

Лабораторная работа № 7

Тема: Разработка пользовательских элементов управления в WinForms

Цель: Освоить создание Custom Controls с использованием GDI+, настройку интеграции с дизайнером и систему событий.

Общие требования для обоих вариантов:

- Элемент должен наследоваться от базового класса Control.
- Должна быть включена двойная буферизация для устранения мерцания.
- Все графические инструменты (Pen, Brush) должны удаляться через using.
- Свойства должны иметь атрибуты [Category] и [Description].

Вариант №1.

Элемент «Кнопочный переключатель (Toggle Switch)»

Этот элемент представляет собой горизонтальный переключатель (как в iOS/Android).

Задание 1: Отрисовка и Свойства

Создайте свойство bool Checked (включен/выключен).

В OnPaint отрисуйте:

- Скругленный фон (фон серый, если Checked = false, и зеленый, если true).
- Белый круг (бегунок), который находится слева при false и справа при true.
- Включите сглаживание (AntiAlias).

Задание 2: Интерактивность

Реализуйте переключение состояния Checked при клике мышкой (переопределите OnClick или OnMouseDown).

Добавьте визуальный эффект: при наведении мыши (OnMouseEnter) цвет фона должен становиться чуть светлее.

Задание 3: События и Дизайнер

Создайте собственное событие CheckedChanged. Оповещайте форму каждый раз, когда переключатель меняет состояние.

Добавьте свойство Color AccentColor, которое позволит программисту через окно Properties выбрать цвет включенного состояния (вместо стандартного зеленого).

Вариант №2.

Элемент «Круговой индикатор выполнения (Round Progress)»

Этот элемент отображает прогресс в виде кольца.

Задание 1: Отрисовка и Свойства

Создайте свойства `int Value` (от 0 до 100) и `int Thickness` (толщина кольца).

В `OnPaint` отрисуйте:

- Серое кольцо (фон шкалы).
- Цветную дугу поверх него, длина которой соответствует значению `Value` (используйте метод `DrawArc`).
- В центре кольца отобразите текст с процентами (используйте `DrawString`).

Задание 2: Интерактивность

Реализуйте «быструю установку»: при клике мышкой в любую точку контроля, `Value` должен устанавливаться в 50 (или в случайное число) для проверки перерисовки.

Добавьте реакцию на мышь: при наведении на контрол толщина кольца `Thickness` должна временно увеличиваться на 2 пикселя.

Задание 3: События и Дизайнер

Создайте событие `ValueChanged`. Оно должно срабатывать только тогда, когда значение прогресса действительно изменилось.

Скройте системное свойство `Text` из окна `Properties`, используя атрибут `[Browsable(false)]`, так как текст внутри кольца формируется автоматически из `Value`.

Отчет должен содержать (см. образец):

- номер и тему лабораторной работы;
- фамилию, номер группы студента и вариант задания;
- скриншоты окна `Visual Studio` с исходным кодом программы и комментариями;
- скриншоты с результатами выполнения программ;
- пояснения, если необходимо;
- выводы.

Отчеты в формате **pdf** отправлять на email:

colledge20education23@gmail.com

Шпаргалка, содержит основные «кирпичики» кода, которые понадобятся для выполнения любого из вариантов лабораторной работы.

1. Подготовка класса и движка

```
using System;
using System.Collections.Generic;
using System.Text;

namespace classcontrol
{
    internal class MyCustomControl : Control
    {
        public MyCustomControl()
        {
            // 1. Включаем двойную буферизацию для плавности
            this.DoubleBuffered = true;

            // 2. Устанавливаем размер по умолчанию
            this.Size = new Size(150, 50);
        }
    }
}
```

2. Свойства и События (Логика)

Шаблон для создания «умного» свойства, которое уведомляет форму об изменениях.

```
private bool _checked = false;

[Category("Appearance")]
[Description("Определяет состояние переключателя")]
[DefaultValue(false)]
public bool Checked
{
    get => _checked;
    set
    {
        if (_checked != value) // Проверяем, изменилось ли значение
        {
            _checked = value;
            OnCheckedChanged(); // Вызываем событие
            Invalidate();       // Перерисовываем контрол
        }
    }
}

// Событие
public event EventHandler CheckedChanged;

protected virtual void OnCheckedChanged()
{
    CheckedChanged?.Invoke(this, EventArgs.Empty);
}

public MyCustomControl()
{
    // 1. Включаем двойную буферизацию для плавности
    this.DoubleBuffered = true;

    // 2. Устанавливаем размер по умолчанию
    this.Size = new Size(150, 50);
}
```

3. Отрисовка (GDI+)

Основные методы рисования, которые пригодятся в OnPaint.

```
protected override void OnPaint(PaintEventArgs e)
{
    base.OnPaint(e);
    Graphics g = e.Graphics;

    // Включаем высокое качество
    g.SmoothingMode = SmoothingMode.AntiAlias;
    g.TextRenderingHint =
System.Drawing.Text.TextRenderingHint.ClearTypeGridFit;

    // Прямоугольник всей области контроля с отступом
    Rectangle rect = new Rectangle(0, 0, Width - 1, Height - 1);

    // ПРИМЕР: Рисование с использованием using (обязательно!)
    using (Pen pen = new Pen(Color.Black, 2))
    using (SolidBrush brush = new SolidBrush(Color.SkyBlue))
    {
        // Заливка
        g.FillEllipse(brush, rect);
        // Контур
        g.DrawEllipse(pen, rect);
    }

    // ПРИМЕР: Вывод текста по центру
    TextRenderer.DrawText(g, "Text", Font, rect, ForeColor,
TextFormatFlags.HorizontalCenter | TextFormatFlags.VerticalCenter);
}
```

4. Интерактивность (Мышь)

Как менять состояние при взаимодействии.

```
private bool _isHovered = false;

protected override void OnMouseEnter(EventArgs e)
{
    base.OnMouseEnter(e);
    _isHovered = true;
    Invalidate(); // Перерисовать, чтобы применить цвет "наведения"
}

protected override void OnMouseLeave(EventArgs e)
{
    base.OnMouseLeave(e);
    _isHovered = false;
    Invalidate();
}

protected override void OnMouseDown(MouseEventArgs e)
{
    base.OnMouseDown(e);
    // Для первого варианта (Toggle):
    Checked = !Checked;
}
```

Полезные методы GDI+ для задач

Задача	Метод GDI+	Подсказка
Круг / Овал	<code>g.FillEllipse</code>	Передайте <code>Rectangle</code> в качестве границ.
Дуга (для Progress)	<code>g.DrawArc</code>	Требует <code>Pen</code> , границы, начальный угол и угол поворота.
Скругленный прямоугольник	<code>GraphicsPath</code>	Нужно добавить дуги и линии в путь и вызвать <code>g.FillPath</code> .
Цвет с прозрачностью	<code>Color.FromArgb(100, Color.Red)</code>	Первый параметр (0-255) — уровень прозрачности.