25.1.3 Cairo – Methoden

Die Klasse Cairo (gb.cairo) besitzt nur (statische) Methoden. Diese Methoden werden in den u.a. Tabellen gruppiert dokumentiert. Nach den Tabellen finden Sie Hinweise zur Verwendung ausgewählter Methoden.

25.1.3.1 Ausgewählte Methoden

Die Klasse *Cairo* verfügt über diese Methoden, bei denen optionale Argumente farbig hervorgehoben sind:

Methode	Beschreibung
Begin (Device As Object)	Startet das Zeichnen auf dem angegebenen Device (Zeichenflächen: Image, CairoPdfSurface, CairoPsSurface und CairoSvgSurface). Der Aufruf kann verschachtelt werden.
End ( )	Beendet das Zeichnen. Sie müssen diese Methode genauso oft aufrufen, wie die Cairo.Begin(..)-Methode → Finish-Methode der Klasse Surface.
NewPath ( )	Löscht den aktuellen Pfad. Nach dem Aufruf existiert weder ein aktueller (Start-)Punkt noch ein Pfad.
NewSubPath ( )	Beginnt einen neuen Unter-Pfad. Beachten Sie, dass der vorhandene, aktuelle Pfad nicht beeinträchtigt wird. Nach diesem Aufruf existiert kein aktueller Punkt. In vielen Fällen wird dieser Aufruf nicht erforderlich sein, da neue (Teil-)Pfade häufig mit Cairo.MoveTo(..) gestartet werden. Ein Aufruf von Cairo.NewSubPath ist besonders dann nützlich, wenn zu Beginn ein neuer Unter-Pfad mit *Cairo.Arc* oder *Cairo.ArcNegative* erfolgt. Dies macht die Dinge einfacher, da es nicht mehr erforderlich ist, Anfangskoordinaten des Kreisbogens für einen Aufruf manuell zu berechnen.
ClosePath ( )	Fügt ein Linien-Segment zum aktuellen Pfad hinzu und zwar vom aktuellen Punkt bis zum Startpunkt des aktuellen Teil-Pfades. Das ist der Punkt, der zuletzt an Cairo.MoveTo() übergeben wurde. So wird der aktuelle Pfad geschlossen und der aktuelle Punkt wird zum gemeinsamen Anfangs- und Endpunkt des Pfades.
Scale (SX As Float, SY As Float)	Ändert die *aktuelle Transformationsmatrix* (CTM) durch Skalierung der x- und y-User-Space-Achsen mit den Faktoren SX und SY. SX: Skalierungsfaktor für die x-Richtung SY: Skalierungsfaktor für die y-Richtung
Translate (TX As Float, TY As Float)	Ändert die *aktuelle Transformationsmatrix* durch Verschieben des Koordinatenursprungs um TX und TY. TX: Verschiebungsweite in x-Richtung TY: Verschiebungsweite in y-Richtung
Rotate (Angle As Float)	Ändert die *aktuelle Transformationsmatrix* durch Drehen der Koordinaten-Achsen bei gegebenem Drehwinkel *Angle* (Bogenmaß).
Clip ([Preserve As Boolean])	Erstellt eine neue Clip-Region, indem die aktuelle Clip-Region mit dem aktuellen Pfad geschnitten wird – unter Beachtung der aktuellen Füll-regeln (FillRule). Nach der Anwendung von Clip() wird der aktuelle Pfad gelöscht, ausgenommen das Preserve-Argument ist TRUE. Die aktuelle Clip-Region beeinflusst alle Zeichenoperationen, wobei alle Änderungen außerhalb dieser Region ignoriert werden. Ein Aufruf von Clip() kann die Clip-Region nur kleiner machen, niemals größer. Der aktuelle Clip ist eine Eigenschaft des Cairo-Zustands. Eine temporäre Clip-Einschränkung erreichen Sie, wenn Sie innerhalb eines Cairo.Save()-/Cairo.Restore()-Paares clippen. Nur Cairo.ResetClip() vergrößert die Clip-Region wieder.
ResetClip ( )	Setzt die aktuelle Clip-Region auf ihren ursprünglichen, uneingeschränkten Zustand zurück.
Save ( )	Erstellt eine Kopie des aktuellen Cairo-Zustands und speichert diese in einem (internen) Stapel der gespeicherten Zustände. Wenn *Restore()* aufgerufen wird, so wird der Zeichnung-Zustand aus dem gespeicherten Zustand wieder hergestellt. Mehrere Aufrufe zum Speichern und Wieder-

Methode	Beschreibung
	herstellen können verschachtelt werden.
Restore ( )	Stellt den Cairo-Zustand wieder her, indem der gespeicherte Zustand abgerufen wird. Anschließend wird der gespeicherte Zustand aus dem Stapel der gespeicherten Zustände gelöscht.
CopyPage ( )	Emittiert die aktuelle Seite für Backends – wie zum Beispiel ein Drucker – die mehrere Seiten unterstützen. Der Inhalt der aktuellen Seite wird für die nächste Seite beibehalten und übernommen.
ShowPage ( )	Emittiert die aktuelle Seite und leert die nachfolgende Seite dann für Backends, die mehrere Seiten unterstützen. Verwenden Sie die Methode → *Cairo.CopyPage()*, wenn der emittierte Seiteninhalt auf die nachfolgende Seite übernommen werden soll.

25.1.3.2 Methoden – Farbe und Muster

Methode	Beschreibung
ColorPattern (Color As Integer) As CairoPattern	Erstellt einen neuen Pinsel entsprechend einer durchscheinenden Farbe. Die Farbe 'Color' wird definiert wie für alle GUI-Komponenten → Kapitel 25.3.5 Arbeit mit Farben.
Function SolidPattern (Red As Float, Green As Float, Blue As Float [, Alpha As Float]) As CairoPattern	Generiert ein neues Farb-Muster. Die Farbanteile RGB und der Alpha-Kanal sind Zahlen vom Typ Float im Bereich von 0.0 bis 1.0.
ImagePattern (Image As Image [, X As Float, Y As Float, Extend As Integer, Filter As Integer]) As CairoPattern	Erstellt ein neues Muster aus einem Bild. Image ist das Bildobjekt und X sowie Y sind die (optionalen) Werte (Datentyp Float) aus der Pattern-Matrix und geben die initiale Translation des Bildes an. Siehe auch → *Extend* und *Filter*.
LinearGradient (X0 As Float, Y0 As Float, X1 As Float, Y1 As Float, Colors As Float [ ][ ]) As CairoPattern	Erstellt einen neuen *linearen* Farbverlauf-Pinsel entlang der durch (X0|Y0) und (X1|Y1) definierten Linie und definiert Farb-Stops aus den Farb- und Positionsargumenten im *mehrdimensionalen* Colors-Array.
RadialGradient (CX0 As Float, CY0 As Float, Radius0 As Float, CX1 As Float, CY1 As Float, Radius1 As Float, Colors As Float [ ][ ]) As CairoPattern	Erstellt einen neuen *radialen* Farbverlauf-Pinsel, wobei die Farben zwischen einem Brennpunkt (FX|FY) und dem Endpunkt auf einer von (CX0|CY0, Radius0) definierten Kreisfläche interpoliert werden und definiert Farb-Stops aus den Farb- und Positionsargumenten im *mehrdimensionalen* Colors-Array.

25.1.3.3 Methoden – Linien, Flächen und ausgewählte Operatoren

Methode	Beschreibung
MoveTo (X As Float, Y As Float)	Beginnt einen neuen (Unter-)Pfad. Nach diesem Aufruf hat der aktuelle Punkt die Koordinaten P(X|Y).
RelMoveTo(DX, DY)	Beginnt einen neuen (Unter-)Pfad. Nach diesem Aufruf besitzt der aktuelle Punkt P' ein Offset von DX und DY gegenüber dem Ausgangspunkt P. Bei einem Start-Punkt P(x|y) hat der aktuelle Punkt nach Anwendung von RelMoveTo(DX,DY) die Koordinaten P'(x+DX | y+DY).
LineTo (X As Float, Y As Float)	Fügt dem Pfad eine Strecke (Geradenabschnitt) vom existierenden Start-Punkt P(x0|y0) zum Punkt P'(X|Y) in User-Space-Koordinaten hinzu. Existiert kein Startpunkt, so muss vorher die MoveTo-Methode aufgerufen werden!
RelLineTo (DX As Float, DY As Float)	Fügt dem Pfad eine Strecke (Geradenabschnitt) vom Start-Punkt P(x0|y0) zum aktuellen Punkt P'(x0+DX | y0+DY) in User-Space-Koordinaten hinzu.
CurveTo (X1 As Float, Y1 As Float, X2 As Float, Y2 As Float, X3 As Float, Y3 As	Fügt eine kubische Bezier-Kurve (Spline) von der aktuellen Position P0(X0|Y0) bis zur Position (X3|Y3) in User-Space-Koordinaten dem

Methode	Beschreibung
Float)	Pfad hinzu, wobei (X1|Y1), (X2|Y2) und (X3|Y3) Stütz-Punkte sind. Nach diesem Aufruf ist (X3|Y3) der aktuelle Punkt.
RelCurveTo (X1 As Float, Y1 As Float, X2 As Float, Y2 As Float, X3 As Float, Y3 As Float)	Fügt – genau wie CurveTo() – eine kubische Bezier-Kurve hinzu. Hier werden aber alle Argumente als Offsets relativ zum aktuellen Punkt aufgefasst.
Rectangle (X0 As Float, Y0 As Float, Width As Float, Height As Float)	Fügt für das Rechteck der vorgegebenen Größe einen geschlossenen Unter-Pfad zum bestehenden Pfad in User-Space-Koordinaten hinzu. Existiert kein Startpunkt S(X0|Y0), so muss vorher die MoveTo-Methode aufgerufen werden.
Arc (XM As Float, YM As Float, Radius As Float [, Angle1 As Float, Angle2 As Float])	Fügt dem Pfad einen Kreisbogen mit dem gegebenen Radius hinzu. Der Bogen wird bei (XM, YM) zentriert und beginnt mit dem (Start-)Winkel Angle1 und dreht den Zentriwinkel (im originalen Cairo-Koordinatensystem) in Richtung steigender Winkel, bis der (End-)Winkel Angle2 erreicht ist. Wenn ein aktueller Punkt existiert, so wird ein Liniensegment zu dem Pfad hinzugefügt, um den aktuellen Punkt mit dem Anfang des Bogens zu verbinden. Ist diese Linie unerwünscht, kann sie durch den Aufruf von *NewSubPath()* vor dem Aufruf von Arc vermieden werden. Die Bezugsachse für 0 Grad ist die positive x-Achse. Alle Winkelangaben müssen im Bogenmaß erfolgen. Setzen Sie die Rad-Funktion ein, um ein Gradmaß in das Bogenmaß zu konvertieren.
ArcNegative (XC As Float, YC As Float, Radius As Float [, Angle1 As Float, Angle2 As Float])	Unterschied zur Dokumentation Arc(): Es wird von Angle1 nach Angle2 (nach eventueller Verschiebung von Angle2 in die Periode von Angle1) in Richtung absteigender Winkel gelaufen.
Stroke ([Preserve As Boolean])	Der (Linien-)Pfad wird mit der aktuellen Linien-Definition (Strichstärke, Strich-Form und Linien-Endform) nachgezeichnet. Der Pfad wird nach Cairo.Stroke() gelöscht – es sei denn, das Preserve-Argument ist auf den Wert TRUE gesetzt.
InStroke (X As Float, Y As Float) As Boolean	Überprüft, ob der angegebene Punkt P(X|Y) innerhalb der Region liegt, die bei einer Cairo.Stroke-Operation in Bezug auf Pfad und Parameter beim Zeichnen betroffen wäre. Zeichenflächen-Dimensionen und Clipping werden nicht berücksichtigt.
Fill ([Preserve As Boolean])	Füllt die durch den aktuellen Pfad begrenzte Fläche anhand der aktuellen Flächen-Definition (→ FillRule). Jeder Teil-Pfad wird vorher *automatisch* geschlossen. Nach Cairo.Fill(...) wird der Pfad gelöscht – aber nur dann, wenn das optionale Preserve-Argument auf nicht TRUE gesetzt ist.
InFill (X As Float, Y As Float) As Boolean	Überprüft, ob der angegebene Punkt P(X|Y) innerhalb der Region liegt, die bei einer Cairo.Fill-Operation in Bezug auf Pfad und Füll-Parameter betroffen wäre. Zeichenflächen-Dimensionen und Clipping werden nicht berücksichtigt.
Mask (Pattern As CairoPattern)	Ein Zeichnungsoperator, der mit dem Alpha-Kanal von *Pattern* als Maske malt. Undurchsichtige Muster-Bereiche werden mit den Pinsel-Werten gezeichnet, transparente Bereiche dagegen nicht.
Paint ([Alpha As Float])	Zeichnungsoperator, der mit den aktuellen Pfad-Werten innerhalb der aktuellen Clip-Region zeichnet. Wenn der (optionale) Alpha-Wert angegeben ist, so wird eine konstante Alpha-Maske verwendet. Das Ergebnis des Zeichnens wird unter Verwendung des Alpha-Wertes verblasst dargestellt.

25.1.3.4 Methoden – Text und Bilder

Methode	Beschreibung
Text (Text As String [, X As Float, Y As Float, Width As Float, Height As Float, Alignment As Integer])	Fügt den angegebenen Text zum Pfad hinzu. Die *optionalen* Argumente bestimmen eine (Text-)Box, in der der Text gezeichnet wird. Es wird der aktuelle Font benutzt, der vorher über die Font-Eigenschaft gesetzt ist. Der Standard-Font bei Cairo ist "Sans Serif".

Methode	Beschreibung
TextExtents (Text As String) As CairoExtents	Ermittelt die Ausmaße (Datentyp CairoExtents) für die Text-Zeichenfolge. Die Ausmaße beschreiben ein User-Space-Rechteck, das den tatsächlich gezeichneten Text umschließt, wie es von *Cairo.Text* und *Cairo.Fill* ohne Transformationsmatrix erstellt worden wäre.
DrawText (Text As String [, X As Float, Y As Float, Width As Float, Height As Float, Alignment As Integer])	Zeichnet den angegebenen Text. Wenn Sie die optionalen Parameter angegeben, so wird der Text durch das angegebene Rechteck begrenzt und nach dem Alignment-Parameter ausgerichtet. Die Methode ist schneller, als den Text mit *Cairo.Text* und anschließendem *Cairo.Fill* zu zeichnen. Als Alignment-Parameter sind die Konstanten der Alignment-Klasse zu verwenden. Es reicht, einen der beiden Parameter Height bzw. Width anzugeben, wodurch jeweils nur ein vertikaler bzw. horizontaler Abstand als Orientierung für den Alignment-Parameter festgelegt wird. Sobald ein Height- oder Width-Parameter angegeben wird, wird nicht mehr die Grundlinie des Textes zur Positionierung am y-Wert verwendet, sondern dessen obere Grenzlinie.

25.1.3.5 Hinweise

Die folgenden Hinweise ergänzen die Inhalte der o.a. Tabellen. Für *ausgewählte* Methoden wird je ein Beispiel mit dem verwendeten Quelltext vorgestellt. Alle Beispiele finden Sie im Download-Bereich in einem Projekt.

- Kreis und Kreisteile

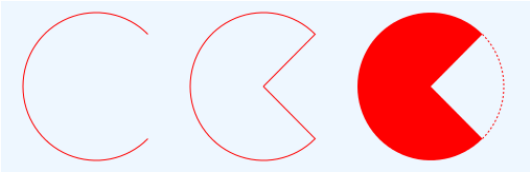

Abbildung 25.1.3.5.1: Kreisbogen - Kreissektoren

Quelltext:

```
Public Sub CairoScriptArcs()
  Dim fAngle1, fAngle2 As Float

  GenerateNewImage()
  SetImageBorder()

  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    DrawCoordinateSystem()
    Cairo.LineWidth = 1
    Cairo.Source = Cairo.ColorPattern(Color.Red)
  ' Kreisbogen 270° - offen
    Cairo.NewPath()
    fAngle1 = Rad(45) ' Start-Winkel
    fAngle2 = fAngle1 + 3 * Pi / 2 ' End-Winkel
    Cairo.Arc(100, 140, 70, fAngle1, fAngle2)
    Cairo.Stroke
  ' Kreisbogen 270° - geschlossen
    fAngle1 = Pi / 4
    fAngle2 = fAngle1 + Rad(270)
    Cairo.MoveTo(260, 140)
    Cairo.Arc(260, 140, 70, fAngle1, fAngle2)
    Cairo.LineTo(260, 140)
    Cairo.Stroke
  ' Kreissektor 270° - gefüllt
    fAngle1 = Pi / 4
    fAngle2 = fAngle1 + 3 * Pi / 2
    Cairo.MoveTo(420, 140)
    Cairo.Arc(420, 140, 70, fAngle1, fAngle2)
    Cairo.LineTo(420, 140)
    Cairo.Fill
  ' Punktierte Bogen-Linie für den 90°-Sektor
    Cairo.Dash = [2, 2]
    fAngle1 = Pi / 4 + 3 * Pi / 2
    fAngle2 = fAngle1 + Rad(90)
    Cairo.Arc(420, 140, 70, fAngle1, fAngle2)
```

```
    Cairo.Stroke
    Cairo.Dash = []  ' Alternative: Cairo.Dash = Null

  Cairo.End

End ' CairoScriptArcs()
```

- Ellipse

In diesem Beispiel wird eine Ellipse gezeichnet, die um 30° geneigt ist und deren schmaler Rand blau ist. Die Fläche hat die Farbe rot.

Quelltext-Ausschnitt:

```
  Cairo.Save()
    Cairo.Translate(400, 160)
    Cairo.Rotate(Pi() / 6)
    Cairo.Scale(0.5, 1.0)
    Cairo.Arc(0, 0, 130, 0, Pi(2))
    Cairo.Source = Cairo.ColorPattern(Color.Blue)
    Cairo.LineWidth = 5
    Cairo.Stroke(True) ' Rand
    Cairo.Source = Cairo.ColorPattern(Color.Red)
    Cairo.Fill() ' Fläche
  Cairo.Restore()
```

- Füll-Muster

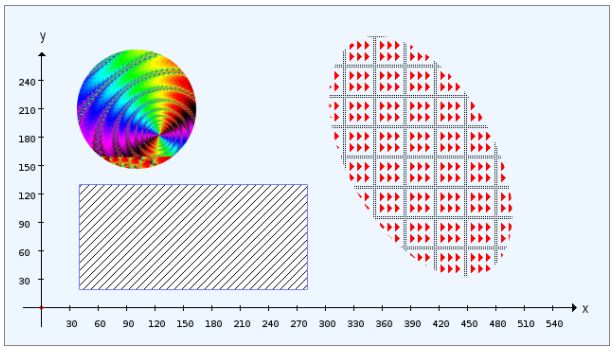

Abbildung 25.1.3.5.2: Füllmuster

Quelltext:

```
Public Sub CairoScriptPattern()
  Dim hPattern As Image

  GenerateNewImage()
  SetImageBorder()

  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    DrawCoordinateSystem()
    Cairo.AntiAlias = Cairo.AntiAliasDefault

    hPattern = Image.Load("Pattern/18.png")
    Cairo.Source = Cairo.ImagePattern(hPattern, 30, 160, 1)
    Cairo.Arc(100, 210, 63)
    Cairo.Fill

  ' Um 30° geneigte Ellipse mit speziellem Füllmuster
    hPattern = Image.Load("Pattern/17.png")
    Cairo.Source = Cairo.ImagePattern(hPattern, 0, 0, 1)
    Cairo.Save()
      Cairo.Translate(400, 160)
      Cairo.Rotate(Pi / 6)
      Cairo.Scale(0.55, 1.0)
      Cairo.Arc(0, 0, 140, 0, Pi(2))
      Cairo.Fill
    Cairo.Restore()

    Cairo.LineWidth = 1
    Cairo.Source = Cairo.ColorPattern(Color.Blue)
    Cairo.Rectangle(40, 20, 240, 110)
```

```
    Cairo.Stroke(True)
    hPattern = Image.Load("Pattern/13.png")
    Cairo.Source = Cairo.ImagePattern(hPattern, 0, 0, 1)
    Cairo.Rectangle(40, 20, 240, 110)
    Cairo.Fill
  Cairo.End
End ' CairoScriptPattern()
```

Beachten Sie die beiden Methoden *Cairo.Save* und *Cairo.Restore* für das Zeichnen der Ellipse mit Muster. Wenn Sie diesen Quelltext-Teil als letzten gesetzt hätten, dann könnten Sie auf die beiden Methoden verzichten, weil dann das Skript zum Zeichnen endet und sich die Methoden *Cairo.Rotate* und *Cairo.Scale* nicht auf das weitere Zeichnen auswirken!

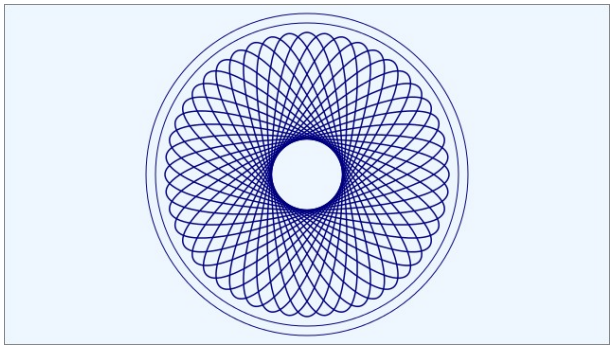

Abbildung 25.1.3.5.3: Zwei Kreise und 30 Ellipsen

Quelltext:

```
Public Sub CairoScriptEllipses()
  Dim i, iLoops As Integer

  GenerateNewImage()
  SetImageBorder()
  Cairo.Begin(hImage)
    Cairo.Translate(daCanvas.W / 2, daCanvas.H / 2)
    Cairo.Scale(xScale, yScale) ' +y ▲

    Cairo.AntiAlias = 0
    Cairo.LineWidth = 1
    Cairo.Source = Cairo.ColorPattern(Color.DarkBlue)
    Cairo.Arc(0, 0, 160) ' Kreis 1
    Cairo.Stroke
    Cairo.Arc(0, 0, 170) ' Kreis 2
    Cairo.Stroke
    Cairo.LineWidth = 1.5 * 1
    iLoops = 2 * 12
    For i = 1 To iLoops
      Cairo.Save()
        Cairo.Rotate(i * Pi / iLoops)
          Cairo.Scale(0.25, 1)
        Cairo.Arc(0, 0, 150, 0, Pi(2))
      Cairo.Restore()
      Cairo.Stroke()
    Next
  Cairo.End
End ' CairoScriptEllipses()
```

- Polygone – Vielecke

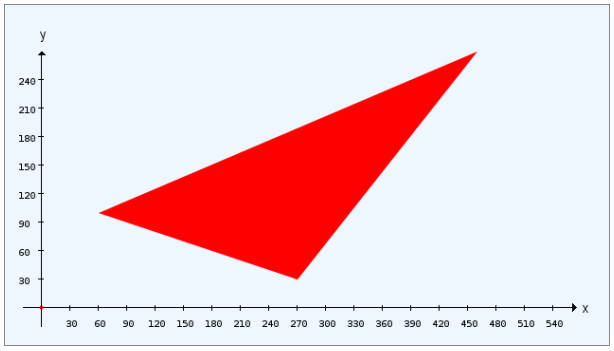

Abbildung 25.1.3.5.4: Konvexes Polygon – Dreieck

Quelltext:

```
Public Sub CairoScriptTriangle()
  GenerateNewImage()
  SetImageBorder()
  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    Cairo.AntiAlias = False
    DrawCoordinateSystem()
  ' Polygon als konvexes Dreieck
    Cairo.Source = Cairo.ColorPattern(Color.Red)
    Cairo.MoveTo(60, 100)
    Cairo.LineTo(460, 270)
    Cairo.LineTo(270, 30)
  ' Man muss nicht zum Anfangspunkt zurück gehen, weil Endpunkt automatisch der Anfangspunkt ist.
    Print Cairo.InFill(210, 210) ' → False
    Print Cairo.InFill(300, 160) ' → True
    Cairo.Fill()
  Cairo.End
End ' CairoScriptTriangle()
```

Mit den beiden *Print-Anweisungen* können Sie feststellen, ob zwei vorgegebene Punkte *in der Fläche* liegen.

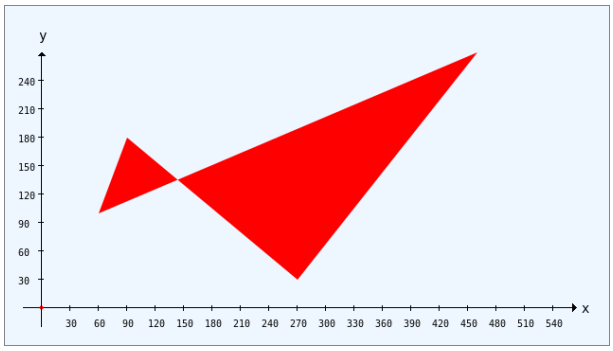

Abbildung 25.1.3.5.5: Nicht-konvexes Polygon

Quelltext:

```
Public Sub CairoScriptPolygon()
  GenerateNewImage()
  SetImageBorder()
  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    Cairo.AntiAlias = False
    DrawCoordinateSystem()
  ' Polygon als nicht-konvexes Vieleck
    Cairo.Source = Cairo.ColorPattern(Color.Red)
    Cairo.MoveTo(60, 100)
    Cairo.LineTo(460, 270)
    Cairo.LineTo(270, 30)
    Cairo.LineTo(90, 180)
  ' Man muss nicht zum Anfangspunkt zurück gehen, weil Endpunkt automatisch der Anfangspunkt ist.
    Print Cairo.InFill(120, 90)
    Print Cairo.InFill(300, 150)
    Cairo.Fill()
  Cairo.End
End ' CairoScriptPolygon()
```

- Diagramme – Kreis, Kreisteile, Rechtecke und Text

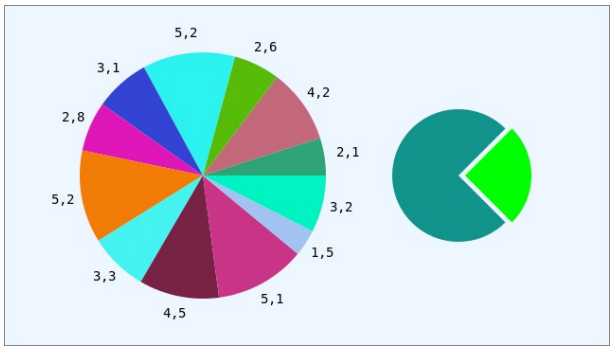

Abbildung 25.1.3.5.6: Kreis-Diagramm

Quelltext Kreis-Diagramm:

```
Public Sub CairoScriptChart()
  Dim i As Integer
  Dim fStartAngle, fTotal, fPx, fPy, fRadius, fMx, fMy, fRadiusOffset As Float
  Dim fSumme As Float = 0, fAngle As Float
  Dim textDimension As RectF
  Dim aData, aAngle As Float[]

  fMx = 170
  fMy = 140
  fRadius = 130
  fRadiusOffset = 20

' Inline-Array mit den darzustellenden Werten
  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]
  For i = 0 To aData.Max
    fTotal = fTotal + aData[i]
  Next

' aAngle => Array für die Daten-Winkel-Äquivalente (Bogenmaß)
  aAngle = New Float[aData.Count]

  For i = 0 To aData.Max
  ' Umrechnung 'Wert' in sein relatives(!) 'Winkel-Äquivalent' mit 360°≡1
    aAngle[i] = (aData[i] / fTotal) * Pi(2)
  Next

  GenerateNewImage()
  SetImageBorder()

  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲

    fStartAngle = 0
  ' Kreis-Sektoren zeichnen - Anzahl: aData.Max
    For i = 0 To aData.Max
      Cairo.Source = Cairo.ColorPattern(Color.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255)))
      Cairo.MoveTo(fMx, fMy)
      Cairo.Arc(fMx, fMy, fRadius, fStartAngle, fStartAngle + aAngle[i]) ' Tortenstück
      Cairo.LineTo(fMx, fMy)
      Cairo.Fill
      fStartAngle = fStartAngle + aAngle[i] ' Neuer Startwinklel
    Next

  ' Werte an den Sektor zeichnen - Anzahl: aData.Max
    For i = 0 To aData.Max
      fStartAngle = fSumme + aAngle[i] / 2
      fSumme = fSumme + aAngle[i]
      fPx = (fRadius + fRadiusOffset) * Cos(fStartAngle) + fMx
      fPy = (fRadius + fRadiusOffset) * Sin(fStartAngle) + fMy

      fPx = fPx - (Cairo.TextExtents(aData[i]).Width / 2)
      fPy = fPy - (Cairo.TextExtents(aData[i]).Height / 3)

      Cairo.Scale(1, -1) ' +y ▼
        Cairo.Source = Cairo.ColorPattern(Color.Black)
        Cairo.Font.Name = "Monospace"
        Cairo.Font.Size = 14
        Cairo.MoveTo(fPx, - fPy)
        Cairo.DrawText(Str(aData[i]))
        Cairo.Scale(1, -1) ' +y ▲
```

```
    Next ' Sektor

  ' 2. Bild mit hervorgehobenem Sektor
    Cairo.Source = Cairo.ColorPattern(Color.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255)))
    Cairo.MoveTo(440, fMy)
    fAngle = Pi / 4
    Cairo.Arc(440, fMy, 70, fAngle, fAngle + 3 * Pi / 2)
    Cairo.LineTo(440, fMy)
    Cairo.Fill()
    Cairo.Source = Cairo.ColorPattern(Color.Green)
    Cairo.MoveTo(440 + 7, fMy)
    fAngle = - Pi(0.25)
    Cairo.Arc(440 + 7, fMy, 70, fAngle, fAngle + Rad(90))
    Cairo.LineTo(440 + 7, fMy)
    Cairo.Fill()

  Cairo.End

End ' CairoScriptChart()
```

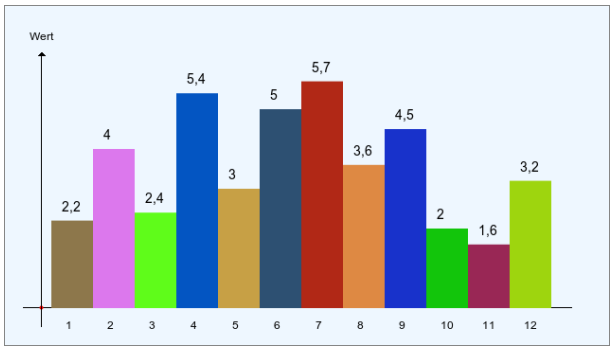

Abbildung 25.1.3.5.7: Säulen-Diagramm

Quelltext Säulen-Diagramm:

```
Public Sub CairoScriptBarChart()
  Dim i As Integer
  Dim fDeltaX, fDeltaY, fBeginX, fEndX, fEndY, fMaxValue As Float
  Dim aData, aDataC As Float[]
  Dim sCaption As String

' Inline-Array mit den darzustellenden Werten
  aData = [2.2, 4, 2.4, 5.4, 3, 5.0, 5.7, 3.6, 4.5, 2.0, 1.6, 3.2]
  aDataC = aData.Copy() ' Kopie des Arrays der Original-Werte
  aDataC.Sort(gb.Descent) ' Absteigende Sortierung der Elemente
  fMaxValue = aDatac[0] ' Das 1. Element ist jetzt der größte Wert im Array
  fBeginX = 10 ' Festlegung Zeichenbereich Abszisse
  fEndX = 540
  fEndY = 240 ' Festlegung Zeichenbereich Ordinate
  fDeltaX = Round((fEndX - fBeginX) / aData.Count, 0) ' Normierte Streifenbreite (Einheit)
  fDeltaY = Round(fEndY / fMaxValue, 0) ' Normierte Streifenhöhe (Einheit)

  GenerateNewImage()
  SetImageBorder()
  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    Draw_X_Axis()
    SetOrigin()
    Draw_Y_Axis()
    Draw_Y_AxisArrow()

    For i = 0 To aData.Max
    ' Berechnung und Anzeige aller Streifen (Rechtecke) im Diagramm mit zufälliger Streifen-Farbe und Wert
      Cairo.Rectangle(fBeginX + i * fDeltaX, 0, fDeltaX, fDeltaY * aData[i])
      Cairo.Source = Cairo.ColorPattern(Color.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255)))
      Cairo.Fill
      Cairo.Scale(1, -1) ' +y ▼
        Cairo.Source = Cairo.ColorPattern(Color.Black)
        Cairo.Font.Name = "Arial"
        Cairo.Font.Size = 14
        Cairo.MoveTo(fBeginX + 0.25 * fDeltaX + i * fDeltaX, - (fDeltaY * aData[i] + 10))
        Cairo.DrawText(Str(aData[i]))
        Cairo.Font.Name = "Arial"
        Cairo.Font.Size = 12
        Cairo.MoveTo(fBeginX + 0.33 * fDeltaX + i * fDeltaX, 22)
```

```
        Cairo.DrawText(Str(i + 1))
      Cairo.Scale(1, -1) ' +y ▲
    Next

  ' Ordinate beschriften
    Cairo.Scale(1, -1) ' +y ▼
      sCaption = ("Value")
      If Cairo.TextExtents(sCaption).Width / 2 > xTranslate Then
         Cairo.MoveTo(0 - xTranslate / 2, -283)
         Cairo.DrawText(sCaption)
      Else
         Cairo.MoveTo(0 - Cairo.TextExtents(sCaption).Width / 2, -283)
         Cairo.DrawText(sCaption)
      Endif
        Cairo.Font.Name = "Monospace"
        Cairo.Font.Size = 10
    Cairo.Scale(1, -1) ' +y ▲
  Cairo.End

End ' CairoScriptBarChart()
```

- LinearGradient und Radial-Gradient

Für den Einsatz der beiden Methoden *Cairo.LinarGradient* und *Cairo.RadialGradient* wird zur Festlegung der Farbverläufe und ColorStops eine Funktion von Tobias Boege verwendet:

```
Private Function BuildColorStops(aColorsG As Integer[], Optional aPositionsG As Float[]) As Float[][]
  Dim iIndex As Integer, fPosition As Float
  Dim aColors As New Float[][]

  For iIndex = 0 To aColorsG.Max
    With Color[aColorsG[iIndex]]
      If Not aPositionsG Then
        fPosition = iIndex / aColorsG.Count ' uniformly distributed
      Else
        fPosition = aPositionsG[iIndex]
      Endif
      aColors.Add([fPosition, .Red, .Green, .Blue])
    End With
  Next
  Return aColors
End ' BuildColorStops(..)
```

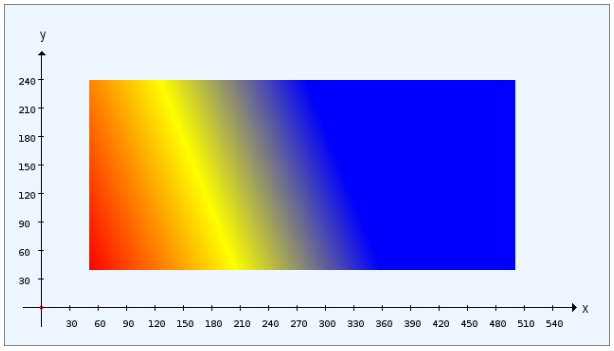

Abbildung 25.1.3.5.8: Linear-Gradient

Quelltext LinearGradient:

```
Public Sub CairoScriptLinearGradient()
  Dim aColorsLG As Integer[] = [Color.Red, Color.Yellow, Color.Blue]
  Dim aColors As Float[][]

  aColors = BuildColorStops(aColorsLG, Null)
  GenerateNewImage()
  SetImageBorder()

  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    DrawCoordinateSystem()
    Cairo.MoveTo(50, 40)
    Cairo.Source = Cairo.LinearGradient(50, 40, 450, 200, aColors)
    Cairo.Rectangle(50, 40, 450, 200)
    Cairo.Fill()
  Cairo.End
End ' CairoScriptLinearGradient()
```

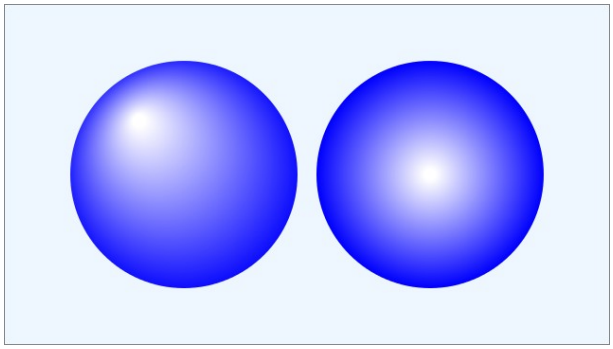

Abbildung 25.1.3.5.9: Radial-Gradient

Quelltext Radial-Gradient:

```
Public Sub CairoScriptRadialGradient()
  Dim aColorsRG As Float[][]

  aColorsRG = BuildColorStops([Color.Blue, Color.White], [1, 0.05])
  GenerateNewImage()
  SetImageBorder()

  Cairo.Begin(hImage)

    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
          DrawCoordinateSystem()
  ' Leucht- oder Spiegelpunkt links oben
    Cairo.AntiAlias = True
    Cairo.Source = Cairo.RadialGradient(100, 200, 0, 150, 140, 130, aColorsRG)
    Cairo.Arc(150, 140, 120)
    Cairo.Fill

  ' Leucht- oder Spiegelpunkt in der Objekt-Mitte
    Cairo.Source = Cairo.RadialGradient(410, 140, 0, 410, 140, 120, aColorsRG)
    Cairo.Arc(410, 140, 120)
    Cairo.Fill
    Cairo.AntiAlias = False

  Cairo.End

End ' CairoScriptRadialGradient()
```

- Clip und Text

Das folgende Beispiel demonstriert den Einsatz der Methode *Cairo.Clip* im Zusammenhang mit den beiden Methoden *Cairo.Save* und *Cairo.Restore*, die notwendigerweise eingesetzt werden müssen, weil auch noch Text in mehreren Zeilen eingefügt werden soll.

Das Ergebnis mit einem Bild im Clip-Bereich und passendem Text ist ansprechend:

Abbildung 25.1.3.5.10: Bild im Clip-Bereich und Text

Quelltext:

```
Public Sub CairoScriptImageClip()
  Dim imgImage As Image
  Dim fMx, fMy, fRadius, fYOffset As Float
  Dim i, X0, Y0 As Integer
  Dim aText As New String[]

  aText.Add("Käfer, du gefällst mir sehr,")
  aText.Add("Wo hast du die Punkte her?")
  aText.Add("Eins und zwei und drei und vier")
  aText.Add("Käferlein, komm, sag es mir!")
  aText.Add("Barbara Henze")

  GenerateNewImage()
  SetImageBorder()
  Cairo.Begin(hImage)
    Cairo.Translate(xTranslate, yTranslate)
    Cairo.Scale(xScale, yScale) ' +y ▲
    fMx = 150
    fMy = 150
    fRadius = 145
    Cairo.AntiAlias = Cairo.AntiAliasDefault ' 0

  ' Bild - Clip-Bereich
    Cairo.Save
      Cairo.Arc(fMx, fMy, fRadius)
      Cairo.Clip()
      imgImage = Image.Load("ladybird_1.png")
      imgImage = imgImage.Rotate(Pi(0.5))
      imgImage = imgImage.Stretch(imgImage.W / 2, imgImage.H / 2)
      Cairo.Source = Cairo.ImagePattern(imgImage, -30, -80)
      Cairo.Paint()
    Cairo.Restore

  ' Mehrzeiliger Text – außerhalb des Clip-Bereichs
    Cairo.Scale(1, -1) ' +y ▼
      Cairo.Source = Cairo.ColorPattern(Color.DarkBlue)
      Cairo.Font.Name = "Arial"
      Cairo.Font.Size = 18
      Cairo.Font.Bold = True

      x0 = 305
      Y0 = 90
      fYOffset = 0
      Cairo.MoveTo(X0, - Y0)
      Cairo.DrawText(aText[0])
      Cairo.MoveTo(X0, - Y0 + 1 * Cairo.Font.Extents.Height + fYOffset)
      Cairo.DrawText(aText[1])
      Cairo.MoveTo(X0, - Y0 + 2 * Cairo.Font.Extents.Height + fYOffset)
      Cairo.DrawText(aText[2])
      Cairo.MoveTo(X0, - Y0 + 3 * Cairo.Font.Extents.Height + fYOffset)
      Cairo.DrawText(aText[3])
      Cairo.MoveTo(X0, - Y0 + 4 * Cairo.Font.Extents.Height + fYOffset)
      Cairo.Font.Size = 12
      Cairo.DrawText(aText[4])
    Cairo.Scale(1, -1)
    Cairo.AntiAlias = Cairo.AntiAliasNone ' 1
  Cairo.End

End ' CairoScriptImageClip()
```