Denis Ramskiy II 11

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

namespace LakyDog
{
    public partial class Form1 : Form
    {
        int generationCount;
        Random rnd ;
        Generation work ;
        Points wPoint;
        public Form1()
        {
            InitializeComponent();
            generationCount = 0;
            wPoint = new Points();
            rnd = new Random(DateTime.Now.Millisecond);
            work = new Generation(rnd, wPoint);
            pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

            File.WriteAllText("BestFitness.txt","");
            File.WriteAllText("AveregFitness.txt", "");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            generationCount++;
            Bitmap new_map = new Bitmap(1000,1000);
            Pen myPen = new Pen(Color.Red);
            Graphics graph = Graphics.FromImage(new_map);

            work.new_generation(rnd, wPoint);
            for (int i = 0; i < 20; i++)
            {
                int x = 500, y = 500;
                for (int j = 0; j < 1000; j++)
                {
                    int nx = x, ny = y;
                    switch (work.get(i).get()[j])
                    {
                        case 0: nx--; break;
                        case 1: nx++; break;
                        case 2: ny++; break;
                        case 3: ny--; break;
                    }
                    graph.DrawLine(myPen, x, y, nx, ny);
                    x = nx; y = ny;
                }
            }
            myPen.Color = Color.Green;
            myPen.Width = 8;

            for (int i = 0; i < wPoint.getCount(); i++)
                graph.DrawEllipse(myPen, wPoint.getPoint(i).getPoints()[0] - 2,
wPoint.getPoint(i).getPoints()[1] - 2, 4, 4);
            pictureBox1.Image = new_map;
```

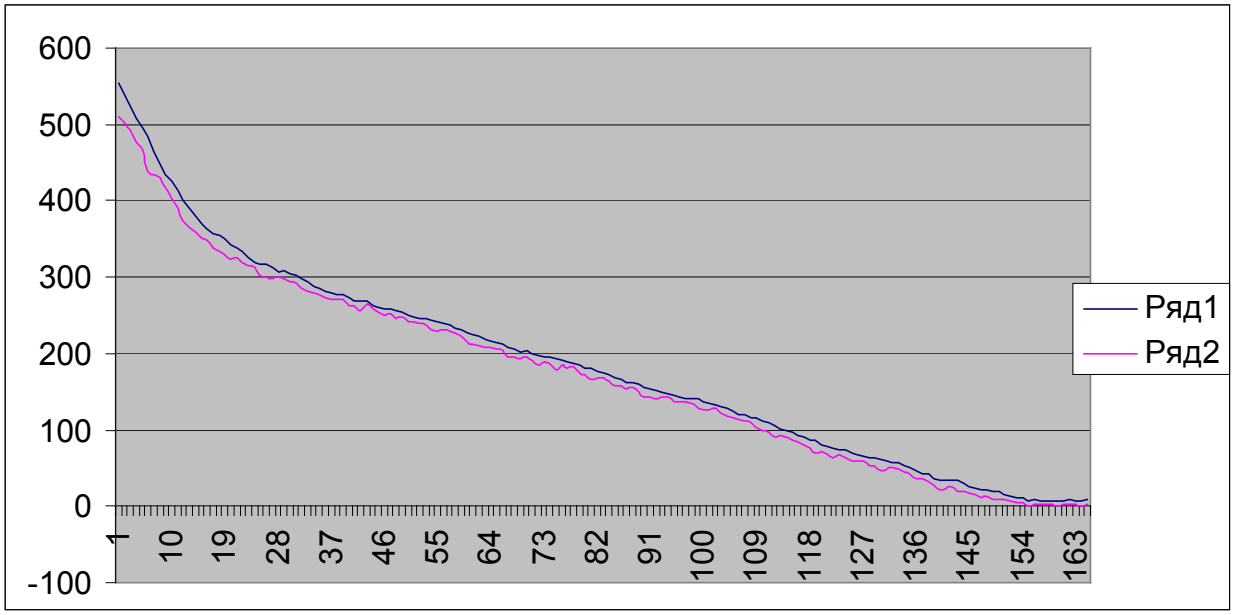
```
            File.AppendAllText("BestFitness.txt",
work.get()[0].get_shFitness().ToString()+"\r\n");
            File.AppendAllText("AveregFitness.txt", work.get_fitness().ToString()
+ "\r\n");

            textBox1.Text = work.get_fitness().ToString();
            textBox2.Text = generationCount.ToString();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LakyDog
{
    public class Gen
    {
        private List<int> hromosom;
        private double fitness;
        private double shFitness;
        private const int max=1000;

        public Gen(Random rnd,Points mypnt)
        {
            shFitness = 0;
            hromosom = new List<int>();
            for (int i = 0; i < max; i++)
                hromosom.Add(rnd.Next() % 4);
            fitness = calcFit(mypnt);
        }
        public Gen(Gen obj)
        {
            this.hromosom = new List<int>();
            for (int i = 0; i < max; i++)
                this.hromosom.Add(obj.get()[i]);
            this.fitness = obj.get_fitness();
            this.shFitness = obj.get_shFitness();
        }
        public Gen(List<int> obj, Points mypnt)
        {
            shFitness = 0;
            this.hromosom = new List<int>();
            for (int i = 0; i < max; i++)
                this.hromosom.Add(obj[i]);
            fitness = calcFit(mypnt);
        }
        public void mutation(Random rnd)
        {
            for (int i = 0; i < max; i++)
                if (rnd.Next() % 1000 < 3)
                    hromosom[i] = rnd.Next() % 4;
        }
        private int calcFit(Points mypnt)
        {
            int result = 1000;
            for (int i = 0; i < mypnt.getCount(); i++)
            {
                int temp1 = Math.Abs(mypnt.getPoint(i).getPoints()[0] -
corOfEnd()[0]);
                int temp2 = Math.Abs(mypnt.getPoint(i).getPoints()[1] -
corOfEnd()[1]);
                if (temp1 + temp2 < result)
                    result = temp1 + temp2;
            }
```

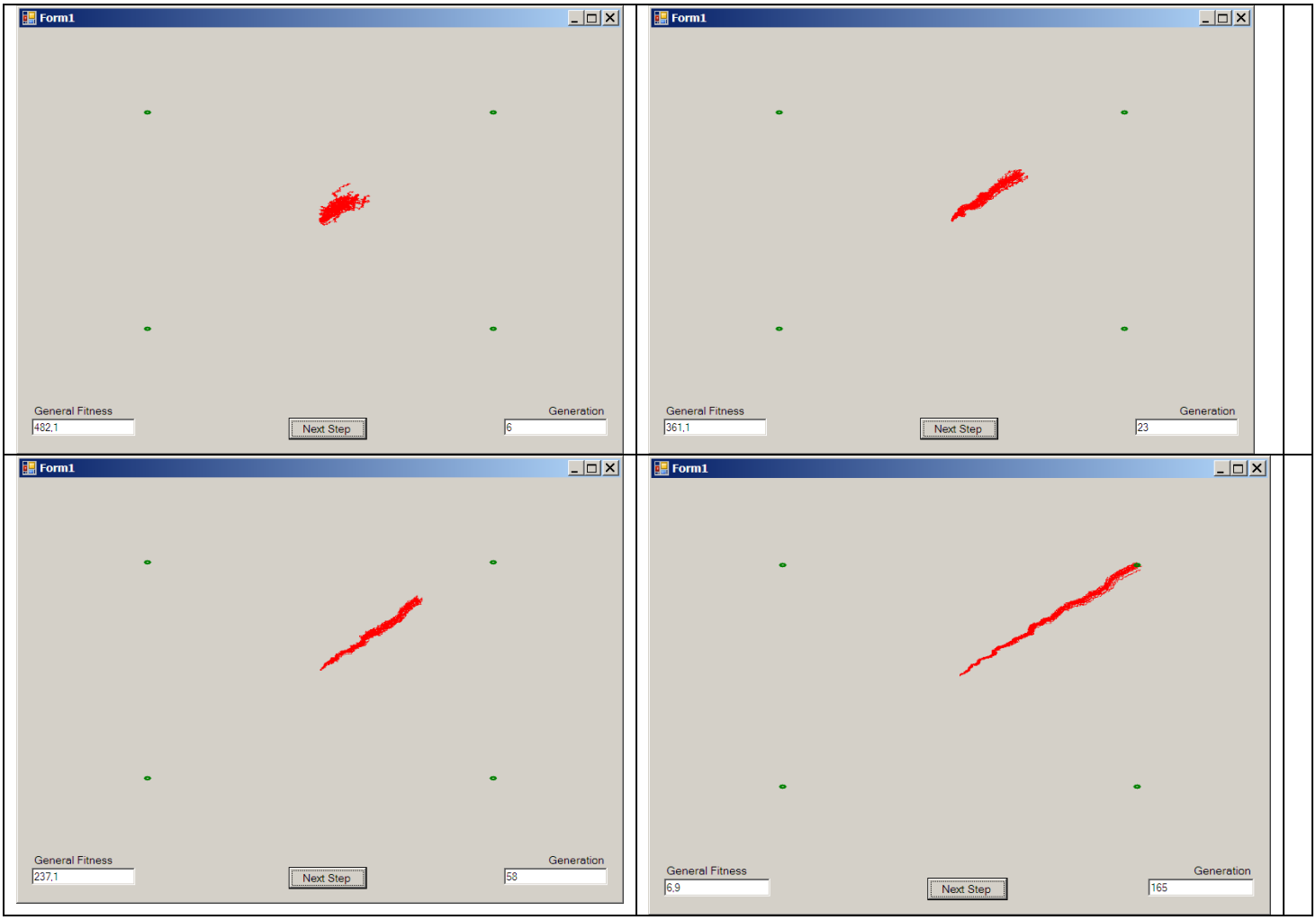
```
            return result;
        }
        static public Gen operator +(Gen obj1, Gen obj2)
        {
            obj1.get().Clear();
            for (int i = 0; i < max; i++)
                obj1.get().Add(obj2.get()[i]);
            obj1.set_fitness(obj2.get_fitness());
            obj1.set_shFitness(obj2.get_shFitness());
            return obj1;
        }
        public void set_fitness(double a)
        {
            this.fitness = a;
        }
        public List<int> get()
        {
            return hromosom;
        }
        public double get_fitness()
        {
            return fitness;
        }
        public double get_shFitness()
        {
            return shFitness;
        }
        public void set_shFitness(double a)
        {
            shFitness = a;
        }
        public List<int> corOfEnd()
        {
            List<int> temp = new List<int>();
            int x = 500, y = 500;
            for (int i = 0; i < max; i++)
            {
                switch (hromosom[i])
                {
                    case 0: x--; break;
                    case 1: x++; break;
                    case 2: y++; break;
                    case 3: y--; break;
                }
            }
            temp.Add(x);
            temp.Add(y);

            return temp;
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LakyDog
{
    public class Generation
    {
        private List<Gen> generation;
        private double fitness;
        private const int max = 1000;
```

```
        public Generation(Random rnd, Points mypnt)
        {
            generation = new List<Gen>();
            for (int I = 0; I < 20; I++)
            {
                Gen temp = new Gen(rnd, mypnt);
                generation.Add(temp);
                fitness += (double)temp.get_fitness() / 20;
            }
            for (int I = 0; I < 20; I++)
                calcShFitness(rnd, mypnt, generation[I]);

            this.shSort();
        }
        public void calcShFitness(Random rnd, Points mypnt,Gen obj)
        {
                double temp=0;
                for (int j = 0; j < 20; j++)
                {
                    int cor1 = Math.Abs(obj.corOfEnd()[0] -
generation[j].corOfEnd()[0]);
                    int cor2 = Math.Abs(obj.corOfEnd()[1] -
generation[j].corOfEnd()[1]);
                    if ( cor1 + cor2 < 50)
                        temp += Math.Abs(1 - (double)generation[j].get_fitness() /
(cor1 + cor2));
                }

                obj.set_shFitness((double)obj.get_fitness() / temp);
        }
        public void shSort()
        {
            for (int i = 0; i < 20; i++)
                for (int j = 0; j < 19; j++)
                    if (generation[j].get_fitness() > generation[j +
1].get_fitness())
                    {
                        Gen temp = new Gen(generation[j]);
                        generation[j] += generation[j + 1];
                        generation[j + 1] += temp;
                    }
        }
        public List<Gen> get_parents(Random rnd)
        {
            List<Gen> TEMP = new List<Gen>();
            TEMP.Add(generation[rnd.Next() % 10]);
            TEMP.Add(generation[rnd.Next() % 10]);
            return TEMP;
        }
        public Gen get_child(Random rnd, List<Gen> par, Points mypnt)
        {
            List<int> TEMP = new List<int>();
            for (int i = 0; i < max; i++)
                if (rnd.Next() % 2 == 0)
                    TEMP.Add(par[0].get()[i]);
                else
                    TEMP.Add(par[1].get()[i]);
            Gen child = new Gen(TEMP, mypnt);
            child.mutation(rnd);

            return child;
        }
        public void new_generation(Random rnd, Points mypnt)
        {
            List<Gen> ngeneration = new List<Gen>();
            double nfitness = 0;
            for (int i = 0; i < 20; i++)
```

```
                ngeneration.Add(this.get_child(rnd, this.get_parents(rnd),
mypnt));

            for (int I = 0; I < 20; I++)
            {
                calcShFitness(rnd, mypnt, ngeneration[I]);
                nfitness += (double)ngeneration[I].get_fitness() / 20;
            }
            for (int i = 0; i < 20; i++)
                this.generation[i] += ngeneration[i];
            this.fitness = nfitness;
            this.shSort();
        }
        public double get_fitness()
        {
            return fitness;
        }
        public List<Gen> get()
        {
            return generation;
        }
        public Gen get(int i)
        {
            return generation[i];
        }
    }
}


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

namespace LakyDog
{
    public class Points
    {
        private List<Point> list;

        public Points()
        {
            list = new List<Point>();
            list.Add(new Point(200, 200));
            list.Add(new Point(200, 800));
            list.Add(new Point(800, 200));
            list.Add(new Point(800, 800));
        }
        public int getCount()
        {
            return list.Count;
        }
        public Point getPoint(int i)
        {
            return list[i];
        }
        public bool checkDistans(Gen obj)
        {
            for (int i = 0; i < list.Count; i++)
                if ((obj.corOfEnd()[0] - list[i].getPoints()[0] +
                     obj.corOfEnd()[1] - list[i].getPoints()[1]) < 50)
                    return true;

                return false;
        }
    }
```

```
}
```


600
500
400
300
200
100
0
-100
1
10
19
28
37
46
55
64
73
82
91
100
109
118
127
136
145
154
163
Ряд1
Ряд2


Form1
General Fitness
482,1
Next Step
Generation
6
Form1
General Fitness
361,1
Next Step
Generation
23
Form1
General Fitness
237,1
Next Step
Generation
58
Form1
General Fitness
6,9
Next Step
Generation
165