Лабораторная работа №8

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МЕХАТРОНИКИ ТРАНСПОРТНЫХ СРЕДСТВ

Кафедра информатики

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению лабораторных работ по дисциплине «Программирование» для студентов специальности 6.050201 “Системная инженерия”

Разработчик - доцент кафедры информатики
кандидат технических наук,
старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2011

Лабораторная работа №8
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений по обработке изображений.

Цель работы – исследовать возможностей интегрированной среды разработки Visual C# и получить практические навыки по созданию приложений по обработке изображений.

1. Теоретические сведения

Для формирования сложных изображений используют битовые образы. Би товый образ — это небольшая картинка, которая находится в оперативной памяти компьютера Так как битовый образ находится в оперативной памяти, то его можно очень быстро вывести на экран. Именно поэтому битовые обра зы используются для формирования картинок в играх.

1.1. Возможности класса Bitmap

Создать битовый образ (объект Bitmap) можно путем загрузки из файла (bmp, gif, jpg, png и др.), ресурса или путем копирования из другого графического объекта Image. Для этой цели используется класс Bitmap, расширяющий возможности класса Image за счет дополнительных методов для загрузки, сохранения и использования растровых изображений.

Класс Bitmap инкапсулирует точечный рисунокGDI+, состоящий из данных пикселей графического изображения и атрибутов рисунка. Объект Bitmap используется для работы с изображениями, определяемыми данными пикселей.

Можно создать изображения из файлов, потоков и других источников, используя один из конструкторов Bitmap (см. табл. 1), и сохранить их в поток или файловую систему с помощью метода Save(). Изображения прорисовываются на экране или в памяти с помощью метода DragImage() объекта Graphics.

Таблица 1. Конструкторы класса Bitmap

Имя	Описание
Bitmap(Image)	Инициализирует новый экземпляр класса Bitmap из указанного существующего изображения.
Bitmap(String)	Инициализирует новый экземпляр класса Bitmap из указанного файла.
Bitmap(Image, Size)	Инициализирует новый экземпляр класса Bitmap из указанного существующего изображения, масштабированного до заданного размера.
Bitmap(Int32, Int32)	Инициализирует новый экземпляр класса Bitmap с заданным размером.
Bitmap(Image, Int32, Int32)	Инициализирует новый экземпляр класса Bitmap из указанного существующего изображения, масштабированного до заданного размера.
Bitmap(Int32, Int32, Graphics)	Инициализирует новый экземпляр класса Bitmap с заданным размером и с разрешением указанного объекта Graphics.
Bitmap(Int32, Int32, PixelFormat)	Инициализирует новый экземпляр класса Bitmap заданными значениями размера и формата.

Основные свойства объектов класса Bitmap представлены в таблице 2, а методы – в таблице 3.

Таблица 2. Свойства класса Bitmap

Имя	Описание
Height	Получает высоту объекта Image в пикселях.
PhysicalDimension	Получает ширину и высоту данного изображения.
PixelFormat	Получает формат пикселей объекта Image.
Size	Получает ширину и высоту данного изображения в пикселях.
Width	Получает ширину объекта Image в пикселях.

Таблица 3. Методы класса Bitmap

Имя	Описание
Clone()	Создает точную копию данного объекта Image.
Clone(Rectangle, PixelFormat)	Создает копию раздела этого объекта Bitmap, определяемого структурой Restangle и с указанным цифровым обозначением PixelFormat.
Clone(RectangleF, PixelFormat)	Создает копию части этого изображения Bitmap, определяемого заданным перечислением PixelFormat.
Dispose()	Освобождает все ресурсы, используемые Image.
GetBounds()	Получает границы изображения в заданных единицах измерения.
GetHBitmap()	Создает объект точечного рисунка GDI из данного изображения Bitmap.
GetHBitmap(Color)	Создает объект точечного рисунка GDI из данного изображения Bitmap.
GetPixel()	Получает цвет указанной точки в этом изображении Bitmap.
MakeTransparent()	Делает стандартно прозрачные цвета прозрачными для данного изображения Bitmap.
MakeTransparent(Color)	Делает заданный цвет прозрачным для данного изображения Bitmap.
RotateFlip()	Поворачивает, зеркально отражает, либо поворачивает и зеркально отражает объект Image.
Save(String)	Сохраняет объект Image в указанный файл или поток.
Save(String, ImageFormat)	Сохраняет объект Image в указанный файл в указанном формате.
SetPixel()	Задает цвет указанного пикселя в этом изображении Bitmap.

Существующее изображение очень просто нарисовать на экране. Сначала необходимо создать объект Bitmap с помощью конструктора растрового изображения Bitmap(String), который принимает имя файла. Этот конструктор поддерживает изображения нескольких форматов, включая BMP, GIF, JPEG, PNG, TIFF. После создания объекта Bitmap, передайте такой объект Bitmap в метод DrawImage() объекта Graphics. Метод DrawImage() перегружен, поэтому он поддерживает различные варианты передачи аргументов (см. табл. 4).

Таблица 4. Варианты метода DrawImage()

Имя	Описание
DrawImage(Image, Point)	Рисует заданный объект Image в заданном месте, используя его исходный фактический размер. Объект Image для рисования. Структура Point, представляющая местоположение верхнего левого угла изображения
DrawImage(Image, Point[])	Рисует заданный объект Image в заданном месте, используя указанные форму и размер. Массив из трех структур Point, определяющих параллелограмм.
DrawImage(Image, Rectangle)	Рисует заданный объект Image в заданном месте, используя

	указанный размер. Структура Rectangle, которая задает расположение и размер создаваемого изображения.
DrawImage(Image, Int32, Int32)	Рисует заданное изображение, используя его исходный фактический размер, в месте, задаваемом парой координат. Int32 - координата X верхнего левого угла выводимого изображения. Int32 - координата Y верхнего левого угла выводимого изображения.
DrawImage(Image, Point[], Rectangle, GraphicsUnit)	Рисует заданную часть указанного объекта Image в заданном месте, используя заданный размер. Структура Rectangle, которая задает часть объекта Image для рисования. Член перечисления GraphicsUnit, задающий единицы измерения: World - Задает в качестве единицы измерения единицу мировой системы координат; Display - Задает единицу измерения устройства отображения. Обычно это точки для видеодисплеев и 1/100 дюйма для принтеров; Pixel – Задает в качестве единицы измерения точка устройства; Point – Задает в качестве единицы измерения пункт (1/72 дюйма); Inch – Задает в качестве единицы измерения дюйм; Document – Задает в качестве единицы измерения единицу документа (1/300 дюйма); Millimeter – Задает в качестве единицы измерения миллиметр.
DrawImage(Image, Rectangle, Rectangle, GraphicsUnit)	Рисует заданную часть указанного объекта Image в заданном месте, используя заданный размер. 2-й параметр-структура Rectangle, которая задает расположение и размер создаваемого изображения. Изображение масштабируется по размерам прямоугольника. 3-й параметр - структура Rectangle, которая задает часть объекта Image для рисования.
DrawImage(Image, Int32, Int32, Rectangle, GraphicsUnit)	Рисует часть изображения в заданном месте.
DrawImage(Image, Int32, Int32, Int32, Int32)	Рисует заданный объект Image в заданном месте, используя указанный размер.
DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit)	Рисует заданную часть указанного объекта Image в заданном месте, используя заданный размер.

Метод DrawImage(Image, Point) рисует заданный объект Image в заданном месте, используя его исходный фактический размер. Объект Image для рисования. Структура Point, представляющая местоположение верхнего левого угла изображения.

Объект Image сохраняет значение ширины, выраженной в точках, и значение горизонтального разрешения (в точках на дюйм). Фактическая ширина рисунка, измеряемая в дюймах, вычисляется делением ширины рисунка в точках на горизонтальное разрешение. Например, рисунок, имеющий ширину 216 точек, при горизонтальном разрешении 72 точки на дюйм будет иметь фактическую ширину 3 дюйма. Аналогичным образом вычисляются высота в точках и фактическая высота.

Этот метод создает изображение с его фактическими размерами. Таким образом, размер изображения в дюймах не зависит от разрешения (в точках на дюйм) устройства отображения. Например, пусть ширина рисунка равна 216 точек, а горизонтальное разрешение — 72 точки на дюйм. Если этот метод вызывается для отображения рисунка на устройстве, разрешение которого равно 96

точкам на дюйм, то ширина созданного рисунка (в точках) будет равна (216/72)*96 = 288.

```
private void DrawImagePoint(PaintEventArgs e)
{
   Image newImage = Image.FromFile("SampImag.jpg");
   Point ulCorner = new Point(100, 100);
   e.Graphics.DrawImage(newImage, ulCorner);
}
```

1.2. Приемы работы с изображениями

1.2.1. Рисование существующего точечного рисунка на экране

Пример 1. В этом примере создается объект Bitmap на основе файла в формате JPEG и на экране рисуется соответствующее растровое изображение с верхним левым углом в точке (20, 50) (рис. 1).

```
private void button1_Click(object sender, EventArgs e)
    {
       Graphics im = pictureBox1.CreateGraphics();
       Bitmap bitmap = new Bitmap("auto1.jpg");
       im.DrawImage(bitmap, 20, 50);
    }
```

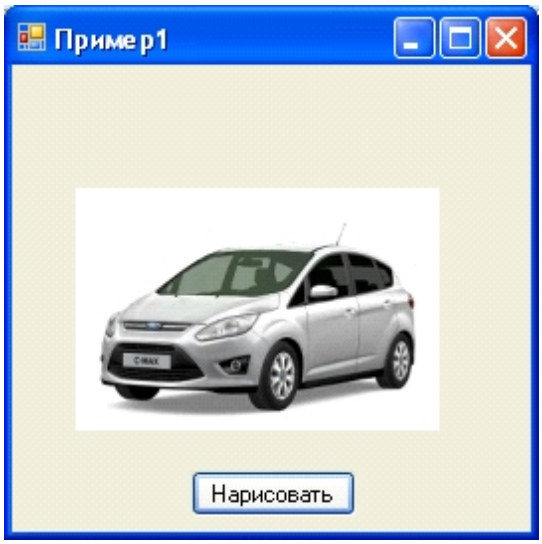

Рис. 1. Результат выполнения примера 1.

Вместо параметров x и y в инструкции вызова метода DrawImage() можно ука зать структуру типа Point.

Для битового образа можно задать прозрачный цвет. Точки рисунка, цвет ко торых совпадает с "прозрачным", при выводе битового образа не отобража ются. Прозрачный цвет задает метод MakeTransparent(). В инструкции вызова метода необходимо указать цвет, который следует рассматривать как про зрачный. Например, инструкция

```
bitmap. MakeTransparent(Color.Magenta);
```

задает, что прозрачным является цвет Magenta (пурпурный).

В инструкции вызова метода MakeTransparent() цвет можно не указывать. В этом случае прозрачным будет цвет, которым окрашена левая нижняя точ ка битового образа.

1.2.2. Обрезка и масштабирование изображений в GDI+

Метод DrawImage() класса Graphics позволяет рисовать и размещать векторные и растровые изображения.

Пример 2. Один из вариантов метода DrawImage() принимает объекты Bitmap и Rectangle. Прямоугольник задает область, в которой должно быть нарисовано изображение. Если размер прямоугольника назначения отличается от размеров исходного изображения, изображение масштабируется, чтобы соответствовать прямоугольнику назначения. В приведенном ниже примере кода демонстрируется три способа рисования одного изображения: рисование без масштабирования, рисование с увеличением и рисование со сжатием (рис. 2).

```
private void button1_Click(object sender, EventArgs e)
    {
     Graphics im = pictureBox1.CreateGraphics();
     Bitmap myBitmap = new Bitmap("auto1.jpg");
     Rectangle expansionRectangle = new Rectangle(210, 10,
                                                  (int)(myBitmap.Width*1.2),
                                                  (int)(myBitmap.Height*1.2));

    Rectangle compressionRectangle = new Rectangle(450, 10, myBitmap.Width / 2,
                                                   myBitmap.Height / 2);
     im.DrawImage(myBitmap, 10, 10);
     im.DrawImage(myBitmap, expansionRectangle);
     im.DrawImage(myBitmap, compressionRectangle);
    }
```

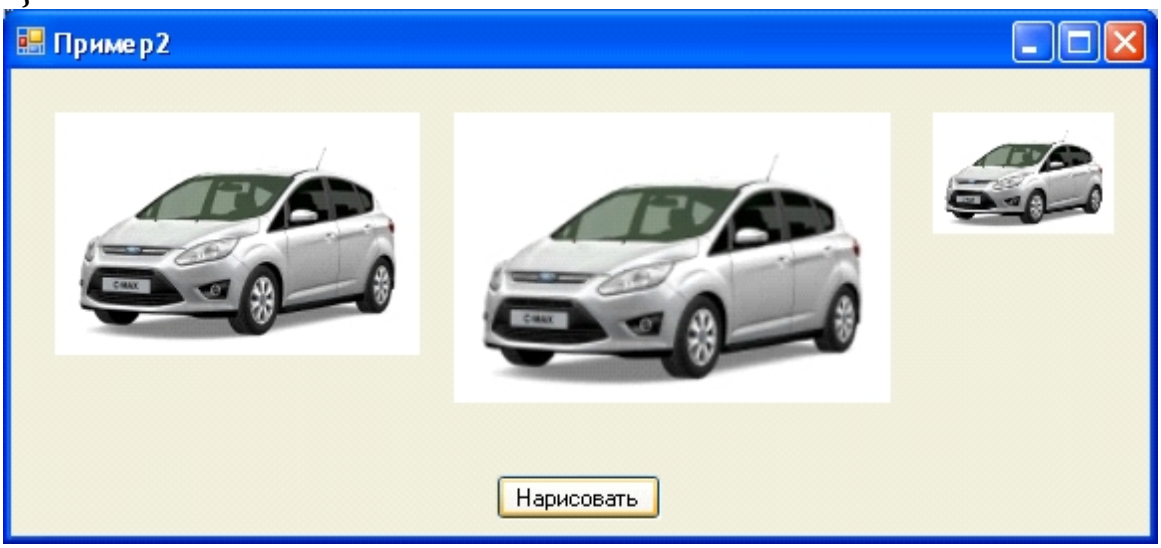

Рис. 2. Результат выполнения примера 2.

Некоторые варианты метода DrawImage() получают в качестве параметров не только конечный, но и исходный прямоугольник. Исходный прямоугольник задает часть исходного изображения, которая должна быть нарисована. Прямоугольник назначения задает прямоугольник, в котором должна быть нарисована эта часть изображения. Если размер прямоугольника назначения отличается от размера исходного прямоугольника, изображение масштабируется, чтобы соответствовать размеру прямоугольника назначения.

Пример 3. Приведенный ниже пример кода демонстрирует создание объекта Bitmap из файла auto1.jpg. Изображение из файла рисуется целиком, без масштабирования, с привязкой к точке с координатами (0, 0). Затем небольшой фрагмент изображения отображается дважды: один раз — со сжатием, второй раз — с увеличением (рис. 3).

```
private void button1_Click(object sender, EventArgs e)
    {
```

```
    Graphics im = pictureBox1.CreateGraphics();
    Bitmap myBitmap = new Bitmap("auto1.jpg");
    Rectangle sourceRectangle = new Rectangle(80, 70, 80, 45);
    Rectangle destRectangle1 = new Rectangle(200, 10, 20, 16);
    Rectangle destRectangle2 = new Rectangle(200, 40, 200, 160);
    // Рисуем в (0, 0)
    im.DrawImage(myBitmap, 0, 0);
    // Рисуем сжатое колесо
    im.DrawImage(myBitmap, destRectangle1, sourceRectangle,
                 GraphicsUnit.Pixel);
    // Рисуем увеличенное колесо
    im.DrawImage(myBitmap, destRectangle2, sourceRectangle,
                 GraphicsUnit.Pixel);
}
```

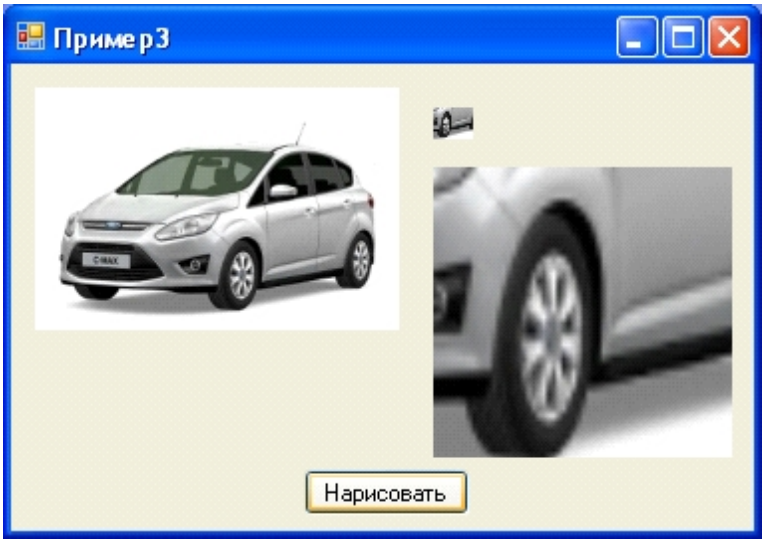

Рис. 3. Результат выполнения примера 3.

3.4.3. Поворот, отражение и наклон изображений

Изображение можно поворачивать, отражать и наклонять, указывая точки назначения для верхнего левого, верхнего правого и нижнего левого углов исходного изображения Эти три точки назначения определяют аффинное преобразование, которое отображает исходное прямоугольное изображение в параллелограмм.

Пример 4. Например, предположим, что исходное изображение представляет собой прямоугольник с верхним левым углом в точке (0, 0), верхним правым углом в точке (100, 0) и нижним левым углом в точке (0, 50). Допустим, эти три точки отображаются в точки назначения следующим образом.

Исходная точка	Точка назначения
Верхний левый угол (0, 0)	(200, 20)
Верхний правый угол (100, 0)	(110, 100)
Нижний левый угол (0, 50)	(250, 30)

На рисунке 4 показаны как исходное изображение, так и отображение этого изображения в параллелограмм. Исходное изображение было наклонено, отражено, повернуто и сдвинуто. Ось x, расположенная вдоль верхнего края исходного изображения, отображается в линию, проходящую

через точки (200, 20) и (110, 100). Ось y, расположенная вдоль левого края исходного изображения, отображается в линию, проходящую через точки (200, 20) и (250, 30).

```
Point[] destinationPoints = {
      new Point(200, 20),    // Координата левой верхней точки оригинала
      new Point(110, 100),  // Координата для правой верхней точки оригинала
      new Point(250, 30)};  // Координата для левой нижней точки оригинала
Image image = new Bitmap("figure.bmp");
e.Graphics.DrawImage(image, 0, 0);
e.Graphics.DrawImage(image, destinationPoints);
```

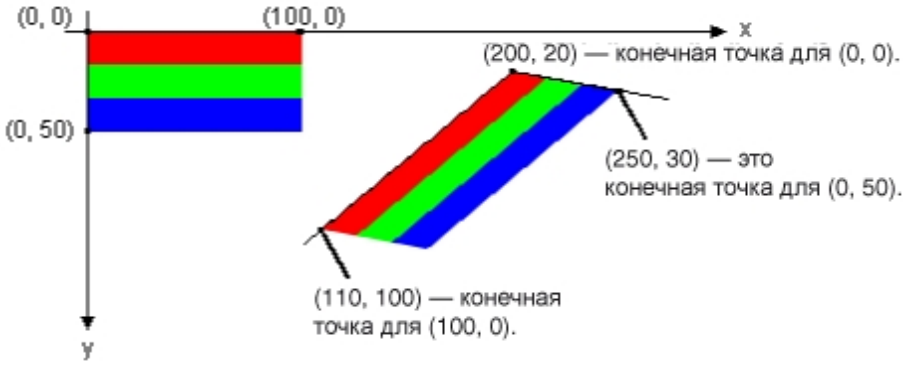

Рис. 4. Результат выполнения примера 4.

Пример 5. На рисунке 5 показано преобразование, примененное к изображению, хранящемуся в файле «auto1.jpg». Код реализации приложения «Автомобиль» приведен ниже:

```
private void Form1_Paint(object sender, PaintEventArgs e)
    {
      Point[] destinationPoints = {  new Point(280, 0),
                                     new Point(170, 140),
                                     new Point(380, 70)};
      Image image = new Bitmap("auto1.jpg");
      e.Graphics.DrawImage(image, 0, 0);
       e.Graphics.DrawImage(image, destinationPoints);
    }
```

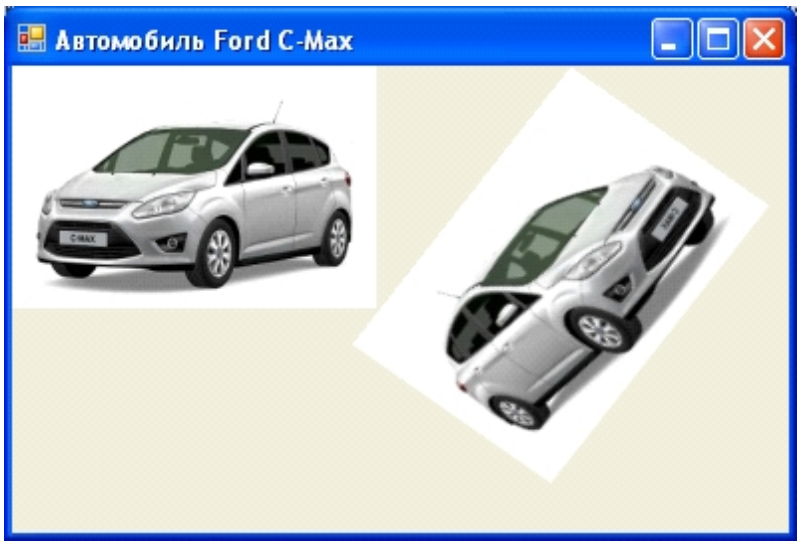

Рис. 5. Результат выполнения приложения «Автомобиль».

2. Рабочее задание

Задание 1. Разработать приложение «Выбор рисунка», с помощью которого можно отобразить любой рисунок из списка рисунков. Список рисунков формируется в результате поиска графических файлов с расширением **.bmp** на дисках компьютера.

Исследовать возможности отображения рисунка в различных видах, для чего использовать кнопки, расположенные под рисунком (Normal, StretchImage, …, Zoom).

При разработке интерфейса приложения использовать компоненты **Label, Button, TextBox, ListBox, PictureBox** (один из вариантов интерфейса представлен на рис. 6).

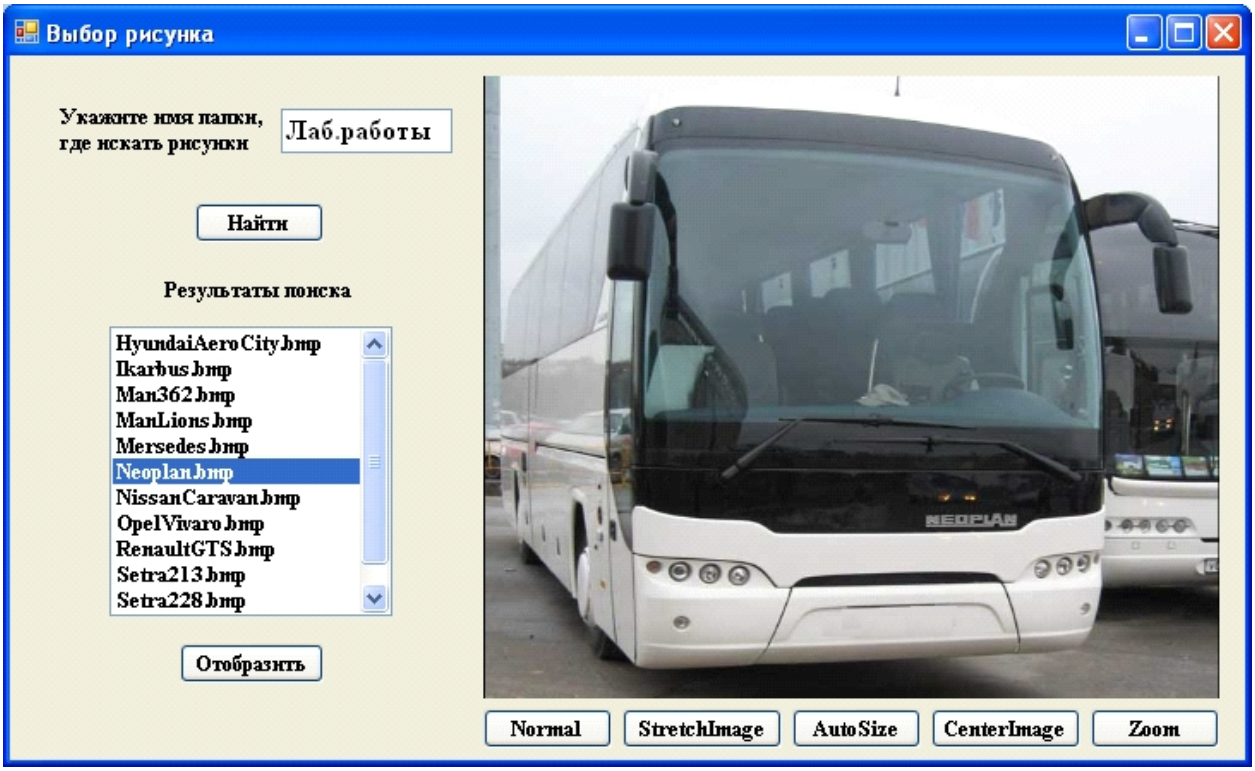

Рис. 6. Внешний вид приложения «Выбор рисунка»

Задание 2. Разработать приложение «Вращение рисунка», с помощью которого можно, отобразив любой рисунок из списка рисунков, манипулировать изображением (уменьшить, увеличить, развернуть на заданный угол). Список рисунков формируется в результате поиска графических файлов с расширением **.bmp** на дисках компьютера.

При разработке интерфейса приложения использовать компоненты **Label, Button, TextBox, ListBox, GroupBox, PictureBox** (один из вариантов интерфейса представлен на рис. 7).

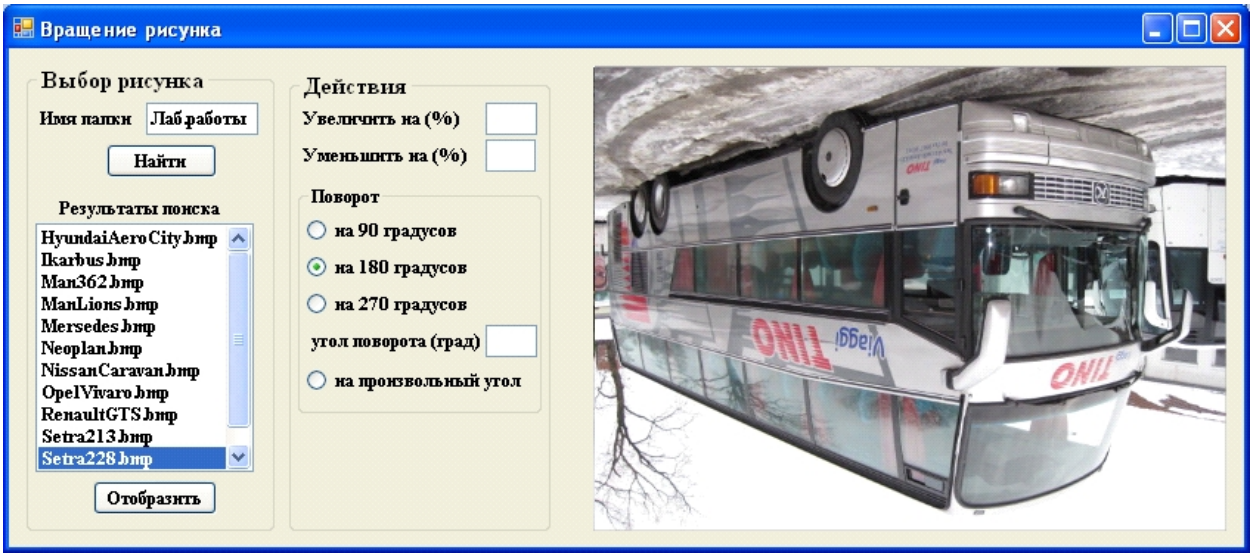

Рис. 7. Внешний вид приложения «Вращение рисунка»

3. Контрольные вопросы

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – СПб.: БХВ-Петербург, 2009. – 320 с.: ил.
3. Лабор В.В. Си Шарп: Создание приложений для Windows. – Мн.: Харвест, 2003. – 384 с.
4. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
5. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
6. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
7. Фаронов В.В. Программирование на языке C#. – СПб.: Питер, 2007. – 240 с.: ил.
8. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.