25.1.6 CairoExtents

Die Klasse *CairoExtents* stellt einen erweiterten Begrenzungsrahmen zur Verfügung, wie er von den Eigenschaften *Cairo.ClipExtents* oder *Cairo.FillExtents* zurückgegeben wird.

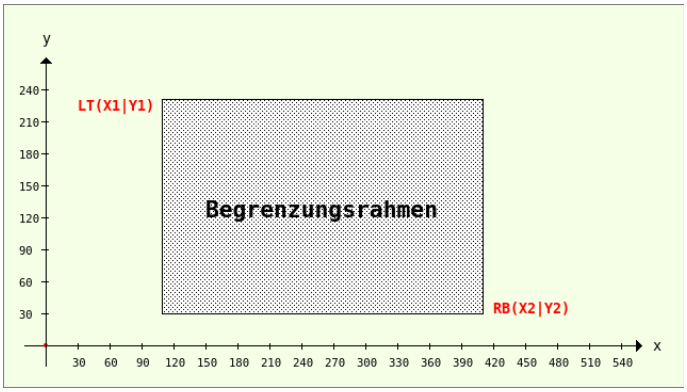

Abbildung 25.1.6.1: Begrenzungsrahmen

25.1.6.1 Eigenschaften

Die Klasse *CairoExtents* verfügt über diese vier Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
X1	Float	Gibt die x-Koordinate des linken oberen Punktes (LT) des Begrenzungsrahmens zurück.
Y1	Float	Gibt die y-Koordinate des linken oberen Punktes (LT) des Begrenzungsrahmens zurück.
X2	Float	Gibt die x-Koordinate des rechten unteren Punktes (RB) des Begrenzungsrahmens zurück.
Y2	Float	Gibt die y-Koordinate des rechten unteren Punktes (RB) des Begrenzungsrahmens zurück.

Tabelle 25.1.6.1.1 : Eigenschaften der Klasse CairoExtents

25.1.6.2 Methode

Die Klasse *CairoExtents* besitzt nur eine Methode:

```
Function Merge ( Extents As CairoExtents ) As CairoExtents
```

Die Methode Merge() liefert das *kleinste Rechteck* vom Typ *CairoExtents* zurück, in dem die Vereinigung der beiden Extents Platz hat.

Es führt beide Extents E (aktueller Begrenzungsrahmen) und F (Argument der Merge-Funktion) zu einem Extent G zusammen. G ist das kleinste Rechteck, das sowohl E als auch F enthält. Eine fiktive Implementation der Merge-Funktion könnte in Gambas so formuliert werden:

```
G.X  = Min(E.X,  F.X)
G.Y  = Min(E.Y,  F.Y)
G.X2 = Max(E.X2, F.X2)
G.Y2 = Max(E.Y2, F.Y2)
```

Danach wird der aktuelle Begrenzungsrahmen E durch den neuen Begrenzungsrahmen G ersetzt.

25.1.6.3 Beispiel 1 – Cairo.ClipExtents

Die Clip-Methode schneidet den aktuellen Clip-Bereich mit dem aktuellen Pfad (anhand der aktiven FillRule), um einen neuen Clip-Bereich zu erzeugen. Wenn Sie einen Clip-Bereich für die verwendete

Oberfläche definieren, dann können Sie diesen Bereich über die Eigenschaft *Cairo.ClipExtents* vom Typ *CairoExtents* jederzeit auslesen. Im Beispiel 1 wird zuerst der Standard-Clip-Bereich ausgelesen, dann nur die Wirkung der Methode Cairo.Clip bei leerem (aktuellen) Pfad untersucht und abschließend ein (rechteckiger) Clip-Bereich definiert und dessen Bereich ausgelesen.

Im Quelltext-Ausschnitt werden nur 3 Prozeduren vorgestellt:

```
[1]  Private Sub GeneratePDF()
[2]    Dim PDFSurface As CairoPdfSurface
[3]
[4]    PDFSurface = New CairoPdfSurface(sPfadPDFDatei, 210, 297) ' -->> DIN A4-Oberfläche
[5]
[6]    Cairo.Begin(PDFSurface)
[7]      Cairo.Matrix = Cairo.Matrix.Translate(MMToPoints(20), MMToPoints(20))
[8]      Cairo.Matrix = Cairo.Matrix.Scale(1, 1) ' Zoom-Faktor = 1
[9]
[10]     'Cairo.Rectangle(MMToPoints(0), MMToPoints(0), MMToPoints(175), MMToPoints(257))
[11]     'Cairo.Clip
[12]
[13]     Print "X1 = "; Round(PointsToMM(Cairo.ClipExtents.X1), 0)
[14]     Print "Y1 = "; Round(PointsToMM(Cairo.ClipExtents.Y1), 0)
[15]     Print "X2 = "; Round(PointsToMM(Cairo.ClipExtents.X2), 0)
[16]     Print "Y1 = "; Round(PointsToMM(Cairo.ClipExtents.Y2), 0)
[17]
[18]     Cairo.Arc(MMToPoints(50), MMToPoints(70), MMToPoints(80))
[19]     Cairo.Source = Cairo.ColorPattern(&HC3DDFF)
[20]     Cairo.AntiAlias = True
[21]     Cairo.Fill
[22]
[23]     Cairo.AntiAlias = False
[24]     Cairo.Rectangle(MMToPoints(0), MMToPoints(0), MMToPoints(175), MMToPoints(257))
[25]     Cairo.Source = Cairo.ColorPattern(Color.Black)
[26]     Cairo.LineWidth = 1
[27]     Cairo.Dash = [1, 1]
[28]     Cairo.Stroke()
[29]     Cairo.Dash = []
[30]   Cairo.End
[31]
[32]   PDFSurface.Finish()
[33]
[34] End ' GeneratePDF()
[35]
[36] Private Function MMToPoints(Value As Float) As Float
[37]   Return Value * 2.83527
[38] End ' MMToPoints(..)
[39]
[40] Private Function PointsToMM(Value As Float) As Float
[41]   Return Value * 0.3527
[42] End ' MMToPoints(..)
```

Kommentar:

- Über die beiden letzten Funktionen können Sie sehr komfortabel User-Koordinaten in Millimeter und Koordinaten in Millimetern in User-Koordinaten konvertieren.
- Interessant sind die Änderungen in den Zeilen 10 und 11. Im ersten Fall sind beide Zeilen auskommentiert und es ergeben sich die folgenden Koordinaten für den Clip-Begrenzungsrahmen:

```
Clip-Bereich = originale Oberfläche (Device) mit X1 = -20, Y1 = -20, X2 = +190, Y2 = +277
```

Die Ergebnisse korrespondieren mit den Einstellungen für die Translation des Cairo-Koordinatensystems: 20mm in x-Richtung und 20mm in y-Richtung (die nach unten zeigt!). Somit ergeben sich genau die Weite von 210mm und die Höhe von 297mm für eine A4-Oberfläche.

- Im zweiten Fall ist nur die Zeile 10 auskommentiert und es ergeben sich die folgenden Koordinaten für den aktuellen Clip-Begrenzungsrahmen:

```
Clip-Bereich = leeres Rechteck mit X1 = -20, Y1 = -20, X2 = -20, Y2 = -20
```

Damit sind alle weiteren Anweisungen zum Zeichnen natürlich ohne Wirkung – Sie sehen nichts auf der Oberfläche, obgleich korrekt und ohne Fehler gezeichnet wurde!

- Im dritten Fall sind die Zeilen 10 und 11 aktiv und es ergeben sich die folgenden Koordinaten für den aktuellen Clip-Begrenzungsrahmen:

```
Clip-Begrenzungsrahmen = definierter Clip-Bereich X1 = 0, Y1 = 0, X2 = 175, Y2 = 257
```

Der Clip-Begrenzungsrahmen ist der definierte Clip-Bereich in der Zeile 10.

Für den zweiten Fall wird der Kreis zum Beispiel nur im definierten Clip-Bereich (punktiertes Rechteck) – dem aktuellen Bereich zum Zeichen – eingezeichnet:

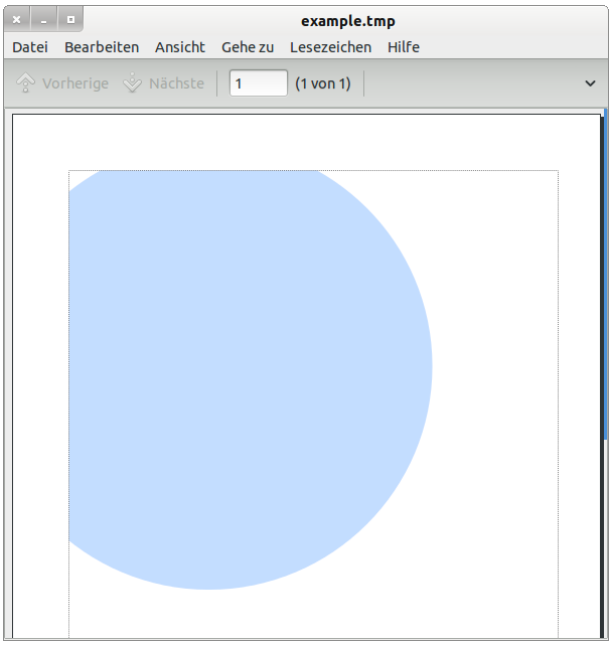

Abbildung 25.1.6.3.1: Kreis im definierten Clip-Bereich

Das folgende Bild des Kreises sehen Sie im *ersten* Fall: Der Kreis wird auf der Oberfläche dargestellt, selbstverständlich an der linken Seite abgeschnitten, denn dort endet der originale A4-Clip-Bereich:

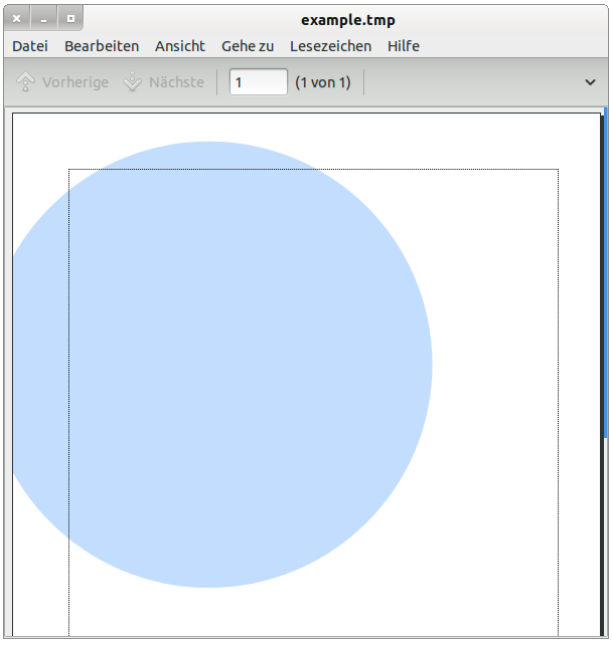

Abbildung 25.1.6.3.2: Kreis im definierten Clip-Bereich A4

25.1.6.4 Beispiel 2 – Cairo.FillExtents

Im Beispiel 2 wird eine Kreisfläche mit hellblauer Farbe gefüllt. Der Begrenzungsrahmen ist in diesem Fall ein Quadrat. Das folgende Bild zeigt den Kreis, den Begrenzungsrahmen (blau) und ausgewählte Punkte von Kreis und Begrenzungsrahmen:

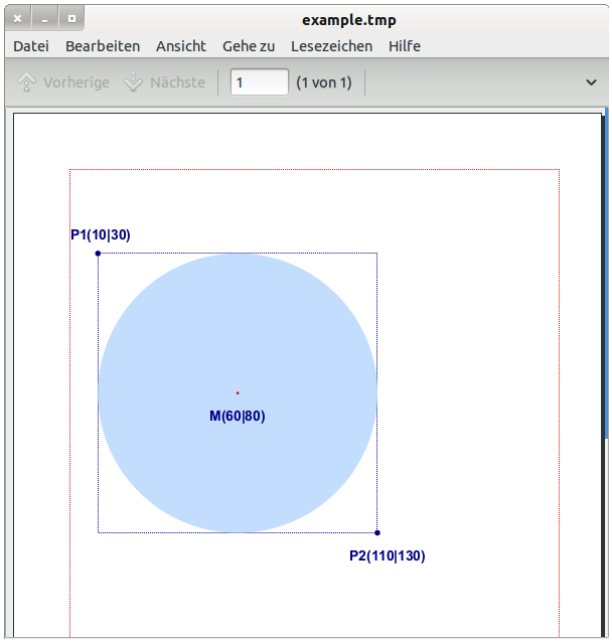

Abbildung 25.1.6.4.1: Kreis im definierten Clip-Bereich A4

Im u.a. Quelltext-Ausschnitt sind vor allem die Zeilen 30 bis 37 interessant. Es werden auch im 2. Beispiel die beiden Funktionen zur Konvertierung der Koordinaten verwendet.

```
[1]  Private Sub GeneratePDF2()
[2]    Dim PDFSurface As CairoPdfSurface
[3]
[4]    PDFSurface = New CairoPdfSurface(sPfadPDFDatei, 210, 297) ' -->> DIN A4
[5]
[6]    Cairo.Begin(PDFSurface)
[7]      Cairo.Matrix = Cairo.Matrix.Translate(MMToPoints(20), MMToPoints(20))
[8]      Cairo.Matrix = Cairo.Matrix.Scale(1, 1) ' Zoom-Faktor = 1
[9]
[10]     Cairo.Rectangle(MMToPoints(0), MMToPoints(0), MMToPoints(175), MMToPoints(257))
[11]     Cairo.Source = Cairo.ColorPattern(Color.Red)
[12]     Cairo.AntiAlias = False
[13]     Cairo.LineWidth = 1
[14]     Cairo.Dash = [1, 1]
[15]     Cairo.Stroke
[16]     Cairo.Dash = []
[17]
[18]     Cairo.Source = Cairo.ColorPattern(Color.DarkBlue)
[19]     Cairo.Arc(MMToPoints(10), MMToPoints(30), MMToPoints(1))
[20]     Cairo.Arc(MMToPoints(110), MMToPoints(130), MMToPoints(1))
[21]     Cairo.Fill
[22]
[23]     Cairo.Rectangle(MMToPoints(10), MMToPoints(30), MMToPoints(100), MMToPoints(100))
[24]     Cairo.AntiAlias = False
[25]     Cairo.LineWidth = 1
[26]     Cairo.Dash = [1, 1]
[27]     Cairo.Stroke
[28]     Cairo.Dash = []
[29]
[30]     Cairo.Source = Cairo.ColorPattern(&HC3DDFF)
[31]     Cairo.AntiAlias = True
[32]     Cairo.Arc(MMToPoints(60), MMToPoints(80), MMToPoints(50))
[33]        Print "X1 = "; Round(PointsToMM(Cairo.FillExtents.X1), 0)
[34]        Print "Y1 = "; Round(PointsToMM(Cairo.FillExtents.Y1), 0)
[35]        Print "X2 = "; Round(PointsToMM(Cairo.FillExtents.X2), 0)
[36]        Print "Y2 = "; Round(PointsToMM(Cairo.FillExtents.Y2), 0)
[37]     Cairo.Fill
[38]
[39]     Cairo.Source = Cairo.ColorPattern(Color.Red)
[40]     Cairo.Arc(MMToPoints(60), MMToPoints(80), MMToPoints(0.5))
```

```
[41]     Cairo.Fill
[42]
[43]   ' TEXTE
[44]     Cairo.Source = Cairo.ColorPattern(Color.DarkBlue)
[45]     Cairo.Font.Name = "Arial"
[46]     Cairo.Font.Size = 14
[47]     Cairo.Font.Bold = True
[48]     Cairo.MoveTo(MMToPoints(50), MMToPoints(90))
[49]     Cairo.DrawText("M(60|80)")
[50]     Cairo.MoveTo(MMToPoints(0), MMToPoints(25))
[51]     Cairo.DrawText("P1(10|30)")
[52]     Cairo.MoveTo(MMToPoints(100), MMToPoints(140))
[53]     Cairo.DrawText("P2(110|130)")
[54]   Cairo.End
[55]
[56]   PDFSurface.Finish()
[57]
[58] End ' btnGeneratePDF2_Click()
```