Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования «Национальный

исследовательский Нижегородский государственный

университет им. Н.И. Лобачевского»

**Институт информационных технологий, математики и механики**

Лабораторная работа №3 по курсу «Технология визуального программирования»:

«Реализация игры средствами Windows Forms и WPF»

Выполнил: студент группы 381808-1

Оганян Роберт Владимирович

Преподаватель: доцент кафедры ПИнж

Борисов Н. А.

Нижний Новгород

2019 г.

# Содержание

[Содержание 2](#_Toc4101358)

[Описание работы программы 3](#_Toc4101359)

[Приложение](#_Toc4101361).............................................................................................7

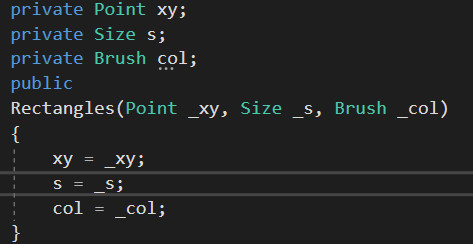
## Описание работы программы

Программа выполнена в среде разработки microsoft visual studio 2017 с помощью Windows Form и WPF (Windows Presentation Foundation**)**. Приложение написано для операционной системы Windows.

**Суть игры:**

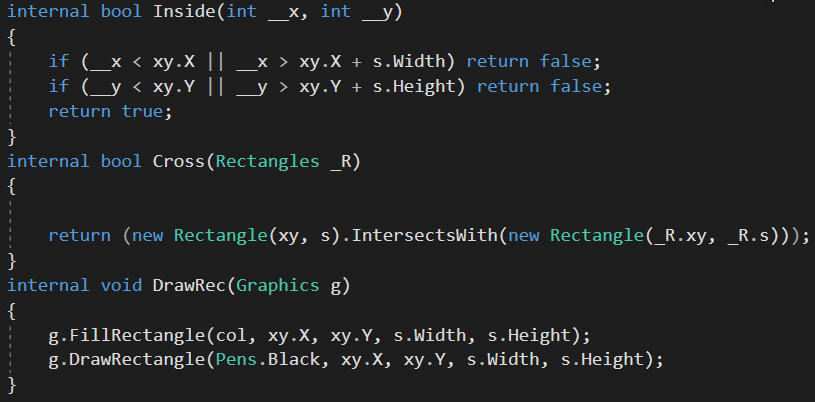
На экране случайным образом с заданным интервалом появляются разноцветные полоски фиксированного размера, которые могут перекрывать друг друга. Поле появления первых N (0.5N) полосок пользователю разрешается убирать их щелчками мышки, причем полоска убирается только если она НЕ перекрыта сверху другими полосками. При убирании всех полосок игра считается выигранной, при появлении 2\*N (N) полосок - проигранной.

**Реализация:**

Сам Form.cs и написанный мною класс Rectangles с тремя полями: координата, размер, цвет.

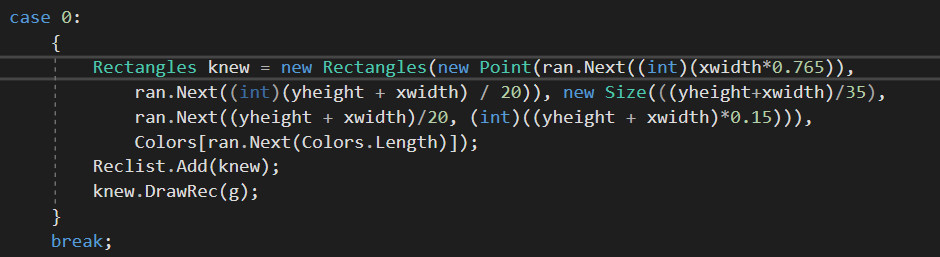
В классе есть 3 метода:

* Bool Inside - принимает координаты клика мыши и узнает, попадаем ли мы в прямоугольник
* Bool Cross - принимает Rectangles и узнает, пересекается ли переданный прямоугольник с текущим
* Void DrawRec - исходя из полей класса рисует прямоугольник

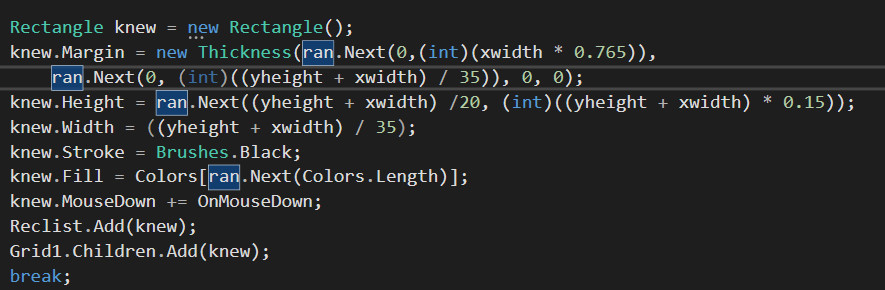


Теперь рассмотрим Form.cs:

Часть блока, рандомно генерирующего объект Rectangles выглядит так:

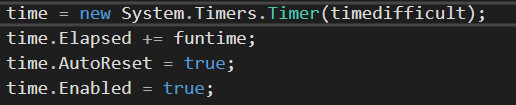


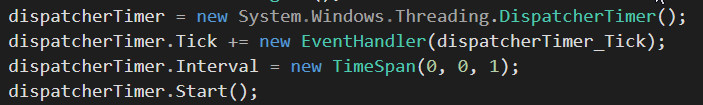
Для сравнения, в WPF это делается иначе. Например, координата задается отступом от всех рамок, а в самом WPF существует свой класс Rectangle, у которого и так есть методы, задающие размер (Height, Width), цвет(Fill), обводку(Stroke).



Таймер задается тоже по-разному:

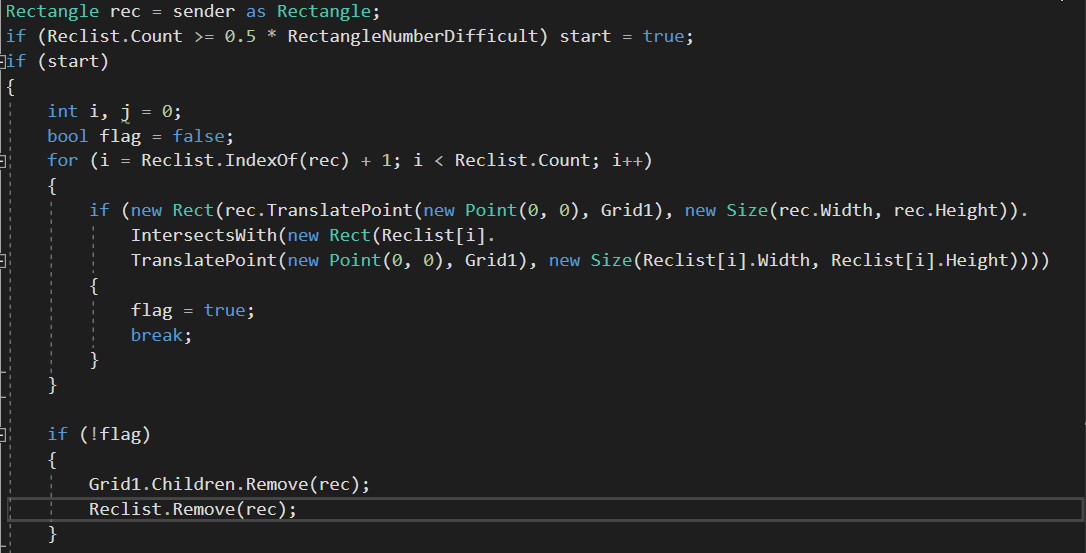
Windows Form:



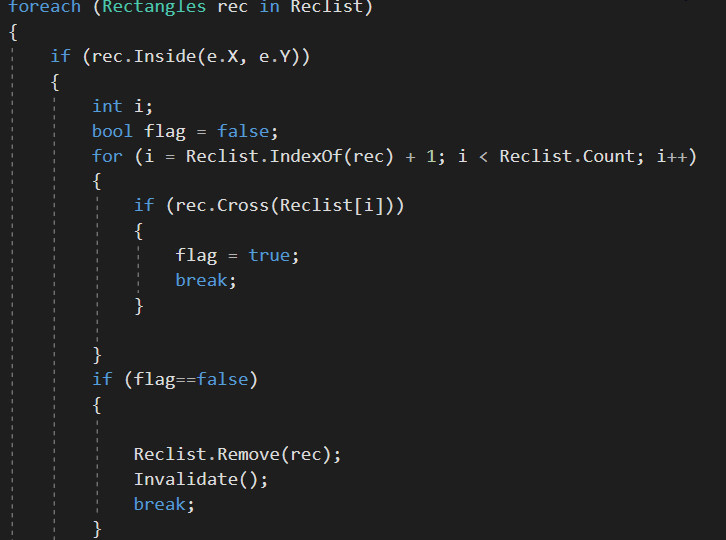
WPF :

Также WPF сам определяет объект Rectangle, так что мы просто используем метод IntersectWith, в отличии от Windows Form, где нужно вручную (написанным нами методом) сравнивать элементы массива Reclist по координатам:

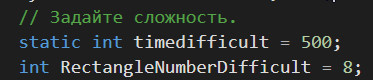
WPF:



Windows form:



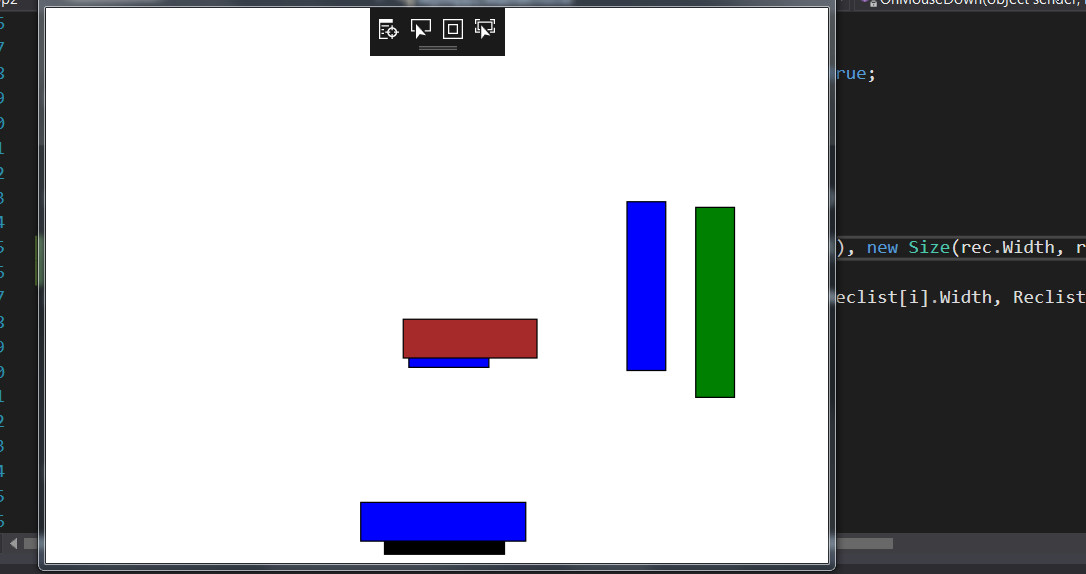
В дополнение скажу, что пользователь может регулировать сложность:



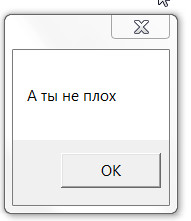
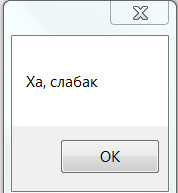
Чем меньше timedifficult, тем меньше задержка, т.е сложнее.

Чем меньше RectangleNumberDifficult, тем меньше порог прямугольников, при котором вы проиграете.

P.s: Задержка 1000 - легко. 500 - нужно попотеть. Если вы выигрываете на <400, то вы сверхчеловек.

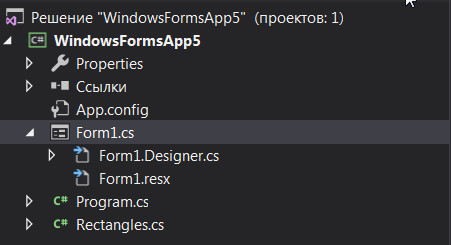


Проигрыш и выигрыш соответственно:



**Приложение**

**Windows Forms:**



Form1.cs:

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.Timers;

namespace WindowsFormsApp5

{

public partial class Form1 : Form

{

Random ran;

Brush[] Colors = { Brushes.Black, Brushes.White,Brushes.Yellow, Brushes.Red, Brushes.Blue,

Brushes.Pink, Brushes.Pink, Brushes.Green, Brushes.Brown };

List<Rectangles> Reclist;

bool start = false;

System.Timers.Timer time;

// Задайте сложность.

static int timedifficult = 500;

int RectangleNumberDifficult = 8;

// Задайте размер окна

int xwidth = 1280;

int yheight = 720;

public Form1()

{

InitializeComponent();

ran = new Random();

this.Width = xwidth;

this.Height = yheight;

Reclist = new List<Rectangles>();

time = new System.Timers.Timer(timedifficult);

time.Elapsed += funtime;

time.AutoReset = true;

time.Enabled = true;

}

private void funtime(object sender, ElapsedEventArgs e)

{

if (Reclist.Count > RectangleNumberDifficult)

{

time.Stop();

MessageBox.Show("Ха, слабак");

Application.Exit();

}

if (start == true && Reclist.Count == 0)

{

time.Stop();

MessageBox.Show("А ты не плох");

Application.Exit();

}

Graphics g = this.CreateGraphics();

switch (ran.Next(2)) // 0 - вертикальные прямоугольники, 1 - горизонтальные

{

case 0:

{

Rectangles knew = new Rectangles(new Point(ran.Next((int)(xwidth\*0.765)),

ran.Next((int)(yheight + xwidth) / 20)), new Size(((yheight+xwidth)/35),

ran.Next((yheight + xwidth)/20, (int)((yheight + xwidth)\*0.15))),

Colors[ran.Next(Colors.Length)]);

Reclist.Add(knew);

knew.DrawRec(g);

}

break;

case 1:

{

Rectangles knew = new Rectangles(new Point(ran.Next((int)(yheight + xwidth) / 20), ran.Next((int)(yheight \* 0.58))), new Size(ran.Next((yheight + xwidth) / 20, (int)((yheight + xwidth) \* 0.15)),((yheight + xwidth) / 35)), Colors[ran.Next(Colors.Length)]);

Reclist.Add(knew);

knew.DrawRec(g);

}

break;

}

}

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

{

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = this.CreateGraphics();

foreach (Rectangles rec in Reclist)

{

rec.DrawRec(g);

}

}

private void Form1\_MouseClick(object sender, MouseEventArgs e)

{

// if (!start) start = true;

if (Reclist.Count >= 0.5\*RectangleNumberDifficult) start = true;

if (start) {

foreach (Rectangles rec in Reclist)

{

if (rec.Inside(e.X, e.Y))

{

int i;

bool flag = false;

for (i = Reclist.IndexOf(rec) + 1; i < Reclist.Count; i++)

{

if (rec.Cross(Reclist[i]))

{

flag = true;

break;

}

}

if (flag==false)

{

Reclist.Remove(rec);

Invalidate();

break;

}

}

}

}

}

}

}

Rectangles.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

namespace WindowsFormsApp5

{

class Rectangles

{

private Point xy;

private Size s;

private Brush col;

public

Rectangles(Point \_xy, Size \_s, Brush \_col)

{

xy = \_xy;

s = \_s;

col = \_col;

}

internal bool Inside(int \_\_x, int \_\_y)

{

if (\_\_x < xy.X || \_\_x > xy.X + s.Width) return false;

if (\_\_y < xy.Y || \_\_y > xy.Y + s.Height) return false;

return true;

}

internal bool Cross(Rectangles \_R)

{

return (new Rectangle(xy, s).IntersectsWith(new Rectangle(\_R.xy, \_R.s)));

}

internal void DrawRec(Graphics g)

{

g.FillRectangle(col, xy.X, xy.Y, s.Width, s.Height);

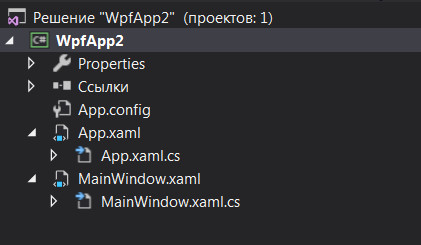
g.DrawRectangle(Pens.Black, xy.X, xy.Y, s.Width, s.Height);

}

}

}

WPF:



MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Threading;

namespace WpfApp2

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Random ran;

Brush[] Colors = { Brushes.Black,Brushes.Yellow, Brushes.Red, Brushes.Blue,

Brushes.Pink, Brushes.Pink, Brushes.Green, Brushes.Brown };

List<Rectangle> Reclist;

bool start = false;

DispatcherTimer dispatcherTimer;

// Задайте сложность.

static int timedifficult = 500;

int RectangleNumberDifficult = 8;

// Задайте размер окна

int xwidth = 640;

int yheight = 480;

public MainWindow()

{

InitializeComponent();

ran = new Random();

this.Width = xwidth;

this.Height = yheight;

Reclist = new List<Rectangle>();

dispatcherTimer = new System.Windows.Threading.DispatcherTimer();

dispatcherTimer.Tick += new EventHandler(dispatcherTimer\_Tick);

dispatcherTimer.Interval = new TimeSpan(0, 0, 1);

dispatcherTimer.Start();

}

private void dispatcherTimer\_Tick(object sender, EventArgs e)

{

if (Reclist.Count > RectangleNumberDifficult)

{

dispatcherTimer.Stop();

MessageBox.Show("Ха, слабак");

App.Current.MainWindow.Close();

}

if (start == true && Reclist.Count == 0)

{

dispatcherTimer.Stop();

MessageBox.Show("А ты не плох");

App.Current.MainWindow.Close();

}

switch (ran.Next(2)) // 0 - вертикальные прямоугольники, 1 - горизонтальные

{

case 0:

{

Rectangle knew = new Rectangle();

knew.Margin = new Thickness(ran.Next(0,(int)(xwidth \* 0.765)),

ran.Next(0, (int)((yheight + xwidth) / 35)), 0, 0);

knew.Height = ran.Next((yheight + xwidth) /20, (int)((yheight + xwidth) \* 0.15));

knew.Width = ((yheight + xwidth) / 35);

knew.Stroke = Brushes.Black;

knew.Fill = Colors[ran.Next(Colors.Length)];

knew.MouseDown += OnMouseDown;

Reclist.Add(knew);

Grid1.Children.Add(knew);

break;

}

case 1:

{

Rectangle knew = new Rectangle();

knew.Margin = new Thickness(ran.Next(0, (int)((yheight + xwidth) / 20)), ran.Next(0, (int)(xwidth \* 0.765)), 0, 0);

knew.Height = ((yheight + xwidth) / 35) ;

knew.Width = ran.Next((yheight + xwidth) / 20, (int)((yheight + xwidth) \* 0.15));

knew.Stroke = Brushes.Black;

knew.Fill = Colors[ran.Next(Colors.Length)];

knew.MouseDown += OnMouseDown;

Reclist.Add(knew);

Grid1.Children.Add(knew);

}

break;

}

}

private void OnMouseDown(object sender, MouseEventArgs e)

{

Rectangle rec = sender as Rectangle;

if (Reclist.Count >= 0.5 \* RectangleNumberDifficult) start = true;

if (start)

{

int i, j = 0;

bool flag = false;

for (i = Reclist.IndexOf(rec) + 1; i < Reclist.Count; i++)

{

if (new Rect(rec.TranslatePoint(new Point(0, 0), Grid1), new Size(rec.Width, rec.Height)).

IntersectsWith(new Rect(Reclist[i].

TranslatePoint(new Point(0, 0), Grid1), new Size(Reclist[i].Width, Reclist[i].Height))))

{

flag = true;

break;

}

}

if (!flag)

{

Grid1.Children.Remove(rec);

Reclist.Remove(rec);

}

}

}

}

}