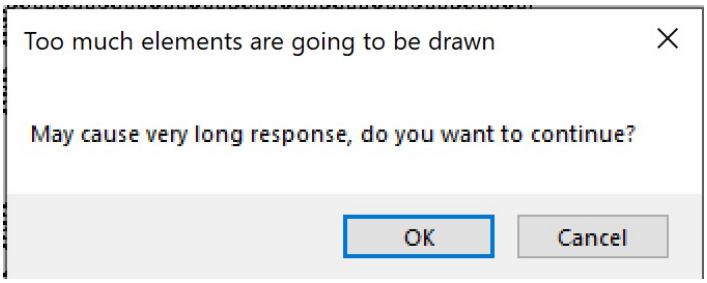


Потом пользователь нашел сохраненную картинку и открыл её.

7. Сообщения пользователю

7.1 Предупреждение о возможных проблемах с производительностью

Рис. 14: Предупреждение



Возникает, когда при изменении количества итераций слишком большое число элементов должно будет отрисоваться.

8. Текст (код) программы

8.1 Form1.cs

```
 using  System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Encoder = System.Drawing.Imaging.Encoder;

namespace Benua_21
{
    public partial class Form1 : Form
    {
        /// <summary>
        /// Previous numericUpDown element value
        /// </summary>
        private int prevNumericUpDownValue = 1;
        /// <summary>
        /// Arrays of Zooms factors
        /// </summary>
        private int[] selectedIndexToZoom = {1, 2, 3, 5};
        /// <summary>
        /// ComplexityPowers of fractal
        /// </summary>
        private int[] complexityFactor = {3, 4, 7};
        /// <summary>
        /// Image for saving fractal on
        /// </summary>
        private Bitmap lastImage;

        private double mEndX, mEndY;

        /// <summary>
        /// move on x-axis
        /// </summary>
        private double moveDx;
        /// <summary>
        /// move on y-axis
        /// </summary>
        private double moveDy;
        private double mStartX, mStartY;

        /// <summary>
        /// Height diff between picturebox and form
        /// </summary>
        private double heightDiff = 0;
        /// <summary>
        /// width diff between picturebox and form
        /// </summary>
```

```
53          private double widthDiff = 0;
54          /// <summary>
55          /// Start color for linear gradient
56          /// </summary>
57          private Color startColor = System.Drawing.Color.Black;
58          /// <summary>
59          /// end color for linear gradient
60          /// </summary>
61          private Color endColor = System.Drawing.Color.Black;
62          /// <summary>
63          /// User started to drag
64          /// </summary>
65          private bool isMoving = false;
66
67          public Form1()
68          {
69              InitializeComponent();
70              //this.Resize += OnResize;
71              //this.OnResizeEnd += OnResize;
72              //picCanvas.SizeMode = PictureBoxSizeMode.StretchImage;
73              //picCanvas.SizeMode = PictureBoxSizeMode.AutoSize;
74              //this.MinimumSize = new Size(Screen.PrimaryScreen.Bounds.Width / 2, saveImageButton.Size.
                    Height + saveImageButton.Location.Y + 40);
75              this.selectZoomComboBox.Items.AddRange(new string[] {
76                  "1X",
77                  "2X",
78                  "3X",
79                  "5X"});
80              picCanvas.BackColor = System.Drawing.Color.LightGray;
81              this.ResizeEnd += OnResizeEnd;
82              this.Resize += OnResize;
83              this.Load += OnLoad;
84              fractalComboBox.SelectedIndex = 0;
85              fractalComboBox.SelectedIndexChanged += FractalComboBoxOnSelectedIndexChanged;
86              selectZoomComboBox.SelectedIndex = 0;
87              selectZoomComboBox.SelectedIndexChanged += SelectZoomComboBoxOnSelectedIndexChanged;
88              picCanvas.MouseUp += PicCanvasOnMouseUp;
89              picCanvas.MouseDown += PicCanvasOnMouseDown;
90              saveImageButton.Click += SaveImageButtonOnClick;
91              iterationsUpDown.Maximum = 8;
92          }
93
94          /// <summary>
95          /// Handles tap on maximize button
96          /// </summary>
97          /// <param name="sender"></param>
98          /// <param name="e"></param>
99          private void OnResize(object sender, EventArgs e)
100         {
101             if (this.WindowState == FormWindowState.Maximized)
102             {
103                 this.OnResizeEnd(EventArgs.Empty);
104             }
105         }
106
107         /// <summary>
```

```
108         /// Saves Image of selected fractal
109         /// </summary>
110         /// <param name="sender"></param>
111         /// <param name="e"></param>
112         private void SaveImageButtonOnClick(object sender, EventArgs e)
113         {
114             SaveFileDialog dialog = new SaveFileDialog();
115 
116             dialog.Filter = "Images (*.png)|*.png";
117             if (dialog.ShowDialog() == DialogResult.OK)
118             {
119                 string filename = dialog.FileName;
120                 {
121                     double dx = moveDx;
122                     double dy = moveDy;
123                     moveDx = 0;
124                     moveDy = 0;
125                     Fractal.imageQualityFactor = 10;
126                     Fractal.eraser.Color = Color.White;
127                     Fractal.startThickness = 4f;
128                     DrawSelectedFractal(fractalComboBox.SelectedIndex, false);
129                     moveDx = dx;
130                     moveDy = dy;
131                 }
132                 Fractal.startThickness = 1f;
133 
134                 Fractal.eraser.Color = Color.LightGray;
135                 Fractal.imageQualityFactor = selectedIndexToZoom[selectZoomComboBox.SelectedIndex];
136                 lastImage.Save(filename, ImageFormat.Png);
137             }
138         }
139         /// <summary>
140         /// Handling dragging: Begin of Drag
141         /// </summary>
142         /// <param name="sender"></param>
143         /// <param name="e"></param>
144         private void PicCanvasOnMouseDown(object sender, MouseEventArgs e)
145         {
146             mStartX = e.X;
147             mStartY = e.Y;
148             Console.WriteLine("start dragging");
149             isMoving = true;
150         }
151         /// <summary>
152         /// Handling Dragging : End of Drag
153         /// </summary>
154         /// <param name="sender"></param>
155         /// <param name="e"></param>
156         private void PicCanvasOnMouseUp(object sender, MouseEventArgs e)
157         {
158             if (!isMoving)
159             {
160                 return;
161             }
162             int zoom = selectedIndexToZoom[selectZoomComboBox.SelectedIndex];
163 
```

```
164             mEndX = e.X;
165             mEndY = e.Y;
166             //moving in diffrent zooms has diffrent impact on offset point
167             moveDx += (mStartX - mEndX) / zoom;
168             moveDy += (mStartY - mEndY) / zoom;
169             Console.WriteLine("MOVING␣␣" + moveDx + "␣" + moveDy);
170             DrawSelectedFractal(fractalComboBox.SelectedIndex);
171             isMoving = false;
172         }
173         /// <summary>
174         /// Sets diffrent maximum value for numericUpDown element
175         /// </summary>
176         private void ValidateDepth()
177         {
178             if (fractalComboBox.SelectedIndex == 2)
179             {
180                 iterationsUpDown.Maximum = 6;
181                 iterationsUpDown.Value = Math.Min(iterationsUpDown.Value, iterationsUpDown.Maximum);
182             }
183             else if (fractalComboBox.SelectedIndex == 1)
184             {
185                 iterationsUpDown.Maximum = 9;
186             }
187             else
188             {
189                 iterationsUpDown.Maximum = 8;
190                 iterationsUpDown.Value = Math.Min(iterationsUpDown.Value, iterationsUpDown.Maximum);
191             }
192         }
193         /// <summary>
194         /// Handling change of zoom
195         /// </summary>
196         /// <param name="sender"></param>
197         /// <param name="e"></param>
198         private void SelectZoomComboBoxOnSelectedIndexChanged(object sender, EventArgs e)
199         {
200             float prevZoom = Fractal.imageQualityFactor;
201             int zoom = selectedIndexToZoom[selectZoomComboBox.SelectedIndex];
202             Fractal.imageQualityFactor = zoom;
203             Point centerPoint = new Point(moveDx + picCanvas.Width / prevZoom / 2, moveDy + picCanvas.
                    Height / 2 / prevZoom);
204             //to zoom in center
205             Point newOrigin = centerPoint - new Point(picCanvas.Width / zoom / 2, picCanvas.Height / zoom
                    / 2);
206             // moveDx = (picCanvas.Width) / 4 * (zoom - 1) ;
207             //moveDy = (picCanvas.Height) / 4 * (zoom - 1) ;
208             moveDx = newOrigin.X;
209             moveDy = newOrigin.Y;
210 
211             DrawSelectedFractal(fractalComboBox.SelectedIndex);
212 
213             //moveDy = moveDx = 0;
214         }
215         /// <summary>
216         /// handling change of Fractal
217         /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void FractalComboBoxOnSelectedIndexChanged(object sender, EventArgs e)
        {
            int selectedState = fractalComboBox.SelectedIndex;
            ValidateDepth();
            moveDx = moveDy = 0;

            DrawSelectedFractal(selectedState);
        }

        private void OnLoad(object sender, EventArgs e)
        {
            SetDifferenceInWidthAndHeight();
            this.OnResizeEnd(EventArgs.Empty);

            DrawSelectedFractal(0);
        }

        /// <summary>
        /// Draws fractal depending on selected index in combobox
        /// </summary>
        /// <param name="index">selected index in combobox</param>
        /// <param name="flag">refresh pictureBox image?</param>
        private void DrawSelectedFractal(int index, bool flag = true)
        {
            Fractal.offsetPoint = new Point(moveDx, moveDy);
            if (index == 0)
            {
                DrowKochCurve(flag);
            }
            else if (index == 1)
            {
                DrawHFractal(flag);
            }
            else
            {
                DrawCircleFractal(flag);
            }
        }
        /// <summary>
        /// handles end of form's resize
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void OnResizeEnd(object sender, EventArgs e)
        {
            Console.WriteLine(widthDiff);
            picCanvas.Width = this.Width - (int)widthDiff;
            picCanvas.Height = this.Height - (int)heightDiff;
            //DrowKochCurve();
            DrawSelectedFractal(fractalComboBox.SelectedIndex);
        }
        /// <summary>
        /// calculates Difference in width and height between from and pictureBox
        /// </summary>
```

```
274         private void SetDifferenceInWidthAndHeight()
275         {
276             heightDiff = this.Height - picCanvas.Height;
277             widthDiff = this.Width - picCanvas.Width;
278         }
279         /// <summary>
280         /// Draws Koch Curve
281         /// </summary>
282         /// <param name="flag">refresh image in imageBox</param>
283         private void DrowKochCurve(bool flag = true)
284         {
285             Console.WriteLine("Redraw");
286             Console.WriteLine("Size is ");
287             Console.WriteLine(picCanvas.Width + " " + picCanvas.Height);
288             Console.WriteLine(this.Width.ToString() + " " + this.Height.ToString());
289             //mBm = new Bitmap(this.ClientSize.Width, this.ClientSize.Height);
290             double width = this.picCanvas.Width;
291             double height = this.picCanvas.Height;
292             double len = Math.Min((height-20) * 2 / Math.Sqrt(3), (width - 20) / 3);
293             Console.WriteLine($"Len is {len}");
294             lastImage = new Bitmap((int)(width * Fractal.imageQualityFactor), (int)(height * Fractal.
                    imageQualityFactor));
295             Point startPoint = new Point((width - 3 * len) / 2, height / 2 + len / 4 * Math.Sqrt(3));
296             Point endPoint = new Point(width - (width - 3 *len) / 2, startPoint.Y);
297             Console.WriteLine(startPoint);
298             Console.WriteLine(endPoint);
299             KochCurve curve = new KochCurve(0, startColor, endColor, (int)iterationsUpDown.Value);
300             curve.Draw(lastImage, startPoint, endPoint);
301             if (flag)
302             {
303                 (picCanvas.Image)?.Dispose();
304                 picCanvas.Image = lastImage;
305             }
306         }
307         /// <summary>
308         /// Draws H-Fractal
309         /// </summary>
310         /// <param name="flag">refresh image on PictureBox?</param>
311         private void DrawHFractal(bool flag = true)
312         {
313             Console.WriteLine("H-fractal");
314             double width = this.picCanvas.Width;
315             double height = this.picCanvas.Height;
316 
317             double maxLen = (double) Math.Min(picCanvas.Width, picCanvas.Height) / 2.1;
318             Console.WriteLine($"MaxLen is {maxLen}");
319 
320             Point midPoint = new Point(width / 2, height / 2);
321             lastImage = new Bitmap((int)(width * Fractal.imageQualityFactor), (int)(height * Fractal.
                    imageQualityFactor));
322 
323             Console.WriteLine("MidPoint " + midPoint);
324 
325             HFractal fractal = new HFractal(maxLen, startColor, endColor, (int)iterationsUpDown.Value);
326             fractal.Draw(lastImage, midPoint);
327             if (flag)
```

```
328             {
329                 (picCanvas.Image)?.Dispose();
330                 picCanvas.Image = lastImage;
331             }
332         }
333         /// <summary>
334         /// Draws Circle Fractal
335         /// </summary>
336         /// <param name="flag"> refresh image on pictureBox?</param>
337         private void DrawCircleFractal(bool flag = true)
338         {
339             double width = this.picCanvas.Width;
340             double height = this.picCanvas.Height;
341 
342             double maxLen = (double)Math.Min(picCanvas.Width, picCanvas.Height) / 2.1;
343             Point midPoint = new Point(width / 2, height / 2);
344             lastImage = new Bitmap((int)(width * Fractal.imageQualityFactor), (int)(height * Fractal.
                    imageQualityFactor));
345             Console.WriteLine(iterationsUpDown.Value);
346             CircleFractal fractal = new CircleFractal(maxLen, startColor, endColor, (int)iterationsUpDown.
                    Value);
347             fractal.Draw(lastImage, midPoint);
348             if (flag)
349             {
350                 (picCanvas.Image)?.Dispose();
351                 picCanvas.Image = lastImage;
352             }
353         }
354         /// <summary>
355         /// Number of iterations changed
356         /// </summary>
357         /// <param name="sender"></param>
358         /// <param name="e"></param>
359         private void IterationsUpDown_ValueChanged(object sender, EventArgs e)
360         {
361             if (Math.Pow((int) iterationsUpDown.Value, complexityFactor[fractalComboBox.SelectedIndex]) >
                    5000 && prevNumericUpDownValue < (int)iterationsUpDown.Value)
362             {
363                 var result = MessageBox.Show("May␣cause␣very␣long␣response,␣do␣you␣want␣to␣continue?",
364                     "Too␣much␣elements␣are␣going␣to␣be␣drawn", MessageBoxButtons.OKCancel);
365                 if (result == DialogResult.Cancel)
366                 {
367                     iterationsUpDown.Value = prevNumericUpDownValue;
368                     return;
369                 }
370                 prevNumericUpDownValue = Math.Max(prevNumericUpDownValue, (int)iterationsUpDown.Value);
371 
372             }
373             DrawSelectedFractal(fractalComboBox.SelectedIndex);
374             prevNumericUpDownValue = (int) iterationsUpDown.Value;
375         }
376         /// <summary>
377         /// Opens ColorDialog for choosing startColor
378         /// </summary>
379         /// <param name="sender"></param>
380         /// <param name="e"></param>
```

```
        private void StartColorButton_Click(object sender, EventArgs e)
        {
            ColorDialog startColorDialog = new ColorDialog();
            startColorDialog.AllowFullOpen = false;
            if (startColorDialog.ShowDialog() == DialogResult.OK)
            {
                startColor = startColorDialog.Color;
                DrawSelectedFractal(fractalComboBox.SelectedIndex);
                //DrowKochCurve();
            }
        }
        /// <summary>
        /// Opens Colordialog for choosing endColor
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void StartEndColorButton_Click(object sender, EventArgs e)
        {
            ColorDialog endColorDialog = new ColorDialog();
            endColorDialog.AllowFullOpen = false;
            if (endColorDialog.ShowDialog() == DialogResult.OK)
            {
                endColor = endColorDialog.Color;
                if (startColor != null)
                    DrawSelectedFractal(fractalComboBox.SelectedIndex);
            }
        }
    }
}
```

8.2 Form1.cs

```
 using  System;
using System.CodeDom;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Benua_21
{

    public abstract class Fractal
    {
        /// <summary>
        /// Pen for erasing
        /// </summary>
        public static Pen eraser = new Pen(Color.LightGray, (float)3.1);
        /// <summary>
        /// for zoom and reating image with good quality
        /// </summary>
        public static float imageQualityFactor = (float)1;
        /// <summary>
        /// Thickness of Pen on first iteration
```

```
25          /// </summary>
26          public static float startThickness = 1;
27          /// <summary>
28          /// Drawing left high point - start point
29          /// </summary>
30          public static Point offsetPoint = new Point();
31          /// <summary>
32          /// Start Len/radius for elements in Fractal
33          /// </summary>
34          public double StartLen { get;protected set; }
35          /// <summary>
36          /// Color for first iteration
37          /// </summary>
38          public Color StartColor { get; protected set; }
39          /// <summary>
40          /// Color for last iteration
41          /// </summary>
42          public Color EndColor { get; protected set; }
43          /// <summary>
44          /// Maximal depth for fractal
45          /// </summary>
46          public int MaxDepth { get; protected set; }
47          /// <summary>
48          /// Current dept for drawing subfractal
49          /// </summary>
50          public int CurDepth { get; protected set; }
51          /// <summary>
52          /// Thickness decrease by iteration
53          /// </summary>
54          //public static float thicknessChangePerIteration = 0.1f;
55
56          /// <summary>
57          /// empty Contructor
58          /// </summary>
59          protected Fractal()
60          {
61              StartLen = MaxDepth = CurDepth = 0;
62              StartColor = EndColor = Color.Black;
63          }
64
65          /// <summary>
66          /// Full constructor
67          /// </summary>
68          /// <param name="startLen">Len or radius for first iteration</param>
69          /// <param name="startColor">Color for first iteration</param>
70          /// <param name="endColor">Maximal depth for fractal</param>
71          /// <param name="maxDepth">Maximal depth for fractal</param>
72          /// <param name="curDepth">Current dept for drawing subfractal</param>
73          protected Fractal(double startLen, Color startColor, Color endColor, int maxDepth, int curDepth =
                0)
74          {
75              StartLen = startLen;
76              StartColor = startColor;
77              EndColor = endColor;
78              MaxDepth = maxDepth;
79              CurDepth = curDepth;
```

```
        }
        /// <summary>
        /// Abstract function to draw current part of fractal
        /// </summary>
        /// <param name="image">Image to draw on</param>
        /// <param name="pt1">leftmost point</param>
        /// <param name="pt2">rightmost point</param>
        public abstract void Draw(Bitmap image, Point pt1, Point pt2 = null, int helper = 0);

        /// <summary>
        /// Generates Color for given iteration
        /// </summary>
        /// <param name="start">color for first iteration</param>
        /// <param name="end"> color for last iteration</param>
        /// <param name="depth"> current depth of fractal's part</param>
        /// <param name="maxDepth"> maximum depth for fractal</param>
        /// <returns>Gradient color for given iteration</returns>
        public static Color GetGradientColor(Color start, Color end, int depth, int maxDepth)
        {
            int rMin = start.R;
            int rMax = end.R;

            int gMin = start.G;
            int gMax = end.G;

            int bMin = start.B;
            int bMax = end.B;

            int neededR = rMin + (int)((double) (rMax - rMin) * depth / maxDepth);

            int neededG = gMin + (int)((double)(gMax - gMin) * depth / maxDepth);

            int neededB = bMin + (int)((double)(bMax - bMin) * depth / maxDepth);

            return Color.FromArgb(neededR, neededG, neededB);
        }
    }



}
```

8.3 HFractal.cs

```
 using  System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Benua_21
{
```

```
10      /// <summary>
11      /// Class for describing H-Fractal
12      /// </summary>
13      public class HFractal : Fractal
14      {
15          /// <summary>
16          /// Empty constuctor
17          /// </summary>
18          public HFractal()
19          {
20          }
21          /// <summary>
22          /// Full Contructor
23          /// </summary>
24          /// <param name="startLen">len or radius of items on first iteration</param>
25          /// <param name="startColor"> first iteration color</param>
26          /// <param name="endColor">last iteration color</param>
27          /// <param name="maxDepth">Maximum depth for fractal's parts</param>
28          /// <param name="curDepth">current depth of subfractal</param>
29          public HFractal(double startLen, Color startColor, Color endColor, int maxDepth = 10, int curDepth
                = 0) : base(startLen, startColor,
30              endColor, maxDepth, curDepth)
31          {
32          }
33          /// <summary>
34          /// method for drawing H-Fractal on image
35          /// </summary>
36          /// <param name="image">image to draw on</param>
37          /// <param name="A">leftmost Point</param>
38          /// <param name="E">rightmost Point</param>
39          public override void Draw(Bitmap image, Point pt1, Point pt2 = null, int helper = 0)
40          {
41              if (MaxDepth == CurDepth)
42              {
43                  return;
44              }
45              double curLen = StartLen / (Math.Pow(2, CurDepth));
46              /*
47               *
48               *C |   |D
49               * A|____|B
50               *  |   |
51               *E |   |F
52               */
53              //pt1 -= offsetPoint;
54
55              Point A = new Point(pt1.X - curLen / 2, pt1.Y);
56              Point B = new Point(pt1.X + curLen / 2, pt1.Y);
57              Point C = new Point(A.X, A.Y + curLen / 2);
58
59              Point D = new Point(B.X, B.Y + curLen / 2);
60
61              Point E = new Point(A.X, A.Y - curLen / 2);
62
63              Point F = new Point(B.X, B.Y - curLen / 2);
64
```

```

            Pen GradientPen = new Pen(Fractal.GetGradientColor(StartColor, EndColor, CurDepth, MaxDepth),
                (startThickness));
            using (var graphics = Graphics.FromImage(image))
            {
                var offA = (A - offsetPoint) * imageQualityFactor;
                var offB = (B - offsetPoint) * imageQualityFactor;
                var offC = (C - offsetPoint) * imageQualityFactor;
                var offD = (D - offsetPoint) * imageQualityFactor;
                var offF = (F - offsetPoint) * imageQualityFactor;
                var offE = (E - offsetPoint) * imageQualityFactor;


                graphics.DrawLine(GradientPen, (float)offA.X, (float)offA.Y, (float)offE.X, (float)offE.Y);
                graphics.DrawLine(GradientPen, (float)offA.X, (float)offA.Y, (float)offC.X, (float)offC.Y);
                graphics.DrawLine(GradientPen, (float)offA.X, (float)offA.Y, (float)offB.X, (float)offB.Y);
                graphics.DrawLine(GradientPen, (float)offB.X, (float)offB.Y, (float)offF.X, (float)offF.Y);
                graphics.DrawLine(GradientPen, (float)offB.X, (float)offB.Y, (float)offD.X, (float)offD.Y);
            }

            Point[] arr = { C, E, D, F };
            for (int i = 0; i < arr.Length; ++i)
            {
                HFractal nextStepFractal = new HFractal(StartLen, StartColor, EndColor, MaxDepth, CurDepth
                    + 1);
                nextStepFractal.Draw(image, arr[i]);
            }

        }

    }
}
```

8.4 KochCurve.cs

```
 using  System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Benua_21
{
    /// <summary>
    /// Class for describing H-Fractal
    /// </summary>
    public class HFractal : Fractal
    {
        /// <summary>
        /// Empty constuctor
        /// </summary>
        public HFractal()
        {
        }
        /// <summary>
```

```
22          /// Full Contructor
23          /// </summary>
24          /// <param name="startLen">len or radius of items on first iteration</param>
25          /// <param name="startColor"> first iteration color</param>
26          /// <param name="endColor">last iteration color</param>
27          /// <param name="maxDepth">Maximum depth for fractal's parts</param>
28          /// <param name="curDepth">current depth of subfractal</param>
29          public HFractal(double startLen, Color startColor, Color endColor, int maxDepth = 10, int curDepth
                = 0) : base(startLen, startColor,
30              endColor, maxDepth, curDepth)
31          {
32          }
33          /// <summary>
34          /// method for drawing H-Fractal on image
35          /// </summary>
36          /// <param name="image">image to draw on</param>
37          /// <param name="A">leftmost Point</param>
38          /// <param name="E">rightmost Point</param>
39          public override void Draw(Bitmap image, Point pt1, Point pt2 = null, int helper = 0)
40          {
41              if (MaxDepth == CurDepth)
42              {
43                  return;
44              }
45              double curLen = StartLen / (Math.Pow(2, CurDepth));
46              /*
47               *
48               *C |   |D
49               * A|____|B
50               *  |    |
51               *E |   |F
52               */
53              //pt1 -= offsetPoint;
54
55              Point A = new Point(pt1.X - curLen / 2, pt1.Y);
56              Point B = new Point(pt1.X + curLen / 2, pt1.Y);
57              Point C = new Point(A.X, A.Y + curLen / 2);
58
59              Point D = new Point(B.X, B.Y + curLen / 2);
60
61              Point E = new Point(A.X, A.Y - curLen / 2);
62
63              Point F = new Point(B.X, B.Y - curLen / 2);
64
65
66              Pen GradientPen = new Pen(Fractal.GetGradientColor(StartColor, EndColor, CurDepth, MaxDepth),
                    (startThickness));
67              using (var graphics = Graphics.FromImage(image))
68              {
69                  var offA = (A - offsetPoint) * imageQualityFactor;
70                  var offB = (B - offsetPoint) * imageQualityFactor;
71                  var offC = (C - offsetPoint) * imageQualityFactor;
72                  var offD = (D - offsetPoint) * imageQualityFactor;
73                  var offF = (F - offsetPoint) * imageQualityFactor;
74                  var offE = (E - offsetPoint) * imageQualityFactor;
75
```

```

            graphics.DrawLine(GradientPen, (float)offA.X, (float)offA.Y, (float)offE.X, (float)offE.Y);
            graphics.DrawLine(GradientPen, (float)offA.X, (float)offA.Y, (float)offC.X, (float)offC.Y);
            graphics.DrawLine(GradientPen, (float)offA.X, (float)offA.Y, (float)offB.X, (float)offB.Y);
            graphics.DrawLine(GradientPen, (float)offB.X, (float)offB.Y, (float)offF.X, (float)offF.Y);
            graphics.DrawLine(GradientPen, (float)offB.X, (float)offB.Y, (float)offD.X, (float)offD.Y);
        }

        Point[] arr = { C, E, D, F };
        for (int i = 0; i < arr.Length; ++i)
        {
            HFractal nextStepFractal = new HFractal(StartLen, StartColor, EndColor, MaxDepth, CurDepth
                + 1);
            nextStepFractal.Draw(image, arr[i]);
        }

    }

}
```

8.5 Point.cs

```
 using  System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Benua_21
{
    public class Point
    {
        /// <summary>
        /// X axis coord
        /// </summary>
        public double X { get; private set; }

        /// <summary>
        /// Y axis coord
        /// </summary>
        public double Y { get; private set; }

        /// <summary>
        /// Empty Constructor
        /// </summary>
        public Point() { }

        /// <summary>
        /// Constructor with two coordinates
        /// </summary>
        /// <param name="x"> X axis coordinate</param>
        /// <param name="y"> Y axis coordinate</param>
        public Point(double x, double y)
        {
            X = x;
```

```
34              Y = y;
35          }
36          /// <summary>
37          /// Overridden multiplication for Point and scalar
38          /// </summary>
39          /// <param name="a">Point to be multiplied</param>
40          /// <param name="b">scalar to be multiplied on</param>
41          /// <returns></returns>
42          public static Point operator *(Point a, double b)
43          {
44              return new Point(a.X * b, a.Y * b);
45          }
46          /// <summary>
47          /// Overriden division for Point and scalar
48          /// </summary>
49          /// <param name="a">Point to be divide </param>
50          /// <param name="b">scalar to be divide on</param>
51          /// <returns></returns>
52          public static Point operator /(Point a, double b)
53          {
54              return new Point(a.X / b, a.Y / b);
55          }
56          /// <summary>
57          /// Overridden sum for two Points
58          /// </summary>
59          /// <param name="a">first Point</param>
60          /// <param name="b">second Point</param>
61          /// <returns></returns>
62          public static Point operator +(Point a, Point b)
63          {
64              return new Point(a.X + b.X, a.Y + b.Y);
65          }
66          /// <summary>
67          /// Overriden subtraction for two Points
68          /// </summary>
69          /// <param name="a">first Point</param>
70          /// <param name="b"> second Point</param>
71          /// <returns></returns>
72          public static Point operator -(Point a, Point b)
73          {
74              return new Point(a.X - b.X, a.Y - b.Y);
75          }
76          /// <summary>
77          /// Rotates Vector, represented by Point by angle
78          /// </summary>
79          /// <param name="a">Vector to be rotated</param>
80          /// <param name="angle"> angle to be rotated On</param>
81          /// <returns></returns>
82          public static Point Rotate(Point a, double angle)
83          {
84              return new Point(a.X * Math.Cos(angle) - a.Y * Math.Sin(angle),
85                  a.X * Math.Sin(angle) + a.Y * Math.Cos(angle));
86          }
87          /// <summary>
88          /// Overriden TOtring method
89          /// </summary>
```

```
90          /// <returns></returns>
91          public override string ToString()
92          {
93              return $"({X}, {Y})";
94          }
95      }
96  }
```

9. Список литературы

1. Документация C Sharp от Microsoft.- Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/
2. Windows Forms Programming in C#, Chris Sells, Addison-Wesley Professional, 2004. 680 стр.
3. The Object-oriented Thought Process, Matt A. Weisfeld, 2000. 311 стр.