CHAPTER 14

A Better Painting Framework (GDI+)

The previous chapter introduced you to the fine art of building a traditional main window using various types contained within the System.Windows.Forms namespace. Now that you can assemble a Form to represent the shell of your GUI-based applications, the next task is to understand the details of rendering graphical data (including stylized text and image data) onto the Form's client area.

We begin by taking a high-level view of the numerous drawing-related namespaces, and examine the process of responding to (and initiating) paint sessions. You will also discover various ways of obtaining (and configuring) a Graphics object. Once you understand the general layout of the GDI+ landscape, the remainder of this chapter covers how to manipulate colors, fonts, geometric shapes, and graphical images. This entails understanding related types such as Brush, Pen, Color, Point, and Rectangle (among others). This chapter also explores a number of GDI+-centric programming techniques such as nonrectangular hit testing and GUI drag-and-drop logic.

The chapter concludes by exploring the new .NET resource format. While technically not part of GDI+ proper, it does involve the manipulation of graphical data (which in my opinion is "GDI+-enough" to be presented here). Here, you learn how to embed your application's external resources directly into a .NET assembly to ship a more portable binary image. During the process, you explore the System.Resources namespace and learn how to perform read/write operations on the underlying *.resx and *.resources files by hand, as well as pull resources from an assembly at runtime using the System.Resources.ResourceManager type.

NOTE If you are a Web programmer by trade, you may think that GDI+ is of no use to you. However, as you will see later in this text during our examination of ASP.NET, GDI+ is not limited to traditional desktop applications and is extremely relevant for Web applications.

Survey of the GDI+ Namespaces

The .NET Framework provides a number of namespaces devoted to two-dimensional graphical rendering. In addition to the basic functionality you would expect to find in a graphics package (color, font, pen, brush, and image manipulation), you also find types
that enable geometric transformations, antialiasing, palette blending, and document printing support. Collectively speaking, these namespaces make up the .NET facility we call GDI+, which is a vast improvement over the traditional Win32 Graphical Device Interface (GDI) API. Table 14-1 gives a high-level view of each major player.

_Table 14-1. The Core GDI+ Namespaces_

<table>
<thead>
<tr>
<th>GDI+ Namespace</th>
<th>Meaning in Life</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.Drawing</td>
<td>This is the key GDI+ namespace, which defines numerous types for basic rendering (fonts, pens, basic brushes, etc.) as well as the almighty Graphics type.</td>
</tr>
<tr>
<td>System.Drawing.Drawing2D</td>
<td>This namespace provides types used for more advanced two-dimensional graphics functionality (e.g., gradient brushes, pen-caps, geometric transforms, etc.).</td>
</tr>
<tr>
<td>System.Drawing.Imaging</td>
<td>This namespace defines types that allow you to directly manipulate graphical images (e.g., change the palette, extract image metadata, manipulate metafiles, and so forth).</td>
</tr>
<tr>
<td>System.Drawing.Printing</td>
<td>This namespace defines types that allow you to render images to the printed page, interact with the printer itself, and format the overall appearance of a given print job.</td>
</tr>
<tr>
<td>System.Drawing.Text</td>
<td>This (rather small) namespace allows you to manipulate collections of fonts. For example, as you see in this chapter, the InstalledFontCollection type allows you to dynamically discover the set of installed fonts on the target machine.</td>
</tr>
</tbody>
</table>

_Configuring a GDI+ Project Workspace_

When you wish to make use of GDI+, you must set a reference to the System.Drawing.dll assembly. This single binary contains definitions of the types for each of the core GDI+ namespaces. Be aware that if you select a new Windows Application Project Workspace using VS .NET, this reference is set on your behalf automatically. Other project types, however, may require you to set this assembly reference explicitly. In any case, once you have set this reference, just make use of the C# “using” keyword and you are ready to render:

```csharp
// Don't forget to reference System.Drawing.dll!
using System.Drawing;
```

To begin the GDI+ journey, let’s examine the functionality defined by the System.Drawing namespace.