Relleno de formas gráficas, texturas, degradados y otros efectos avanzados con GDI+

Luis Miguel Blanco Ancos

El conjunto de clases incluido en el espacio de nombres System.Drawing, provee al programador de todas aquellas características necesarias para dotar a los gráficos de efectos adicionales, que permitan aportarles una mayor vistosidad. A lo largo de este artículo, realizaremos un repaso de los principales elementos existentes en GDI+, de forma que sirvan de base al lector para la elaboración de sus propios efectos.

Pintando con trazo grueso. La clase Brush

Cuando dibujamos figuras en la superficie de un formulario utilizando la clase Pen, habremos comprobado que el interior de las mismas queda "hueco", es decir, vemos el fondo del formulario.

Si además de dibujar una figura, queremos que su interior quede relleno de un color o efecto basado en una combinación de colores, necesitamos realizar las siguientes operaciones:

En primer lugar instanciaremos un objeto de alguna de las clases derivadas del tipo Brush, como SolidBrush, TextureBrush, HatchBrush, etc., dependiendo del efecto de relleno que queramos lograr. Brush es una clase abstracta, y como tal, no podemos instanciar objetos directamente de ella.

Seguidamente deberemos obtener del formulario el objeto Graphics, que representa el contexto de dispositivo gráfico sobre el que vamos a dibujar. De esto se encarga el método Form.CreateGraphics.

Por último llamaremos a alguno de los métodos del objeto Graphics que comienzan por el nombre Fill, lo que dibujará la correspondiente figura, rellenándola con el color o efecto correspondiente al método usado. La siguiente tabla muestra alguno de los métodos disponibles. Para un mayor detalle, recomendamos al lector la consulta de la documentación sobre los tipos Brush disponible en la documentación de la plataforma .NET.

Método	**Tipo de figura**
FillRectangle	Rectángulo
FillEllipse	Elipse
FillClosedCurve	Curva cerrada, creada a partir de un array de tipos Point
FillPolygon	Polígono

El siguiente código fuente, crea una figura con forma ovalada, y rellena su interior de un color, utilizando los pasos descritos anteriormente.

```
' definir color
Dim oColor As Color
oColor = Color.FromName("Green")

' crear objeto brush
Dim oSolidB As New SolidBrush(oColor)

' obtener el dispositivo gráfico y crear dibujo
Dim oGraphics As Graphics = Me.CreateGraphics()
oGraphics.FillEllipse(oSolidB, New Rectangle(25, 45, 150, 50))
oGraphics.Dispose()
```

Como comentarios adicionales sobre el anterior código, cabe destacar que hemos utilizado un objeto SolidBrush, que es el más básico de todos los objetos de este tipo disponibles. El color lo hemos creado usando el método Color.FromName, pasando una cadena con el nombre del color a usar. No es el modo más directo ni habitual, ya que lo más sencillo es utilizar la lista de miembros de esta estructura, que contiene de modo inmediato los colores. Muy importante también es la llamada a Graphics.Dispose al finalizar el proceso, para liberar los recursos gráficos que hayamos estado utilizando. El resultado de ejecutar este ejemplo lo podemos ver en la siguiente imagen.

La clase TextureBrush. Gráficos con tapices

Supongamos que tenemos un archivo con una imagen que nos gustaría incluir como relleno o textura para una figura que dibujemos en el formulario.

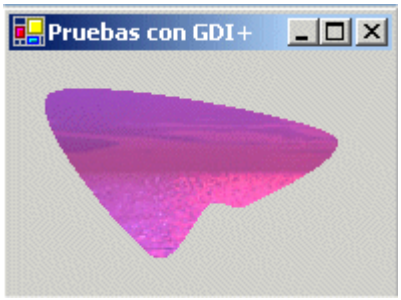
Con GDI+ nada más fácil. En primer lugar, para obtener una referencia hacia la imagen, debemos crear un objeto Bitmap, en cuyo constructor pasaremos la ruta del archivo. A continuación, crearemos un objeto TextureBrush, pasándole como parámetro el objeto Bitmap que apunta al archivo que usaremos como base para el relleno de la figura. El resto de pasos son los ya conocidos por el lector: obtener el objeto Graphics del formulario y llamar a cualquiera de sus métodos Fill.

Como ejemplo de uso de este tipo de objeto, en el código fuente mostrado a continuación, creamos una curva cerrada a partir de un array de estructuras Point,

usando el método Graphics.FillClosedCurve, rellenándola seguidamente con el contenido de una imagen proporcionada por un objeto TextureBrush.

```
' obtener imagen de archivo
Dim oBitmap As New Bitmap("E:\Pruebas\MiTextura.gif")

' crear textura de la imagen
Dim oTextureB As New TextureBrush(oBitmap)

' definir coordenadas de la figura
Dim aCoordenadas(4) As Point
aCoordenadas(0) = New Point(20, 20)
aCoordenadas(1) = New Point(70, 100)
aCoordenadas(2) = New Point(100, 75)
aCoordenadas(3) = New Point(120, 75)
aCoordenadas(4) = New Point(160, 40)

' obtener el dispositivo gráfico y crear dibujo
Dim oGraphics As Graphics = Me.CreateGraphics()
oGraphics.FillClosedCurve(oTextureB, aCoordenadas)

oGraphics.Dispose()
```

El resultado obtenido al ejecutar el anterior fuente será similar al que mostramos en la siguiente imagen.



Efectos avanzados con las clases Brush. Tramas y degradados

El espacio de nombres System.Drawing.Drawing2D contiene varias clases derivadas de Brush que permiten, a partir de la combinación de varios colores, la aplicación de efectos tales como tramas y degradados en el proceso de creación de formas gráficas.

La primera de estas clases con la que vamos a tratar será HatchBrush, mediante la cual, podemos crear una figura que muestre un efecto de trama combinando uno o dos colores.

El siguiente fuente muestra la creación de dos figuras con este tipo de clase, a las que aplicamos sendos efectos de tramado. Para seleccionar el tramado usaremos la

enumeración HatchStyle. Recuerde el lector que para usar este tipo de objeto es necesario importar el espacio de nombres System.Drawing.Drawing2D.

```
Imports System.Drawing.Drawing2D
'....
'....
Dim oHatchB1 As New HatchBrush(HatchStyle.HorizontalBrick, _
   Color.Aquamarine, Color.DarkMagenta)

Dim oHatchB2 As New HatchBrush(HatchStyle.SolidDiamond, _
   Color.Aquamarine, Color.DarkMagenta)

Dim oGraphics As Graphics = Me.CreateGraphics()
oGraphics.FillRectangle(oHatchB1, New Rectangle(40, 40, 100, 50))
oGraphics.FillEllipse(oHatchB2, New Rectangle(40, 110, 100, 50))

oGraphics.Dispose()
```

La siguiente imagen muestra el resultado.

La siguiente clase con la que trabajaremos será LinearGradientBrush, la cual efectúa un degradado o fundido de dos colores, entre dos coordenadas o una zona rectangular, esto depende del constructor utilizado a la hora de instanciar este objeto. El siguiente código fuente muestra un ejemplo de uso de esta clase.

```
Imports System.Drawing.Drawing2D
'....
'....
Dim oLinearGradB As New LinearGradientBrush(New Point(15, 10), _
   New Point(100, 75), _
   Color.DarkGreen, Color.LightSkyBlue)

Dim oGraphics As Graphics = Me.CreateGraphics()
oGraphics.FillRectangle(oLinearGradB, New Rectangle(15, 10, 80, 65))
oGraphics.Dispose()
```

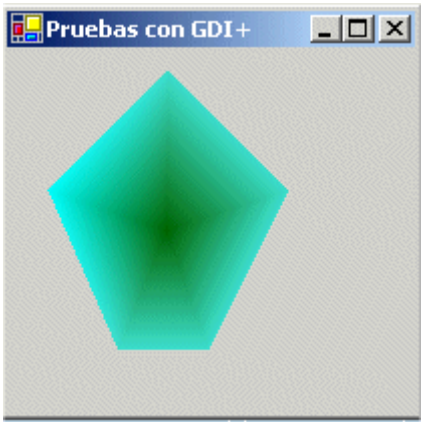
La figura, con el efecto de degradado aplicado a los dos colores del objeto, se muestra en la siguiente imagen.

Para terminar con este conjunto de clases pasaremos a PathGradientBrush, cuya finalidad consiste en rellenar figuras con un color para la zona central, y un array de colores para los bordes. Esta clase incorpora un mecanismo que se ocupa de aplicar el fundido entre los colores central y exteriores. El siguiente código de ejemplo crea un objeto Brush de este tipo.

```
Imports System.Drawing.Drawing2D
'....
'....
' coordenadas para realizar el degradado
Dim pntPuntos(4) As Point
pntPuntos(0) = New Point(20, 70)
pntPuntos(1) = New Point(80, 10)
pntPuntos(2) = New Point(140, 70)
pntPuntos(3) = New Point(100, 150)
pntPuntos(4) = New Point(55, 150)

' array de colores para los bordes de la figura
Dim aColores(1) As Color
aColores(0) = Color.Aqua
aColores(1) = Color.Turquoise

' crear objeto para el degradado
Dim oPathGradB As New PathGradientBrush(pntPuntos)
oPathGradB.SurroundColors = aColores
oPathGradB.CenterColor = Color.DarkGreen

Dim oGraphics As Graphics = Me.CreateGraphics()
' dibujar figura usando el objeto
' para el degradado
oGraphics.FillPolygon(oPathGradB, pntPuntos)

oGraphics.Dispose()
```

La propiedad SurroundColors nos permite establecer el array de colores para el borde de la figura, mientras que en la propiedad CenterColor asignamos el color central hacia el que se realizará el fundido o degradado de los colores exteriores.

La figura obtenida podemos verla en la siguiente imagen.



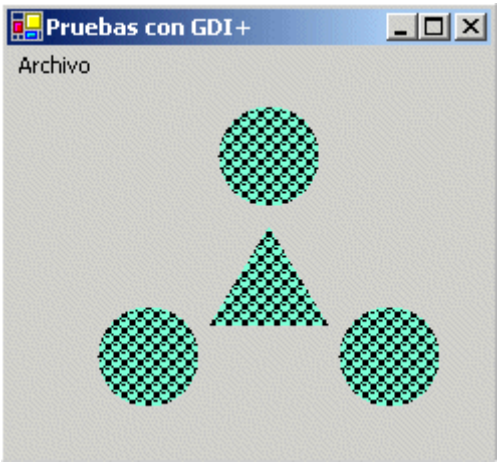
La clase GraphicsPath. Creación de una figura compleja a partir de varias simples

Esta clase, situada en el espacio de nombres System.Drawing.Drawing2D, representa a un contenedor de figuras. Tras crear un objeto de este tipo, utilizaremos los métodos que comienzan por Add para añadir figuras al contenedor. Una vez terminada la composición del gráfico, y dependiendo de si vamos a dibujarlo con un objeto Pen, o un subtipo de Brush, emplearemos el método DrawPath o FillPath respectivamente de la clase Graphics para plasmar la figura en la superficie del formulario. El siguiente código muestra un ejemplo de este tipo de objeto.

```
Imports System.Drawing.Drawing2D
'....
'....
' crear el objeto GraphicsPath
Dim oGPath As New GraphicsPath()

' añadir figuras dentro del objeto
Dim aPuntos(2) As Point
aPuntos(0) = New Point(100, 120)
aPuntos(1) = New Point(130, 70)
aPuntos(2) = New Point(160, 120)

oGPath.AddPolygon(aPuntos)

oGPath.AddEllipse(New Rectangle(105, 10, 50, 50))
oGPath.AddEllipse(New Rectangle(45, 110, 50, 50))
oGPath.AddEllipse(New Rectangle(165, 110, 50, 50))

Dim oGph As Graphics = Me.CreateGraphics
```

```
oGph.FillPath(New HatchBrush(HatchStyle.Sphere, Color.Aquamarine), _
   oGPath)

oGph.Dispose()
```

La combinación de figuras obtenidas se muestra en la siguiente imagen.



Dibujo de texto en formato gráfico. El método DrawString

Para dibujar en el formulario un texto con características gráficas, debemos utilizar el método Graphics.DrawString, especificando como parámetros, la cadena con el texto a mostrar, un objeto Font con el tipo de letra a visualizar, un objeto derivado de Brush que tenga el tipo de trazo para pintar, y una estructura PointF con las coordenadas en donde comenzará a dibujarse el texto. El siguiente código fuente realiza este proceso.

```
' objeto Font con el tipo de letra
Dim oTipoLetra As New Font("Lucida Console", _
   45, FontStyle.Italic, GraphicsUnit.Pixel)

' objeto SolidBrush
Dim oSBrush As New SolidBrush(Color.Olive)

Dim oGraphics As Graphics = Me.CreateGraphics()
' dibujar texto en modo gráfico
oGraphics.DrawString("Hola mundo desde GDI+", _
   oTipoLetra, oSBrush, New PointF(10, 10))

oGraphics.Dispose()
```

El texto dibujado en el formulario tras ejecutar este código podemos verlo en la siguiente imagen.

Por supuesto que el ejemplo anterior muestra la forma más *aburrida* de pintar texto usando GDI+. Como hemos dicho, al usar DrawString estamos empleando un objeto derivado de Brush para el dibujo del texto. Si en lugar de utilizar la clase SolidBrush, tomamos por ejemplo HatchBrush, no sólo dibujaremos el texto, sino que aplicaremos al mismo el efecto proporcionado por dicha clase. Además, podemos pasar a DrawString un objeto RectangleF, para definir con más precisión el área del formulario en la que vamos a dibujar. Veámoslo en el siguiente fuente.

```
Imports System.Drawing.Drawing2D
'....
'....
Dim oTipoLetra As New Font("Comic Sans MS", _
   55, FontStyle.Bold, GraphicsUnit.Pixel)

Dim oHBrush As New HatchBrush(HatchStyle.NarrowVertical, _
   Color.Cornsilk)

Dim oGraphics As Graphics = Me.CreateGraphics()
oGraphics.DrawString("Hola mundo desde GDI+", _
   oTipoLetra, oHBrush, New RectangleF(10, 10, 250, 400))

oGraphics.Dispose()
```

Con estas ligeras variaciones, el texto obtenido sería como el mostrado a continuación.

Como acabamos de comprobar, con un poco de imaginación y las clases de GDI+, podemos conseguir efectos realmente notables con menor esfuerzo del que requería la creación gráfica a base de llamadas al API de Windows, gracias al modelo robusto y cohesionado de clases que proporciona la plataforma .NET Framework.