Graphics 2

Working with Images

Formatting Text

By Pakita Shamoi

Intro

- Often developers need to display, create, or modify images.
- The .NET Framework provides tools to work with a variety of image formats, enabling you to perform many common image-editing tasks.

Image class

- This abstract class gives you the ability to create, load, modify, and save images such as .BMP files, .JPG files, and .TIF files
- You can:
  - Create a drawing or chart, and save the results as an image file.
  - Use text to add copyright information or a watermark to a picture.
  - Resize JPEG images so that they consume less space and can be downloaded faster.

Image class (cont.)

- ***Image.FromFile -*** accepts a path to an image file as a parameter
- *Image.FromStream* - accepts a *System.IO.Stream* object as a parameter

- You can also use two classes that inherit *Image*:
  - *System.Drawing.Bitmap* for still images
  - *System. Drawing.Imaging.Metafile* for animated images.

Bitmap class

- The most commonly used class for working with new or existing images.
- You can create *Bitmap* from an existing *Image*, file, or stream, or to create a blank bitmap of a specified *height* and *width*.
- *Bitmap* contains two particularly useful methods that *Image* lacks:
  - ***GetPixel*** Returns a *Color* object describing a particular pixel in the image. A pixel is a single colored dot in the image, consisting of a red, green, and blue component (RGB)
  - ***SetPixel*** Sets a pixel to the specified color.
- More complex image editing requires you to create a *Graphics* object by calling *Graphics.FromImage*.

Displaying images

- To display an image that is saved to the disk do the following:
  - load it with *Image.FromFile*
  - create a *PictureBox* control
  - use the *Image* to define *PictureBox.Background Image*.

```
Image i = Image.FromFile(@"C:\windows\g.bmp");
pictureBox1.BackgroundImage = i;
Bitmap b = new Bitmap(@"C:\windows\k.bmp");
pictureBox2.BackgroundImage = b;
```

Displaying images

- Alternatively, you can display an image as the background for a form or control by using the *Graphics.DrawImage* method.

```
Bitmap bm = new Bitmap(@"C:\Wall\Azul.jpg");
Graphics g = this.CreateGraphics();
g.DrawImage(bm, 1, 1, this.Width, this.Height);
```

How to create a picture

- To create a new, blank picture, create an instance of the *Bitmap* class
- You can then edit it using the *Bitmap.SetPixel* method, or you can call *Graphics.FromImage* and edit the image using the *Graphics* drawing methods.

```
Bitmap bm = new Bitmap(600, 600);
Graphics g = Graphics.FromImage(bm);
Brush b = new LinearGradientBrush
    (new Point(1, 1), new Point(600, 600), Color.YellowGreen, Color.Red);
Point[] points = new Point[]
                {new Point(10, 10),
                 new Point(77, 500),
                 new Point(590, 100)};
g.FillPolygon(b, points);
```

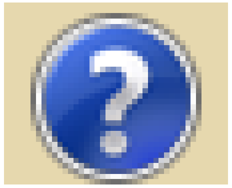
How to save a picture

- To save a picture, call *Bitmap.Save*.
- Two of the overloads accept a parameter of type *System.Drawing.Imaging.ImageFormat*, to describe the file type: Bmp, Emf, Exif, Gif, Icon, Jpeg, MemoryBmp, Png, Tiff, or Wmf.
  - Jpeg is the most common format for photographs
  - Gif is the most common format for charts, screen shots, and drawings.
- bm.Save(@"C:\bm.jpg", ImageFormat.Jpeg);

Using icons

- Icons are transparent bitmaps of specific sizes that are used by Windows to convey status.
- The .NET Framework provides standard 40-by-40 system icons as properties of the *SystemIcons* class, including icons for exclamation, information, and question.
- You can call *Icon.ToBitmap* to create a *Bitmap* object that can be edited.

```
Graphics g = this.CreateGraphics();
g.DrawIcon(SystemIcons.Question, 40, 40);
```



Adding text to Graphics

- Developers often add text to images to label objects or create reports.
- To add text to Graphics:
  - Create a *Graphics* object

```
Graphics g = this.CreateGraphics();
```

  - Create a *Font* object.

```
Font f = new Font("Arial", 12, FontStyle.Bold);
```

  - Optionally, create a *Brush* object.
  - Call *Graphics.DrawString* and specify the location for the text.

```
g.DrawString("Hello, World!", f, Brushes.Blue, 10, 10);
```

Controlling Text Formatting

- You can control the alignment and direction of text using the *StringFormat* class.
- After creating and configuring a *StringFormat* object, you can provide it to the *Graphics.DrawString* method to control how text is formatted
- You can customize
  - Alignment

```
f1.Alignment = StringAlignment.Center;
```

  - Direction (FormatFlags)

```
f2.FormatFlags = StringFormatFlags.DirectionVertical;
```

```
g.DrawString("Format2", this.Font, Brushes.Red, (RectangleF)r, f2)
```

Summary

- The *Image* and *Bitmap* classes enable you to edit or create pictures, and save the results as a file.
- To display a picture in a Windows Form, load the picture into an instance of the *Image* or *Bitmap* class, create an instance of the *PictureBox* control, and then use the *Image* or *Bitmap* object to define the *PictureBox.BackgroundImage* property.
- To create and save a picture, create a *Bitmap* object, edit it using a *Graphics* object, and then call the *Bitmap.Save* method.

Summary

- To add text to graphics, create a *Graphics* object, create a *Font* object, optionally create a *Brush* object, and then call the *Graphics.DrawString* method.
- To create a *Font* object, pass the font family name, font size, and font style.
- Write text by calling the *Graphics.DrawString* method. The *DrawString* method requires
  - a *Font* object
  - a *Brush* object that specifies the color of the text
  - location to draw the text.
- Use the *StringFormat* class to control the formatting of text. You can use this class to change the direction of the text, or to change the alignment of text.

Home work

- Page 364, lab – adding logo to an image and legend to a pie chart