**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по учебной практике**

**(****технологической (проектно-технологической) практике)**

**Тема: Создание приложений в Windows Forms**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9302 |  | Гелета А.И. |
| Руководитель |  | Калмычков В.А. |

Санкт-Петербург

2021

**ЗАДАНИЕ**

**на «Учебную (технологическую (проектно-технологическую))» практику**

|  |  |  |
| --- | --- | --- |
| Студентка Гелета А.И. | | |
| Группа 9302 | | |
| Тема практики: «Создание приложений в Windows Forms» | | |
| Задание на практику:  Ознакомится с методами работы с Windows Forms, написать две программы с пользовательским интерфейсом. Первая должна имитировать движение объекта по траектории, вторая – рисовать фракталы. | | |
| Сроки прохождения практики: 01.07.2021 – 14.07.2021 | | |
| Дата сдачи отчета: 12.07.2021 | | |
| Дата защиты отчета: 13.07.2021 | | |
|  | | |
| Студентка |  | Гелета А.И. |
| Руководитель |  | Калмычков В.А. |

**Аннотация**

Цель практики - изучение и освоение базовых понятий, методов и приемов использования инструментальных средств и технологий программирования при решении практических задач с выбором различных структур данных и организацией программного графического интерфейса пользователя.

Во вводной части практики было выполнено 7 пробных заданий. В первой части практики была разработана программа, позволяющая с разными настройками запускать движение параллелограмма по траектории котангенса. Во второй части практики программа рисует фрактал (множество Кантора) заданного уровня.

**Summary**

Purpose of practice is to learn basic concepts, methods and techniques of using programming tools and technologies to solve practical tasks with a choice of different data structures and organization of graphic users’ interface.

In the introductory part of practice the 7 test tasks were completed. In the first part of practice the program, that allow to imitate moving a parallelogram in trajectory of cotangent with various settings, was developed. In the second part program is drawing a fractal (Cantor set) of given level.

**содержание**

[Введение 5](#_Toc76982869)

[Введение в Windows Forms 6](#_Toc76982870)

[Задание № 1. Элементы button, textbox и label 6](#_Toc76982871)

[Задание № 2. Элемент MessageBox, Подсказка ToolTip 7](#_Toc76982872)

[Задание № 3. Изменение шрифта текста и цвета формы и элементов 8](#_Toc76982873)

[Задание № 4. Элемент MenuStrip и свойство Anchor, Открытие и запись текстового файла 9](#_Toc76982874)

[Задание № 5. Рисование линий, треугольника, эллипса и окружности в PictureBox 11](#_Toc76982875)

[Задание № 6. Событие MauseHover 13](#_Toc76982876)

[Формирование траектории для движения простого объекта 13](#_Toc76982877)

[Индивидуальное задание 1 15](#_Toc76982878)

[Формулировка задания 15](#_Toc76982879)

[Математическая постановка 15](#_Toc76982880)

[Описание пользовательского интерфейса 16](#_Toc76982881)

[Описание графических примитивов 19](#_Toc76982882)

[Пример работы программы 19](#_Toc76982883)

[Текст программы 22](#_Toc76982884)

[Индивидуальное задание 2 27](#_Toc76982885)

[Формулировка задания 27](#_Toc76982886)

[Математическая постановка 27](#_Toc76982887)

[Описание пользовательского интерфейса 28](#_Toc76982888)

[Описание графических примитивов 29](#_Toc76982889)

[Пример работы программы 29](#_Toc76982890)

[Текст программы 32](#_Toc76982891)

# Введение

Цели практики: изучение и освоение базовых понятий, методов и приемов использования инструментальных средств и технологий программирования при решении практических задач с выбором различных структур данных и организацией программного графического интерфейса пользователя, закрепление и приобретение новых знаний и практических навыков программирования.

Задачи практики: формирование базовых практических понятий, лежащих в основе процесса разработки программного графического интерфейса пользователя, получение навыков применения средств визуализации при решении практических задач и использовании различных структур данных, освоение способов реализации программ на языке программирования C# с учётом особенностей реализации в конкретной системе программирования.

# Введение в Windows Forms

## Задание № 1. Элементы button, textbox и label

Эти элементы – один из самых используемых в Windows Forms. В настройках кнопки (рисунок 1) можно изменить текст. Так, вместо стандартного button1 можно написать «Копировать»:

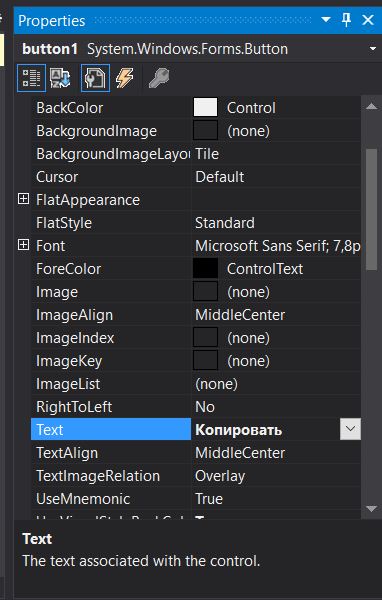


Рисунок 1

С тремя элементами при запуске на форме видно только пустую строчку для ввода и кнопку копировать, на label текста никакого нет (рисунок 2)

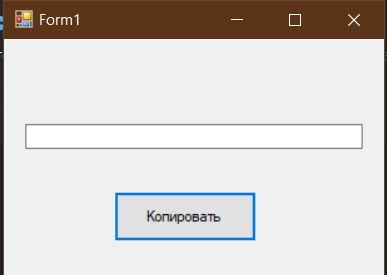


Рисунок 2

После написания какого-либо текста в поле и нажатия кнопки весь текст отображается на элементе label, который расположен выше (рисунок 3):

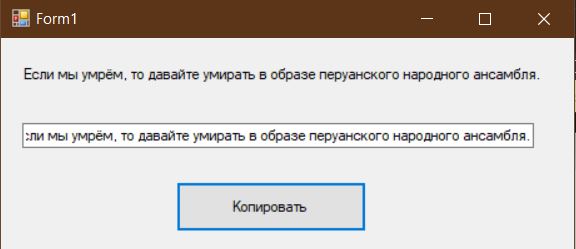


Рисунок 3

## Задание № 2. Элемент MessageBox, Подсказка ToolTip

На рисунке 4 представлена работа элемента ToolTip: при наведении на поле появляется подсказка в виде всплывающего окна.

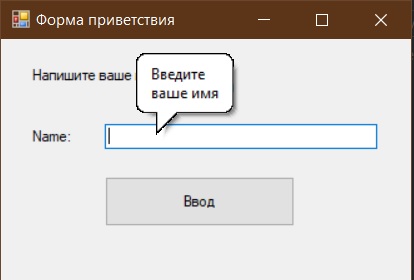


Рисунок 4

На рисунке 5 представлена работа всплывающего окна Message Box. Так, на форму был добавлен TextBox. При введении имени (текста) в поле и нажатии кнопки «Ввод» появляется окно «Приветствие», в котором пользователя приветствуют по написанному им в поле имени.

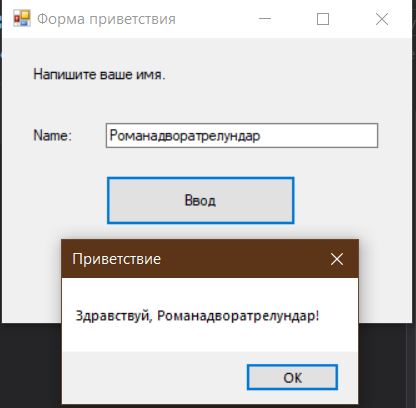


Рисунок 5

## Задание № 3. Изменение шрифта текста и цвета формы и элементов

В настройках (рисунок 6) можно изменить шрифт, размер и цвет текста, как для простого текста, так и для кнопки.

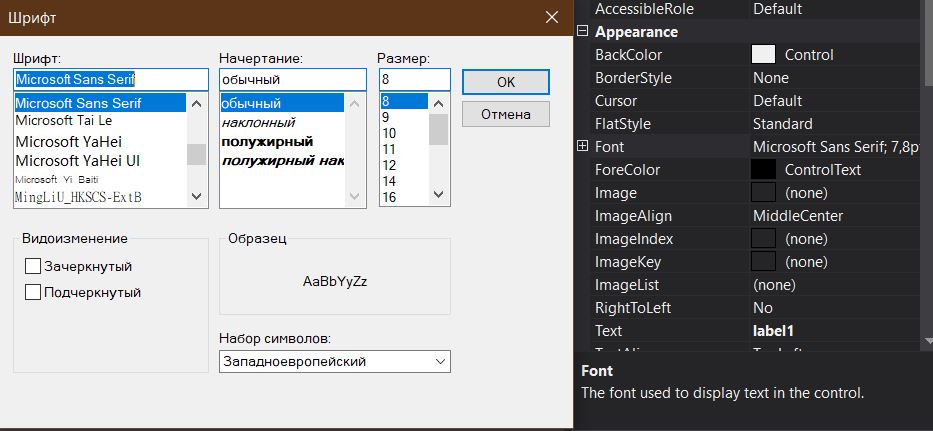


Рисунок 6

Также можно изменять фон формы. Изменения шрифта и фона показаны на рисунке 7:

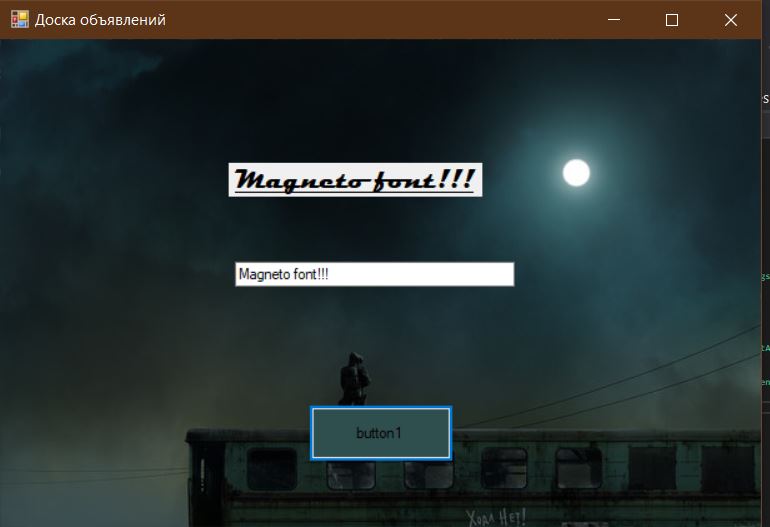


Рисунок 7

Также с помощью комбинации ComboBox (выбор из коллекции элементов) и PictureBox (отображение картинки) можно дать пользователю возможность выбрать картинку, которая будет отображаться на экране (рисунок 8):

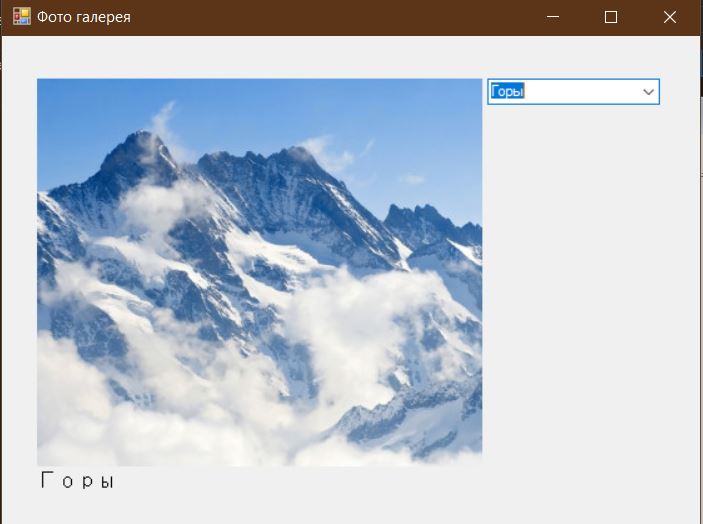


Рисунок 8

PictureBox также может использоваться для рисования текста любого размера, шрифта и цвета, в любом месте (рисунок 9):

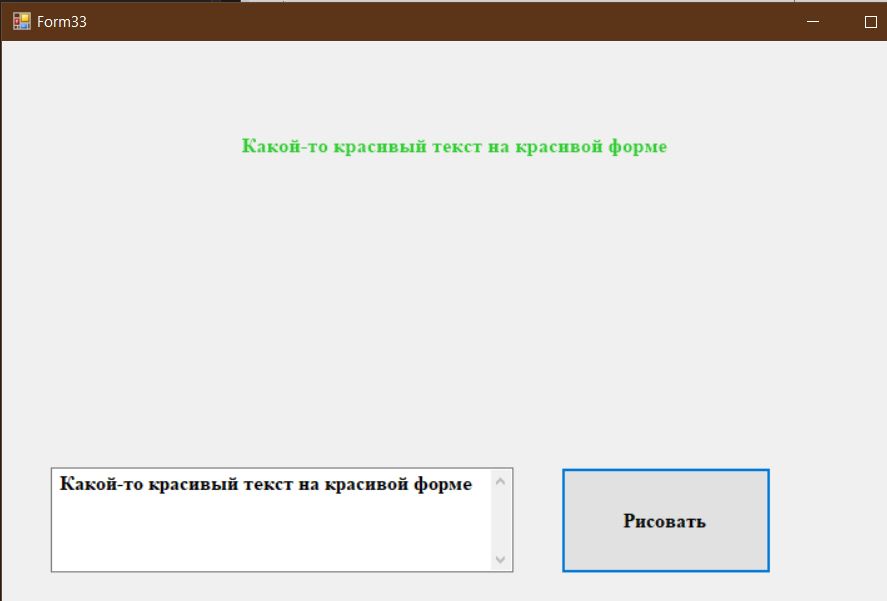


Рисунок 9

## Задание № 4. Элемент MenuStrip и свойство Anchor, Открытие и запись текстового файла

Элемент MenuStrip (рисунок 10) представляет собой привычное открывающееся меню, где можно выбрать какое-либо действие (в данном задании – открыть файл, сохранить как и выход).

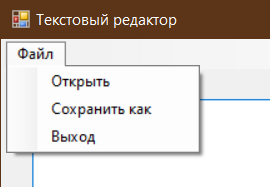


Рисунок 10

При выборе «Открыть» открывается файл, если он есть, и копируется все содержимое в текстовое поле (рисунок 11), если нет, то выдается соответствующее предупреждение (рисунок 12). При выборе «Сохранить как» изменения, внесенные в текстовом окне, сохраняются в файл.

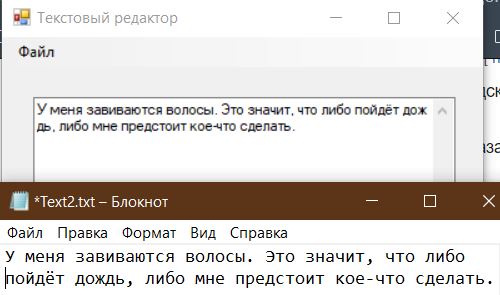


Рисунок 11

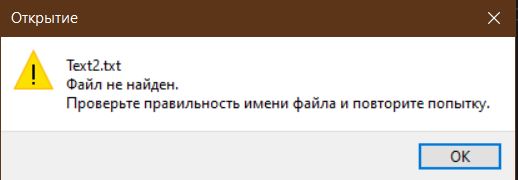


Рисунок 12

При выходе (через элемент меню «Выход» или с помощью красного крестика в правом углу) выводится всплывающее окно, которое уточняет, хочет ли пользователь закрыть редактор, хотя там есть несохраненные изменения:

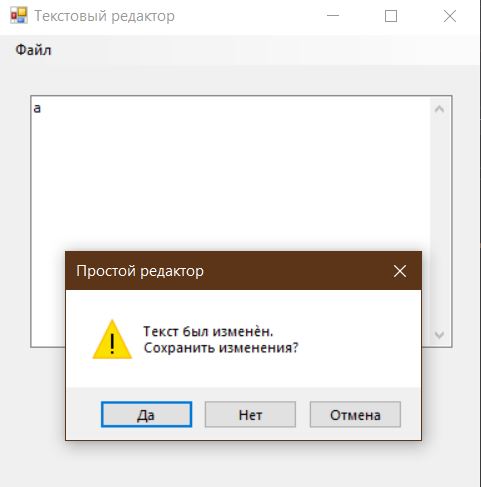


Рисунок 13

## Задание № 5. Рисование линий, треугольника, эллипса и окружности в PictureBox

PictureBox с помощью специальных методов позволяет рисовать. Примеры простых рисунков показаны на рисунках 14, 15, 16 (линия, нарисованная по двум точкам с координатами, треугольник – три линии, нарисованные по трем точкам с координатами и эллипс, нарисованной по одной точке с координатами и двумя радиусами)

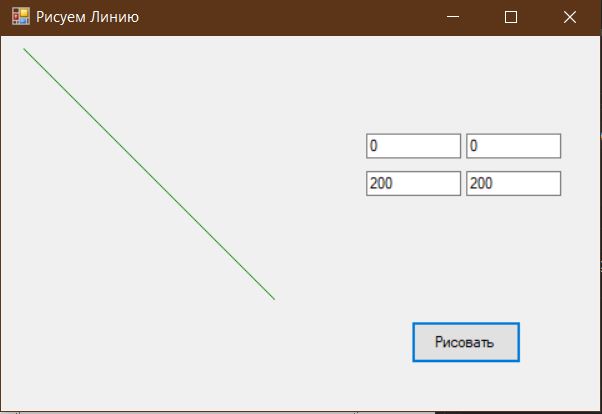


Рисунок 14

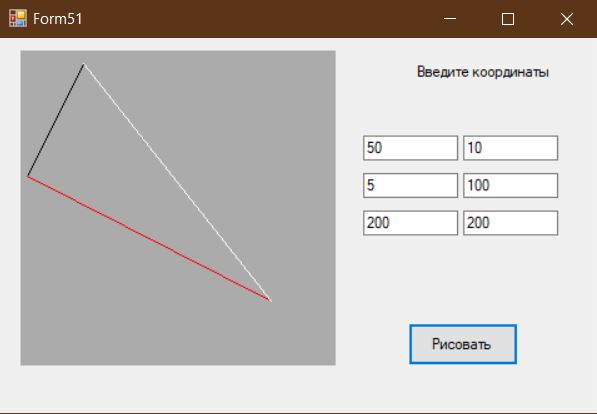


Рисунок 15

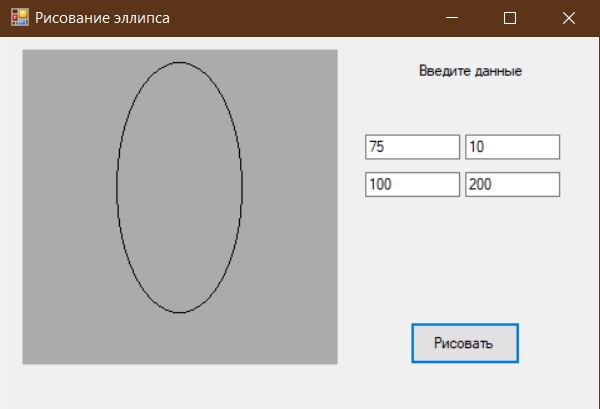


Рисунок 16

Можно также использовать заливку фигуры, как показано на рисунке:

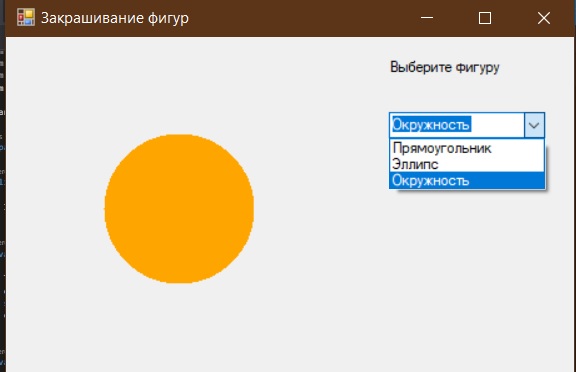


Рисунок 17

## Задание № 6. Событие MauseHover

Событие MauseHover позволяет при наведении на какой-либо объект (в данном случае простой текст с надписью «Не трогать» выполнять какие-либо действие. Как показано на рисунке 18 при наведении текст меняется на красны текст “Error” и открывается окно с сообщением:

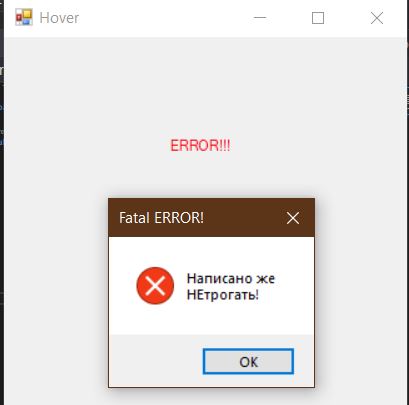


Рисунок 18

## Формирование траектории для движения простого объекта

Движение показано на рисунках 19 и 20:

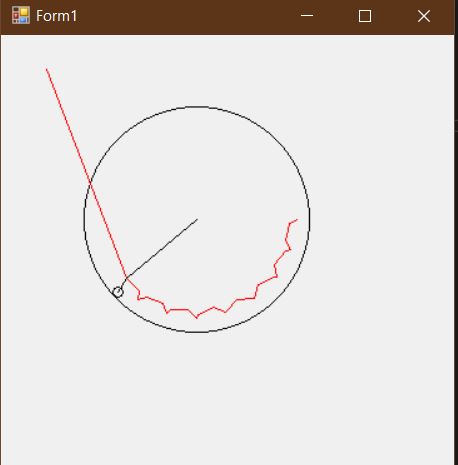


Рисунок 19

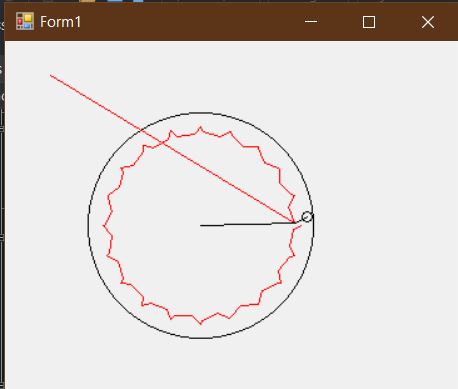


Рисунок 20

# Индивидуальное задание 1

## Формулировка задания

Вариант 42-8-4. Написать программу в Windows Forms, которая отображает динамическое движение объекта по заданной траектории. Траектория – котангенс, объект – параллелограмм, условие – заданные разные цвета контура и заливки.

## Математическая постановка

В задании требуется построить движение объекта по траектории – ctgx. Для понятности и компактности представления было решено взять только одну «ветку» котангенса – в промежутке от 0 до (не закрашенная область на рисунке 1.1), так как экран ограничен, а все «ветки» по своей структуре одинаковы. Если число повторов движения больше одного, то это демонстрирует движение по нескольким веткам, только без сдвига.

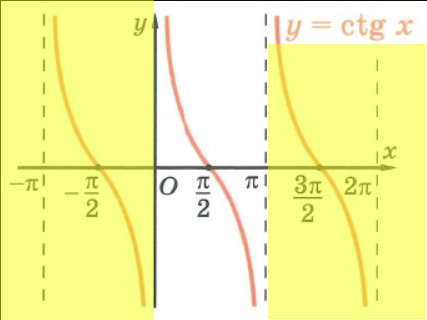


Рисунок 1.1

Объект – параллелограмм, который задается длинной, высотой и (в данном случае) большим углом (рисунок 1.2). Так, при подборе правильных параметров может получиться и прямоугольник, и квадрат, так как по определению они – разновидность параллелограмма.

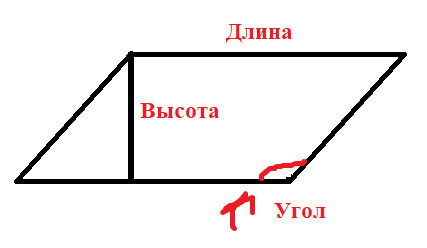


Рисунок 1.2

Для прорисовки параллелограмма на вход подается условный «центр», который привязан к траектории – пересечение диагоналей квадрата с координатой (х, у) – рисунок 3. Тогда левый верхний угол можно рассчитать как:

где a - длина, h – высота, angle – больший угол (рисунки 1.2 и 1.3).

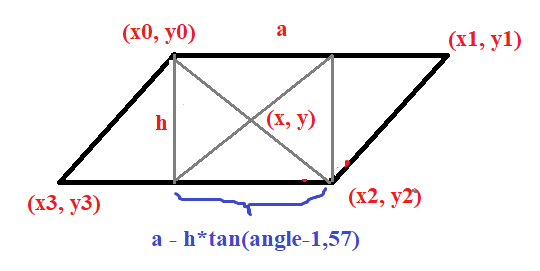


Рисунок 1.3

Тогда, очевидно, что последующие координаты точек рассчитываются как:

\*Примечание: все координаты считаются исходя из координат экрана, где ось оу направлена вниз.

## Описание пользовательского интерфейса

Пользовательский интерфейс представлен на рисунке 1.4. Слева большое окно picturebox для выведения результатов (графика и движения по траектории). Справа – настройка различных параметров. Кнопка «Рисовать» запускает движение с заданными параметрами.

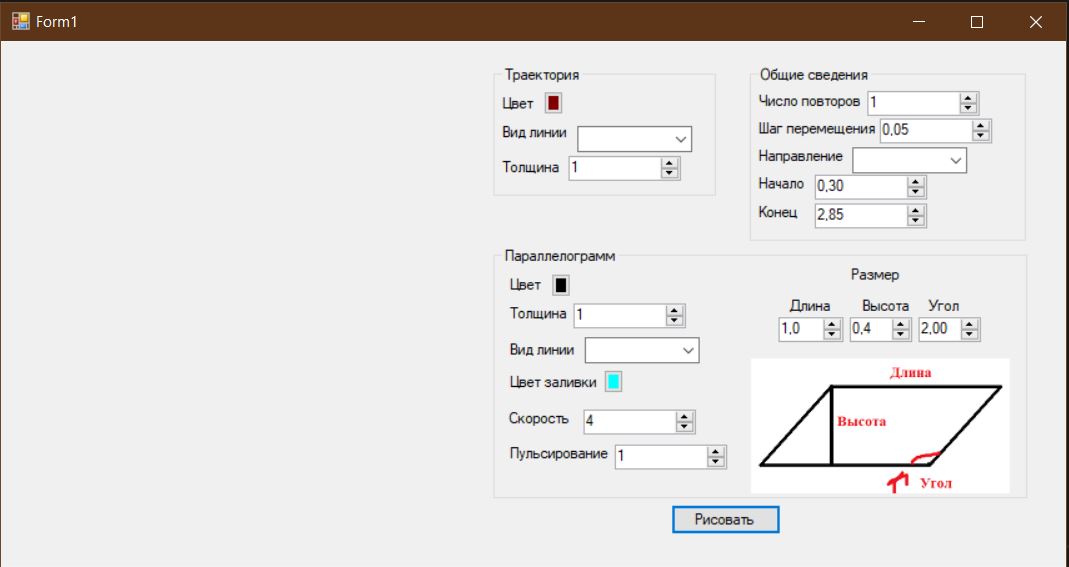


Рисунок 1.4

Настройки разделены на 3 блока (подробнее об ограничениях и назначении каждого элемента написано в таблице 1.1):

«Траектория» устанавливает ее цвет, вид линии (простая, пунктирная, из точек, точка-пунктир), толщина линии.

«Общие сведения» устанавливает число повторов, шаг перемещения, начало и конец траектории (по оси икс), направление движения (вперед, назад и туда-обратно).

«Параллелограмм» устанавливает настройки объекта: цвет контура, толщину и вид линии контура, цвет заливки, скорость движения и пульсирование (1 – нет пульсации, чем больше значение, тем сильнее увеличивается при пульсации объект). Также можно устанавливать размер (для наглядности представлен рисунок, чтобы пояснить, какой параметр за что отвечает): длину, высоту и угол в радианах (больше – 90 градусов, но меньше – 180 градусов).

Таблица 1.1

|  |  |  |
| --- | --- | --- |
| **Название элемента** | **Назначение** | **Принимаемые значения** |
| pictureBox1 | Вывод результатов | - |
| button1 | «Рисовать» (Запуск движения) | - |
| ***groupBox1*** | ***Группирует объекты «Траектория»*** | ***-*** |
| button2 | Цвет траектории | Вызывается colorDialogue1 с выбором любого цвета |
| comboBox1 | Вид линии траектории | Сплошная, Пунктирная, Из точек, Точка-тире |
| numericUpDown1 | Толщина линии траектории | От 1 до 20 с шагом 1 |
| ***groupBox2*** | ***Группирует объекты «Общие сведения»*** | ***-*** |
| numericUpDown4 | Число повторов | От 1 до 500 с шагом 1 |
| numericUpDown5 | Шаг перемещения (зависит точность прорисовки) | От 0,01 до 0,5 с шагом 0,01 |
| numericUpDown6 | Начало (первая точка прорисовки траектории по оси х) | От 0,05 до 2 с шагом 0,05 |
| numericUpDown7 | Конец (последняя точка прорисовки по оси х) | От 2 до 3,1 с шагом 0,05 |
| comboBox2 | Направление движения | Слева направо, Справа налево, Туда-сюда |
| ***groupBox3*** | ***Группирует объекты «Параллелограмм»*** | ***-*** |
| pictureBox2 | Отображение примера объекта (для помощи задания параметров) | - |
| button3 | Цвет контура объекта | Вызывается colorDialogue2 с выбором любого цвета |
| button4 | Цвет заливки объекта | Вызывается colorDialogue3 с выбором любого цвета |
| comboBox4 | Вид линии контура | Сплошная, Пунктирная, Из точек, Точка-тире |
| numericUpDown2 | Толщина линии контура объекта | От 1 до 20 с шагом 1 |
| numericUpDown3 | Скорость движения объекта (зависит задержка между каждым шагом) | От 1 до 100 с шагом 1 |
| numericUpDown8 | Пульсирование (увеличение и уменьшение объекта по ходу движения) | От 1 до 10 с шагом 1 |
| numericUpDown9 | Длина объекта | От 0,1 до 2 с шагом 0,1 |
| numericUpDown10 | Высота объекта | От 0,1 до 2 с шагом 0,1 |
| numericUpDown11 | Угол (в радианах) | От 1,57 (90 градусов) до 3 (172 градуса) с шагом 0,01 |

## Описание графических примитивов

Для построения графиков в данной программе использовались следующие графические примитивы: DrawLine – строит линию, соединяя две заданные точки DrawLines – получает на вход массив точек и последовательно соединяет две соседние точки, DrawPolygon – рисует замкнутую фигуру (полигон), последовательно соединяя переданные точки, FillPolygon – делает тоже самое, но с заливкой. При правильно рассчитанных точках DrawPolygon рисует параллелограмм, который и требуется в задании.

## Пример работы программы

Так как показать движение сложно на картинке, приведем две картинки (рисунок 1.5 и 1.6), на которых при одних и тех же параметрах за один цикл параллелограмм находится в разных положениях.

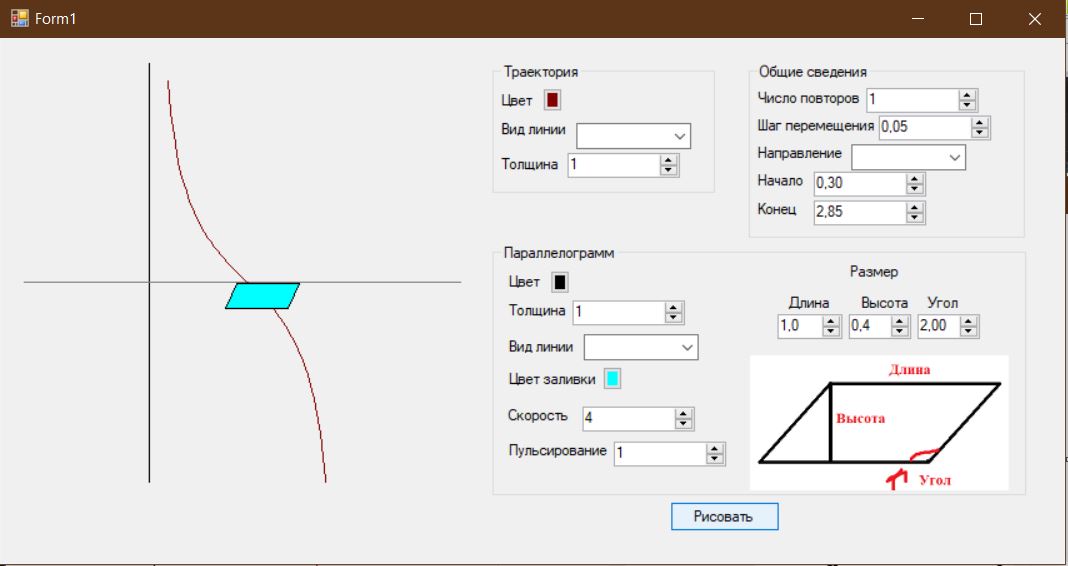


Рисунок 1.5

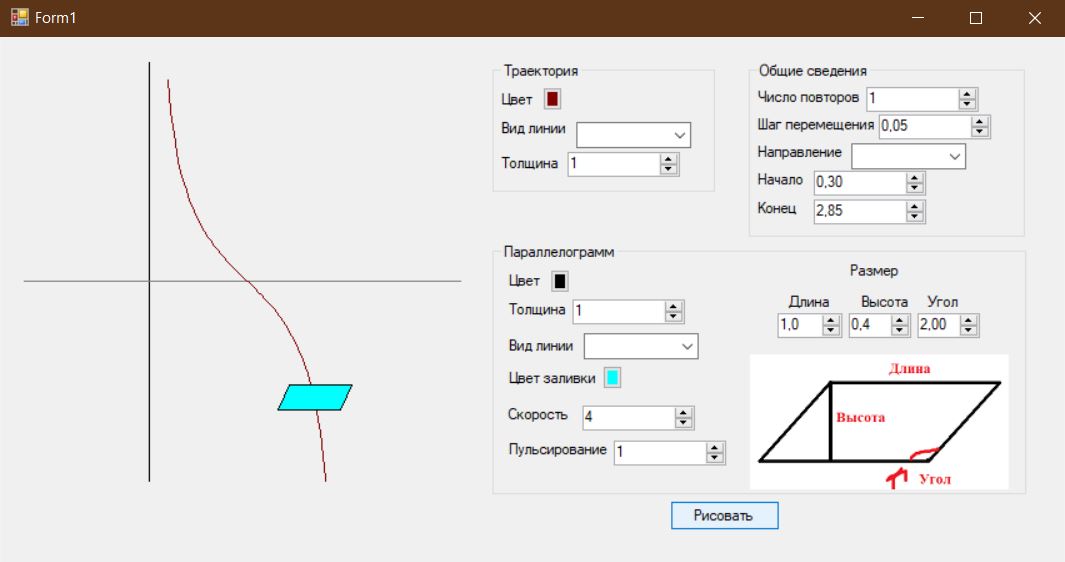


Рисунок 1.6

На рисунке 1.7 показан пример пульсации: по сравнению с рисунком 6 размер не изменился, но параллелограмм больше, так как включена пульсация, т.е. на каждом шаге объект увеличивается до определённого размера, потом уменьшается. Также на рисунке 1.7 изменили параметры заливки, вид и толщину контура и траектории.

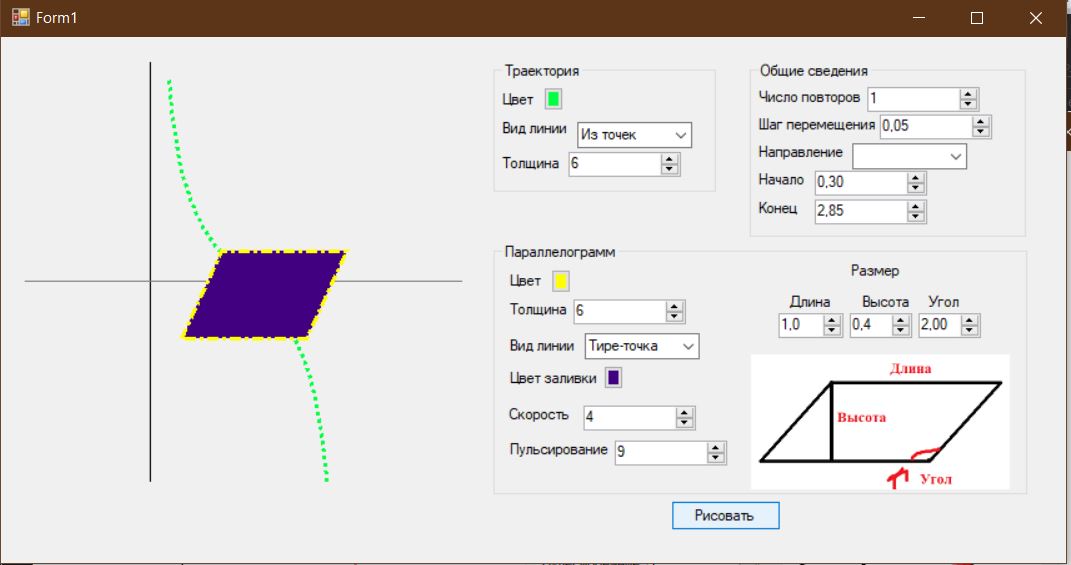


Рисунок 1.7

На рисунке 1.8 представлен пример, когда параллелограмм становится квадратом (угол = 1,57 радиан = 90 градусов, высота и длина равны). Также изменены начало и конец траектории.

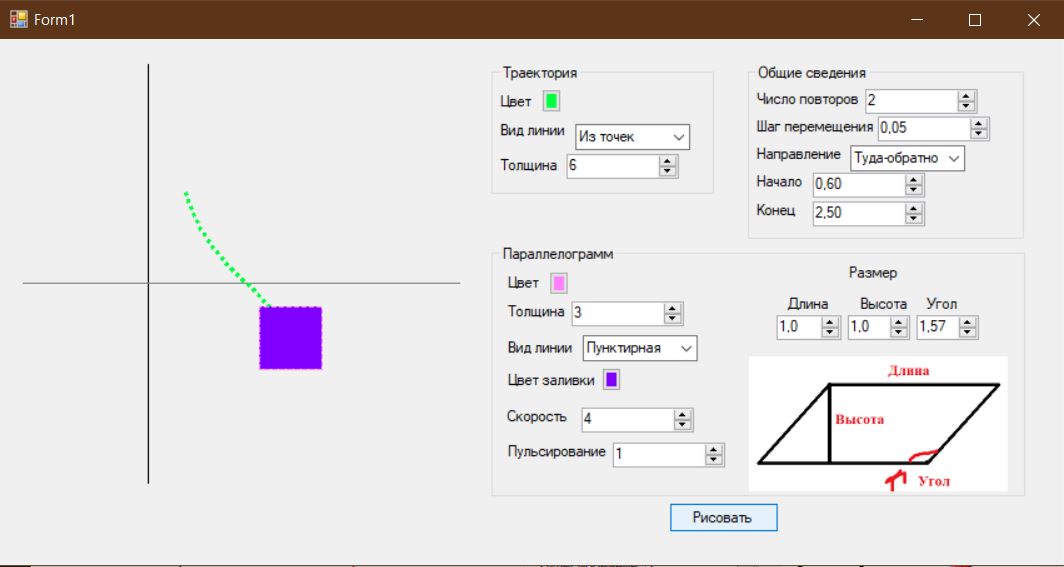


Рисунок 1.8

На рисунке 1.9 представлено то, как выглядит максимальная толщина линии – 20.

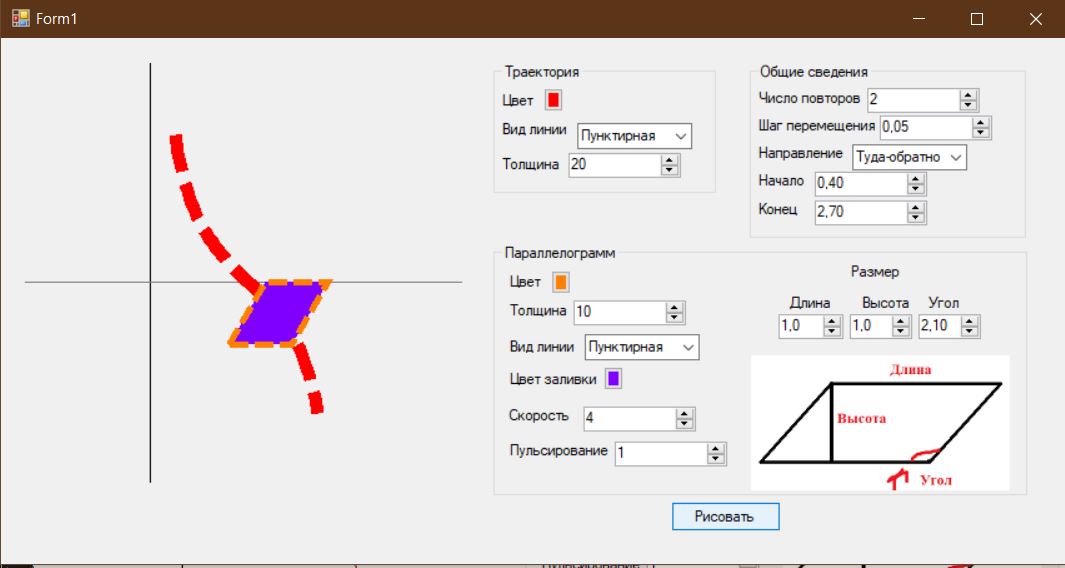


Рисунок 1.9

На рисунке 1.10 представлен максимально возможный угол параллелограмма. Как видно, хотя объект и очень сжатый, это все еще параллелограмм.

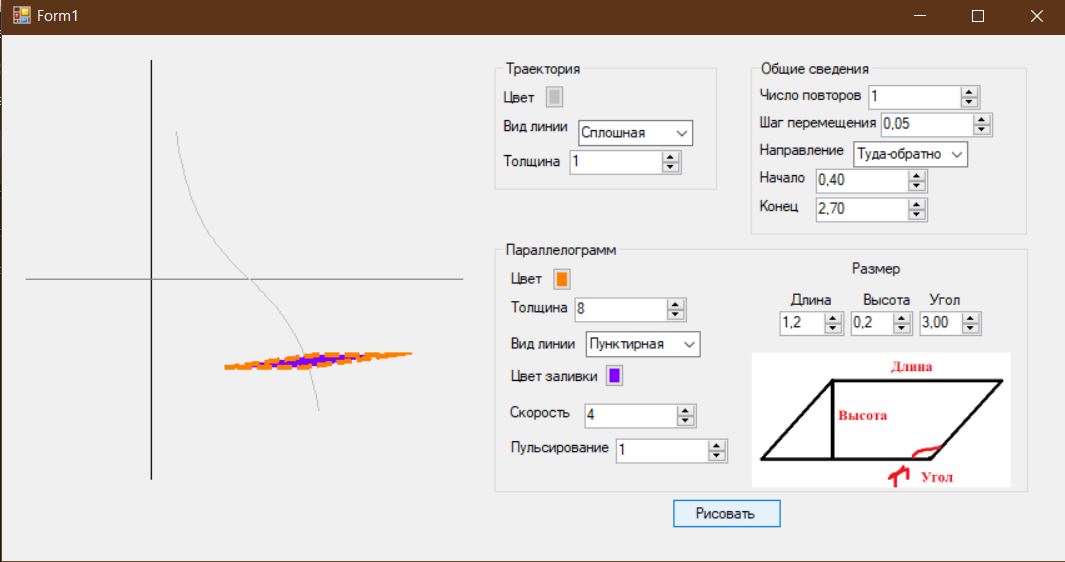


Рисунок 1.10

Так как по заданию требуется установить разные цвета контура и заливки, то при попытке сделать их одинаковыми выпадает следующее окно:

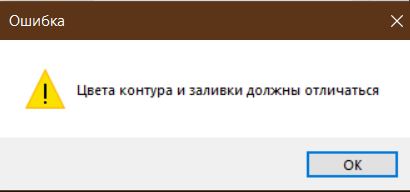


Рисунок 1.11

## Текст программы

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Threading;

using System.Drawing.Drawing2D;

namespace Part2

{

public partial class Form1 : Form

{

int osX = 100, osY = 175;

int timeToSleep = 120;

double InitT = 0.05, LastT = 3.1; // оборот в 360 градусов (6,28 радиан)

double Step = 0.05, curPoint;

int scale = 50;

int pointsToDraw;

int direction = 0;

int a = 50, h = 20; double angle = 2;

//draw styles

Color traektColor, borderColor, fillColor; //buttons 2, 3, 4

DashStyle styleGR = DashStyle.Solid, stylePr = DashStyle.Solid;

//puls

int pulsStep = 1, puls;

bool Increase = true;

public Form1()

{

InitializeComponent();

}

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

{

button2.BackColor = colorDialog1.Color;

button3.BackColor = colorDialog2.Color;

button4.BackColor = colorDialog3.Color;

traektColor = colorDialog1.Color;

borderColor = colorDialog2.Color;

fillColor = colorDialog3.Color;

}

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

{

pointsToDraw = (int)Math.Round((LastT - InitT) / Step) + 1;

Step = (double)numericUpDown5.Value;

InitT = (double)numericUpDown6.Value;

LastT = (double)numericUpDown7.Value;

a = (int)(numericUpDown9.Value \* scale);

h = (int)(numericUpDown10.Value \* scale);

angle = (double)numericUpDown11.Value;

puls = (int)numericUpDown8.Value;

pulsStep = 1; Increase = true;

for (int i = 0; i < numericUpDown4.Value; i++)

switch (direction)

{

case 0: Paint\_MoveForward(); break;

case 1: Paint\_MoveBack(); break;

case 2: Paint\_MoveForward(); Paint\_MoveBack(); break;

}

}

private void Paint\_Osi(int x0, int y0)

{

Graphics gr = pictureBox1.CreateGraphics();

gr.DrawLine(Pens.Black, x0, 0, x0, 1000);

gr.DrawLine(Pens.Gray, 0, y0, 1000, y0);

}

private void Paint\_Parallelogr(Point center)

{

Graphics gr = pictureBox1.CreateGraphics();

Point[] points = new Point[4];

double b = h \* Math.Tan(-1.57 + angle);

Point startPoint = new Point(center.X - (int)((a - h \* Math.Tan(-1.57 + angle))/2), center.Y - h / 2);

points[0] = startPoint;

points[1] = new Point(startPoint.X + a, startPoint.Y);

points[2] = new Point(startPoint.X - (int)b + a, startPoint.Y + h);

points[3] = new Point(startPoint.X - (int)b, startPoint.Y + h);

Pen border = new Pen(borderColor);

border.DashStyle = stylePr;

border.Width = (float)numericUpDown2.Value/2;

SolidBrush inside = new SolidBrush(fillColor);

gr.FillPolygon(inside, points);

gr.DrawPolygon(border, points);

}

private void Paint\_FullGraphic()

{

Graphics gr = pictureBox1.CreateGraphics();

gr.Clear(BackColor);

Paint\_Osi(osX, osY);

PointF[] p = new PointF[pointsToDraw];

double cur = InitT;

for (int i = 0; i < pointsToDraw; i++)

{

double x = cur \* scale;

double y = (1.0 / Math.Tan(cur)) \* scale;

p[i] = new PointF(osX + (int)x, osY - (int)y); // расчет очередной точки траектории

cur += Step;

if (cur > LastT) break;

}

Pen pen = new Pen(traektColor);

pen.DashStyle = styleGR;

pen.Width = (float)numericUpDown1.Value / 2;

gr.DrawLines(pen, p); // траектория

}

private void Paint\_Graphic(PointF[] p)

{

Graphics gr = pictureBox1.CreateGraphics();

gr.Clear(BackColor);

Paint\_Osi(osX, osY);

Pen pen = new Pen(traektColor);

pen.DashStyle = styleGR;

pen.Width = (float)numericUpDown1.Value/2;

gr.DrawLines(pen, p); // траектория

}

private void Paint\_MoveForward()

{

//set parameters

double x, y;

int i = 0; // количество точек прорисовки

pointsToDraw = (int)Math.Round((LastT - InitT) / Step) + 1;

PointF[] p = new PointF[pointsToDraw]; // точки для прорисовки (LastT/Step)

curPoint = InitT;

int aInit = a, hInit = h;

//draw

while (curPoint <= LastT)

{

x = curPoint \* scale;

y = (1.0 / Math.Tan(curPoint)) \* scale;

p[i] = new PointF(osX + (int)x, osY - (int)y); // расчет очередной точки траектории

Paint\_FullGraphic();

//Paint\_Graphic(p);

Pulse();

Point point = new Point((int)p[i].X, (int)p[i].Y);

Paint\_Parallelogr(point);

curPoint += Step;

Thread.Sleep(timeToSleep/(int)numericUpDown3.Value); //время приостановки прорисовки

i++;

}

}

private void Pulse()

{

//парам пам пам пульсация

if (puls != 1)

{

if (Increase)

{

a += 10; h += 10;

pulsStep++;

}

else

{

a -= 10; h -= 10;

pulsStep--;

}

if (pulsStep > puls) Increase = false;

else if (pulsStep == 1) Increase = true;

}

}

private void Paint\_MoveBack()

{

//set parameters

double x, y;

int i = 0; // количество точек прорисовки

PointF[] p = new PointF[pointsToDraw]; // точки для прорисовки (LastT/Step)

curPoint = LastT;

int aInit = a, hInit = h;

//draw

while (curPoint >= InitT)

{

x = curPoint \* scale;

y = (1.0 / Math.Tan(curPoint)) \* scale;

p[i] = new PointF(osX + (int)x, osY - (int)y); // расчет очередной точки траектории

Paint\_FullGraphic();

//Paint\_Graphic(p);

Pulse();

Point point = new Point((int)p[i].X, (int)p[i].Y);

Paint\_Parallelogr(point);

curPoint -= Step;

Thread.Sleep(timeToSleep / (int)numericUpDown3.Value); //время приостановки прорисовки

i++;

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

//вид линии траектории

styleGR = (DashStyle)Enum.ToObject(typeof(DashStyle), comboBox1.SelectedIndex);

}

private void comboBox4\_SelectedIndexChanged(object sender, EventArgs e)

{

//вид линии объекта

stylePr = (DashStyle)Enum.ToObject(typeof(DashStyle), comboBox4.SelectedIndex);

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

//направление 0 - вперед, 1 - назад, 2 - туда обратно

direction = comboBox2.SelectedIndex;

}

private void button2\_Click(object sender, EventArgs e)

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

button2.BackColor = colorDialog1.Color;

traektColor = colorDialog1.Color;

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (colorDialog2.ShowDialog() == DialogResult.OK)

{

if (colorDialog3.Color != colorDialog2.Color)

{

button3.BackColor = colorDialog2.Color;

borderColor = colorDialog2.Color;

}

else

{

MessageBox.Show("Цвета контура и заливки должны отличаться", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (colorDialog3.ShowDialog() == DialogResult.OK)

{

if (colorDialog3.Color != colorDialog2.Color)

{

button4.BackColor = colorDialog3.Color;

fillColor = colorDialog3.Color;

}

else MessageBox.Show("Цвета контура и заливки должны отличаться", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

}

# Индивидуальное задание 2

## Формулировка задания

Вариант 20–1. Формирование изображения фракталов с отображением формирующихся уровней на дереве. Вид фрактала – множество Кантора. Необходима прорисовка n фракталов на разных участках и дерево в одном окне.

## Математическая постановка

Множество Кантора (рисунок 2.1) – размещение 4 уменьшенных квадратов под базовым квадратом в его вершинах (далее используются только три квадрата).

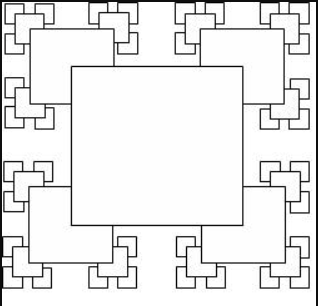


Рисунок 2.1

Для хранения фрактала используется дерево. Узлом дерева является квадрат, его дети – три или четыре квадрата, в зависимости от уровня. Каждый узел хранит длину квадрата, координаты квадрата, информацию о текущем уровне и позицию (слева сверху, справа снизу от родителя и т.д.)

При создании дерева передается информация о центре первого квадрата и его длина, исходя из этого можно рассчитать информацию для «детей»:

,

где х, у – координаты центра родителя, а – длина стороны родителя, а вершины квадратов рассчитываются в следующем порядке: верхний левый угол, верхний правый угол, нижний правый угол, нижний левый угол.

После первого уровня отпадает необходимость считать 4 квадрата-ребенка, потому что теперь один из них находится под другим квадратом, его не видно, значит, он нам не интересен. Так, если индекс родителя делится на два без остатка (0 и 2), то нам не нужен тот квадрат, индекс которого в сумме с индексом родителя дает 2 (2 и 0 соответственно). В другом случае сумма должна быть равна 4.

## Описание пользовательского интерфейса

На пользовательском интерфейсе (рисунок 2.2) не много деталей: слева окно для вывода результатов, кнопка для запуска прорисовки фрактала и дерева.

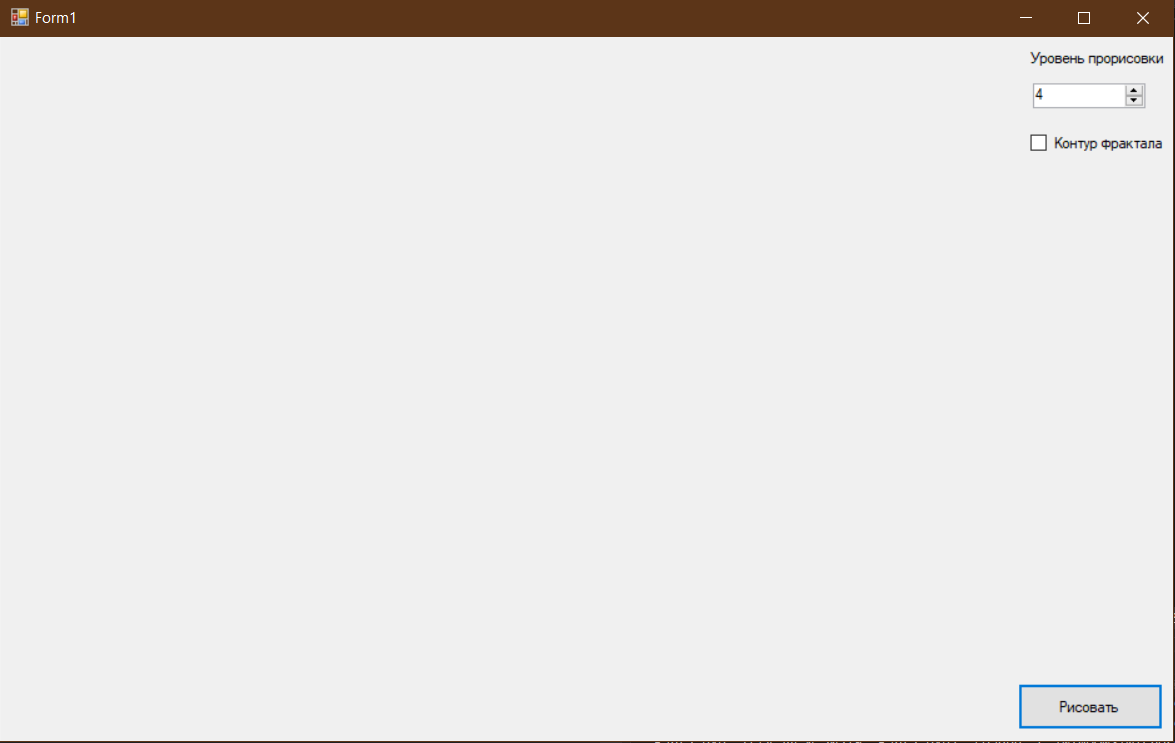


Рисунок 2.2

Из настроек – выбор уровня прорисовки фрактала (от 0 – простой квадрат, до 9 – 10 рисунков фракталов, в последнем из которых прорисовано 9 уровней). Также можно выбрать, прорисовывать ли контур квадратов на фрактале, поставив галочку в квадрат или убрав ее (без контура уменьшается скорость работы программы)

Таблица 2.1

|  |  |  |
| --- | --- | --- |
| **Название элемента** | **Назначение** | **Принимаемые значения** |
| pictureBox1 | Вывод результатов | - |
| button1 | «Рисовать» (Отображение фрактала и дерева) | - |
| numericUpDown1 | Уровень прорисовки фрактала | От 0 до 9 с шагом 1 |
| checkBox1 | Прорисовывать ли контур квадратов у фрактала | True, false |

## Описание графических примитивов

Для построения графиков в данной программе использовались следующие графические примитивы: DrawLine – строит линию, соединяя две заданные точки. DrawEllipse – рисует эллипс (при одинаковых радиусах – окружность), DrawRectangle – рисует прямоугольник (при одинаковых длинах сторон рисует квадрат), FillRectangle – аналогично DrawRectangle, но с заливкой. Эллипс и линии используются для отрисовки дерева, прямоугольник для отрисовки самого фрактала

## Пример работы программы

На рисунке 2.3 пример отрисовки двух уровней фрактала с конутром.

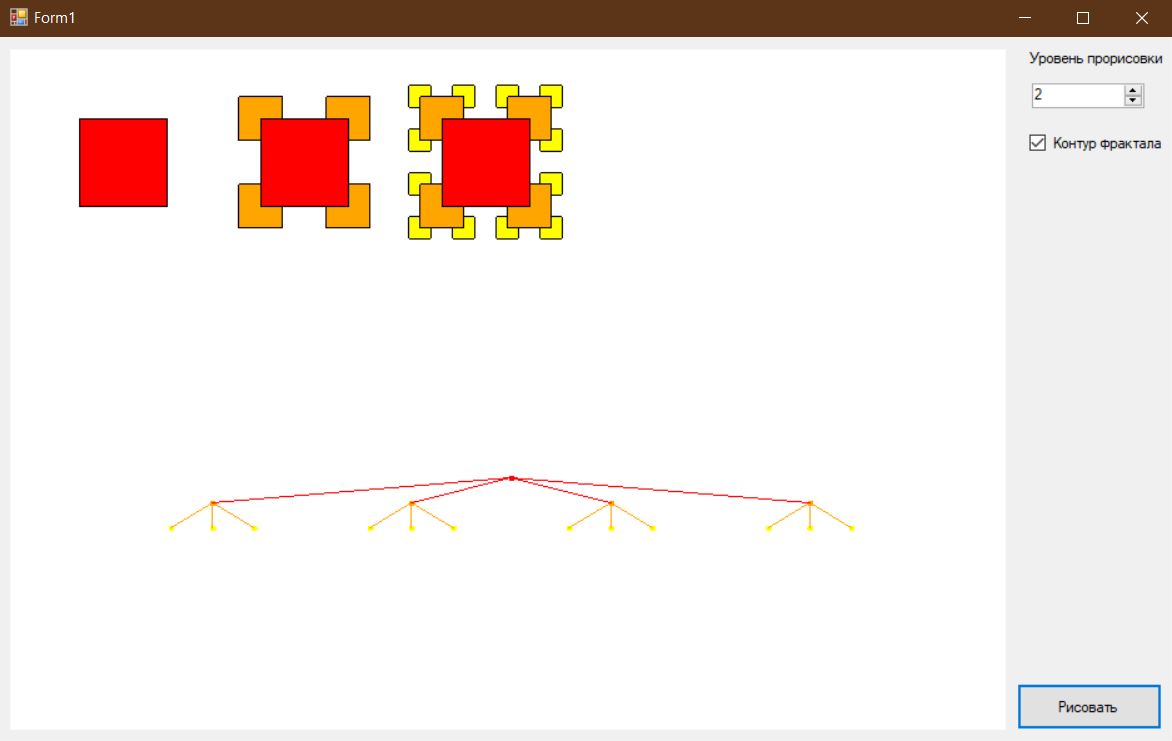


Рисунок 2.3

Как видно из примера, программа рисует n фракталов на всех уровнях от 0 до выбранного пользователем. Дерево строится для последнего. Уровень на фрактале раскрашен в тот же цвет, что и соответствующий уровень на дереве. На рисунке 2.4 представлен фрактал 4 уровня без контура.

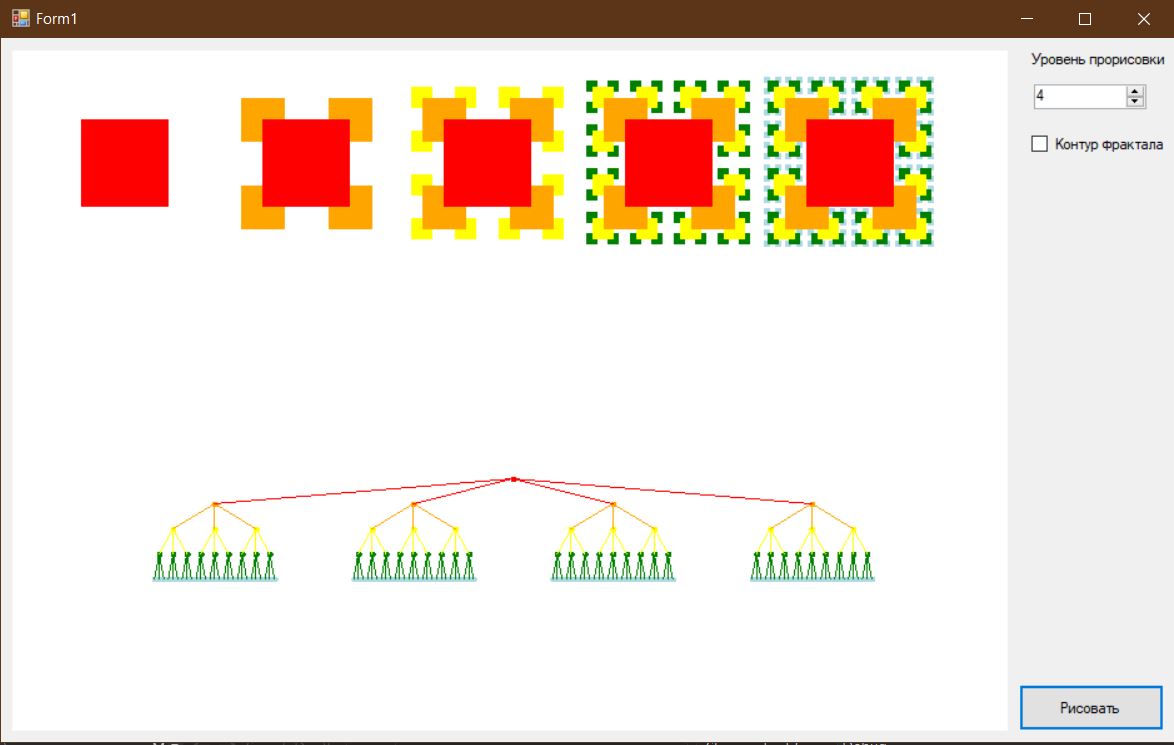


Рисунок 2.4

Максимальный уровень – 9, представлен на рисунке 2.5.

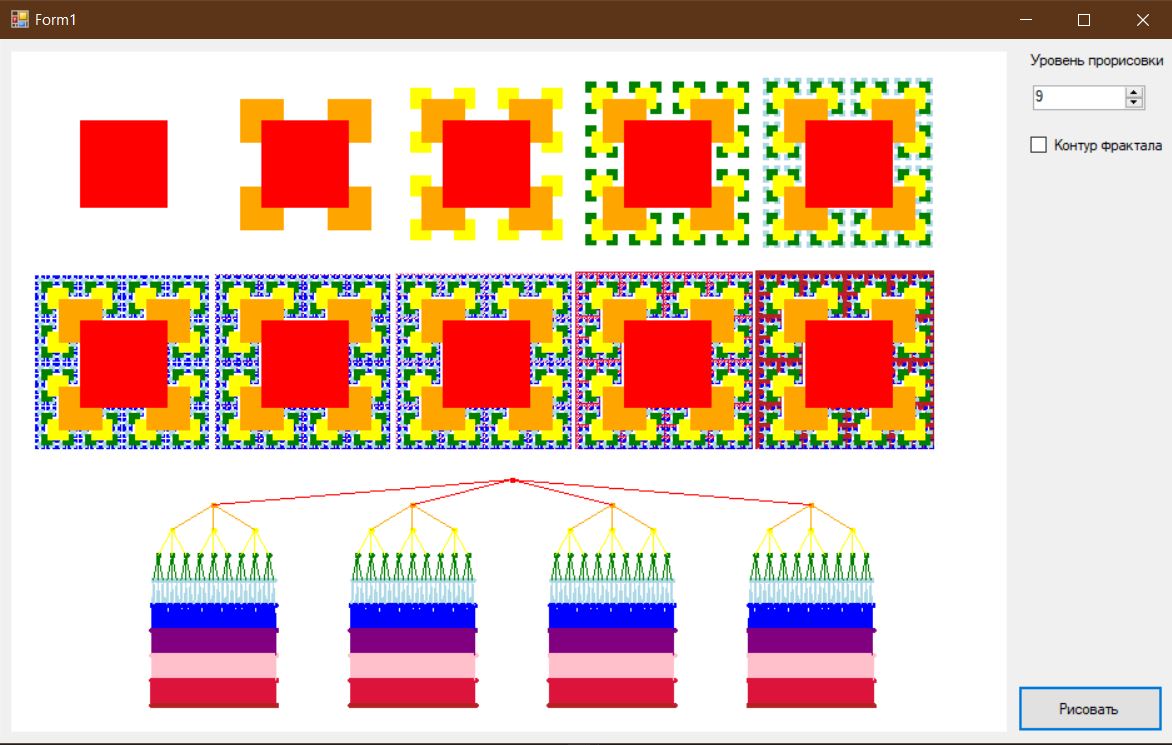


Рисунок 2.5

При выборе 9 уровня с контуром отрисовка фрактала занимает некоторое время, поэтому можно проследить процесс построения (рисунок 2.6). Результат такого построения представлен на рисунке 2.7.



Рисунок 2.6

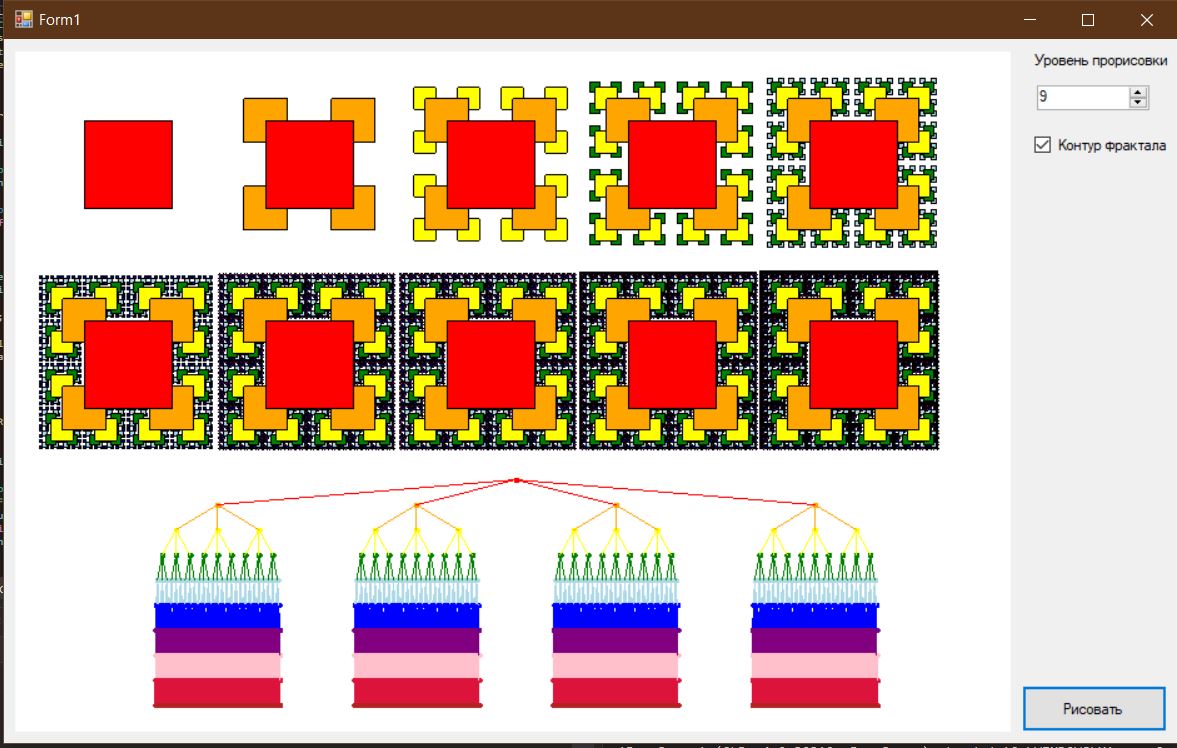


Рисунок 2.7

## Текст программы

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Part3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

FracTree tree = new FracTree(70, new Point(90, 90));

bool drawLine = checkBox1.Checked;

tree.BuildFracTree((int)numericUpDown1.Value);

Graphics graphics = pictureBox1.CreateGraphics();

tree.Draw(graphics, checkBox1.Checked);

}

}

public class FracTree

{

public class TreeNode

{

public int position { get; } //left up, right up, right down, left down

public int level;

public Point center;

public float a { get; set; }

public TreeNode parent { get; set; }

public TreeNode[] children;

public TreeNode(TreeNode parent, int pos)

{

children = new TreeNode[4];

position = pos;

this.parent = parent;

if (parent != null)

{

a = parent.a / 2;

switch (position)

{

case 0: center = new Point((int)(parent.center.X - parent.a / 2), (int)(parent.center.Y - parent.a / 2)); break;

case 1: center = new Point((int)(parent.center.X + parent.a / 2), (int)(parent.center.Y - parent.a / 2)); break;

case 2: center = new Point((int)(parent.center.X + parent.a / 2), (int)(parent.center.Y + parent.a / 2)); break;

case 3: center = new Point((int)(parent.center.X - parent.a / 2), (int)(parent.center.Y + parent.a / 2)); break;

}

level = parent.level + 1;

}

}

}

TreeNode root;

int levels;

bool drawLine = false;

public FracTree(int a, Point center)

{

root = new TreeNode(null, -1);

root.a = a; root.center = center;

root.level = 0;

levels = 0;

}

public void BuildFracTree(int levels)

{

this.levels = levels;

if (levels > 0) BuildRecurs(root);

}

private void BuildRecurs(TreeNode cur)

{

for (int i = 0; i < 4; i++)

{

bool needToDraw = !((cur != root) && ((cur.position % 2 == 0 && (i + cur.position == 2)) || (cur.position % 2 != 0 && (i + cur.position == 4))));

if (needToDraw)

{

cur.children[i] = new TreeNode(cur, i);

if (cur.children[i].level != levels) BuildRecurs(cur.children[i]);

}

else cur.children[i] = null;

}

}

public void Draw(Graphics gr, bool drawLine = false)

{

this.drawLine = drawLine;

gr.Clear(Color.White);

DrawRecurs(root, gr);

float centerX = gr.VisibleClipBounds.Width / 2, centerY = 2 \* (2 \* root.a + 20) + 20;

DrawTreeRecurs(gr, root, new PointF(centerX, centerY));

}

void DrawRecurs(TreeNode cur, Graphics gr)

{

for (int i = 0; i < 4; i++)

{

TreeNode child = cur.children[i];

if (child != null)

{

bool needToDraw = !((cur != root) && ((cur.position % 2 == 0 && child.position + cur.position == 2) || (cur.position % 2 != 0 && child.position + cur.position == 4)));

if (needToDraw) DrawRecurs(child, gr);

}

}

float moveX = 2 \* root.a + 5, moveY = 0, k;

for (int i = cur.level; i <= levels; i++)

{

k = i;

if (i > 4) { moveY = 2 \* root.a + 20; k = k % 5; }

gr.FillRectangle(new SolidBrush(ChooseColor(cur)), (float)(cur.center.X + moveX\*k - cur.a / 2), (float)(cur.center.Y + moveY - cur.a / 2), cur.a, cur.a);

if(drawLine) gr.DrawRectangle(new Pen(Color.Black), (float)(cur.center.X + moveX\*k - cur.a / 2), (float)(cur.center.Y + moveY - cur.a / 2), cur.a, cur.a);

}

}

void DrawTreeRecurs(Graphics gr, TreeNode cur, PointF parCenter)

{

int koeff = -1; float width = gr.VisibleClipBounds.Width / 2;

for (int i = 0; i < 4; i++)

{

TreeNode child = cur.children[i];

PointF newCenter = new PointF(0, 0);

if (cur == root) newCenter = new PointF(parCenter.X \* 2 \* (i + 1) / 5, parCenter.Y + 20);

else if (child != null) newCenter = new PointF(parCenter.X + koeff\*width/(4\*(float)Math.Pow(3, cur.children[i].level-1)), parCenter.Y + 20);

if (child != null) { koeff++; DrawTreeRecurs(gr, child, newCenter); gr.DrawLine(new Pen(ChooseColor(cur)), parCenter.X + 2, parCenter.Y + 2, newCenter.X + 2, newCenter.Y + 2); }

}

SolidBrush brush = new SolidBrush(ChooseColor(cur));

gr.FillEllipse(brush, parCenter.X, parCenter.Y, 5, 5);

}

Color ChooseColor(TreeNode node)

{

//Random rand = new Random();

//return Color.FromArgb(rand.Next(0, 255), rand.Next(0, 255), rand.Next(0, 255));

switch (node.level)

{

case 0: return Color.Red;

case 1: return Color.Orange;

case 2: return Color.Yellow;

case 3: return Color.Green;

case 4: return Color.LightBlue;

case 5: return Color.Blue;

case 6: return Color.Purple;

case 7: return Color.Pink;

case 8: return Color.Crimson;

case 9: return Color.Firebrick;

default: return Color.Black;

}

}

}

}