**29. Анимация**

Задание 1. Работа с таймером

Решение:

namespace Task1

{

public partial class Form1 : Form

{

private double x1, y1, x2, y2;

private int alpha = 90;

private readonly Pen pen = new Pen(Color.DarkRed, 2);

private System.Windows.Forms.Timer timer1 = new System.Windows.Forms.Timer(); // Объявление и инициализация таймера

public Form1()

{

InitializeComponent();

// Начинаем таймер

timer1.Interval = 1000; // Установка интервала таймера (в миллисекундах)

timer1.Start();

// Добавляем обработчики событий

this.Paint += new PaintEventHandler(Form1\_Paint);

timer1.Tick += new EventHandler(timer1\_Tick);

}

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

{

//...

}

private void timer1\_Tick(object sender, EventArgs e)

{

alpha += -6; // увеличиваем угол на 6°

if (alpha >= 360) alpha -= 360; // начинаем с начала если прошли полный круг

double radian = alpha \* Math.PI / 180;

x1 = ClientSize.Width / 2;

y1 = ClientSize.Height / 2;

x2 = x1 + (int)(100 \* Math.Cos(radian));

y2 = y1 - (int)(100 \* Math.Sin(radian));

Invalidate(); // Принудительный вызов события Paint

}

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

{

Graphics g = e.Graphics;

g.DrawLine(pen, (float)x1, (float)y1, (float)x2, (float)y2);

}

}

}

Анализ результатов:

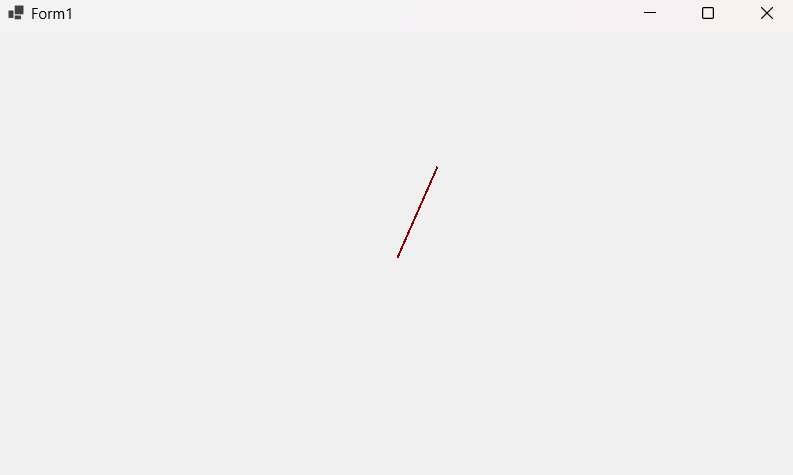


Рисунок 29.1 – Результат задания

Задание 2: Создание баннера

Решение:

namespace Task2

{

public partial class Form1 : Form

{

// переменные для работы с баннером

Bitmap banner;

// область вывода баннера

Rectangle rect;

// таймер

private System.Windows.Forms.Timer timer1 = new System.Windows.Forms.Timer();

// расстояние между баннерами

private int spacing = 50;

public Form1()

{

InitializeComponent();

try

{

// попытка загрузить баннер из файла

banner = new Bitmap("C:\\Users\\USER\\Desktop\\36ТП\\3 курс\\Практика Толочко\\Ivanov\_Practice29\\Practice29\\source\\baner.png");

}

catch (Exception ex)

{

// если произошла ошибка, выводим сообщение

MessageBox.Show("Ошибка загрузки баннера: " + ex.ToString(), "Ошибка");

this.Close();

return;

}

// определение области вывода баннера

rect = new Rectangle(0, 0, banner.Width, banner.Height);

// настройка таймера

timer1.Interval = 50;

timer1.Tick += Timer1\_Tick;

// включение двойной буферизации

this.DoubleBuffered = true;

// подписываемся на события

this.Load += Form1\_Load;

this.Paint += Form1\_Paint;

this.MouseMove += Form1\_MouseMove;

}

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

{

timer1.Start();

}

private void Timer1\_Tick(object sender, EventArgs e)

{

if (banner == null) return;

rect.X -= 1;

if (Math.Abs(rect.X) > rect.Width + spacing)

rect.X += rect.Width + spacing;

Invalidate(); // запрос на перерисовку формы

}

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

{

if (banner == null) return;

Graphics g = e.Graphics;

for (int i = 0; i <= this.ClientSize.Width / (rect.Width + spacing) + 1; i++)

{

g.DrawImage(banner, rect.X + i \* (rect.Width + spacing), rect.Y);

}

}

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

{

if ((e.Y < rect.Y + rect.Height) && (e.Y > rect.Y))

{

if (!timer1.Enabled)

timer1.Enabled = true;

}

else

{

if (timer1.Enabled)

timer1.Enabled = false;

}

}

}

}

Анализ результатов:

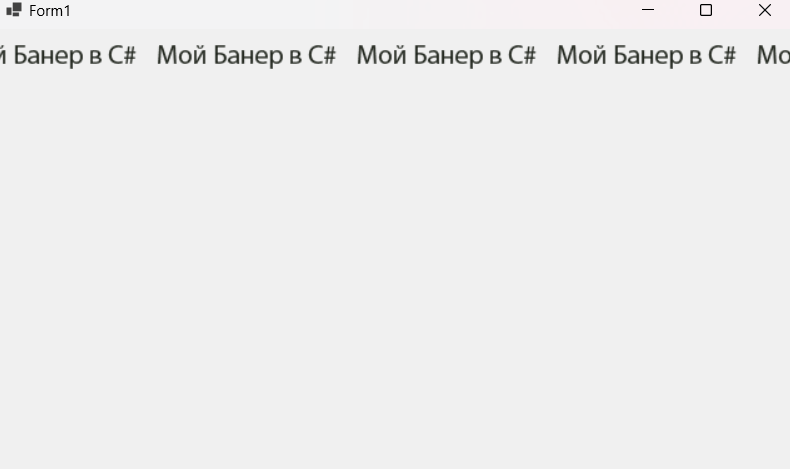


Рисунок 29.2 – Результат задания

Задание 3: Движение по траектории

Решение:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task3

{

public partial class Form1 : Form

{

private System.Windows.Forms.Timer timer = new System.Windows.Forms.Timer();

private float t = 0;

private int a = 100; // изменение размера и формы кривой

public Form1()

{

InitializeComponent();

this.DoubleBuffered = true;

this.Paint += new PaintEventHandler(Form1\_Paint);

timer.Interval = 50; // управление скоростью анимации

timer.Tick += new EventHandler(Timer\_Tick);

timer.Start();

}

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

{

t += 0.01f; // управление скоростью анимации

this.Invalidate(); // Перерисовать форму

}

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

{

if (t >= -1 && t <= 1)

{

float x = 3 \* a \* t / (1 + t \* t \* t) + this.ClientSize.Width / 2;

float y = 3 \* a \* t \* t / (1 + t \* t \* t) + this.ClientSize.Height / 2;

e.Graphics.FillEllipse(Brushes.Red, x, y, 10, 10); // Рисуем окружность

}

}

}

}

Анализ результатов:

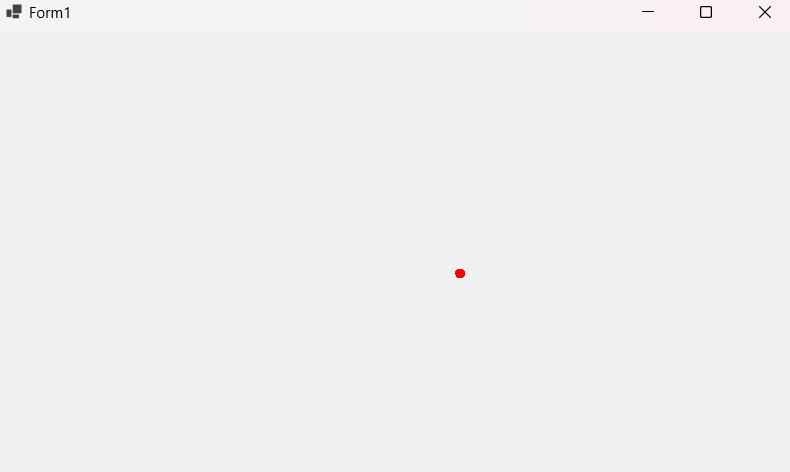


Рисунок 29.3 – Результат задания

Задание 4: Анимация полёта самолёта

Решение:

using System;

namespace Task4

{

public partial class Form1 : Form

{

// битовые образы: небо и самолет

System.Drawing.Bitmap sky, plane;

Graphics g; // рабочая графическая поверхность

// приращение координаты X,

// определяет скорость полета

int dx;

// область, в которой находится самолет

Rectangle rct;

// true - самолет скрывается в облаках

Boolean demo = true;

// генератор случайных чисел

System.Random rnd;

// таймер

private System.Windows.Forms.Timer timer2 = new System.Windows.Forms.Timer();

public Form1()

{

InitializeComponent();

try

{

// загрузка битовых образов из файлов

sky = new Bitmap("C:\\Users\\USER\\Desktop\\36ТП\\3 курс\\Практика Толочко\\Ivanov\_Practice29\\Practice29\\source\\sky.bmp"); // небо

plane = new Bitmap("C:\\Users\\USER\\Desktop\\36ТП\\3 курс\\Практика Толочко\\Ivanov\_Practice29\\Practice29\\source\\plane.bmp"); // самолет

// загрузить и задать фоновый рисунок формы

this.BackgroundImage = new Bitmap("C:\\Users\\USER\\Desktop\\36ТП\\3 курс\\Практика Толочко\\Ivanov\_Practice29\\Practice29\\source\\sky.bmp");

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString(),

"Полет",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

this.Close(); // закрыть приложение

return;

}

// сделать прозрачным фон вокруг объекта

plane.MakeTransparent();

// задать размер формы в соответствии

// с размером фонового рисунка

this.ClientSize =

new System.Drawing.Size(

new Point(BackgroundImage.Width,

BackgroundImage.Height));

// задать вид границы окна

this.FormBorderStyle =

System.Windows.Forms.FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

// g - графическая поверхность, на которой

// будем формировать рисунок.

// в качестве поверхности

// будем использовать BackgroundImage формы

g = Graphics.FromImage(BackgroundImage);

// инициализация генератора случ. чисел

rnd = new System.Random();

// исходное положение самолета

rct.X = -40;

rct.Y = 20 + rnd.Next(20);

rct.Width = plane.Width;

rct.Height = plane.Height;

/\*

скорость полета определяется периодом следования

сигналов от таймера (значение свойства Timer1.Interval)

и величиной приращения координаты по X

\*/

dx = 2; // скорость полета - 2 пикселя/тик\_таймера

timer2.Interval = 20;

timer2.Enabled = true;

timer2.Tick += new EventHandler(timer2\_Tick);

}

private void timer2\_Tick(object sender, EventArgs e)

{

// стираем изображение самолета путем копирования

// области фона на рабочую поверхность

g.DrawImage(sky, new Point(0, 0));

// изменяем положение самолета

if (rct.X < this.ClientRectangle.Width)

rct.X += dx;

else

{

// если достигли границы, задаем заново

// положение самолета

rct.X = -40;

rct.Y = 20 +

rnd.Next(this.ClientSize.Height - 40 - plane.Height);

// скорость полета от 2 до 5 пикселей/тик\_таймера

dx = 2 + rnd.Next(4);

}

// рисуем самолет на рабочей поверхности

// (фактически на поверхности формы),

// но для того, чтобы изменеия появились,

// надо инициировать обновление формы

g.DrawImage(plane, rct.X, rct.Y);

/\*

Метод Refresh инициирует перирисовку всей формы

Метод Invalidate позволяет иницировать перерисовку

только той области формы, которая указана

в качестве параметра метода.

\*/

if (!demo)

// обновить область формы,

// в которой находится объект

this.Invalidate(rct);

else

{

// если объект находится вне области,

// указанной в качестве параметра метода

// Invalidate, то от не будет виден

Rectangle reg =

new Rectangle(20, 20,

sky.Width - 40, sky.Height - 40);

// показать обновляемую область

g.DrawRectangle(Pens.Black,

reg.X, reg.Y, reg.Width - 1, reg.Height - 1);

this.Invalidate(reg); // обновить область

}

}

}

}

Анализ результатов:

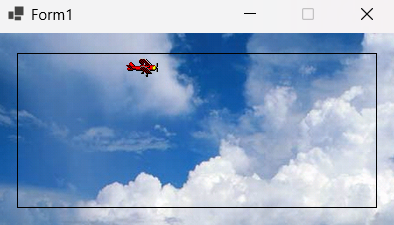


Рисунок 29.4 – Результат задания

Задание 5: Разработайте программу анимации взлета ракеты. Старт осуществляется по нажатию специальной «красной» кнопки.

Решение:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task5

{

public partial class Form1 : Form

{

private PictureBox rocketPictureBox;

private System.Windows.Forms.Timer timer;

private Button launchButton;

private int rocketPosition;

public Form1()

{

InitializeComponent();

// Создаем PictureBox для ракеты

rocketPictureBox = new PictureBox

{

Image = Image.FromFile("C:\\Users\\USER\\Desktop\\36ТП\\3 курс\\Практика Толочко\\Ivanov\_Practice29\\Practice29\\source\\Rocket.png"), // путь к изображению ракеты

SizeMode = PictureBoxSizeMode.AutoSize,

Location = new Point(this.ClientSize.Width / 2, this.ClientSize.Height)

};

// Создаем кнопку запуска

launchButton = new Button

{

Text = "Запуск!",

BackColor = Color.Red,

Location = new Point(this.ClientSize.Width / 2 - 200, this.ClientSize.Height - 50)

};

launchButton.Click += LaunchButton\_Click;

// Создаем таймер для анимации

timer = new System.Windows.Forms.Timer { Interval = 50 };

timer.Tick += Timer\_Tick;

// Добавляем контролы на форму

this.Controls.Add(rocketPictureBox);

this.Controls.Add(launchButton);

}

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

{

timer.Start();

}

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

{

rocketPosition += 5;

rocketPictureBox.Location = new Point(rocketPictureBox.Location.X, this.ClientSize.Height - rocketPosition);

if (rocketPictureBox.Location.Y < 0)

{

timer.Stop();

MessageBox.Show("Ракета запущена!");

}

}

}

}

Анализ результатов:

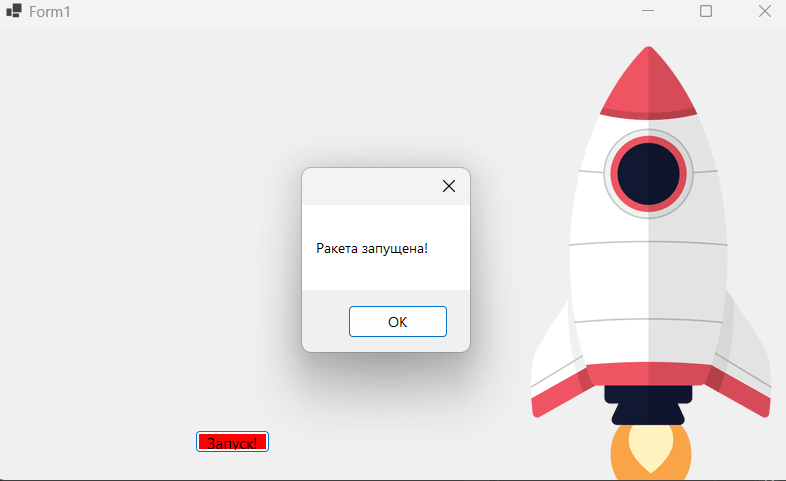


Рисунок 29.5 – Результат задания

Задание 6: Разработайте программу Clock моделирующую движение секундной стрелки.

Решение:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task6

{

public partial class Form1 : Form

{

private System.Windows.Forms.Timer timer;

public Form1()

{

InitializeComponent();

// Создаем таймер, который будет вызывать перерисовку каждую секунду

timer = new System.Windows.Forms.Timer { Interval = 1000 };

timer.Tick += new EventHandler(Timer\_Tick);

timer.Start();

}

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e);

int clockRadius = Math.Min(ClientSize.Width, ClientSize.Height) / 2;

Point clockCenter = new Point(ClientSize.Width / 2, ClientSize.Height / 2);

// Рисуем циферблат

e.Graphics.DrawEllipse(Pens.Black, clockCenter.X - clockRadius, clockCenter.Y - clockRadius, clockRadius \* 2, clockRadius \* 2);

// Рисуем секундную стрелку

double angle = DateTime.Now.Second \* 6 \* Math.PI / 180; // 6 градусов на секунду

int handRadius = (int)(clockRadius \* 0.9);

Point handEnd = new Point(clockCenter.X + (int)(handRadius \* Math.Sin(angle)), clockCenter.Y - (int)(handRadius \* Math.Cos(angle)));

e.Graphics.DrawLine(Pens.Red, clockCenter, handEnd);

}

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

{

this.Invalidate(); // Перерисовываем форму

}

}

}

Анализ результатов:

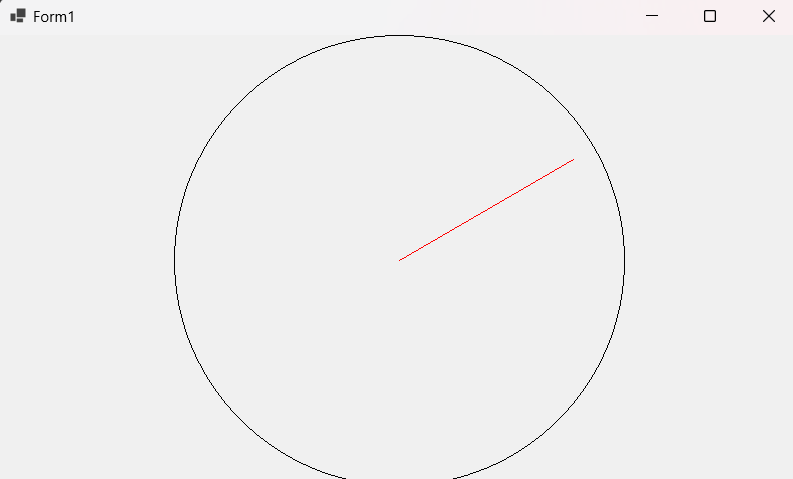


Рисунок 29.6 – Результат задания

Задание 7: Создайте программу, показывающую движение окружности по синусоиде.

Решение:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Task7

{

public partial class Form1 : Form

{

private System.Windows.Forms.Timer timer;

private float t = 0;

public Form1()

{

InitializeComponent();

// Создаем таймер, который будет вызывать перерисовку каждые 5 мс

timer = new System.Windows.Forms.Timer { Interval = 5 };

timer.Tick += new EventHandler(Timer\_Tick);

timer.Start();

}

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e);

// Рисуем ось X

e.Graphics.DrawLine(Pens.Black, 0, this.ClientSize.Height / 2, this.ClientSize.Width, this.ClientSize.Height / 2);

// Рисуем окружность, движущуюся по синусоиде

float x = t;

float y = (float)Math.Sin(t) \* this.ClientSize.Height / 4 + this.ClientSize.Height / 2;

e.Graphics.FillEllipse(Brushes.Red, x, y, 10, 10);

}

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

{

t += 0.1f; // управления скоростью анимации

if (t > this.ClientSize.Width)

t = 0;

this.Invalidate(); // Перерисовываем форму

}

}

}

Анализ результатов:

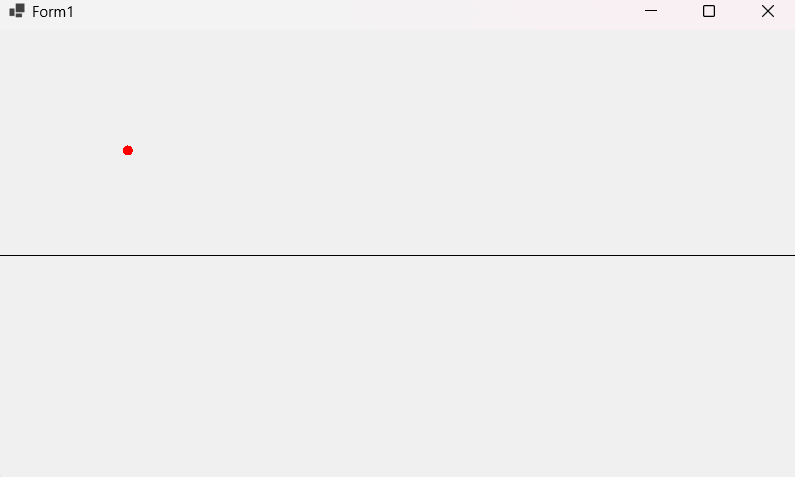


Рисунок 29.7 – Результат задания