23.3.4.1 Projekt 1 – Zeichnen mit Paint

Im Projekt 1 werden Ihnen verschiedene Möglichkeiten gezeigt, wie Sie unterschiedliche Formen und Text mit Methoden der Klasse Paint zeichnen. Dabei wird in allen Beispielen ein Picture als Zeichenfläche eingesetzt, wobei die Begriffe Picture und Picture-Objekt hier synonym verwendet werden. Die Vorteile dieses Vorgehens wurden bereits im → Kapitel 23.3.3 Zeichen mit Paint vorgestellt. Neben dem gezeichneten Bild wird der verwendete Quelltext vorgestellt und kurz kommentiert, wenn das erforderlich ist. Diese Variablen gelten global im Projekt:

```
Public hFile As File
Public hPicture As Picture
Public sPicturePath As String = Application.Path &/ "Pictures"
Public xTranslate As Float = 40  ' Verschiebung Koordinatensystem x-Achse
Public yTranslate As Float = 320 ' Verschiebung Koordinatensystem y-Achse
Public xScale As Float = +1      ' Skalierung x-Richtung
Public yScale As Float = -1      ' Skalierung y-Richtung → +y ▲
```

Diese Prozeduren gelten für fast alle Beispiele:

```
Public Sub GenerateNewPicture()
  hPicture = New Picture(daCanvas.Width, daCanvas.Height, True) ' Ein Picture-Objekt erzeugen
  hPicture.Fill(&H00F5FFE6&) ' Option: Hintergrundfarbe für das Picture festlegen --> hellgrün
End ' GenerateNewPicture()
```

```
Public Sub SetPictureBorder()
  Paint.Begin(hPicture)
    Paint.LineWidth = 2 ' Dünner, grauer Bildrand
    Paint.Brush = Paint.Color(Color.Gray)
    Paint.Rectangle(0, 0, daCanvas.Width, daCanvas.Height) ' 640, 360
    Paint.Stroke
  Paint.End
End ' SetPictureBorder()
```

```
Public Sub daCanvas_Draw()
  Paint.Begin(daCanvas)
    If hPicture Then Paint.DrawPicture(hPicture, 0, 0) ' Das Picture wird in eine DrawingArea gezeichnet
  Paint.End
End ' daCanvas_Draw()
```

Die letzte Prozedur dient dazu, das Picture – das ja nur im Speicher existiert – anzuzeigen.

Die Auslagerung von Quelltext in einzelne Prozeduren beantwortet auch die Frage: Kann man einzelnen (Teil-)Zeichnungen in Prozeduren auslagern mit einem klaren JA. Zwischen *Paint.Begin(Device)* und *Paint.End()* ist die verwendete Zeichenfläche bekannt und jeder Paint-Aufruf wirkt auf diese Zeichenfläche. Es ist sogar von Vorteil, (Teil-)Zeichnungen in Prozeduren auszulagern, denn so können diese Prozeduren später aufgerufen werden, um zum Beispiel Koordinatenachsen zu zeichnen, wenn das erforderlich ist. Unterstützt wird dieses Vorgehen durch die Möglichkeit, einzelne *Paint.Begin(Device)*- und *Paint.End()*-Aufrufe zu verschachteln. Das Einbinden der Prozedur SetPictureBorder() in jedes Beispiel nutzt diese Verschachtelung.

Beispiel 1 – Zeichnen von Text

Abbildung 23.3.4.1.1: Gezeichneter Text

Als Text können Sie einfachen Text oder RichText verwenden. Im ersten Beispiel wird die Verwendung von RichText als Alternative vorgestellt.

Quelltext:

```
[1]  Public Sub PaintScriptText()
[2]    Dim textDimension As RectF
[3]    Dim sText1, sText2, sText As String
[4]    Dim X, Y, W, H As Float
[5]
[6]    hPicture = New Picture(daCanvas.Width, daCanvas.Height, True)
[7]    hPicture.Fill(&H00C3DDFF&) ' hellblau
[8]
[9]    Paint.Begin(hPicture)
[10]   ' Original-Koordinatensystem → +y-Richtung nach unten!
[11]     Paint.Brush = Paint.Color(&C3CCFF)
[12]     Paint.Rectangle(40, 40, 560, 280, 32)
[13]     Paint.Fill(True)
[14]     Paint.Brush = Paint.Color(Color.White)
[15]     Paint.LineWidth = 20
[16]     Paint.Stroke
[17]
[18]     Paint.Font.Name = "FreeSans"
[19]     Paint.Font.Size = 60
[20]     Paint.Font.Bold = True
[21]   ' Paint.Font = Font["FreeSans, 60, bold"] ' Kompakte Font-Beschreibung
[22]
[23]     sText1 = ("Drawing with")
[24]     sText2 = ("Paint")
[25]
[26]   ' Text 1
[27]     textDimension = Paint.TextSize(sText1)
[28]     X = daCanvas.W / 2 - textDimension.W / 2 ' zentriert
[29]     Y = 85
[30]     W = textDimension.W
[31]     H = textDimension.H
[32]     Paint.Brush = Paint.Color(Color.DarkBlue)
[33]     Paint.DrawTextShadow((sText1), X, Y, W, H,, 50, 0.8) ' Ausrichtung ,, wird nicht verwendet
[34]     Paint.Brush = Paint.Color(Color.DarkBlue)
[35]     Paint.DrawText((sText1), X, Y, W, H)
[36]
[37]   ' Text 2
[38]     textDimension = Paint.TextSize(sText2)
[39]     X = daCanvas.W / 2 - textDimension.W / 2
[40]     Y = 85 + Paint.TextSize(sText1).H - 20
[41]     W = textDimension.W
[42]     H = textDimension.H
[43]     Paint.Brush = Paint.Color(Color.DarkBlue)
[44]     Paint.DrawTextShadow((sText2), X, Y, W, H,, 50, 0.8)
[45]     Paint.Brush = Paint.Color(Color.DarkBlue)
[46]     Paint.DrawText((sText2), X, Y, W, H)
[47]
[48]   ' Alternative mit RichText und viel probieren - bis alles passt
[49]   ' Paint.Brush = Paint.Color(Color.Blue)
[50]   ' Paint.DrawRichTextShadow(("Drawing with Paint"), 0, 0, 640, 340, Align.Center, 50, 0.8)
[51]   ' Paint.Brush = Paint.Color(Color.DarkBlue)
[52]   ' Paint.DrawRichText(("Drawing with Paint"), 0, 0, 640, 340, Align.Center)
[53]
[54]   Paint.End
[55]
[56] End ' PaintScriptIntro()
```

Kommentar:

- Zuerst wird in den Zeilen 6 und 7 ein Picture-Objekt erzeugt und dessen Hintergrund-Farbe auf hellblau gesetzt. Dann wird ein Rechteck mit abgerundeten 'Ecken' in einem anderen Blauton gezeichnet und dieses danach mit einem dicken weißen Rand versehen.
- In den Zeilen 18 bis 20 werden Font-Eigenschaften einzeln zugewiesen. Die Zeile 21 stellt eine kompakte Alternative vor.
- Die Eigenschaft *Paint.TextSize(text)* in der Zeile 27 gibt die Maße für ein Rechteck zurück, in dem der Text Platz findet.
- In der Zeile 28 wird der x-Wert berechnet, so dass der Text zentriert – in Bezug auf die Zeichenfläche – gezeichnet wird. Den y-Wert legen Sie in geeigneter Weise fest. Bei einem Klick in die Zeichenfläche werden stets die Koordinaten des Punktes ausgegeben, den Sie angeklickt hatten. Diese Funktionalität sollten Sie bei der *Erprobung eines Projektes* zum Zeichnen mit den Klassen Paint oder Cairo stets eingeschaltet lassen.
- Die Länge und Höhe des (Text-)Rechtecks werden in den Zeilen 30 und 31 ausgelesen und den beiden Variablen W und H zugewiesen.
- Ein Textschatten wird in der Zeile 33 gezeichnet. Mit den beiden Argumenten Radius (→ 50)

und Opacity (→ 0.8) und der Schattenfarbe sollten Sie unbedingt experimentieren, um einen guten Effekt zu erzielen.

- In der Zeile 35 und 46 zeichnen Sie jeweils den (Teil-)Text.

Sie erkennen bestimmt, dass das Zeichnen von Text zwar möglich ist und gute Ergebnisse liefert, aber sehr aufwendig ist. Deshalb wird Ihnen in den vier Zeilen 49 bis 53 eine schnelle Alternative im Rich-Text-Format vorgestellt, die ähnliche Ergebnisse liefert.

Beispiel 2 – Zeichnen von einzelnen Punkten

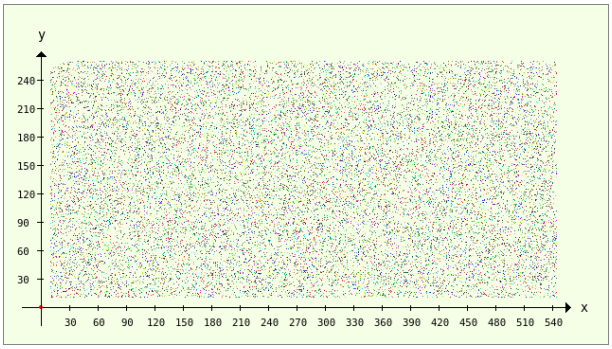

Abbildung 23.3.4.1.2: Muster aus zufälligen, farbigen Punkten

Es gibt in der Klasse Paint keine Methode, um einen einzelnen Punkt zu zeichnen. Mit diesem Ansatz gelingt das ohne Probleme:

```
Public Sub SetPoint(X As Integer, Y As Integer, cColor As Integer)
  Paint.AntiAlias = False
    Paint.FillRect(X, Y, 1, 1, cColor)
  Paint.AntiAlias = True
End ' SetPoint(..)
```

Seien Sie aber nicht enttäuscht, wenn Sie auf der Zeichenfläche den einzelnen Punkt nur schwer ausmachen können – er ist so winzig. Aus diesem Grunde wird Ihnen der Quelltext für eine gut sichtbare Punktwolke vorgestellt.

Quelltext:

```
[1]  Public Sub PaintScriptPointCloud()
[2]    Dim i As Integer
[3]
[4]    GenerateNewPicture()
[5]    SetPictureBorder()
[6]    Paint.Begin(hPicture)
[7]      Paint.Translate(xTranslate, yTranslate)
[8]      Paint.Scale(xScale, yScale) ' +y ▲
[9]      Paint.AntiAlias = False
[10]     DrawCoordinateSystem()
[11]     For i = 1 To 20000
[12]     ' Punkte mit zufälligen Koordinaten (im festen Raster) und einer Zufallsfarbe
[13]       SetPoint(Rnd(10, 550 - 5), Rnd(10, 260), Color.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255)))
[14]     Next
[15]     Paint.AntiAlias = True
[16]   Paint.End
[17]
[18] End ' PaintScriptPointCloud()
```

Kommentar:

- In der Zeile 10 wird mit der Prozedur *DrawCoordinateSystem()* ein komplettes Koordinaten-System eingezeichnet, für dessen 11 (Teil-)Zeichnungen Sie den kompletten Quelltext nachlesen können:

```
Public Sub DrawCoordinateSystem()
    Draw_X_Axis()
    Draw_X_AxisArrow()
    Draw_X_AxisScale()
    Draw_X_AxisLabels()
    Draw_X_AxisCaption()
```

```
      SetOrigin()
    Draw_Y_Axis()
    Draw_Y_AxisArrow()
    Draw_Y_AxisScale()
    Draw_Y_AxisLabels()
    Draw_Y_AxisCaption()
End ' DrawCoordinateSystem()
```

```
Private Sub Draw_X_Axis()
' x-Achse
  Paint.AntiAlias = False
    Paint.MoveTo(-20, 0)
    Paint.LineTo(560, 0)
    Paint.Stroke
  Paint.AntiAlias = True
End ' Draw_X_Axis()

Public Sub Draw_X_AxisArrow()
  Dim i As Integer = 1
' x-Achse-Pfeil
  Paint.AntiAlias = False
    For i = 1 To 5
      Paint.MoveTo(553 + i, 6 - i)
      Paint.LineTo(553 + i, -7 + i)
      Paint.Stroke
    Next
  Paint.AntiAlias = True
End ' Draw_X_AxisArrow()

Public Sub Draw_X_AxisScale()
  Dim i As Integer = 1
' x-Skale
  Paint.AntiAlias = False
    For i = 1 To 18
      Paint.MoveTo(30 * i, 2)
      Paint.LineTo(30 * i, -4)
      Paint.Stroke
    Next
  Paint.AntiAlias = True
End ' Draw_X_AxisScale()

Public Sub Draw_X_AxisLabels()
  Dim i As Integer = 1
' Beschriftung x-Achse
  Paint.Scale(1, -1) ' +y ▼
    Paint.Font = Font["Monospace, 8"]
    Paint.Brush = Paint.Color(Color.Black)
    For i = 1 To 3 ' zweistellig
      Paint.DrawText(Str(30 * i), 25 + (i - 1) * 30, 20) ' Iterationsgleichung
    Next
    For i = 4 To 18 ' dreistellig
      Paint.DrawText(Str(30 * i), 22 + (i - 1) * 30, 20) ' Iterationsgleichung
    Next
  Paint.Scale(1, -1)
End ' Draw_X_AxisLabels()

Public Sub Draw_X_AxisCaption()
  Paint.AntiAlias = False
    Paint.Scale(1, -1)
    Paint.Font = Font["Monospace, 10"]
    Paint.DrawText("x", 570, 5)
    Paint.Font = Font["Monospace, 8"]
    Paint.Scale(1, -1)
  Paint.AntiAlias = True
End ' Draw_X_AxisCaption()

Public Sub Draw_Y_Axis()
' y-Achse
  Paint.AntiAlias = False
    Paint.MoveTo(0, -20)
    Paint.LineTo(0, 270)
    Paint.MoveTo(0, 0)
    Paint.Stroke
  Paint.AntiAlias = True
End ' Draw_Y_Axis()

Public Sub Draw_Y_AxisArrow()
  Dim i As Integer = 1
' y-Achse-Pfeil
  Paint.AntiAlias = False
    For i = 1 To 5
      Paint.MoveTo(-6 + i, 264 + i)
      Paint.LineTo(7 - i, 264 + i)
      Paint.Stroke
```

```
    Next
  Paint.AntiAlias = True
End ' Draw_Y_AxisArrow()

Public Sub Draw_Y_AxisScale()
  Dim i As Integer = 1
' y-Skale
  Paint.AntiAlias = False
    For i = 1 To 8
      Paint.MoveTo(2, 30 * i)
      Paint.LineTo(-4, 30 * i)
      Paint.Stroke
    Next
  Paint.AntiAlias = True
End ' Draw_Y_AxisScale()

Public Sub Draw_Y_AxisLabels()
  Dim i As Integer = 1
' Beschriftung y-Achse
  Paint.Scale(1, -1)
    For i = 1 To 8
      Paint.DrawText(Str(30 * i), -25, -25 - (i - 1) * 30) ' Iterationsgleichung
    Next
  Paint.Scale(1, -1)
End ' Draw_Y_AxisLabels()

Public Sub Draw_Y_AxisCaption()
  Paint.Scale(1, -1)
    Paint.Font = Font["Monospace, 10"]
    Paint.DrawText("y", -3, -283)
    Paint.Font = Font["Monospace, 8"]
  Paint.Scale(1, -1)
End ' Y_AxisCaption()

Private Sub SetOrigin()
' Koordinatenursprung
  Paint.Brush = Paint.Color(Color.Red)
  Paint.MoveTo(0, 0)
  Paint.Arc(0, 0, 2)
  Paint.Fill()
  Paint.Brush = Paint.Color(Color.Black)
End ' SetOrigin()
```

Kommentar:

- Eine Besonderheit zeigt sich nicht nur in der vorletzten Prozedur *Draw_Y_Axis()*: Es wird das Koordinatensystem zwei Mal skaliert. Der Hintergrund ist simpel – sollte aber immer im Hinterkopf bleiben. Zum Zeichnen ist es günstig, das Koordinatensystem geeignet zu verschieben, um Bezeichnungen der Koordinatenachse gut lesbar anzubringen und die y-Achse nach oben zeigen zu lassen. Wenn Sie jetzt aber Text zeichnen wollen, so sehen Sie ihn spiegelverkehrt. Das erste *Paint.Scale(1, -1)* invertiert deshalb – nur zum Text-Zeichnen – die Richtung der y-Achse wieder nach unten und das zweite *Paint.Scale(1, -1)* lässt die positive y-Achse wieder nach oben richten, nachdem der Text gezeichnet wurde.
- Zusätzlich müssen Sie darauf achten, die y-Koordinate für den zu zeichnenden Text ebenso zu invertieren!

Wenn Sie gut sichtbare Punkte auf die Zeichenfläche setzen wollen, dann bietet sich die Möglichkeit, kleine farbige Kreise zu zeichnen:

```
Paint.Brush = Paint.Color(Color.Blue)
  Paint.Arc(Mx, My, 3)
Paint.Fill
```

Beispiel 3 – Darstellung einer Funktion

Die Darstellung einer Funktion y = f(x) erfolgt in diesem Beispiel weitgehend statisch, denn es werden die Funktion, der Definitionsbereich und auch der Wertebereich fest vorgegeben. Es wird erst im Kapitel 23.3.5.7 ein vollständiger Funktionsplotter mit erweiterter Funktionalität vorgestellt.

Sie erkennen im Quelltext, dass nur einige Teil-Zeichnungen des Koordinatensystems verwendet werden, weil durch den gewählten Maßstab für Abszisse und Ordinate eine andere Skalen-Einteilung notwendig wird. Alle Beschreibungen und die Funktionsbezeichnung werden erst *nach* dem Zeichnen des Funktionsbildes eingefügt.

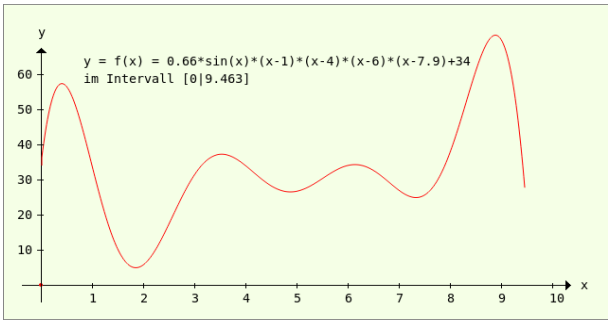

Abbildung 23.3.4.1.3: Bild der Funktion f(x) im angegebenen Intervall

Quelltexte:

```
Public Function f(x As Float) As Float
  Return 0.66 * Sin(x) * (x - 1) * (x - 4) * (x - 6) * (x - 7.9) + 34
End ' f(x)
```

```
Public Sub PaintScriptFunction()
  Dim X0, Y0, X1, Y1 As Float
  Dim i As Integer
  Dim sLabel As String

  GenerateNewPicture()
  SetPictureBorder()
  Paint.Begin(hPicture) ' MOTTO: ERST ZEICHNEN UND DANN BESCHRIFTEN!
  ' Verschiebung des Koordinatensystems
    Paint.Translate(xTranslate, yTranslate)
  ' Skalierung des Koordinatensystems
  ' Positive x-Achse nach rechts - positive y-Achse zeigt nach oben
    Paint.Scale(xScale, yScale) ' +y ▲
    Paint.AntiAlias = False ' False ist Standard
    Paint.LineWidth = 1 ' LinienBreite = 1 ist Standard
    Paint.Brush = Paint.Color(Color.Black) ' Zeichenfarbe schwarz ist Standard

    Draw_X_Axis()
    Draw_X_AxisArrow()
    Draw_X_AxisCaption()
      SetOrigin()
    Draw_Y_Axis()
    Draw_Y_AxisArrow()
    Draw_Y_AxisCaption()

    Paint.AntiAlias = True
    Paint.Brush = Paint.Color(Color.Red)

    X0 = 0
    Y0 = 4 * f(0) ' Passender Maßstab ' f(0) = 34
    Paint.MoveTo(X0 + 1, Y0)

    While i < 511
      Inc i
      X1 = i
      Y1 = 4 * f(i * (1 / 54)) ' Kontrolle Argumente: Print i * (1 / 54)
      Paint.LineTo(X1, Y1)
    Wend
    Paint.Stroke
    Paint.AntiAlias = False
  ' x-Skale
    Paint.Brush = Paint.Color(Color.Black)

    For i = 1 To 10
      Paint.MoveTo(54 * i, 2)
      Paint.LineTo(54 * i, -4)
      Paint.Stroke
    Next
  ' y-Skale
    For i = 1 To 6
      Paint.MoveTo(2, 40 * i)
      Paint.LineTo(-4, 40 * i)
      Paint.Stroke
    Next
  ' TEXT
    Paint.NewPath
    Paint.Scale(1, -1) ' +y ▼
    Paint.Font = Font["Monospace, 10"]
    Paint.Brush = Paint.Color(Color.Black)
```

```
    Paint.DrawText("y = f(x) = 0.66*sin(x)*(x-1)*(x-4)*(x-6)*(x-7.9)+34", 45, -250)
    sLabel = ("in the interval")
    Paint.DrawText(sLabel & " [0|9.463]", 45, -230)

  ' Beschriftung x-Achse
    For i = 1 To 10
      Paint.DrawText(Str(1 * i), 51 + (i - 1) * 54, 20)
    Next
  ' Beschriftung y-Achse
    For i = 1 To 6
      Paint.DrawText(Str(10 * i), -25, -35 - (i - 1) * 40)
    Next
   Paint.Scale(1, -1) ' +y ▲
Paint.End

End ' PaintScriptFunction()
```

Beispiel 4 – Visualisierung von Daten – Kreis-Diagramm

Seine Stärken spielt die Klasse Paint aus, wenn es zum Beispiel um die grafische Darstellung von Daten in einem Kreis- oder Säulen-Diagramm geht.

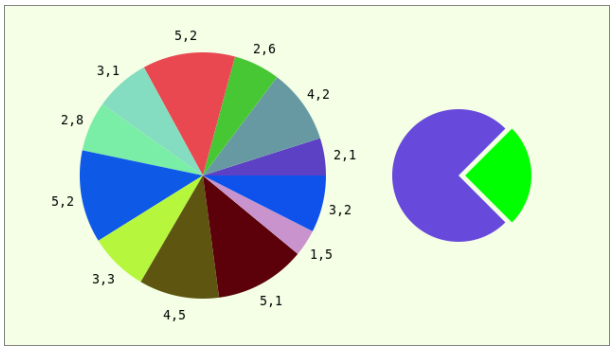

Abbildung 23.3.4.1.4: Kreis-Diagramm

Mit dem u.a. Quelltext können Sie ein Kreis-Diagramm mit Angabe des zu jedem Sektor gehörenden Wertes zeichnen. Wichtige Passagen im Quelltext werden kommentiert.

```
[1]  Public Sub PaintScriptChart()
[2]    Dim i As Integer
[3]    Dim fStartAngle, fTotal, fPx, fPy, fRadius, fMx, fMy, fRadiusOffset As Float
[4]    Dim fSumme As Float = 0
[5]    Dim textDimension As RectF
[6]    Dim aData, aAngle As Float[]
[7]
[8]    fMx = 170
[9]    fMy = 140
[10]   fRadius = 130
[11]   fRadiusOffset = 20
[12]
[13] ' Inline-Array mit den darzustellenden Werten
[14]   aData = [2.1, 4.2, 2.6, 5.2, 3.1, 2.8, 5.2, 3.3, 4.5, 5.1, 1.5, 3.2]
[15]   For i = 0 To aData.Max
[16]       fTotal = fTotal + aData[i]
[17]   Next
[18] ' aAngle => Array für die Daten-Winkel-Äquivalente (Bogenmaß)
[19]   aAngle = New Float[aData.Count]
[20]   For i = 0 To aData.Max
[21]       aAngle[i] = (aData[i] / fTotal) * Pi(2) ' Umrechnung 'Wert' in sein 'Winkel-Äquivalent'
[22]   Next
[23]
[24]   GenerateNewPicture()
[25]   SetPictureBorder()
[26]   Paint.Begin(hPicture)
[27]     Paint.Translate(xTranslate, yTranslate)
[28]     Paint.Scale(xScale, yScale) ' +y ▲
[29]
[30]     fStartAngle = 0
[31]   ' Kreis-Sektoren zeichnen - Anzahl: aData.Max
[32]     For i = 0 To aData.Max
[33]       Paint.Brush = Paint.Color(Color.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255)))
[34]       Paint.Arc(fMx, fMy, fRadius, fStartAngle, aAngle[i], True) ' Kreis-Sektor
[35]       Paint.Fill
[36]       fStartAngle = fStartAngle + aAngle[i] ' Neuer Startwinklel
[37]     Next
```

```
[38]
[39]    ' Werte an den zugehörenden Sektor zeichnen - Anzahl: aData.Max
[40]      For i = 0 To aData.Max
[41]        fStartAngle = fSumme + aAngle[i] / 2
[42]        fSumme = fSumme + aAngle[i]
[43]        fPx = (fRadius + fRadiusOffset) * Cos(fStartAngle) + fMx
[44]        fPy = (fRadius + fRadiusOffset) * Sin(fStartAngle) + fMy
[45]
[46]        textDimension = Paint.TextSize(aData[i])
[47]        fPx = fPx - (textDimension.Width / 2)
[48]        fPy = fPy - (textDimension.Height / 3)
[49]
[50]        Paint.Scale(1, -1) ' +y ▼
[51]          Paint.Brush = Paint.Color(Color.Black)
[52]          Paint.Font = Font["Monospace, 10"]
[53]          Paint.DrawText(Str(aData[i]), fPx, - fPy)
[54]        Paint.Scale(1, -1) ' +y ▲
[55]      Next
[56]
[57]    ' 2. Bild mit hervorgehobenem Sektor - Zusatz
[58]      Paint.Brush = Paint.Color(Color.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255)))
[59]      Paint.Arc(440, fMy, 70, Pi / 4, 3 * Pi / 2, True)
[60]      Paint.Fill
[61]      Paint.Brush = Paint.Color(Color.Green)
[62]      Paint.Arc(440 + 7, fmy, 70, - Pi(0.25), Rad(90), True)
[63]      Paint.Fill
[64]    Paint.End
[65]
[66] End ' PaintScriptChart()
```

Kommentar:

- Als Daten-Quelle für die 12 Werte dient ein Array (Inline-Array) in der Zeile 14.
- Von allen Werten im Array wird in den Zeilen 15 bis 17 die Summe *fTotal* berechnet.
- In den Zeilen 20 bis 22 werden die zu jedem Wert gehörenden Winkel-Äquivalente berechnet und in einem weiteren Array *aAngle* im Bogenmaß gespeichert.
- Das Zeichnen der einzelnen Sektoren mit zufällig berechnetem Farbwert wird in den Zeilen 32 bis 37 realisiert, wobei für jeden Sektor sein Start-Winkel neu errechnet wird (Zeile 36).
- Die Beschriftung der einzelnen Sektoren mit dem dazugehörenden Wert in der Verlängerung der Winkelhalbierenden jedes Sektors – dazu dient der Wert der Variablen *fRadiusOffset* – erfolgt in den Zeilen 40 bis 55. Interessant sind dabei nicht die Anweisungen zum Zeichnen der Werte-Texte (Zeile 53), sondern die Berechnungen der Positionen der Werte-Texte. Über die Eigenschaft *Paint.TextSize(text)* werden die Positionen fPx und fPy berechnet.
- Für eine große Anzahl von Werten müssen Sie die Schriftgröße im Font u.U. stark verkleinern und den Offset erhöhen.
- Die 2. Zeichnung dient nur der Veranschaulichung, wie man einen dominanten Sektor herausheben kann. Diese Teil-Zeichnung steht aber in keinem Zusammenhang mit dem linken Kreisdiagramm.

Beispiel 5 – Visualisierung von Daten – Säulen-Diagramm

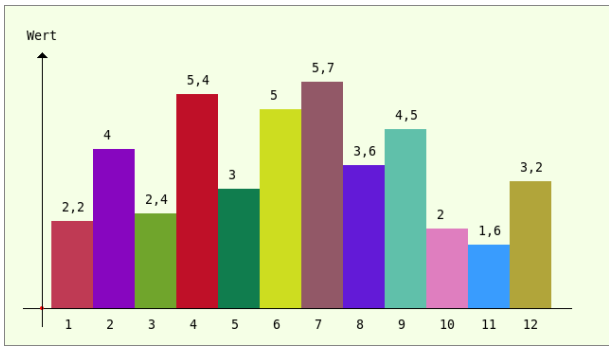

Abbildung 23.3.4.1.5: Säulen-Diagramm

Im Quelltext für das Säulen-Diagramm leistet die Anweisung in der Zeile 32 die Hauptlast – das Zeichnen der einzelnen Rechtecke (Säulen) – deren Höhe ihrem relativen Wert entspricht.

```
[1]  Public Sub PaintScriptBarChart()
[2]  Dim i As Integer
[3]    Dim fDeltaX, fDeltaY, fBeginX, fEndX, fEndY, fMaxValue As Float
```

```
[4]    Dim aData, aDataC As Float[]
[5]    Dim sCaption As String
[6]
[7]  ' Inline-Array mit den darzustellenden Werten
[8]    aData = [2.2, 4, 2.4, 5.4, 3, 5.0, 5.7, 3.6, 4.5, 2.0, 1.6, 3.2]
[9]    aDataC = aData.Copy()    ' Kopie des Arrays der Original-Werte
[10]   aDataC.Sort(gb.Descent) ' Absteigende Sortierung der Elemente
[11]   fMaxValue = aDatac[0]    ' Das 1. Element ist jetzt der größte Wert im Array
[12]
[13]   fBeginX = 10 ' Festlegung des Zeichenbereiches Abszisse
[14]   fEndX = 540
[15]   fEndY = 240  ' Festlegung des Zeichenbereiches Ordinate
[16]   fDeltaX = Round((fEndX - fBeginX) / aData.Count, 0) ' Normierte Streifenbreite (Einheit)
[17]   fDeltaY = Round(fEndY / fMaxValue, 0)               ' Normierte Streifenhöhe (Einheit)
[18]
[19]   GenerateNewPicture()
[20]   SetPictureBorder()
[21]   Paint.Begin(hPicture)
[22]     Paint.Translate(xTranslate, yTranslate)
[23]     Paint.Scale(xScale, yScale) ' +y ▲
[24]
[25]     Draw_X_Axis()
[26]     SetOrigin()
[27]     Draw_Y_Axis()
[28]     Draw_Y_AxisArrow()
[29]
[30]     For i = 0 To aData.Max
[31]   ' Berechnung und Anzeige aller Rechtecke im Diagramm mit zufälliger Streifen-Farbe und Wert
[32]       Paint.FillRect(fBeginX + i * fDeltaX, 0, fDeltaX, fDeltaY * aData[i], Color.RGB(Rnd(0, 255),
                      Rnd(0, 255), Rnd(0, 255)))
[33]       Paint.Scale(1, -1) ' +y ▼
[34]         Paint.Brush = Paint.Color(Color.Black)
[35]         Paint.Font = Font["Monospace, 10"]
[36]         Paint.DrawText(Str(aData[i]), fBeginX + 0.25*fDeltaX + i*fDeltaX, -(fDeltaY * aData[i] + 10))
[37]         Paint.Font = Font["Monospace, 10"]
[38]         Paint.DrawText(Str(i + 1), fBeginX + 0.33 * fDeltaX + i * fDeltaX, 22)
[39]       Paint.Scale(1, -1) ' +y ▲
[40]     Next
[41]
[42]   ' y-Achse beschriften
[43]     Paint.Scale(1, -1) ' +y ▼
[44]       sCaption = ("Value")
[45]       If Paint.TextSize(sCaption).W / 2 > xTranslate Then
[46]          Paint.DrawText(sCaption, 0 - xTranslate / 2, -283)
[47]       Else
[48]          Paint.DrawText(sCaption, 0 - Paint.TextSize(sCaption).W / 2, -283)
[49]       Endif
[50]       Paint.Font = Font["Monospace, 8"]
[51]     Paint.Scale(1, -1) ' +y ▲
[52]   Paint.End
[53]
[54] End ' PaintScriptBarChart()
```

Kommentar:

- Als Daten-Quelle für die 12 Werte dient auch in diesem Säulen-Diagramm ein Array (Inline-Array) (Zeile 8).
- In den Zeilen 9 bis 11 wird der größte Wert im Array *aData* über die Array-Kopie ermittelt und der variablen *fMaxValue* zugewiesen.
- Aus den einzelnen Werten und dem Wert von *fMaxValue* werden die relativen Säulen-Höhen berechnet (Zeile 16).
- In der Zeile 17 erfolgt die Berechnung der Säulenbreite aus den Vorgaben fBeginX und fEndX in den Zeilen 13 und 14.
- Das Zeichnen der einzelnen Säulen mit zufällig berechnetem Farbwert wird in den Zeilen 30 bis 40 realisiert. Anschließend werden die Werte-Texte über der Säule und die Säulen-Nummer eingezeichnet.
- In den Zeilen 42 bis 52 wird ein Bezeichner – hier "Value" – eingezeichnet. Für längere Bezeichner gibt es eine Sonderlösung. In der deutschen Version steht dort die Zeichenkette "Wert".

Beispiel 6 – Füllmuster für Flächen

Es erhöht die Qualität von gezeichneten Flächen, wenn Sie diese mit einem einfarbigen oder auch farbigen Muster füllen. Im Beispiel werden zum Teil die vorhandenen schwarz-weiß-Muster der Klasse Draw oder selbst geschaffene Muster eingesetzt. Im Quelltext werden die Muster aus einzelnen Bild-Dateien ausgelesen. Die entsprechenden Zeilen sind farbig markiert.

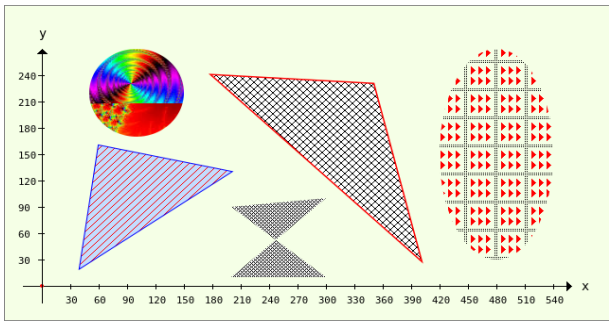

Abbildung 23.3.4.1.6: Säulen-Diagramm

```
[1]  Public Sub PaintScriptPattern()
[2]    Dim hPattern As Image
[3]
[4]    GenerateNewPicture()
[5]    SetPictureBorder()
[6]    Paint.Begin(hPicture)
[7]      Paint.Translate(xTranslate, yTranslate)
[8]      Paint.Scale(xScale, yScale) ' +y ▲
[9]
[10]     DrawCoordinateSystem()
[11]     Paint.AntiAlias = True
[12]     hPattern = Image.Load("Pattern/18.png")
[13]     Paint.Brush = Paint.Image(hPattern)
[14]     Paint.Arc(100, 220, 50)
[15]     Paint.Fill
[16]
[17]     Paint.LineWidth = 2
[18]     Paint.Brush = Paint.Color(Color.Blue)
[19]     Paint.Polygon([40, 20, 60, 160, 200, 130]) ' A(40|20), B(60|160) und C(200|130)
[20]     Paint.Stroke(True)
[21]     hPattern = Image.Load("Pattern/15.png")
[22]     Paint.Brush = Paint.Image(hPattern)
[23]     Paint.Fill
[24]
[25]     Paint.LineWidth = 3
[26]     Paint.Brush = Paint.Color(Color.Red)
[27]     Paint.Polygon([180, 240, 400, 30, 350, 230])
[28]     Paint.Stroke(True)
[29]     hPattern = Image.Load("Pattern/14.png")
[30]     Paint.Brush = Paint.Image(hPattern)
[31]     Paint.Fill
[32]
[33]     hPattern = Image.Load("Pattern/6.png")
[34]     Paint.Brush = Paint.Image(hPattern)
[35]   ' Punkte A(200|10), B(300|10), C(300|100), D(200|90)
[36]     Paint.Polygon([200, 10, 300, 100, 200, 90, 300, 10]) ' Reihenfolge der Punkte beachten ACDB!
[37]     Paint.Fill
[38]
[39]     hPattern = Image.Load("Pattern/17.png")
[40]     Paint.Brush = Paint.Image(hPattern)
[41]     Paint.Ellipse(420, 30, 120, 240)
[42]     Paint.Fill
[43]     Paint.AntiAlias = False
[44]   Paint.End
[45]
[46] End ' PaintScriptPattern()
```

Beispiel 7 – Radial-Gradient

Die Effekte, die sich mit der Methode Paint.RadialGradient erzeugen lassen, werden auch Sie überzeugen:

```
[1]  Public Sub PaintScriptRadialGradient()
[2]    Dim aColors As Integer[] = [Color.Blue, Color.White]
[3]    Dim aPositions As Float[] = [1, 0.05]
[4]    Dim aColors2 As Integer[] = [Color.Blue, Color.White]
[5]    Dim aPositions2 As Float[] = [0.9, 0.01]
[6]
[7]    GenerateNewPicture()
[8]    SetPictureBorder()
[9]    Paint.Begin(hPicture)
[10]     Paint.Translate(xTranslate, yTranslate)
```

```
[11]     Paint.Scale(xScale, yScale) ' +y ▲
[12]
[13]     Paint.AntiAlias = True
[14]     Paint.Brush = Paint.RadialGradient(150, 140, 120, 100, 200, aColors, aPositions)
[15]     Paint.Arc(150, 140, 120)
[16]     Paint.Fill
[17]
[18]     Paint.Brush = Paint.RadialGradient(410, 140, 120, 410, 140, aColors2, aPositions2)
[19]     Paint.Arc(410, 140, 120)
[20]     Paint.Fill
[21]     Paint.AntiAlias = False
[22]   Paint.End
[23]
[24] End ' PaintScriptRadialGradient()
```

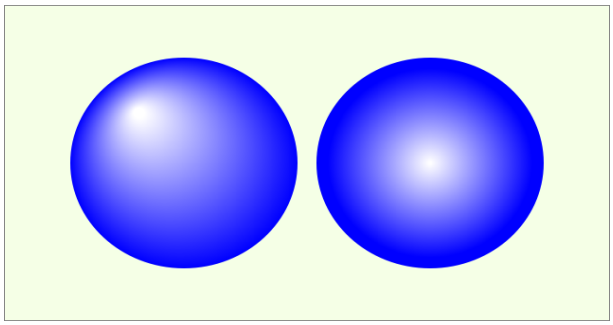

Abbildung 23.3.4.1.7: Effekte mit RadialGradient

Für die linke 'Kugel' scheint die fiktive Beleuchtung von links oben zu kommen und bei der rechten genau von vorn – einfach schön!

Beispiel 8 – Linear-Gradient

Für die Effekte mit LinearGradient werden noch Einsatzmöglichkeiten gesucht. Was fällt Ihnen dazu ein? Das sind die erzielbaren Ergebnisse für einen Drei-Farben-Verlauf. Den Quelltext finden Sie im Projekt und die notwendige Theorie im → Kapitel 23.3.2.

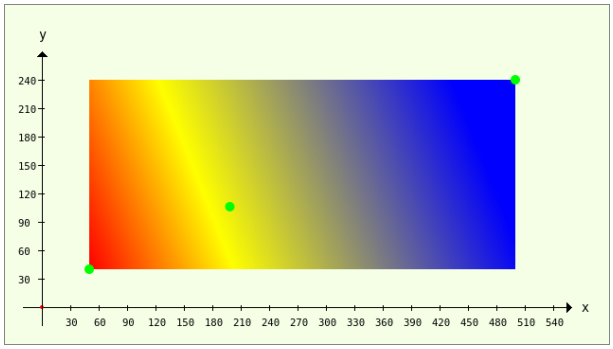

Abbildung 23.3.4.1.8: Effekt mit LinearGradient

Beispiel 9 – Bildbereich – Clip

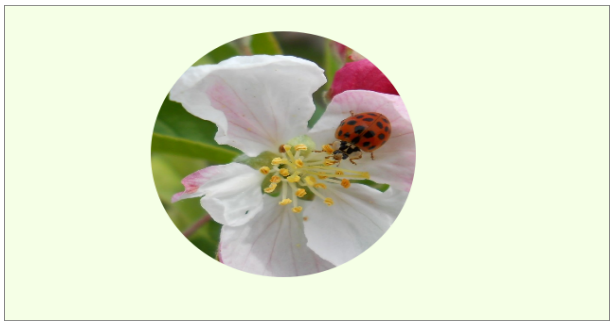

Abbildung 23.3.4.1.9: Kreisförmiger Bildbereich mit hinterlegtem Bild

Das müssen Sie wissen: Die definierte Clip-Region beeinflusst alle weiteren Zeichenoperationen und jede Bild-Änderung *außerhalb dieser Region* wird ignoriert.

Der Umfang des Quelltextes für diesen Effekt ist kurz:

```
[1]  Public Sub PaintScriptImageClip()
[2]    Dim hImage As Image
[3]    Dim fMx, fMy, fRadius As Float
[4]
[5]    GenerateNewPicture()
[6]    SetPictureBorder()
[7]    Paint.Begin(hPicture)
[8]      Paint.Translate(xTranslate, yTranslate)
[9]      Paint.Scale(xScale, yScale) ' +y ▲
[10]
[11]     fMx = 255
[12]     fMy = 150
[13]     fRadius = 140
[14]     Paint.AntiAlias = True
[15]     Paint.Arc(fMx, fMy, fRadius)
[16]     Paint.Clip ' Festlegung des aktuellen Zeichenbereiches
[17]     hImage = Image.Load("ladybird_1.png")
[18]     hImage = hImage.Rotate(Pi)
[19]   ' hImage = hImage.Stretch(fMx + fRadius, fMy + fRadius) ' Alternative mit 2 Zeilen
[20]   ' Paint.DrawImage(himage, fMx - fRadius, fMy - fRadius)
[21]   ' Einzeiler mit Stretch-Funktion inklusive!
[22]     Paint.DrawImage(hImage, fMx - fRadius, fMy - fRadius, fMx + fRadius, fMy + fRadius)
[23]     Paint.AntiAlias = False
[24]   Paint.End
[25]
[26] End ' PaintScriptImageClip()
```

Kommentar:

- Als Form für den Bildbereich wird in der Zeile 15 ein Kreis definiert.
- Genau dieser Kreis wird zum aktiven Bild-Bereich (Zeile 16).
- Ob Sie die beiden Zeilen 19 und 20 verwenden oder nur die Zeile 22 bleibt Ihnen überlassen. Die erzielte Wirkung ist adäquat – es wird das geladene Bild so hinter den kreisförmigen Bild-Bereich gelegt, dass keine Überdeckungslücke entsteht.