Color Figures

**Figure 2.4:** Drawing a line from point (0, 0) to point (120, 80) 13

**Figure 2.5:** Creating a Windows application 13

**Figure 2.6:** Adding a reference to System.Drawing.dll 14

**Figure 2.7:** The System.Drawing namespace in a project 14

**Figure 2.8:** Adding the Form_Paint event handler 15

**Figure 2.9:** Your first GDI+ application 15

**Figure 2.10:** Using Point to draw a line 16

**Figure 2.11:** Using PointF to draw a line 16

**Figure 2.12:** Using Rectangle to create rectangles 17

**Figure 2.13:** Using RectangleF to create rectangles 17

**Figure 2.14:** Using the Round, Truncate, Union, Inflate, Ceiling, and Intersect methods of Rectangle 18

**Figure 3.20:** A curve-drawing application 18

**Figure 3.26:** Drawing a polygon 19

**Figure 4.4:** A sample hatch brush application 19

**Figure 4.5:** The default hatch style rectangle 20

**Figure 4.6:** The LightDownwardDiagonal style with different colors 20

**Figure 4.7:** The DiagonalCross style 21

**Figure 4.8:** The texture brush application 21

**Figure 4.9:** Using texture brushes 22

**Figure 4.10:** Clamping a texture 22

**Figure 4.11:** The TileFlipY texture option 23
Figure 5.22: Using RotateTransform to rotate text 40
Figure 5.23: Using TranslateTransform to translate text 40

Figure 6.3: Using Rectangle methods 41
Figure 6.4: Hit test using the Contains method 41
Figure 6.10: Bounds of an infinite region 42
Figure 6.12: UsingClip methods 42
Figure 6.13: Using TranslateClip 43
Figure 6.18: Client and nonclient areas of a form 43
Figure 6.19: A nonrectangular form and controls 44
Figure 6.20: The nonrectangular forms application 44
Figure 6.21: A circular form 45
Figure 6.22: A triangular form 45

Figure 7.1: A zoomed raster image 46
Figure 7.2: A zoomed vector image 46
Figure 7.3: A simple image viewer application 47
Figure 7.4: Browsing a file 47
Figure 7.5: Viewing an image 48
Figure 7.6: Reading the properties of an image 48
Figure 7.7: A thumbnail image 49
Figure 7.8: Rotate menu items 49
Figure 7.9: Flip menu items 49
Figure 7.10: An image with default settings 50
Figure 7.11: The image of figure 7.10, rotated 90 degrees 50
Figure 7.12: The image of Figure 7.10, rotated 180 degrees 51
Figure 7.13: The image of Figure 7.10, rotated 270 degrees 51
Figure 7.14: The image of Figure 7.10, flipped in the x direction 52
Figure 7.15: The image of Figure 7.10, flipped in the y direction 52
Figure 7.16: The image of Figure 7.10, flipped in both the x and the y directions 53
Figure 7.17: Fit menu items 53
Figure 7.18: An image in ImageViewer 54
Figure 7.19: The image of Figure 7.18 after Fit Width 54
Figure 7.20: The image of Figure 7.18 after Fit Height 55
Figure 7.21: The image of Figure 7.18 after Fit Original 55
Figure 7.22: The image of Figure 7.18 after Fit All 56
Figure 7.23: Zoom menu items 56
Figure 7.24: An image in ImageViewer 57
Figure 7.25: The image of Figure 7.24 with 25 percent zoom 57
Figure 7.26: The image of Figure 7.24 with 50 percent zoom 57
Figure 7.27: The image of Figure 7.24 with 200 percent zoom 58
Figure 7.28: The image of Figure 7.24 with 500 percent zoom 58
Figure 7.29: An animated image with three frames 58
Figure 7.30: An image animation example 59
Figure 7.31: The first frame of an animated image 59
Figure 7.32: The second frame of an animated image 60
Figure 7.33: A bitmap example 60
Figure 7.34: Changing the pixel colors of a bitmap 61
Figure 7.35: Viewing icons 61
Figure 7.36: A skewing application 62
Figure 7.37: Normal view of an image 62
Figure 7.38: Skewed image 63
Figure 7.39: Drawing transparent graphics objects 63
Figure 7.40: Drawing multiple images 64
Figure 7.41: Viewing an image in a picture box 64
Figure 7.42: Saving images with different sizes 65
Figure 7.43: New image, with width of 200 and height of 200 66

Figure 8.1: Using BitmapData to set grayscale 66
Figure 8.2: Changing the pixel format of a partial bitmap 67
Figure 8.3: Viewing a metafile 67
Figure 8.4: A metafile created programmatically 68
Figure 8.5: Reading metafile records 68
Figure 8.6: Reading metafile header attributes 68
Figure 8.7: Applying a color remap table 69
Figure 8.8: Wrapping images 69
Figure 8.9: Drawing semitransparent images 70
Figure 8.10: Applying SetGamma and SetColorKey 70
Figure 8.11: Using the SetNoOp method 70
COLOR FIGURES

Figure 9.4: Reading line caps 71
Figure 9.5: Reading line dash styles 71
Figure 9.6: Getting line dash caps 71
Figure 9.7: A rectangle, an ellipse, and a curve with different line styles 72
Figure 9.8: A line with custom caps 72
Figure 9.9: The line join test application 73
Figure 9.10: The Bevel line join effect 73
Figure 9.11: The Miter line join effect 74
Figure 9.12: The Round line join effect 74
Figure 9.13: Customized starting and ending caps 74
Figure 9.14: Setting customized starting and ending caps 75
Figure 9.15: Adjustable arrow caps 75
Figure 9.16: A simple graphics path 75
Figure 9.17: A filled graphics path 76
Figure 9.18: A shaped form 76
Figure 9.19: Three subpaths 77
Figure 9.20: Nested containers 77
Figure 9.21: Drawing with different PageUnit values 78
Figure 9.22: Saving and restoring graphics states 78
Figure 9.23: Using graphics containers to draw text 79
Figure 9.24: Using graphics containers to draw shapes 79
Figure 9.25: Reading the metadata of a bitmap 79
Figure 9.26: Color blending examples 80
Figure 9.27: Transparent graphics shapes in an image using alpha blending 80
Figure 9.28: Mixed blending effects 81
Figure 9.29: Using linear gradient brushes 81
Figure 9.30: Using a rectangle in the linear gradient brush 81
Figure 9.31: Using the SetBlendTriangularShape method 82
Figure 9.32: Using the SetSigmaBellShape method 82
Figure 9.33: Comparing the effects of SetBlendTriangularShape and SetSigmaBellShape 83
Figure 9.34: Setting the center of a gradient 84
Figure 9.35: A multicolor gradient 85
Figure 9.36: Using blending in a linear gradient brush 85
Figure 9.37: Blending using PathGradientBrush 86
Figure 9.38: Setting the focus scale  86
Figure 9.39: Blending multiple colors  87
Figure 9.40: Using the InterpolationColors property of PathGradientBrush  87
Figure 9.41: Multicolor blending using PathGradientBrush  88
Figure 9.42: Drawing semitransparent graphics shapes  88
Figure 9.43: Drawing semitransparent shapes on an image  89
Figure 9.44: Using CompositingMode.SourceOver  89
Figure 9.45: Blending with CompositingMode.SourceCopy  90
Figure 9.46: A mixed blending example  90
Figure 9.47: Drawing with SmoothingMode set to Default  91
Figure 9.48: Drawing with SmoothingMode set to AntiAlias  91

Figure 10.1: Steps in the transformation process  92
Figure 10.2: Transformation stages  92
Figure 10.3: Drawing a line from point (0, 0) to point (120, 80)  92
Figure 10.4: Drawing a line from point (0, 0) to point (120, 80) with origin (50, 40)  93
Figure 10.5: Drawing with the GraphicsUnit.Inch option  93
Figure 10.6: Drawing with the GraphicsUnit.Inch option and a pixel width  94
Figure 10.7: Combining page and device coordinates  94
Figure 10.8: Drawing a line and filling a rectangle  95
Figure 10.9: Rotating graphics objects  95
Figure 10.10: Using the RotateAt method  96
Figure 10.11: Resetting a transformation  96
Figure 10.12: Scaling a rectangle  97
Figure 10.13: Shearing a rectangle  97
Figure 10.14: Translating a rectangle  98
Figure 10.15: Composite transformation  98
Figure 10.16: Local transformation  99
Figure 10.17: Rotating images  99
Figure 10.18: Scaling images  100
Figure 10.19: Translating images  100
Figure 10.20: Shearing images  101
Figure 10.24: Translating colors 101
Figure 10.25: Scaling colors 102
Figure 10.26: Shearing colors 102
Figure 10.29: Rotating colors 103
Figure 10.30: Using the transformation matrix to transform text 103
Figure 10.31: Using the transformation matrix to shear text 104
Figure 10.32: Using the transformation matrix to reverse text 104
Figure 10.33: Scale \rightarrow Rotate \rightarrow Translate composite transformation 105
Figure 10.34: Translate \rightarrow Rotate \rightarrow Scale composite transformation with Append 105
Figure 10.35: Translate \rightarrow Rotate \rightarrow Scale composite transformation with Prepend 106

Figure 11.6: Creating a Windows application 106
Figure 11.7: Your first printing application 107
Figure 11.8: The printer settings form 107
Figure 11.9: Reading printer properties 108
Figure 11.11: The print events application 108
Figure 11.12: The form with text file printing options 109
Figure 11.13: A graphics-printing application 109
Figure 11.14: Drawing simple graphics items 110
Figure 11.15: Viewing an image 111
Figure 11.16: Print dialogs in the Visual Studio.NET toolbox 111
Figure 11.17: The print dialog application 112
Figure 11.18: Viewing an image and text 113
Figure 11.19: The print preview dialog 113
Figure 11.20: The page setup dialog 114
Figure 11.21: The print dialog 114
Figure 11.22: The custom page settings dialog 115
Figure 11.23: The PageSetupDialog sample in action 115
Figure 11.24: A form for printing multiple pages 116
Figure 11.25: Print preview of multiple pages 116
Figure 11.26: Setting a document name 117
Figure 11.27: Marginal-printing test application 117
Figure 11.29: Print controller test form 118
Figure 11.30: Print controller output

Figure 12.3: The FirstWebApp project

Figure 12.4: The default WebForm1.aspx page

Figure 12.5: The HTML view of WebForm1.aspx

Figure 12.6: An ASP.NET document’s page properties

Figure 12.7: The WebForm1.aspx design mode after the addition of Web Forms controls

Figure 12.8: Viewing an image in an Image control

Figure 12.9: Drawing simple graphics objects on the Web

Figure 12.10: Drawing various graphics objects

Figure 12.11: Drawing an image

Figure 12.12: Using LinearGradientBrush and PathGradientBrush

Figure 12.13: Drawing semitransparent objects

Figure 12.14: Entering points on a chart

Figure 12.15: A line chart in ASP.NET

Figure 12.16: A pie chart–drawing application in ASP.NET

Figure 12.17: The Draw Chart button click in action

Figure 12.18: The Fill Chart button click in action

Figure 13.2: Drawing on a form

Figure 13.3: Drawing on Windows controls

Figure 13.4: Drawing lines in a loop

Figure 13.5: The same result from two different drawing methods

Figure 13.6: Using DrawRectangle to draw rectangles

Figure 13.7: Using system pens and brushes

Figure 15.1: An interactive GUI application

Figure 15.2: Designing transparent controls

Figure 15.3: Drawing a circular form and Windows controls

Figure 15.4: A graphics copyright application

Figure 15.5: Thumbnail view of an image

Figure 15.6: An image after copyright has been added to it

Figure 15.7: Users table schema
Figure 15.8: Reading and writing images in a database form  135
Figure 15.9: Displaying a bitmap after reading data from a database  136
Figure 15.10: An owner-drawn ListBox control  137
Figure 15.11: An owner-drawn ListBox control with images  138

Figure A.1: An error generated from Listing A.1  138
Figure A.2: An exception-handled error message  139
Figure 2.4: Drawing a line from point (0, 0) to point (120, 80)

Figure 2.5: Creating a Windows application
Figure 2.7: The `System.Drawing` namespace in a project
Figure 2.9: Your first GDI+ application
Figure 2.11: Using PointF to draw a line
Figure 2.14: Using the Round, Truncate, Union, Inflate, Ceiling, and Intersect methods of Rectangle

Figure 3.20: A curve-drawing application
**Figure 3.26:** Drawing a polygon

**Figure 4.4:** A sample hatch brush application
Figure 4.5: The default hatch style rectangle
Figure 4.8: The texture brush application
Figure 4.9: Using texture brushes

Figure 4.10: Clamping a texture
**Figure 4.11:** The TileFlipY texture option

**Figure 4.12:** A color gradient
Figure 4.13: A gradient pattern with pattern repetition

Figure 4.14: Our linear gradient brush application
**Figure 4.15:** The default linear gradient brush output
Figure 4.17: Using a rectangle in a linear gradient brush
Figure 4.19: Creating and using pens
Figure 4.20: Displaying pen types

Figure 4.21: Our pen alignment application
Figure 4.22: Drawing with center pen alignment

Figure 4.23: Drawing with inset pen alignment
Figure 4.25: Drawing dashed lines with different cap styles

Figure 4.26: Graphics shapes with cap and dash styles
**Figure 4.27:** Rotation and scaling

**Figure 4.30:** Transformation in path gradient brushes
Figure 4.31: Using system pens and system brushes

Figure 4.32: GDI+Painter with pen and brush support
Figure 4.33: GDI+Painter in action

Figure 5.1: Creating colors using different methods
Figure 5.2: Getting brightness, hue, and saturation components of a color

Figure 5.3: Using system colors to draw graphics objects
Figure 5.4: Converting colors

Figure 5.11: Getting line spacing, ascent, descent, free (extra) space, and height of a font
Figure 5.14: Alignment and trimming options

Figure 5.15: Drawing tabbed text on a form

<table>
<thead>
<tr>
<th>ID</th>
<th>Math</th>
<th>Physics</th>
<th>Chemistry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1002</td>
<td>76</td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td>1003</td>
<td>53</td>
<td>98</td>
<td>90</td>
</tr>
<tr>
<td>1008</td>
<td>99</td>
<td>78</td>
<td>65</td>
</tr>
</tbody>
</table>
**Figure 5.18:** Using a private font collection
Colors, fonts, and text are common elements of graphics programming. In this chapter, you learned about the colors, fonts, and text representations in the .NET Framework class library. You learned how to create these elements and use them in GDI+.

**Figure 5.20:** Drawing text on a form
Colors, fonts, and text are common elements of graphics programming. In this chapter, you learned about the color, font, and text representations in the .NET Framework class. You learned how to use these elements and...
Figure 6.10: Bounds of an infinite region

Figure 6.12: Using Clip methods
Figure 6.18: Client and nonclient areas of a form
Figure 6.19: A nonrectangular form and controls

Figure 6.20: The nonrectangular forms application
Figure 6.21: A circular form

Figure 6.22: A triangular form
Figure 7.1: A zoomed raster image

Figure 7.2: A zoomed vector image
**Figure 7.3:** A simple image viewer application

**Figure 7.4:** Browsing a file
Figure 7.5: Viewing an image

Figure 7.6: Reading the properties of an image
Figure 7.7: A thumbnail image

Figure 7.8: Rotate menu items

Figure 7.9: Flip menu items
Figure 7.10: An image with default settings

Figure 7.11: The image of Figure 7.10, rotated 90 degrees
Figure 7.12: The image of Figure 7.10, rotated 180 degrees

Figure 7.13: The image of Figure 7.10, rotated 270 degrees
Figure 7.14: The image of Figure 7.10, flipped in the x-direction

Figure 7.15: The image of Figure 7.10, flipped in the y-direction
Figure 7.16: The image of Figure 7.10, flipped in both the x- and the y-directions

Figure 7.17: Fit menu items
Figure 7.18: An image in ImageViewer

Figure 7.19: The image of Figure 7.18 after Fit Width
**Figure 7.20:** The image of Figure 7.18 after Fit Height

**Figure 7.21:** The image of Figure 7.18 after Fit Original
Figure 7.22: The image of Figure 7.18 after Fit All

Figure 7.23: Zoom menu items
**Figure 7.24:** An image in ImageViewer

**Figure 7.25:** The image of Figure 7.24 with 25 percent zoom

**Figure 7.26:** The image of Figure 7.24 with 50 percent zoom
Figure 7.27: The image of Figure 7.24 with 200 percent zoom

Figure 7.28: The image of Figure 7.24 with 500 percent zoom

Figure 7.29: An animated image with three frames
**Figure 7.30:** An image animation example

**Figure 7.31:** The first frame of an animated image
Figure 7.32: The second frame of an animated image

Figure 7.33: A bitmap example
**Figure 7.34:** Changing the pixel colors of a bitmap

**Figure 7.35:** Viewing icons
Figure 7.36: A skewing application

Figure 7.37: Normal view of an image
**Figure 7.38:** Skewed image

**Figure 7.39:** Drawing transparent graphics objects
Figure 7.40: Drawing multiple images

Figure 7.41: Viewing an image in a picture box
**FIGURE 7.42:** Saving images with different sizes
Figure 7.43: New image, with width of 200 and height of 200
Figure 8.2: Changing the pixel format of a partial bitmap

Figure 8.3: Viewing a metafile
FIGURE 8.4: A metafile created programmatically

FIGURE 8.5: Reading metafile records

FIGURE 8.6: Reading metafile header attributes
Figure 8.7: Applying a color remap table

Figure 8.8: Wrapping images
Figure 8.9: Drawing semitransparent images
Figure 9.4: Reading line caps

Figure 9.5: Reading line dash styles

Figure 9.6: Getting line dash caps
Figure 9.7: A rectangle, an ellipse, and a curve with different line styles

Figure 9.8: A line with custom caps
Figure 9.9: The line join test application
Figure 9.13: Customized starting and ending caps
Figure 9.14: Setting customized starting and ending caps

Figure 9.15: Adjustable arrow caps

Figure 9.16: A simple graphics path
Figure 9.17: A filled graphics path

Close? Right Click!

Figure 9.18: A shaped form
Figure 9.19: Three subpaths

Graphics Surface (Object)
CompositingQuality = CompositingQuality.HighQuality;
SmoothingMode = SmoothingMode.HighSpeed;

Graphics Container A
SmoothingMode = SmoothingMode.AntiAlias;

Graphics Container B
SmoothingMode = SmoothingMode.HighQuality
InterpolationMode = InterpolationMode.High

Graphics Container C

Figure 9.20: Nested containers
Figure 9.22: Saving and restoring graphics states
Figure 9.23: Using graphics containers to draw text

Figure 9.24: Using graphics containers to draw shapes

Figure 9.25: Reading the metadata of a bitmap
Figure 9.26: Color blending examples

Figure 9.27: Transparent graphics shapes in an image using alpha blending
Figure 9.28: Mixed blending effects

Figure 9.29: Using linear gradient brushes

Figure 9.30: Using a rectangle in the linear gradient brush
Figure 9.33: Comparing the effects of `SetBlendTriangularShape` and `SetSigmaBellShape`
Figure 9.34: Setting the center of a gradient
Figure 9.35: A multicolor gradient

Figure 9.36: Using blending in a linear gradient brush
Figure 9.38: Setting the focus scale
Figure 9.39: Blending multiple colors
Figure 9.42: Drawing semitransparent graphics shapes
Figure 9.43: Drawing semitransparent shapes on an image
Figure 9.46: A mixed blending example
**Figure 10.1:** Steps in the transformation process

**Figure 10.2:** Transformation stages

**Figure 10.3:** Drawing a line from point (0, 0) to point (120, 80)
Figure 10.4: Drawing a line from point (0, 0) to point (120, 80) with origin (50, 40)
Figure 10.7: Combining page and device coordinates
Figure 10.8: Drawing a line and filling a rectangle

Figure 10.9: Rotating graphics objects
Figure 10.11: Resetting a transformation
Figure 10.12: Scaling a rectangle

Figure 10.13: Shearing a rectangle
**Figure 10.14:** Translating a rectangle

**Figure 10.15:** Composite transformation
**Figure 10.16:** Local transformation

**Figure 10.17:** Rotating images
Figure 10.18: Scaling images

Figure 10.19: Translating images
Figure 10.20: Shearing images

Figure 10.24: Translating colors
Figure 10.25: Scaling colors

Figure 10.26: Shearing colors
Figure 10.29: Rotating colors

Figure 10.30: Using the transformation matrix to transform text
Colors, fonts and text are common elements of graphics programming. In this chapter, you learned about the colors, fonts and text representation in the .NET Framework class library. You learned how to create these elements and use in GDI+.

**Figure 10.31:** Using the transformation matrix to shear text

**Figure 10.32:** Using the transformation matrix to reverse text
Figure 10.33: Scale \( \text{fi} \) Rotate \( \text{fi} \) Translate composite transformation
**Figure 11.6:** Creating a Windows application
Figure 11.7: Your first printing application

Figure 11.8: The printer settings form
Figure 11.9: Reading printer properties

Figure 11.11: The print events application
Figure 11.12: The form with text file printing options

Figure 11.13: A graphics-printing application
Figure 11.14: Drawing simple graphics items
Figure 11.15: Viewing an image

Figure 11.16: Print dialogs in the Visual Studio.NET toolbox
Figure 11.17: The print dialog application
Figure 11.18: Viewing an image and text

Figure 11.19: The print preview dialog
Figure 11.20: The page setup dialog

Figure 11.21: The print dialog
Figure 11.22: The custom page settings dialog
Figure 11.24: A form for printing multiple pages

Figure 11.25: Print preview of multiple pages
**Figure 11.26:** Setting a document name

**Figure 11.27:** Marginal-printing test application
Figure 11.29: Print controller test form

Figure 11.30: Print controller output
Figure 12.3: The FirstWebApp project
Figure 12.6: An ASP.NET document’s page properties
Figure 12.9: Drawing simple graphics objects on the Web

Figure 12.10: Drawing various graphics objects
Figure 12.11: Drawing an image
Figure 12.13: Drawing semitransparent objects
Figure 12.14: Entering points on a chart
Figure 12.15: A line chart in ASP.NET
Figure 12.16: A pie chart–drawing application in ASP.NET
Figure 12.17: The Draw Chart button click in action
Figure 12.18: The Fill Chart button click in action

Figure 13.2: Drawing on a form
Figure 13.3: Drawing on Windows controls

Figure 13.4: Drawing lines in a loop
Figure 13.5: The same result from two different drawing methods
**Figure 13.7:** Using system pens and brushes

**Figure 15.1:** An interactive GUI application
**Figure 15.2:** Designing transparent controls

**Figure 15.3:** Drawing a circular form and Windows controls
Figure 15.4: A graphics copyright application

Figure 15.5: Thumbnail view of an image

Figure 15.6: An image after copyright has been added to it
**Figure 15.7:** Users table schema

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>AutoNumber</td>
</tr>
<tr>
<td>UserName</td>
<td>Text</td>
</tr>
<tr>
<td>UserEmail</td>
<td>Text</td>
</tr>
<tr>
<td>UserPhoto</td>
<td>OLE Object</td>
</tr>
<tr>
<td>UserDescription</td>
<td>Memo</td>
</tr>
</tbody>
</table>

**Figure 15.8:** Reading and writing images in a database form
Figure 15.9: Displaying a bitmap after reading data from a database
<table>
<thead>
<tr>
<th>Color</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>Item</td>
</tr>
<tr>
<td>Blue</td>
<td>Item</td>
</tr>
<tr>
<td>Red</td>
<td>Item</td>
</tr>
<tr>
<td>Green</td>
<td>Item</td>
</tr>
<tr>
<td>Yellow</td>
<td>Item</td>
</tr>
</tbody>
</table>
Figure A.1: An error generated from Listing A.1
**Figure A.2:** An exception-handled error message