

Тема 6. Графика. Компоненты для работы с графикой.

Цель занятия:

Ознакомиться с основами работы с графикой в Windows Forms.

Учебные вопросы:

- 1. Основные классы для работы с графикой.**
- 2. Использование класса Graphics для рисования на форме или элементах управления.**
- 2. Компонент PictureBox.**

1. Основные классы для работы с графикой.

Базовые классы и структуры для работы с графикой предоставляет пространство имен `System.Drawing` в .NET.

Вот основные классы, которые оно включает:

1. Graphics. Основной класс для выполнения операций рисования. Он предоставляет методы для рисования линий, фигур, текста, изображений и других графических элементов.

Методы :

- DrawLine(Pen pen, Point pt1, Point pt2) — рисует линию.
- DrawRectangle(Pen pen, Rectangle rect) — рисует прямоугольник.
- FillRectangle(Brush brush, Rectangle rect) — закрашивает прямоугольник.
- DrawEllipse(Pen pen, Rectangle rect) — рисует эллипс.
- DrawString(string s, Font font, Brush brush, PointF point) — рисует текст.
- DrawImage(Image image, Point point) — отображает изображение.

Получение объекта Graphics :

- В обработчике события Paint через параметр PaintEventArgs.
- Через метод CreateGraphics() у элемента управления (например, формы).

2. Pen. Класс, используемый для определения стиля, цвета и толщины линий при рисовании.

3. Brush. Класс, используемый для заливки областей (например, прямоугольников или эллипсов).

Существует несколько типов кистей:

- SolidBrush: Однородная заливка одним цветом.
- HatchBrush: Заливка с текстурой (например, штриховкой).
- LinearGradientBrush: Градиентная заливка.
- TextureBrush: Заливка текстурой (например, изображением).

4. Font. Класс, используемый для определения шрифта, размера и стиля текста при рисовании.

5. Color. Класс, представляющий цвет. Может быть использован для задания цвета линий, заливки, текста и т.д.

6. Point, PointF, Size, Rectangle.

Классы для работы с геометрическими объектами:

Point: Представляет точку с целочисленными координатами.

PointF: Представляет точку с координатами типа float.

Size: Представляет размер (ширину и высоту).

Rectangle: Представляет прямоугольник с координатами и размерами.

7. Bitmap. Класс, представляющий растровое изображение. Используется для загрузки, сохранения и манипулирования изображениями.

8. PaintEventArgs. Класс, передаваемый в обработчик события Paint. Содержит объект Graphics, который можно использовать для рисования.

Пример использования :

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.DrawLine(new Pen(Color.Black), new Point(0, 0), new Point(100, 100));
}
```


Событие **Paint** в Windows Forms — это одно из ключевых событий, которое используется для рисования и перерисовки элементов управления (например, формы или пользовательского элемента управления). Оно вызывается каждый раз, когда необходимо обновить внешний вид элемента управления, например, при его первом отображении, изменении размеров, сворачивании/разворачивании окна или других действиях, которые требуют перерисовки.

Событие Paint возникает в следующих случаях:

- Первичная отрисовка : Когда форма или элемент управления отображается впервые.
- Обновление области экрана : Когда часть элемента управления становится видимой после того, как она была скрыта (например, если другое окно закрывается).
- Изменение размеров : Когда изменяются размеры элемента управления.
- Явный вызов метода `Invalidate()` : Когда программист явно запрашивает перерисовку элемента управления.
- Другие события системы : Например, при изменении темы оформления операционной системы.

2. Использование класса Graphics для рисования на форме или элементах управления.

Класс Graphics используется для рисования на форме или элементах управления в Windows Forms. Он предоставляет методы для рисования линий, фигур, текста и изображений.

Чтобы начать рисование, нужно получить объект Graphics. Это можно сделать несколькими способами:

Основные способы получения объекта Graphics:

- Событие Paint (рекомендуемый способ). Объект Graphics передаётся в обработчик события Paint через параметр PaintEventArgs.
- Метод Control.CreateGraphics(). Используется для временного рисования на элементах управления или форме.
- Создание Graphics из изображения. С помощью метода Graphics.FromImage().

Пример 1:

Рисование на форме через событие Paint

```
public partial class Form1 : Form
{
    Source: 1
    public Form1()
    {
        InitializeComponent();
        Text = "Рисование с использованием Graphics";
        Size = new Size(400, 300);
        Paint += MainForm_Paint;
    }

    Source: 1
    private void MainForm_Paint(object sender, PaintEventArgs e)
    {
        // Получение объекта Graphics
        Graphics g = e.Graphics;

        // Рисование линии
        g.DrawLine(Pens.Blue, 10, 10, 200, 10);

        // Рисование прямоугольника
        g.DrawRectangle(Pens.Red, 50, 50, 100, 50);

        // Заливка эллипса
        g.FillEllipse(Brushes.Green, 200, 100, 100, 50);

        // Рисование текста
        Font font = new Font("Arial", 14);
        g.DrawString("Пример текста", font,
            Brushes.Black, new PointF(10, 150));
    }
}
```

Пояснение:

```
Paint += MainForm_Paint;
```

Назначение обработчика события.

Paint: Это событие, определённое в классе Form (или другом элементе управления). Оно вызывается каждый раз, когда форма требует перерисовки. Например, при:

- Первоначальном отображении формы.
- Изменении размера окна.
- Вызове метода Invalidate() или Update().

+=: Это оператор подписки на событие. С его помощью вы добавляете метод, который будет вызван, когда событие Paint сработает.

Метод MainForm_Paint будет вызван, когда событие Paint произойдёт.

```
Graphics g = e.Graphics;
```

Получение объекта Graphics

`e` — это объект класса `PaintEventArgs`, переданный в метод-обработчик события `Paint`.

Свойство `Graphics` объекта `PaintEventArgs` содержит `Graphics`, связанный с формой или элементом управления, который нужно перерисовать.

Рисование линии

```
g.DrawLine(Pens.Blue, 10, 10, 200, 10);
```

g.DrawLine: Метод для рисования линии.

Pens.Blue: Синий "перо" (Pen) для определения цвета и толщины линии.

10, 10: Начальная точка линии (координаты X и Y).

200, 10: Конечная точка линии.

Рисование прямоугольника

```
g.DrawRectangle(Pens.Red, 50, 50, 100, 50);
```

`g.DrawRectangle`: Метод для рисования контура прямоугольника.

`Pens.Red`: Красное перо для контура.

`50, 50`: Координаты верхнего левого угла прямоугольника.

`100, 50`: Ширина и высота прямоугольника.

Заливка эллипса

```
// Заполнение эллипса  
g.FillEllipse(Brushes.Green, 200, 100, 100, 50);
```

`g.FillEllipse`: Метод для рисования заполненного эллипса.

`Brushes.Green`: Зелёная кисть (`Brush`) для заливки.

`200, 100`: Координаты верхнего левого угла прямоугольника, в который вписывается эллипс.

`100, 50`: Ширина и высота эллипса.

Рисование текста

```
Font font = new Font("Arial", 14);  
g.DrawString("Пример текста", font,  
    Brushes.Black, new PointF(10, 150));
```

`new Font("Arial", 14)`: Создаётся объект шрифта с использованием:

`"Arial"`: Название шрифта.

`14`: Размер шрифта (в пунктах).

`g.DrawString`: Метод для вывода текста.

`"Пример текста"`: Строка, которая будет нарисована.

`font`: Шрифт для текста.

`Brushes.Black`: Чёрная кисть для рисования текста.

`new PointF(10, 150)`: Точка (X, Y), где текст начнётся.

2. Класс Pen.

Класс Pen в C# используется для рисования контуров графических объектов, таких как линии, прямоугольники, эллипсы, и других фигур.

Этот класс является частью пространства имен System.Drawing.

1. Конструкторы Pen

Класс Pen предоставляет несколько способов создания объектов:

Простой конструктор с цветом пера:

```
Pen pen = new Pen(Color.Red);
```

Рисует линии красного цвета.

Толщина по умолчанию: 1 пиксель.

Конструктор с цветом и толщиной:

```
Pen pen = new Pen(Color.Blue, 5);
```

Рисует линии синего цвета с толщиной 5 пикселей.

Создание пера на основе кисти (Brush):

```
Brush brush = new SolidColorBrush(Color.Green);  
Pen pen = new Pen(brush, 3);
```

Используется кисть для определения цвета пера (например, градиент или текстура).

2. Основные свойства

Свойство	Описание
<code>Color</code>	Задаёт или получает цвет пера.
<code>Width</code>	Задаёт толщину пера в пикселях.
<code>DashStyle</code>	Определяет стиль линии (сплошная, пунктир, точечная и т. д.).
<code>StartCap</code>	Указывает стиль начального конца линии (например, плоский или закругленный).
<code>EndCap</code>	Указывает стиль конечного конца линии.
<code>DashPattern</code>	Массив чисел, определяющий пользовательский стиль пунктирной линии.
<code>Brush</code>	Задаёт кисть (<code>Brush</code>), используемую для рисования линий.
<code>LineJoin</code>	Определяет стиль соединений линий (например, острый угол или закругление).

Пример:

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
    // Получение объекта Graphics
    Graphics g = e.Graphics;

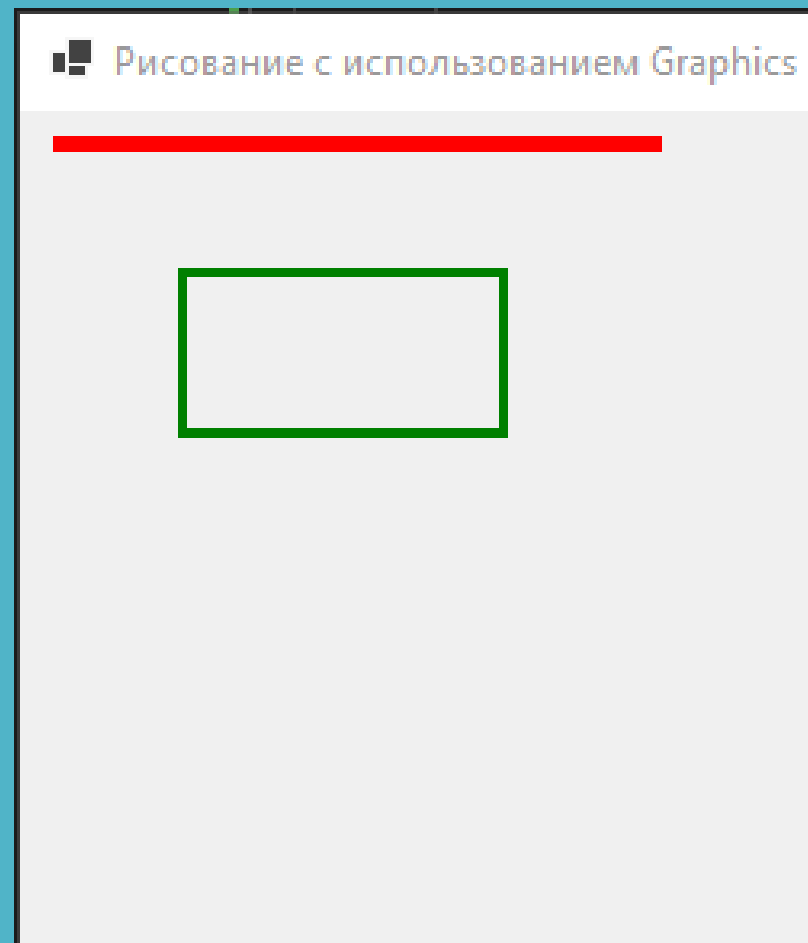
    // // Перо с толщиной 5 пикселей
    Pen thickPen = new Pen(Color.Red, 5);

    // Рисование линии
    g.DrawLine(thickPen, 10, 10, 200, 10);

    // // Перо с толщиной 5 пикселей
    Pen thickPen2 = new Pen(Color.Green, 3);

    // Рисование прямоугольника
    g.DrawRectangle(thickPen2, 50, 50, 100, 50);

    // Освобождение ресурсов пера
    thickPen.Dispose();
}
```



Метод `Dispose()` освобождает ресурсы, которые объект использует.

Конструкция `using` автоматически вызывает `Dispose()` после завершения работы с объектом.

Это наиболее предпочтительный способ работы с объектами, реализующими интерфейс `IDisposable` (включая `Pen`):

Пример:

Ссылка: 1

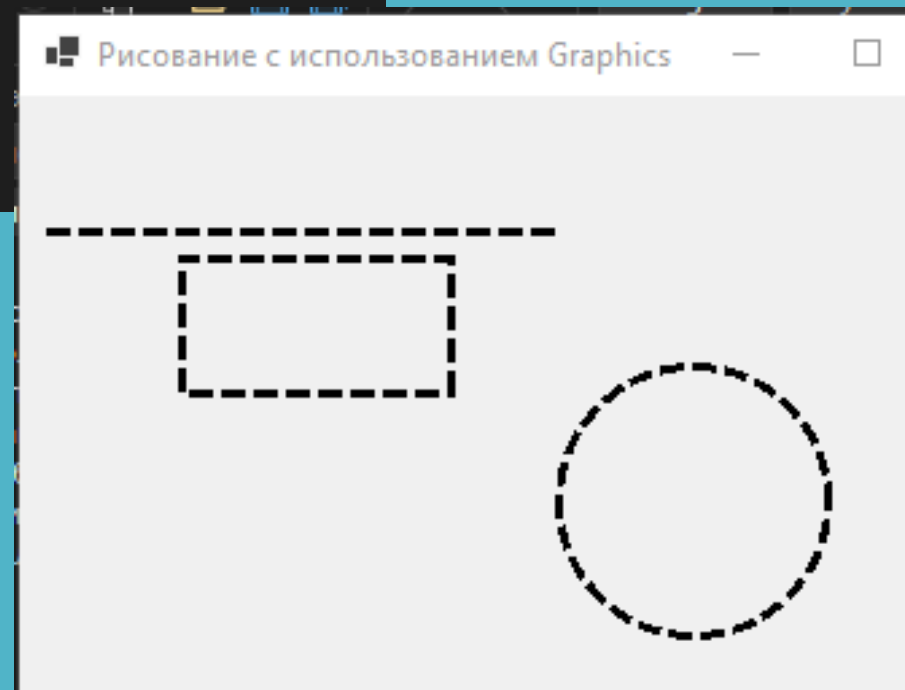
```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
    // Получение объекта Graphics
    Graphics g = e.Graphics;

    using (Pen thickPen = new Pen(Color.Blue, 5))
    {
        // Использование thickPen для рисования
        g.DrawLine(thickPen, 10, 10, 200, 10);
    }
    // После выхода из блока using метод Dispose()
    // будет вызван автоматически
}
```

Пример с пунктирной линией

```
using (Pen pen = new Pen(Color.Black, 3))  
{  
    // Пунктирный стиль  
    pen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  
    e.Graphics.DrawLine(pen, 10, 50, 200, 50);  
    g.DrawRectangle(pen, 60, 60, 100, 50);  
    g.DrawEllipse(pen, 200, 100, 100, 100);  
}
```

DrawEllipse: рисует только контур эллипса, используя Pen.



2. Класс Brush.

Класс Brush в пространстве имен System.Drawing является базовым классом для объектов, используемых для заливки различных графических объектов (например, прямоугольников, эллипсов, многоугольников и других фигур).

Brush не рисует линии или контуры, а именно заливает фигуры внутренним цветом или паттерном.

Основные производные классы от Brush:

- **SolidBrush.** Используется для заливки однотонным цветом.
- **HatchBrush.** Используется для заливки с использованием текстуры (штриховки), которая определяется с помощью паттерна и цвета.
- **LinearGradientBrush.** Используется для создания заливки с линейным градиентом между двумя точками.
- **TextureBrush.** Используется для заливки с использованием текстуры или изображения.
- **RadialGradientBrush.** Используется для создания радиальной заливки (с градиентом).

Пример:

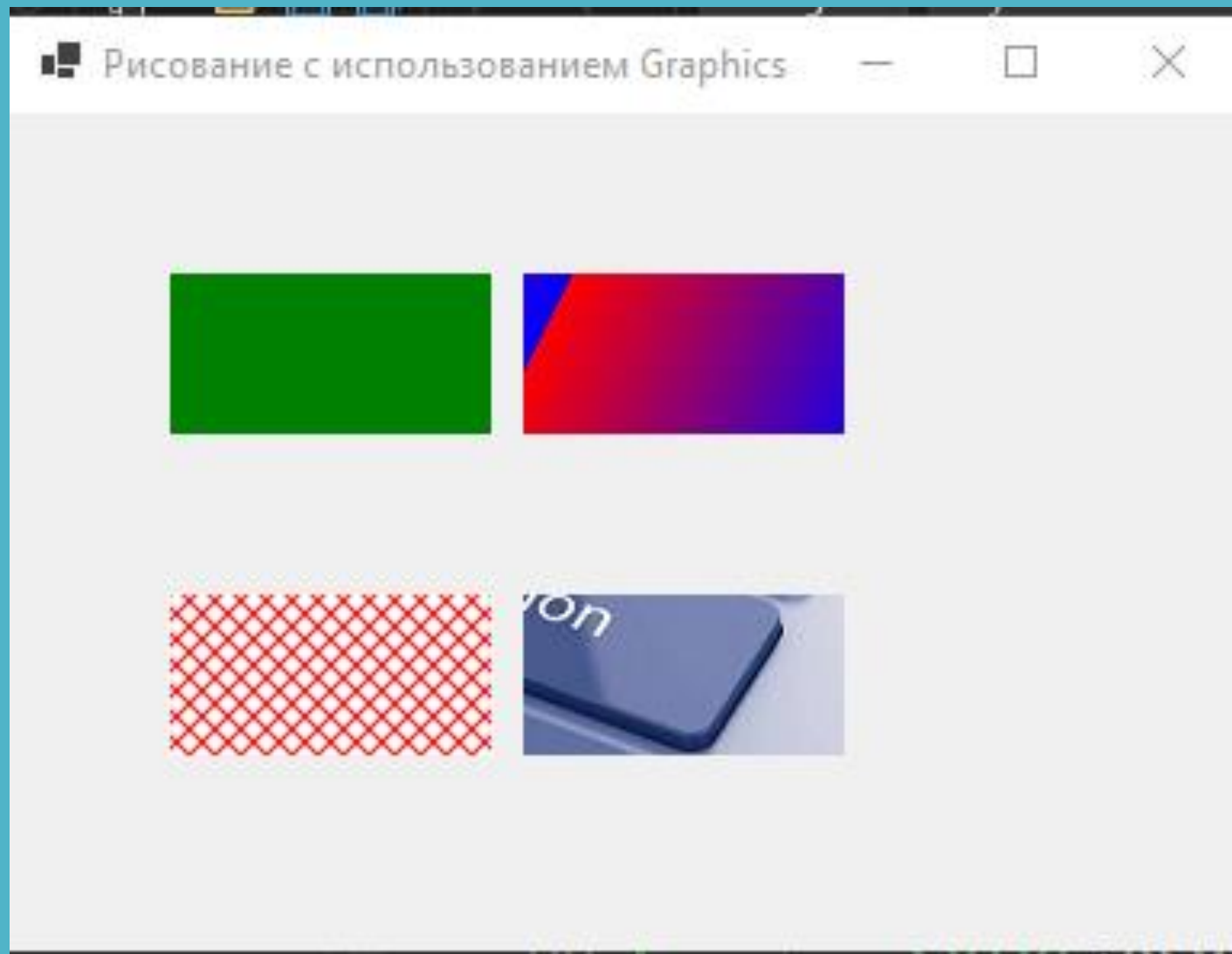
```
// Использование SolidBrush для заливки прямоугольника:  
using (SolidBrush brush = new SolidBrush(Color.Green))  
{  
    g.FillRectangle(brush, 50, 50, 100, 50);  
}  
  
// Использование HatchBrush для заливки прямоугольника с паттерном:  
using (HatchBrush brush = new HatchBrush(HatchStyle.DiagonalCross,  
    Color.Red, Color.White))  
{  
    g.FillRectangle(brush, 50, 150, 100, 50);  
}
```

```
// Использование LinearGradientBrush для заливки с линейным градиентом:
using (LinearGradientBrush brush = new LinearGradientBrush(
    new PointF(50, 50), new PointF(150, 100),
    Color.Red, Color.Blue))
{
    g.FillRectangle(brush, 160, 50, 100, 50);
}
```

```
// Использование TextureBrush для заливки с текстурой:
// Загрузка изображения для текстуры
string path = @"C:\Users\User\Desktop\csarp\Graph1\Graph1\pics\prg.jpg";
Image img = Image.FromFile(path);

// Создание кисти с текстурой
using (TextureBrush brush = new TextureBrush(img))
{
    // Заливка прямоугольника с текстурой
    g.FillRectangle(brush, 160, 150, 100, 50);
}
```

Результат:



2. Класс Color.

Класс Color из пространства имен System.Drawing используется для представления и работы с цветами в Windows Forms и других графических приложениях в C#.

Этот класс предоставляет методы, свойства и статические члены для создания и манипулирования цветами.

Основные характеристики:

- Класс Color является структурой (struct).
- Позволяет работать с цветами в формате ARGB (Alpha, Red, Green, Blue).
- Поддерживает создание пользовательских цветов, а также использование predetermined colors.

Конструкторы и методы создания объектов Color:

- Предопределенные цвета: Класс предоставляет множество предопределенных цветов, таких как Color.Red, Color.Blue, Color.Green и т. д.

```
Color redColor = Color.Red;    // Предопределенный цвет красный
Color blueColor = Color.Blue;  // Предопределенный цвет синий
```

- Создание цвета с использованием ARGB: Можно создать цвет, задав значения для альфа-канала (прозрачности), красного, зеленого и синего компонентов.

```
Color customColor = Color.FromArgb(255, 100, 150, 200);  
// Полностью непрозрачный цвет
```

Параметры:

A (Alpha): степень прозрачности (0 = полностью прозрачный, 255 = полностью непрозрачный).

R (Red): интенсивность красного цвета (0–255).

G (Green): интенсивность зеленого цвета (0–255).

B (Blue): интенсивность синего цвета (0–255).

- Создание цвета из RGB (по умолчанию альфа-канал = 255, полностью непрозрачно):

```
Color rgbColor = Color.FromArgb(100, 200, 150);
```

Класс Color предоставляет набор predetermined цветов, которые можно использовать для установки цвета различных элементов интерфейса, таких как фон, текст, границы и т.д.

Эти цвета доступны как статические свойства класса Color.

Вы можете найти полный список predetermined цветов [в официальной документации](#).

2. Компонент PictureBox.

Компонент PictureBox в Windows Forms используется для отображения изображений на форме.

Это простой и удобный способ работать с графическими файлами, такими как JPG, PNG, BMP, GIF и другими.

PictureBox предоставляет множество настроек для работы с изображениями, включая возможность изменения размера и расположения.

Основные свойства PictureBox:

Image

Позволяет задать или получить изображение, отображаемое в компоненте.

```
pictureBox1.Image = Image.FromFile("path_to_image.jpg");
```


SizeMode

Определяет, как изображение будет отображаться внутри PictureBox.

Варианты:

- Normal (по умолчанию): изображение отображается в натуральном размере.
- StretchImage: изображение растягивается или сжимается, чтобы заполнить компонент.
- AutoSize: размер PictureBox автоматически подстраивается под размер изображения.
- CenterImage: изображение отображается в центре компонента.
- Zoom: изображение масштабируется пропорционально, чтобы вписаться в размеры PictureBox.

```
pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
```

BorderStyle

Определяет стиль границы вокруг PictureBox.

Возможные значения:

- None: без границы.
- FixedSingle: одна тонкая граница.
- Fixed3D: объемная граница.

```
pictureBox1.BorderStyle = BorderStyle.Fixed3D;
```

InitialImage

Изображение, которое отображается, пока основное изображение загружается.

ErrorImage

Изображение, которое отображается, если основное изображение не удалось загрузить.

Основные методы PictureBox:

Load(string path)

Загружает изображение из указанного пути.

```
pictureBox1.Load("path_to_image.jpg");
```

LoadAsync(string path)

Асинхронная загрузка изображения. Удобно для загрузки изображений из сети.

```
pictureBox1.LoadAsync("http://example.com/image.jpg");
```

Пример:

```
private void MainForm_Load(object sender, EventArgs e)
{
    // Создаем компонент PictureBox
    PictureBox pictureBox = new PictureBox
    {
        Size = new Size(200, 200), // Размер PictureBox
        Location = new Point(10, 10), // Позиция на форме
        Image = Image.FromFile("image.jpg"), // Загружаем изображение
       SizeMode = PictureBoxSizeMode.Zoom, // Масштабируем изображение
        BorderStyle = BorderStyle.FixedSingle // Добавляем границу
    };

    // Добавляем PictureBox на форму
    this.Controls.Add(pictureBox);
}
```

Некоторые компоненты , которые позволяют работать с изображениями и графикой.

Panel. Панель может быть использована как контейнер для других элементов управления, но также может быть полезна для рисования пользовательской графики. Она поддерживает событие `Paint`, что позволяет использовать её для отрисовки графики.

Label. Хотя `Label` обычно используется для отображения текста, его можно настроить для отображения изображений через свойство `Image`.

Свойства :

- `Image`: Устанавливает изображение, которое будет отображаться рядом с текстом или вместо него.
- `ImageAlign`: Определяет выравнивание изображения относительно текста.

Button. Кнопка может быть настроена для отображения изображений через свойство `Image`. Это полезно, если вам нужно создать кнопку с иконкой или изображением.

Свойства :

- `Image`: Устанавливает изображение, которое будет отображаться на кнопке.
- `ImageAlign`: Определяет выравнивание изображения относительно текста.
- `TextImageRelation`: Определяет расположение текста относительно изображения.

SplitContainer. Этот компонент позволяет разделить форму на две панели, одна из которых может быть использована для отображения изображений, а другая — для управления ими (например, с помощью кнопок или ползунков).

Список литературы:

1. [Видеокурс C#.](#)
2. <https://metanit.com/sharp/windowsforms/3.1.php>
3. [Видеокурс C# Windows Forms](#)

Материалы лекций:

<https://github.com/ShViktor72/Education>

Обратная связь:

colledge20education23@gmail.com

Задание на дом:

Задание 1. Рисование примитивов

Создайте новый проект Windows Forms.

Добавьте на форму кнопку "Нарисовать фигуры". При нажатии на кнопку, должны отобразиться:

- Синяя горизонтальная линия длиной 200 пикселей.
- Красный прямоугольник (100x50 пикселей).
- Зеленый закрашенный круг (диаметром 80 пикселей).
- Черный текст "Графика в C#" шрифтом Arial, 16 px. с тенью
- Прямоугольник с линейным градиентом от синего к белому
- Желтый треугольник

Требования:

- Рисование должно выполняться в обработчике события Paint.
- Все фигуры должны быть нарисованы с помощью Graphics.

Задание 2. Отображение изображения

Создайте форму с компонентом PictureBox, который загружает изображение из файла.

Добавьте кнопку "Выбрать изображение", при нажатии которой пользователь может выбрать файл (JPG/PNG).

Загруженное изображение должно отображаться в PictureBox в режиме Zoom.

Требования:

- Использовать OpenFileDialog для выбора файла.
- Загружать изображение в PictureBox программно.
- Настроить SizeMode так, чтобы изображение вписывалось в PictureBox без искажений.