Einstieg in VB.NET von René Martin
- Für Programmiereinsteiger -

Kapitel 11 Grafik

Kapitel 11 Grafik

Es passt zwar nicht ganz in das Grundthema des Datenaustauschs, das unser Buch als roter Faden durchzieht, allerdings bildet es einen netten Abschluss und zeigt eine der vielen weiteren Möglichkeiten, was man mit VB.NET noch alles machen kann: Grafikprogrammierung mit GDI+.

An einem einfachen Beispiel wird eine Gerade gezeichnet, die durch zwei beliebige Punkte geht.

Im Paint-Ereignis eines Formulars kann festgelegt werden, dass eine Linie gezeichnet werden soll:

```
Protected Overrides Sub OnPaint(ByVal e As _
System.Windows.Forms.PaintEventArgs)
   Dim g As Graphics
   Dim p As Pen
   p = New Pen(System.Drawing.Color.Black)
   g = e.Graphics
   g.DrawLine(p, 0, 0, 500, 500)
```

Beim Starten des Formulars wird die Linie durch die beiden Punkte (0;0) und (500;500) gezeichnet. Gerechnet wird in Pixeln; die Größen entsprechen den Größen, die für die Userform in den Eigenschaften angegeben werden. Der Ursprung (0;0) befindet sich links oben, gemessen wird nach rechts und nach unten mit positiven Zahlen.

Der Benutzer trägt nun in eine Startmaske zwei Wertepaare ein:

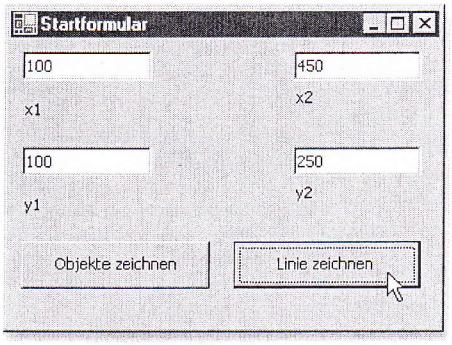

Abbildung 11.1 Das Startformular

Über den Zeichnen-Button wird ein zweites Formular aufgerufen:

```
Public Shared objFormIch As frmStart
Private Sub butZeichnen_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles butZeichnen.Click
   Dim objForm As Form
   objForm = New frmDiagramm()
   objForm.Text = "Liniendiagramm"
   objFormIch = Me
   objForm.Show()
End Sub
```

Das Objekt »objFormIch« wird im zweiten Teil benötigt, wenn das zweite Formular sich die Werte des ersten Formulars holt.

Mit ein wenig Grundkenntnissen der Schulmathematik (y = m*x + b) kann aus den beiden Punkten die Gerade berechnet werden. Damit die x- und y-Achsen nicht auf den Kanten des Formulars liegen, wurden sie um je 20 verschoben. Und so ergibt sich:

```
Protected Overrides Sub OnPaint(ByVal e As _
System.Windows.Forms.PaintEventArgs)
   Dim x1 As Integer = _
      CInt(frmStart.objFormIch.txtX1.Text) + 20
   Dim y1 As Integer = 500 - _
      CInt(frmStart.objFormIch.txtY1.Text) - 20
   Dim x2 As Integer = _
      CInt(frmStart.objFormIch.txtX2.Text) + 20
   Dim y2 As Integer = 500 - _
      CInt(frmStart.objFormIch.txtY2.Text) - 20
   ' y = m*x +b
   ' m = (y1-y2)/(x1-x2)
   ' b = y1 -m*x1
   Dim m As Double = (y1 - y2) / (x1 - x2)
   Dim b As Double = y1 - m * x1
   x1 = -20
   y1 = m * x1 + b
   x2 = 500 - 20
   y2 = m * x2 + b

   Dim g As Graphics
   Dim p As Pen
   p = New Pen(System.Drawing.Color.Black)
   g = e.Graphics
   With g
      .DrawLine(p, 0, 480, 500, 480)
      .DrawLine(p, 20, 500, 20, 0)
      .DrawLine(p, 120, 485, 120, 475)
      .DrawLine(p, 220, 485, 220, 475)
      .DrawLine(p, 320, 485, 320, 475)
      .DrawLine(p, 420, 485, 420, 475)

      .DrawLine(p, 15, 80, 25, 80)
      .DrawLine(p, 15, 180, 25, 180)
      .DrawLine(p, 15, 280, 25, 280)
      .DrawLine(p, 15, 380, 25, 380)

      .DrawLine(New Pen(System.Drawing.Color.Red), _
      x1, y1, x2, y2)
   End With
End Sub
```

Die Linie selbst wird lediglich durch den Befehl

```
      .DrawLine(New Pen(System.Drawing.Color.Red), _
      x1, y1, x2, y2)
```

erzeugt; die übrigen Linien bilden die beiden Achsen oder die Striche, die im Abstand von 100 gesetzt wurden.

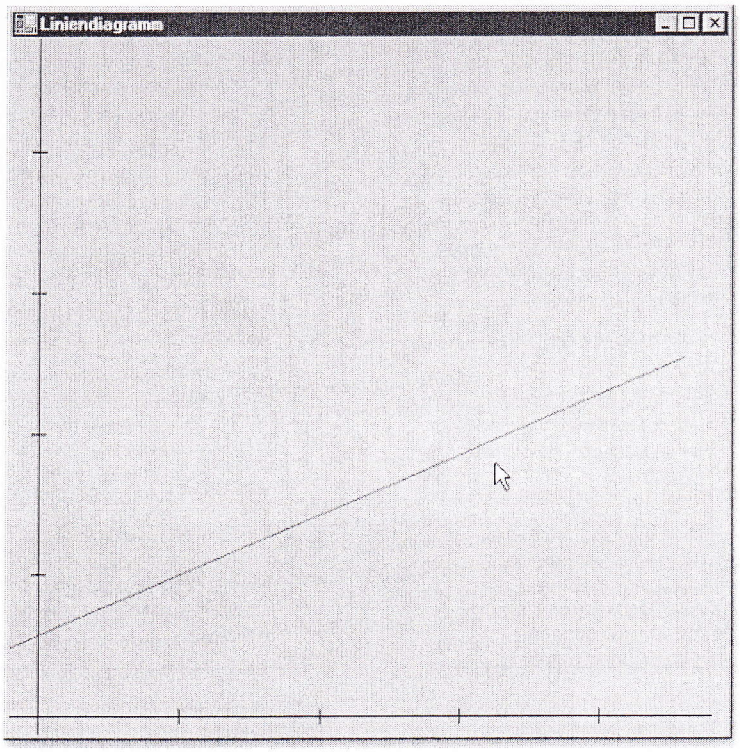

Abbildung 11.2 Die erzeugte Gerade

Man muss nicht nur das Paint-Ereignis eines anderen Formulars bemühen, man kann auch auf »sich selbst« zeichnen. Ein einfacher Kreis wird folgendermaßen erzeugt:

```
Dim g As Graphics
Dim p As Pen
```

```
Dim r As Rectangle

g = Me.CreateGraphics()
p = New Pen(System.Drawing.Color.Red, 5)
r = New Rectangle(10, 10, 200, 200)
g.DrawEllipse(p, r)
```

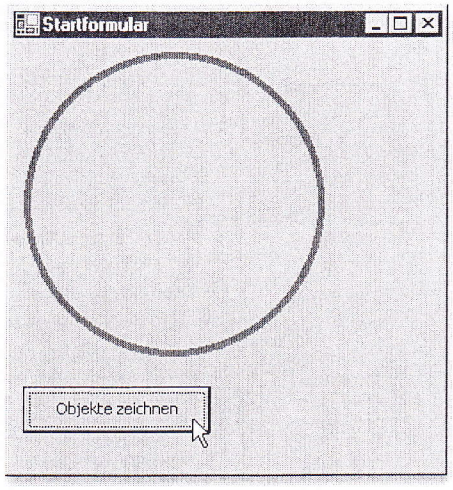

Abbildung 11.3 Der gezeichnete Kreis

Das Graphics-Objekt stellt folgende Methoden zur Verfügung:

Tabelle 11.1 Einige Methoden zum Zeichnen

Methode	Erklärung
Clear	löscht die Zeichenoberfläche
DrawArc	zeichnet einen Bogen
DrawBezier	zeichnet eine Bézierspline-Kurve
DrawClosedCurve	zeichnet eine geschlossene Kurve
DrawCurve	zeichnet eine Kurve
DrawEllipse	zeichnet eine Ellipse
DrawIcon	zeichnet ein Icon
DrawImage	zeichnet ein Bild
DrawLine	zeichnet eine Linie
DrawPie	zeichnet eine Kreisform
DrawPolygon	zeichnet ein Vieleck
DrawRectangle	zeichnet ein Rechteck
DrawString	zeichnet eine Textzeichenfolge
FillClosedCurve, FillEllipse, FillPie, FillPolygon, FillRectangle	füllt das Innere der entsprechenden geschlossenen Kurve
RotateTransform	dreht ein Objekt
Save	speichert den aktuellen Zustand dieses Graphics-Objekts

Die meisten Methoden verlangen vier Koordinaten (x, y, Width und Height), die über das Objekt »Rectangle« eingegeben werden, wie beispielsweise »DrawEllipse«, »DrawRectangle« oder »DrawLine«. Diese Parameter können aus dem Objekt als Eigenschaft wieder ausgelesen werden.

Viele der Methoden verlangen einen Pen. Dieses Objekt legt eine Reihe von Eigenschaften fest, wie beispielsweise »Color«, »Width« (die Breite der Linie), »DashStyle« (eine gestrichelte Linie), »StartCap« und »EndCap« (der Anfang und das Ende einer offenen Linie). Die Farben liegen – wie könnte es anders sein – in einer Klasse. »System.Drawing.Color« stellt beispielsweise Red, Blue, Yellow, Green, Violet, Orange, White, Black, aber auch Navy, Tomato, Lavender, Gold und 128 weitere Farben zur Verfügung. Wem das nicht genügt, der kann alle 16 Millionen Farben mit der Methode »FromArgb« erzeugen. »Rgb« bezeichnet dabei den Rot-, Grün- und Blau-Anteil.

Zwei Objekte sind in diesem Zusammenhang noch erwähnenswert: Mit »Image« wird ein Bild festgelegt, das normalerweise als Datei gespeichert vorliegt. Das Objekt »Brush« kann verschiedene Muster anzeigen und damit auch Bilder:

```
Dim g As Graphics
Dim r As Rectangle
Dim b As Brush
Dim bi As Image
```

```
r = New Rectangle(0, 0, 600, 600)
bi = New System.Drawing.Bitmap("D:\Data\PIC00025.gif")
b = New TextureBrush(bi)
g.FillRectangle(b, r)
```

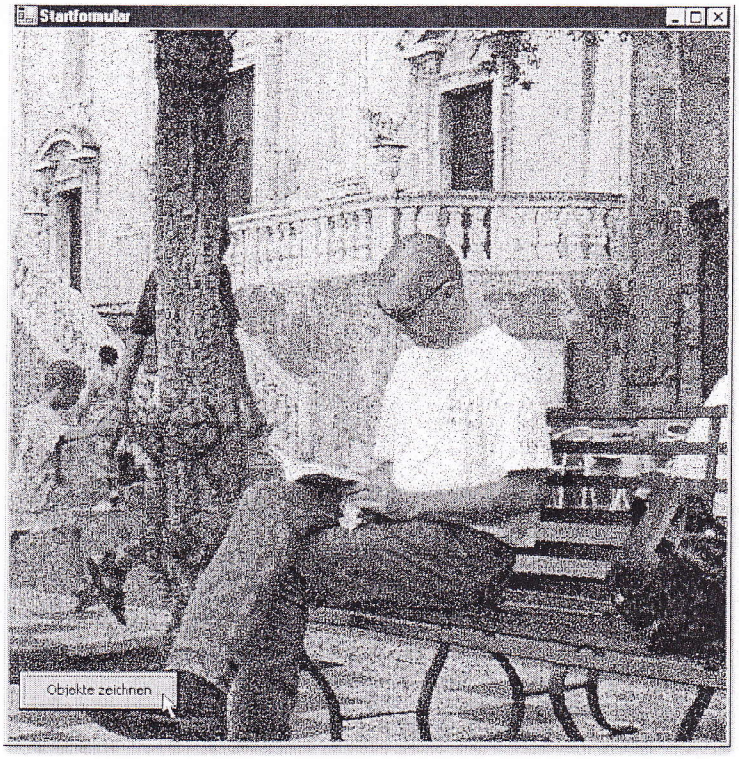

Abbildung 11.4 Auch ohne Steuerelemente können Bilder angezeigt werden.

<< zurück <top> vor >>

[Galileo Computing]

Galileo Press GmbH, Gartenstraße 24, 53229 Bonn, Tel.: 0228.42150.0, Fax 0228.42150.77, info@galileo-press.de