

ЛАБОРАТОРНАЯ РАБОТА 9. ОБРАБОТКА ИЗОБРАЖЕНИЙ

Цель лабораторной работы: изучить возможности Visual Studio по открытию и сохранению файлов. Написать и отладить программу для обработки изображений.

9.1. Отображение графических файлов

Обычно для отображения точечных рисунков, рисунков из метафайлов, значков, рисунков из файлов в формате BMP, JPEG, GIF или PNG используется объект **PictureBox**, т.е. элемент управления **PictureBox** действует как контейнер для картинок. Можно выбрать изображение для вывода, присвоив значение свойству **Image**. Свойство **Image** может быть установлено в окне **Свойства** или в коде программы, указывая на рисунок, который следует отображать.

Элемент управления **PictureBox** содержит и другие полезные свойства, в том числе: **AutoSize** определяющее, будет ли изображение растянуто в элементе **PictureBox**, и **SizeMode**, которое может использоваться для растягивания, центрирования или увеличения изображения в элементе управления **PictureBox**.

Перед добавлением рисунка к элементу управления **PictureBox** в проект обычно добавляется файл рисунка в качестве *ресурса*. После добавления ресурса к проекту можно повторно использовать его. Например, может потребоваться отображение одного и того же изображения в нескольких местах.

Необходимо отметить, что поле **Image** само является классом для работы с изображениями, у которого есть свои методы. Например, метод **FromFile** используется для загрузки изображения из файла. Кроме класса **Image** существует класс **Bitmap**, который расширяет возможности класса **Image** за счет дополнительных методов для загрузки, сохранения и использования растровых изображений. Так метод **Save** класса **Bitmap** позволяет сохранять изображения в разных форматах, а методы **GetPixel** и **SetPixel** позволяют получить доступ к отдельным пикселям рисунка.

9.2. Компоненты *OpenFileDialog* и *SaveFileDialog*

Компонент **OpenFileDialog** является стандартным диалоговым окном. Он аналогичен диалоговому окну «Открыть файл» операционной системы Windows.

Компонент **OpenFileDialog** позволяет пользователям просматривать папки личного компьютера или любого компьютера в сети, а также выбирать файлы, которые требуется открыть. Для вызова диалогового окна для выбора файла можно использовать метод **ShowDialog()** который возвращает **true** при корректном выборе.

Диалоговое окно возвращает путь и имя файла, который был выбран пользователем в специальном свойстве **FileName**.

9.3. Простой графический редактор

Создайте приложение, реализующее простой графический редактор. Функциями этого редактора должны быть: открытие рисунка, рисование поверх него простой кистью, сохранение рисунка в другой файл. Для этого создайте форму и разместите на ней элементы управления **button** и **picturebox** (рис 7.1).

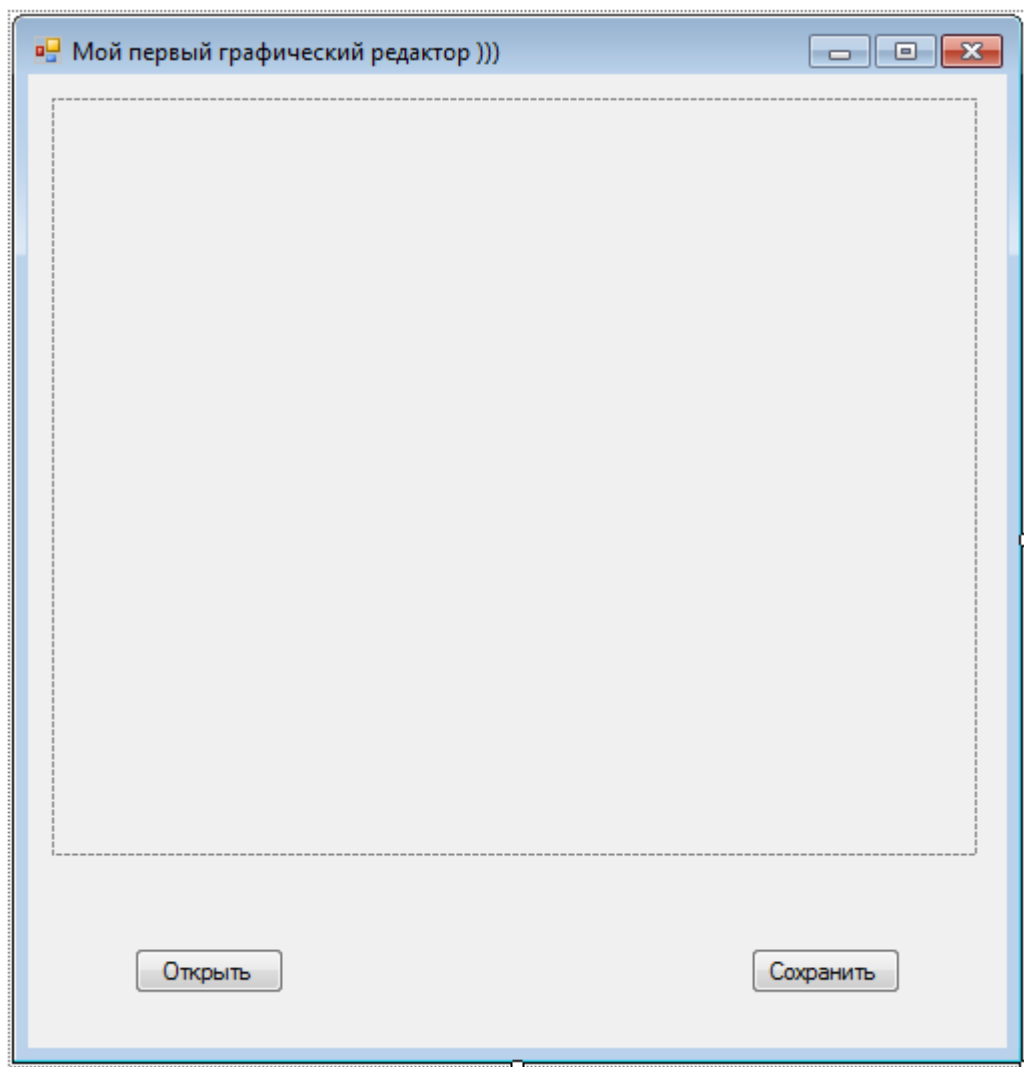


Рис. 7.1. Форма для графического редактора

В этом случае на понадобится из панели элементов размещать на форме компоненты диалоговых окон `OpenFileDialog` и `SaveFileDialog`. Эти элементы будут порождены динамически в ходе выполнения программы с помощью конструктора. Например так:

```
OpenFileDialog dialog = new OpenFileDialog();
```

Далее они будут вызываться с помощью метода **ShowDialog()**.

Для кнопок «Открыть» и «Сохранить» создайте свои обработчики события. Также создайте обработчик события **Load** для формы. Для элемента управления **picturebox1** создайте обработчики события **MouseDown**, **MouseMove**. Код приложения будет выглядеть следующим образом:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    { //Объявляем переменные доступные в каждом обработчике события
        private Point PreviousPoint, point; //Точка до перемещения курсора мыши и текущая точка
        private Bitmap bmp;
        private Pen blackPen;
        private Graphics g;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            blackPen = new Pen(Color.Black, 4); //подготавливаем перо для рисования
        }

        private void button1_Click(object sender, EventArgs e)
        { //открытие файла
            OpenFileDialog dialog = new OpenFileDialog(); //описываем и порождаем объект dialog класса
            OpenFileDialog
            //Задаем расширения файлов
            dialog.Filter = "Image files (*.BMP, *.JPG, *.GIF, *.TIF, *.PNG, *.ICO, *.EMF,
            *.WMF)|*.bmp;*.jpg;*.gif; *.tif; *.png; *.ico; *.emf; *.wmf";
            if (dialog.ShowDialog() == DialogResult.OK)//вызываем диалоговое окно и проверяем выбран ли
            файл
            {
                Image image = Image.FromFile(dialog.FileName); //Загружаем в image изображение из выбранного
                файла
                int width = image.Width;
                int height = image.Height;
                pictureBox1.Width = width;
                pictureBox1.Height = height;

                bmp = new Bitmap(image, width, height); //создаем и загружаем из image изображение в формате
                bmp

                pictureBox1.Image = bmp; //записываем изображение в формате bmp в pictureBox1
                g = Graphics.FromImage(pictureBox1.Image); //подготавливаем объект Graphics для рисования в
                pictureBox1

            }
        }

        private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
        { // обработчик события нажатия кнопки на мыши
            // записываем в предыдущую точку (PreviousPoint) текущие координаты
            PreviousPoint.X = e.X;
            PreviousPoint.Y = e.Y;
        }

        private void pictureBox1_MouseMove(object sender, MouseEventArgs e)

```

```

{ //Обработчик события перемещения мыши по pictureBox1
if (e.Button == MouseButton.Left) //Проверяем нажата ли левая кнопка мыши
{ //запоминаем в point текущее положение курсора мыши
point.X = e.X;
point.Y = e.Y;

//соединяем линией предыдущую точку с текущей
g.DrawLine(blackPen, PreviousPoint, point);

//текущее положение курсора мыши сохраняем в PreviousPoint
PreviousPoint.X = point.X;
PreviousPoint.Y = point.Y;
pictureBox1.Invalidate();//Принудительно вызываем перерисовку pictureBox1
}
}

private void button2_Click(object sender, EventArgs e)
{ //сохранение файла
SaveFileDialog savedialog = new SaveFileDialog();//описываем и порождаем объект savedialog
//задаем свойства для savedialog
savedialog.Title = "Сохранить картинку как ...";
savedialog.OverwritePrompt = true;
savedialog.CheckPathExists = true;
savedialog.Filter =
"Bitmap File(*.bmp)|*.bmp|" +
"GIF File(*.gif)|*.gif|" +
"JPEG File(*.jpg)|*.jpg|" +
"TIF File(*.tif)|*.tif|" +
"PNG File(*.png)|*.png";
savedialog.ShowHelp = true;
// If selected, save
if (savedialog.ShowDialog() == DialogResult.OK)//вызываем диалоговое окно и проверяем задано
ли имя файла
{
// в строку fileName записываем указанный в savedialog полный путь к файлу
string fileName = savedialog.FileName;
// Убираем из имени три последних символа (расширение файла)
string strFileExtn =
fileName.Remove(0, fileName.Length - 3);
// Сохраняем файл в нужном формате и с нужным расширением
switch (strFileExtn)
{
case "bmp":
bmp.Save(fileName, System.Drawing.Imaging.ImageFormat.Bmp);
break;
case "jpg":
bmp.Save(fileName, System.Drawing.Imaging.ImageFormat.Jpeg);
break;
case "gif":
bmp.Save(fileName, System.Drawing.Imaging.ImageFormat.Gif);
break;
case "tif":
bmp.Save(fileName, System.Drawing.Imaging.ImageFormat.Tiff);
break;
case "png":
bmp.Save(fileName, System.Drawing.Imaging.ImageFormat.Png);
break;
default:
break;
}
}
}
}

```

```
}  
}
```

Далее добавим в проект кнопку для перевода изображения в градации серого цвета:

```
private void button3_Click(object sender, EventArgs e)  
{ //циклы для перебора всех пикселей на изображении  
for (int i = 0; i < bmp.Width; i++)  
for (int j = 0; j < bmp.Height; j++)  
{  
int R = bmp.GetPixel(i, j).R; //извлекаем в R значение красного цвета в текущей точке  
int G = bmp.GetPixel(i, j).G; //извлекаем в G значение зеленого цвета в текущей точке  
int B = bmp.GetPixel(i, j).B; //извлекаем в B значение синего цвета в текущей точке  
int Gray = (R + G + B)/3; // высчитываем среднее арифметическое трех каналов  
Color p = Color.FromArgb(255, Gray, Gray, Gray); //переводим int в значение цвета. 255 - по-  
казывает степень прозрачности. остальные значения одинаковы для трех каналов R,G,B  
bmp.SetPixel(i, j, p); //записываем полученный цвет в текущую точку  
}  
Refresh(); //вызываем функцию перерисовки окна  
}
```

Данный код демонстрирует возможность обращения к отдельным пикселям. Цвет каждого пикселя хранится в модели RGB и состоит из трех составляющих: красного, зеленого и синего цвета, называемых каналами. Значение каждого канала может варьироваться в диапазоне от 0 до 255.

9.4. Выполнение индивидуального задания

Добавте в приведенный графический редактор свои функции в соответствии с вариантом. Для вызова новой функции:

- добавить кнопку;
- добавить меню и соответствующий пункт;
- добавить тулбар и иконку.

1. Расширьте приложение путем добавления возможности выбора пользователем цвета и величины кисти.

2. Разработайте функцию, добавляющую на изображение 1000 точек с координатами заданными случайным образом. Цвет, также, задается случайным образом.

3. Создайте функцию, переводящую изображение в черно-белый формат.

4. Разработайте функцию, оставляющую на изображении только один из каналов (R,G,B). Канал выбирается пользователем.

5. Создайте функцию, выводящую на изображение окружность. Центр окружности совпадает с центром изображения. Все точки вне окружности закрашиваются черным цветом. Все точки внутри окружности остаются неизменными. Радиус окружности задается пользователем.

6. Вывести на изображение треугольник. Все точки вне треугольника закрашиваются синим цветом. Все точки внутри треугольника не изменять.

7. Вывести на изображение ромб. Все точки вне ромба переводятся в градации серого цвета. Все точки внутри ромба закрашиваются зеленым цветом.
8. Разработайте функцию, которая выводит на изображение черные горизонтальные линии в каждой четной строке.
9. Разработайте функцию, которая выводит на изображение черные вертикальные линии в каждом столбце с номером, кратным заданному числу.
10. Создайте функцию, разбивающую изображение на четыре равные части. В каждой оставте значение только одного канала R, G и B, а в четвертой выведите градации серого цвета.
11. Разработайте функцию, заменяющую все точки синего цвета на точки красного цвета.
12. Создайте функцию, инвертирующую изображение в градациях серого цвета в негатив.
13. Создайте функцию, изменяющую яркость изображения путем прибавления или уменьшения заданной пользователем величины к каждому каналу.
14. Создайте функцию, переводящую изображение в черно-белый формат в соответствии с пороговым значением, которое ввел пользователь.
15. Разработайте функцию для создания эффекта мозаики. При этом изображения разбивается на прямоугольные фрагменты, в каждом из которых выбирается цвет средней точки и этим же цветом закрашивается весь фрагмент.
16. Разработайте функцию, разбиющую изображение на фрагменты, в каждом из которых остается только один из каналов (R, G, B).
17. Для каждого прямоугольного фрагмента заданного размера вычислить среднюю яркость по случайно взятому каналу (R, G, B) и вычесть ее половину из значения соответствующего канала каждой точки.
18. Инвертировать значение указанного канала для всех точек изображения.
19. Преобразовать каждую точку изображения путем вычисления нового значения каждого канала как средней величины соседних точек.
20. Преобразовать каждую точку изображения путем вычисления нового значения каждого канала как разности между значениями двух других каналов.
21. Преобразовать каждую точку выделенного (указанного) фрагмента изображения путем сброса до нуля наименьшего значения из трех каналов, например: (123, 25, 87) -> (123, 0, 87).
22. Разбить изображение на полосы заданной ширины и каждой из них инвертировать значение одного случайно взятого канала.
23. Первые 16 точек каждой строки закрасить ее усредненным цветом.
24. Закрасить диагональные n точек изображения инверсией усредненного цвета строки и столбца
25. Поменять местами четные и нечетные точки каждой нечетной строки.