**Лабораторная работа № 13** Создание проектов с отрисовкой и движением изображений.

**Цель работы:** получить практические навыки разработки приложений с использованием команд графики, получить практический опыт организации движения фигуры при помощи таймера, передачи информации между программными модулями с использованием свойств.

**Вариант 1**

Маленький кружок должен «метаться» внутри формы, отталкиваясь от его ребер. Окончание работы – нажатие клавиши Esc.

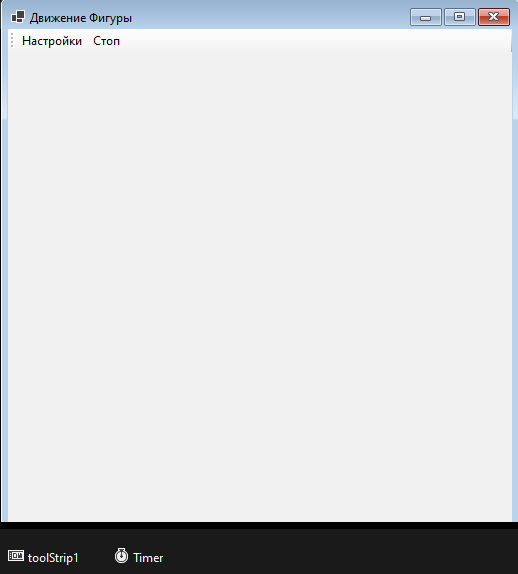
Настройки должны изменять цвет фигуры, вид фигуры (круг, квадрат или треугольник) и скорость движения фигуры.

***Задание 1:***

* Разработать программу движения фигуры по поверхности формы при помощи таймера. Движение фигуры достигается при помощи изменения координат по горизонтали и (или) по вертикали каждый тик таймера. Регулируя интервал работы таймера, можно управлять скоростью движения фигуры.
* Учесть, что при изменении размеров формы фигура все равно должна двигаться в заданных направлениях в пределах формы.
* При нажатии на кнопку «Стоп» фигура останавливается, надпись на кнопке меняется на «Старт». При повторном нажатии кнопки фигура возобновляет движение. Программа должна завершаться по нажатию клавиши Esc.

**Компоненты формы Form1.cs:**

settingsToolStripButton

****

toolStrip

Timer

stopToolStripButton

Рисунок 1 - Форма

Код Form1.cs:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WinFormsApp1

{

    public partial class Form1 : Form

    {

        private float x, y; // Координаты центра фигуры

        private float vx, vy; // Скорость по осям

        private int size = 20; // Размер фигуры (диаметр/сторона)

        private Color shapeColor = Color.Red;

        private string shapeType = "Circle"; // Тип фигуры: Circle, Square, Triangle

        private Random rand = new Random();

        private float speed = 5f; // Базовая скорость

        private bool isRunning = true; // Флаг для отслеживания состояния движения

        public Form1()

        {

            InitializeComponent();

            Timer.Interval = 16; // ~60 FPS

            Timer.Tick += Timer\_Tick;

            Timer.Start();

            // Начальная позиция в центре

            x = ClientSize.Width / 2;

            y = ClientSize.Height / 2;

            // Случайное начальное направление

            float angle = (float)(rand.NextDouble() \* 2 \* Math.PI);

            vx = speed \* (float)Math.Cos(angle);

            vy = speed \* (float)Math.Sin(angle);

            this.DoubleBuffered = true;

            this.KeyDown += Form1\_KeyDown;

            this.Resize += Form1\_Resize;

            // Начальная установка текста кнопки

            stopToolStripButton.Text = "Стоп";

        }

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

        {

            // Ограничиваем позицию при изменении размера формы

            x = Math.Max(size / 2, Math.Min(ClientSize.Width - size / 2, x));

            y = Math.Max(size / 2, Math.Min(ClientSize.Height - size / 2, y));

            Invalidate();

        }

        private void Form1\_KeyDown(object sender, KeyEventArgs e)

        {

            if (e.KeyCode == Keys.Escape)

            {

                Close(); // Завершение программы по Esc

            }

        }

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

        {

            if (!isRunning) return;

            // Обновляем позицию

            x += vx;

            y += vy;

            // Проверка столкновений с границами

            if (x - size / 2 < 0 || x + size / 2 > ClientSize.Width)

            {

                vx = -vx; // Отскок по X

                x = Math.Max(size / 2, Math.Min(ClientSize.Width - size / 2, x)); // Коррекция позиции

            }

            if (y - size / 2 < 0 || y + size / 2 > ClientSize.Height)

            {

                vy = -vy; // Отскок по Y

                y = Math.Max(size / 2, Math.Min(ClientSize.Height - size / 2, y)); // Коррекция позиции

            }

            Invalidate();

        }

        protected override void OnPaint(PaintEventArgs e)

        {

            base.OnPaint(e);

            e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

            using (SolidBrush brush = new SolidBrush(shapeColor))

            {

                if (shapeType == "Circle")

                {

                    e.Graphics.FillEllipse(brush, x - size / 2, y - size / 2, size, size);

                }

                else if (shapeType == "Square")

                {

                    e.Graphics.FillRectangle(brush, x - size / 2, y - size / 2, size, size);

                }

                else if (shapeType == "Triangle")

                {

                    PointF[] points = new PointF[]

                    {

                        new PointF(x, y - size / 2), // Верх

                        new PointF(x - size / 2, y + size / 2), // Низ лево

                        new PointF(x + size / 2, y + size / 2) // Низ право

                    };

                    e.Graphics.FillPolygon(brush, points);

                }

            }

        }

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

        {

            using (SettingsForm settings = new SettingsForm(shapeColor, speed, shapeType))

            {

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

                {

                    shapeColor = settings.SelectedColor;

                    speed = settings.Speed;

                    shapeType = settings.SelectedShape;

                    // Обновляем скорость с учетом текущего направления

                    float currentSpeed = (float)Math.Sqrt(vx \* vx + vy \* vy);

                    if (currentSpeed > 0)

                    {

                        vx = vx / currentSpeed \* speed;

                        vy = vy / currentSpeed \* speed;

                    }

                    else

                    {

                        float angle = (float)(rand.NextDouble() \* 2 \* Math.PI);

                        vx = speed \* (float)Math.Cos(angle);

                        vy = speed \* (float)Math.Sin(angle);

                    }

                    Invalidate();

                }

            }

        }

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

        {

            if (isRunning)

            {

                Timer.Stop();

                isRunning = false;

                stopToolStripButton.Text = "Продолжить";

            }

            else

            {

                Timer.Start();

                isRunning = true;

                stopToolStripButton.Text = "Стоп";

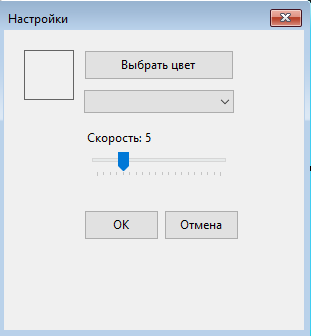
            }

        }

    }

}

**Компоненты формы SettingsForm.cs:**

****

shapeComboBox

colorButton

Label1

speedTrackBar

cancelButton

okButton

colorPreview

Рисунок 2 - Форма

Код SettingsForm.cs:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WinFormsApp1

{

    public partial class SettingsForm : Form

    {

        public Color SelectedColor { get; private set; }

        public float Speed { get; private set; }

        public string SelectedShape { get; private set; }

        public SettingsForm(Color currentColor, float currentSpeed, string currentShape)

        {

            InitializeComponent();

            SelectedColor = currentColor;

            Speed = currentSpeed;

            SelectedShape = currentShape;

            colorPreview.BackColor = currentColor;

            speedTrackBar.Value = (int)currentSpeed;

            speedLabel.Text = $"Скорость: {currentSpeed}";

            shapeComboBox.SelectedItem = currentShape;

        }

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

        {

            using (ColorDialog colorDialog = new ColorDialog())

            {

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

                {

                    SelectedColor = colorDialog.Color;

                    colorPreview.BackColor = SelectedColor;

                }

            }

        }

        private void SpeedTrackBar\_Scroll(object sender, EventArgs e)

        {

            Speed = speedTrackBar.Value;

            speedLabel.Text = $"Скорость: {Speed}";

        }

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

        {

            Speed = speedTrackBar.Value;

            SelectedShape = shapeComboBox.SelectedItem.ToString();

            DialogResult = DialogResult.OK;

            Close();

        }

    }

}

8.1 Протокол испытаний программы движения круга

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность программы обеспечивать движение круга по и изменять направление при достижении границы. | Запустить программу, наблюдать движение круга до достижения границы формы. | Круг движется по горизонтали с заданной скоростью, при достижении границы меняет направление движения. | Круг движется и меняет направление при столкновении с границей. Соответствует ожиданиям(рисунок 3). |
| Способность программы изменять цвет и | Нажать кнопку настроек, выбрать новый цвет | Цвет фигуры изменится | Цвет меняется. Соответствует ожиданиям(рисунок 4-5). |
| Способность программы останавливать и перезапускать движение по нажатию кнопки. | Нажать кнопку "Стоп" во время движения, затем нажать "продолжить" после остановки. | При нажатии "Стоп" движение останавливается, текст кнопки меняется на "Продолжить". | Движение останавливается, текст кнопки изменяется. Соответствует ожиданиям(рисунок 6). |
| Способность программы менять фигуру. | Нажать кнопку настроек, выбрать новую фигуру | Фигура изменится на выбранную. | Фигура меняется. Соответствует ожиданиям(рисунок 6-7) |
| Способность программы менять скорость. | Нажать кнопку настроек, выбрать новую скорость | Скорость изменится на выбранную | Скорость меняется Соответствует ожиданиям(рисунок 6-7) |

8.1 Результаты выполнения протокола испытаний

Итоги тестирования представлены на рисунках (рис. 3-9)

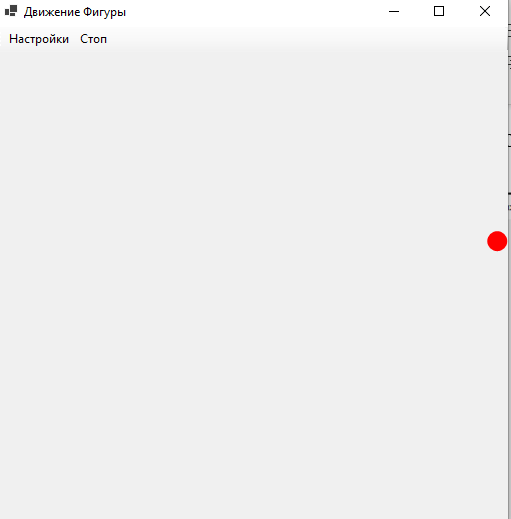
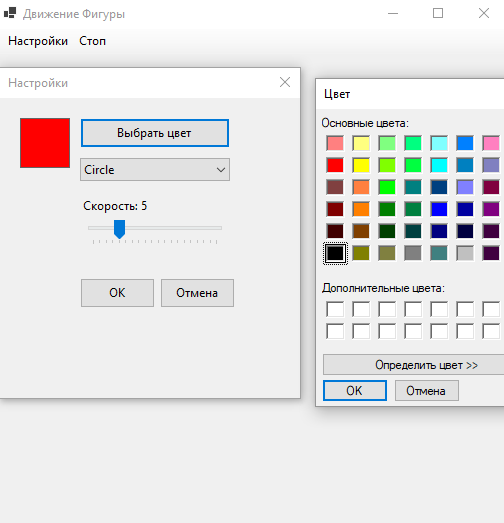
 

Рисунок 3 – Тест 001 Рисунок 4 – тест 002

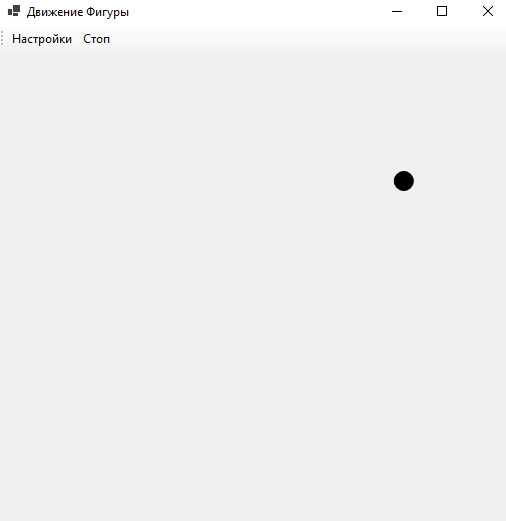
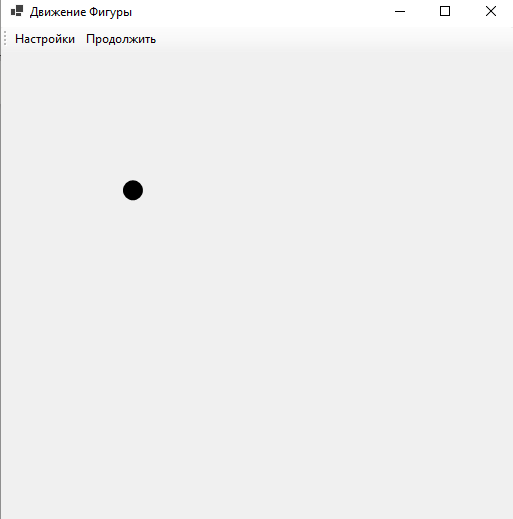
 

Рисунок 5 – Тест 002 Рисунок 4 – тест 003

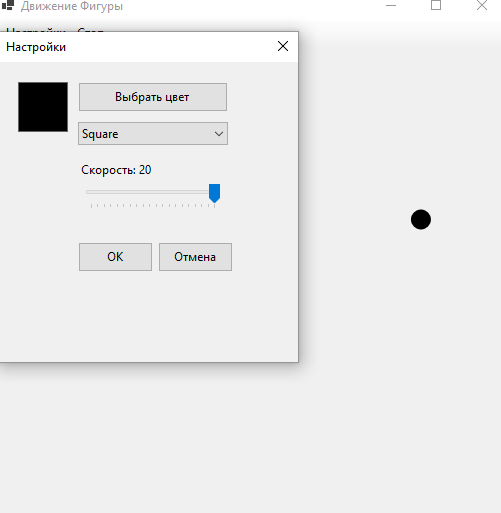
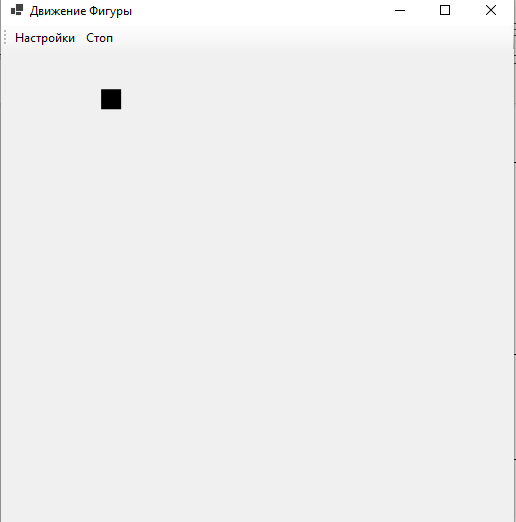
 

Рисунок 6 – Тест 004 Рисунок 7 – тест 004

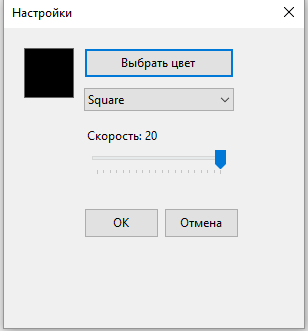
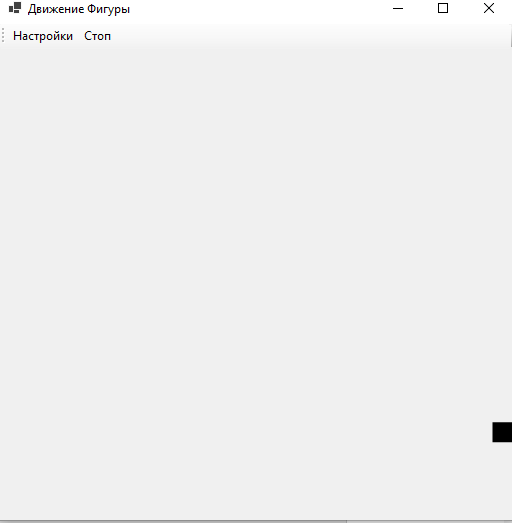
 

Рисунок 8 – Тест 005 Рисунок 9 – тест 005