**29 АНИМАЦИЯ**

Задание №1. Работа с таймером. Создать анимацию секундной стрелки.

Листинг программы:

public partial class Form1 : Form

{

private int x1, y1, x2, y2, R;

private double a;

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

public Form1()

{

InitializeComponent();

}

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

{

Graphics grap = e.Graphics;

grap.DrawLine(pen, x1, y1, x2, y2);

}

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

{

R = 150;

a = 0;

x1 = ClientSize.Width / 2;

y1 = ClientSize.Height / 2;

x2 = x1 + (int)(R \* Math.Cos(a));

y2 = y1 - (int)(R \* Math.Sin(a));

}

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

{

a -= 0.1;

x2 = x1 + (int)(R \* Math.Cos(a));

y2 = y1 - (int)(R \* Math.Sin(a));

Invalidate();

} }

Таблица 29.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 150, 0, x1, x2, y1, y2 | Animation |

Источник: собственная разработка

Анализ результатов представлен на рисунке 29.1.

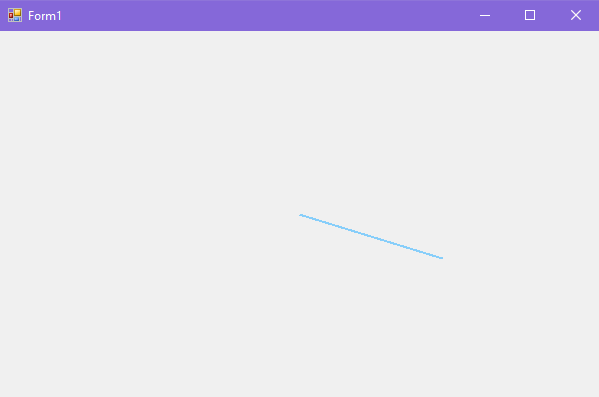


Рисунок 29.1 – Результат работы программы

Источник: собственная разработка

Задание №2. Создать анимацию бегущей строки.

Листинг программы:

public partial class Form1 : Form

{

Graphics grap;

Bitmap baner;

Rectangle rct;

public Form1()

{

InitializeComponent();

try

{

baner = new Bitmap("banner.jpg");

}

catch (Exception e)

{

MessageBox.Show("Ошибка загрузки файла баннера\n" + e.ToString(), "Баннер");

this.Close();

return;

}

rct.X = 0;

rct.Y = 0;

rct.Width = baner.Width;

rct.Height = baner.Height;

timer1.Interval = 50;

timer1.Enabled = true;

}

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

{

rct.X -= 1;

if (Math.Abs(rct.X) > rct.Width)

rct.X += rct.Width;

Invalidate();

}

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

{

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

{

if (timer1.Enabled != false)

timer1.Enabled = false;

}

else

{

if (timer1.Enabled != true)

timer1.Enabled = true;

}

}

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

{

grap = e.Graphics;

for (int i = 0; i < Convert.ToInt16(this.ClientSize.Width / rct.Width) + 1; i++)

grap.DrawImage(baner, rct.X + i \* rct.Width, rct.Y);

}

}

Таблица 29.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 0, 0, 50 | Animation |

Источник: собственная разработка

Анализ результатов представлен на рисунке 29.2.



Рисунок 29.2 – Результат работы программы

Источник: собственная разработка

Задание №3. Реализовать движение окружности по декартову листу.

Листинг программы:

public partial class Form1 : Form

{

private int x1, y1, x2, y2;

private double a, t, fi;

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

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

{

x1 = ClientSize.Width / 2;

y1 = ClientSize.Height / 2;

a = 150;

fi = -0.5;

t = Math.Tan(fi);

x2 = x1 + (int)((3 \* a \* t) / (1 + t \* t \* t));

y2 = y1 - (int)((3 \* a \* t \* t) / (1 + t \* t \* t));

}

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

{

Graphics g = e.Graphics;

g.DrawEllipse(pen, x2, y2, 20, 20);

}

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

{

fi += 0.01;

t = Math.Tan(fi);

x2 = x1 + (int)((3 \* a \* t) / (1 + t \* t \* t));

y2 = y1 - (int)((3 \* a \* t \* t) / (1 + t \* t \* t));

Invalidate();

}

public Form1()

{

InitializeComponent();

}

}

Таблица 29.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| x1, y1, 150, -0.5, t, x2, y2 | Animation |

Источник: собственная разработка

Анализ результатов представлен на рисунке 29.3.



Рисунок 29.3 – Результат работы программы

Источник: собственная разработка

Задание №4. Создать анимацию полёта самолёта.

Листинг программы:

public partial class Form1 : Form

{

Bitmap sky, plane;

Graphics g;

int dx;

Rectangle rct;

Boolean demo = true;

Random rand;

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

{

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

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

rct.X += dx;

else

{

rct.X = -40;

rct.Y = 20 +

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

dx = 2 + rand.Next(4);

}

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

if (!demo)

this.Invalidate(rct);

else

{

Rectangle reg = new Rectangle(20, 20, sky.Width - 40, sky.Height - 40);

g.DrawRectangle(Pens.Black, reg.X, reg.Y, reg.Width - 1, reg.Height - 1);

this.Invalidate(reg);

}

}

public Form1()

{

InitializeComponent();

try

{

sky = new Bitmap("sky.bmp");

plane = new Bitmap("plane.bmp");

this.BackgroundImage = new Bitmap("sky.bmp");

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString(), "Полет", MessageBoxButtons.OK, MessageBoxIcon.Error);

this.Close();

return;

}

plane.MakeTransparent();

this.ClientSize = new Size(new Point(BackgroundImage.Width,

BackgroundImage.Height));

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

g = Graphics.FromImage(BackgroundImage);

rand = new Random();

rct.X = -40;

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

rct.Width = plane.Width;

rct.Height = plane.Height;

dx = 2;

timer1.Interval = 20;

timer1.Enabled = true; } }

Таблица 29.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| rct, dx, g | Animation |

Источник: собственная разработка

Анализ результатов представлен на рисунке 29.4.

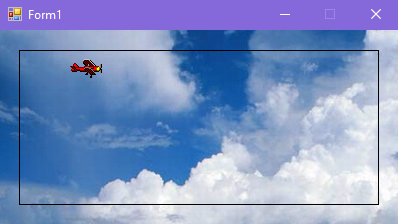


Рисунок 29.4 – Результат работы программы

Источник: собственная разработка

Задание №5. Разработайте программу анимации взлета ракеты. Старт осуществляется по нажатию специальной «красной» кнопки.

Листинг программы:

public partial class Form1 : Form

{

int y = 450;

public Form1() {

InitializeComponent();

}

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

{

Graphics RocketBody = e.Graphics;

Graphics RocketWing1 = e.Graphics;

Graphics RocketWing2 = e.Graphics;

Graphics RocketPorthole = e.Graphics;

SolidBrush RocketBodyColor = new SolidBrush(Color.DarkViolet);

SolidBrush RocketWing1Color = new SolidBrush(Color.Magenta);

SolidBrush RocketWing2Color = new SolidBrush(Color.Magenta);

SolidBrush RocketPortholeColor = new SolidBrush(Color.Aqua);

GraphicsPath gpBody = new GraphicsPath(FillMode.Winding);

GraphicsPath gpWing1 = new GraphicsPath(FillMode.Winding);

GraphicsPath gpWing2 = new GraphicsPath(FillMode.Winding);

GraphicsPath gpPorthole = new GraphicsPath(FillMode.Winding);

gpBody.AddPolygon(new Point[] { new Point(450, y), new Point(465, y + 100), new Point(485, y + 100), new Point(500, y), new Point(475, y - 30), new Point(450, y) });

gpWing1.AddPolygon(new Point[] { new Point(465, y + 100), new Point(445, y + 120), new Point(445, y + 90), new Point(460, y + 70), new Point(465, y + 100) });

gpWing2.AddPolygon(new Point[] { new Point(485, y + 100), new Point(505, y + 120), new Point(505, y + 90), new Point(490, y + 70), new Point(485, y + 100) });

gpPorthole.AddEllipse(465, y + 5, 20, 20);

RocketBody.FillPath(RocketBodyColor, gpBody);

RocketWing1.FillPath(RocketWing1Color, gpWing1);

RocketWing2.FillPath(RocketWing2Color, gpWing2);

RocketPorthole.FillPath(RocketPortholeColor, gpPorthole);

}

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

{

y--;

Invalidate();

}

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

{

timer1.Start();

}

}

Таблица 29.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 450 | Animation |

Источник: собственная разработка

Анализ результатов представлен на рисунке 29.5.



Рисунок 29.5 – Результат работы программы

Источник: собственная разработка

Задание №6. Создайте программу, показывающую движение окружности по синусоиде.

Листинг программы:

public partial class Form1 : Form

{

const float p = (float)Math.PI;

Timer timer = new Timer() { Interval = 100 };

GraphicsPath sinusoid;

int n;

public Form1()

{

InitializeComponent();

button1.Click += button1\_Click;

pictureBox1.Paint += pictureBox1\_Paint;

timer.Tick += timer\_Tick;

sinusoid = new GraphicsPath();

}

void timer\_Tick(object sender, EventArgs e)

{

if (++n >= sinusoid.PointCount) n = 0;

pictureBox1.Refresh();

}

void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

if (sinusoid.PointCount == 0) return;

PointF pt = sinusoid.PathPoints[n];

e.Graphics.ScaleTransform(1, -1);

e.Graphics.TranslateTransform(5, -pictureBox1.Height / 2);

e.Graphics.DrawPath(Pens.Black, sinusoid);

e.Graphics.FillEllipse(Brushes.Yellow, RectangleF.FromLTRB(pt.X - 15, pt.Y - 15, pt.X + 15, pt.Y + 15));

}

void button1\_Click(object sender, EventArgs e)

{

pictureBox2.Visible = true;

pictureBox3.Visible = true;

pictureBox4.Visible = true;

pictureBox5.Visible = true;

pictureBox6.Visible = true;

pictureBox7.Visible = true;

pictureBox8.Visible = true;

pictureBox9.Visible = true;

pictureBox10.Visible = true;

n = 0;

CreateSinusoid(sinusoid, 10f \* p, pictureBox1.ClientSize);

if (timer.Enabled)

timer.Stop();

else

timer.Start();

}

private void CreateSinusoid(GraphicsPath GraphPath, float MaxX, Size size)

{

GraphPath.Reset();

PointF[] points = new PointF[1] { PointF.Empty };

for (float i = 0; i <= MaxX; i += 0.4f)

{

points[points.GetUpperBound(0)] = new PointF(i, (float)Math.Sin(i));

Array.Resize<PointF>(ref points, points.Length + 1);

}

Array.Resize<PointF>(ref points, points.Length - 1);

GraphPath.AddCurve(points);

Matrix m = new Matrix();

m.Scale((float)(size.Width / MaxX), 0.4f \* size.Height);

GraphPath.Transform(m);

}

}

Таблица 29.6 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| n, p, 100 | Animation |

Источник: собственная разработка

Анализ результатов представлен на рисунке 29.6.

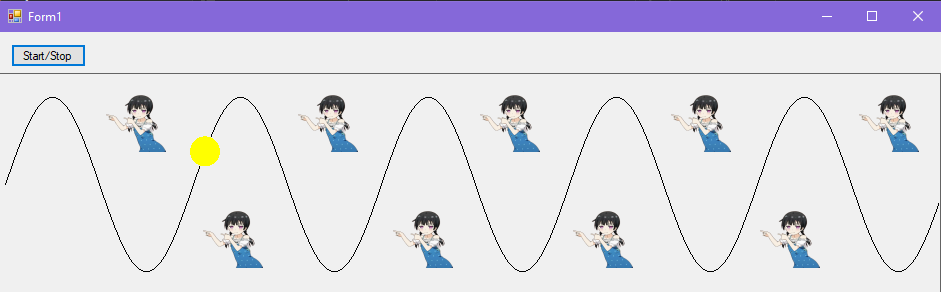


Рисунок 29.6 – Результат работы программы

Источник: собственная разработка