23.3.3 Zeichnen mit Paint

23.3.3.1 Zeichenflächen

Sie benötigen zum Zeichnen mit Paint in Gambas – wie in der realen Welt – eine Zeichenfläche und einen Stift oder Pinsel. Als Zeichenfläche (Device) stehen Ihnen zur Verfügung:

- DrawingArea,
- ScrollArea als spezialisierte DrawingArea,
- Picture,
- Image und SVGImage oder
- Printer.

23.3.3.2 Geometrische Formen

Diese geometrischen Formen (Linien oder Flächen oder Text) in sechs Kategorien können Sie auf die Zeichenfläche zeichnen:

- Punkt
- Linie – Strecke als kürzeste Verbindung zweier Punkte
- Polygon (n-Eck)
  Rechteck (Spezialfall Quadrat),
- Ellipse (Spezialfall Kreis)
  Ellipsen-Bogen – Ellipsen-Sektor
  Kreis-Bogen – Kreissektor
- Bézier-Kurven 3. Grades als krummlinige Verbindung von zwei Punkten
- Text

Ja, auch Text wird gezeichnet, wobei Sie neben dem Text die Informationen zum Font, zur Schriftfarbe und zum (Start-)Punkt angeben müssen, an dem die Schrift nach dem Aufruf spezieller Methoden in die Zeichenfläche eingefügt wird.

23.3.3.3 Paint-Koordinatensystem

Alle Zeichenflächen besitzen ein (nicht sichtbares) Koordinatensystem, auf das sich alle Angaben von Koordinaten in den Eigenschaften oder in den Argumenten der Methoden der Klasse Paint beziehen. Beachten Sie bitte diese Hinweise:

- Der Koordinaten-Ursprung O(0|0) des Paint-Koordinatensystems liegt in der *linken oberen Ecke* der Paint-Zeichenfläche.
- Die positive x-Achse zeigt nach rechts.
- Die positive y-Achse zeigt – im Gegensatz zum gewohnten kartesischen Koordinatensystem aus der Mathematik – nach unten!
- Koordinaten in Paint sind immer vom Datentyp Float.
- Jeder Punkt auf der Zeichenfläche wird durch ein Koordinaten-Paar P(x|y) festgelegt.

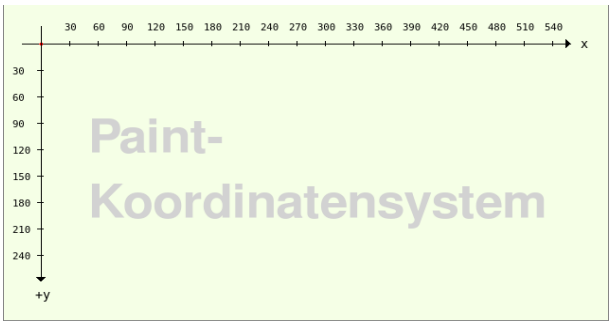

Abbildung 23.3.3.3.1: Paint-Koordinatensystem mit verschobenem Koordinatenursprung

Um die Bezeichnungen der beiden Achsen und die Werte *bei Bedarf* gut lesbar einzuzeichnen, kann man den Koordinatenursprung des Koordinatensystem mit der Translate-Methode der Paint-Klasse permanent verschieben.

```
Paint.Begin(hPicture)
  Paint.Translate(40, 45) ' Verschiebung um +40.0 in x-Richtung und um +45.0 in y-Richtung
  ...
Paint.End
```

Die Verschiebung ändert die *Richtung* der beiden Koordinatenachsen nicht!

Sie könnten in der folgenden Abbildung auch auf die Darstellung des Koordinatensystems verzichten, denn es hat keinen Einfluss auf die *Wirkung* des Bildes:

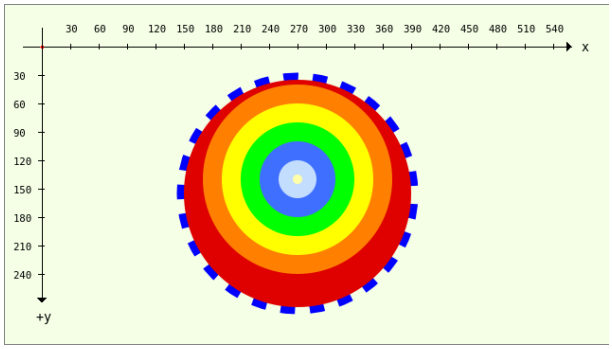

Abbildung 23.3.3.3.2: Paint-Bild

Die nächste Abbildung zeigt das *korrekte Bild* einer Funktion in einem verschobenen Paint-Koordinatensystem mit gut lesbaren Bezeichnungen:

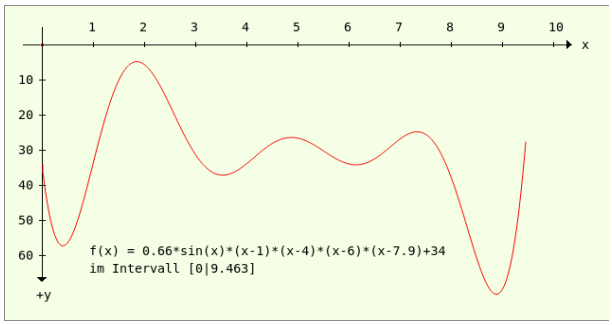

Abbildung 23.3.3.3.3: Bild einer Funktion y = f(x)

Sie erkennen sicher, dass die Funktion im vorgegebenem Intervall keine Nullstellen besitzt und neben dem lokalen Minimum in der Umgebung von 2 auch über einen lokalen Hochpunkt verfügt.
Um die gewohnte Sicht zu realisieren, können Sie das Koordinatensystem so *skalieren*, dass die positive y-Achse nach oben zeigt. Eine Translation des Koordinatenursprungs O(0|0), verbunden mit einer Spiegelung der y-Achse an der Abzisse – verwirklicht durch den Einsatz der Scale-Methode – setzen Sie so um:

```
Paint.Begin(hPicture)
  Paint.Translate(40, 45) ' Verschiebung um +40.0 in x-Richtung und um +45.0 in y-Richtung
  Paint.Scale(1, -1)      ' Die -1 bewirkt eine Invertierung der Richtung der y-Achse → +y ▲
  ...
Paint.End
```

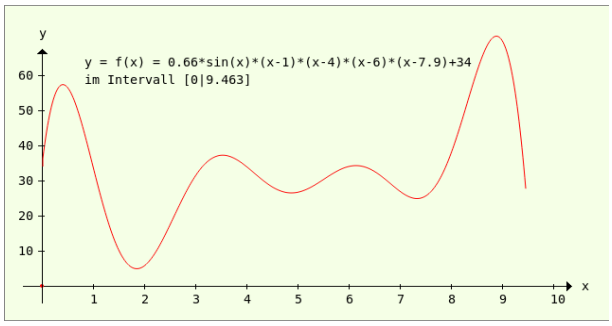

Abbildung 23.3.3.3.4: Bild einer Funktion y = f(x)

23.3.3.4 Zeichnen mit Methoden der Klasse Paint

Zugegeben – das Konzept zum Zeichnen geometrischer Formen mit Paint ist gewöhnungsbedürftig, denn eine geometrische Form wird stets *in drei Schritten* auf die benutzte Zeichenfläche gezeichnet:

1. Zuerst weisen Sie dem Pinsel (*Paint.Brush*) geeignete Eigenschaften wie zum Beispiel Füll-Farbe oder Linien-Farbe durch den Aufruf von Paint.Color, Paint.Image, Paint.LinearGradient oder Paint.RadialGradient zu. Die Linienbreite legen Sie bei Bedarf separat fest.
2. Dann legen Sie den so genannten *Zeichenpfad* fest, der durch einen Start-Punkt und die Angabe einer geometrischen Form beschrieben wird. Einen Start-Punkt legen Sie über die Methode *Paint.MoveTo(..)* fest oder Sie nutzen den Endpunkt einer abgeschlossenen Zeichen-Aktion. Die Form definieren Sie mit Hilfe ausgewählter Zeichen-Methoden wie zum Beispiel Paint.LineTo(..), Paint.Rectangle(..) oder Paint.Arc(..) mit bestimmten Argumenten. Unterscheiden Sie bei *Flächen* stets zwischen dem Rand (Form Linie) und der Fläche innerhalb der Umrandung.
3. Abschließend lassen Sie den Zeichenpfad mit dem definierten Pinsel auf der Zeichenfläche (Device) nachzeichnen, wobei Sie für eine festgelegte Linie die Methode *Paint.Stroke(A)* verwenden und für eine Fläche die Methode *Paint.Fill(A)*. Im Normalfall wird der zuletzt benutzte Zeichenpfad nach (einmaligem) Nachzeichnen gelöscht. Setzen Sie das optionale Argument A auf den Wert True, dann bleibt der Zeichenpfad erhalten und kann weiter genutzt werden.

Das folgende Beispiel greift den letzten Schritt mit dem Hinweis auf das optionale Argument A noch einmal konkret auf. Ein Kreis mit rotem Rand und gelber Kreisfläche soll auf ein *Picture(-Objekt)* als Zeichenfläche (Device) gezeichnet werden.

```
Public hPicture As Picture

Public Sub Form_Open()
  daCanvas.Cached = False ' Standard für eine DrawingArea
  hPicture = New Picture(daCanvas.Width, daCanvas.Height, True)
  hPicture.Fill(&HC3DDFF)
  PaintScriptCircle()
End ' Form_Open()

Public Sub PaintScriptCircle()
  Paint.Begin(hPicture)
    Paint.Translate(30, 300)                  ' Verschiebung des Koordinatensystems
    Paint.Scale(1, -1)                        ' Invertierung der Richtung der y-Achse → +y ▲
    Paint.Brush = Paint.Color(Color.Red)      ' Farbe Kreisbogen
    Paint.LineWidth = 5                       ' Dicke des Kreisbogens
    Paint.Arc(200, 150, 70)                   ' Kreismittelpunkt M(200|150), Radius 70
    Paint.Stroke(True)                        ' Kreisbogen (nach-)zeichnen → Pfad bleibt erhalten!
    Paint.Brush = Paint.Color(Color.Yellow)   ' Farbe Kreisfläche
    Paint.Fill()                              ' Kreisfläche farbig ausfüllen → Pfad wird gelöscht
  Paint.End
End ' PaintScriptCircle()
```

Um sich vom Ergebnis des Zeichnens zu überzeugen, wird das Picture – das sich bisher nur im Speicher befindet – in eine DrawingArea (daCanvas) gezeichnet und somit erst sichtbar:

```
Public Sub daCanvas_Draw()
  Paint.Begin(daCanvas)
    If hPicture Then Paint.DrawPicture(hPicture, 0, 0)
  Paint.End
End ' daCanvas_Draw()
```

Diese flexible Art und Weise zu zeichnen hat sich bewährt. Wollen Sie das Picture ausdrucken, dann reichen folgende zusätzlichen Zeilen, mit denen Sie das Picture auf den Drucker (myPrinter) zeichnen:

```
Public Sub myPrinter_Draw()

  Paint.Begin(myPrinter)
    If hPicture Then Paint.DrawPicture(hPicture, 0, 0)
  Paint.End

End ' myPrinter_Draw()
```

Den Ausdruck des Bildes starten Sie nach einem Drucker-Dialog mit 'myPrinter.Print'.

Wenn Sie dagegen das Bild sichern und weiterverarbeiten wollen, um zum Beispiel Zahlen als Daten in einem Text anschaulich zu präsentieren, so speichern Sie das Picture als Bild-Datei ab:

```
Public Sub btnSaveCurPicture_Click()
  Dim sPictureFileName As String

  sPictureFileName = Lower(cmbPictures.Text & ".png")
  hPicture.Save(sPicturePath &/ sPictureFileName)
End ' btnSaveCurPicture_Click()
```

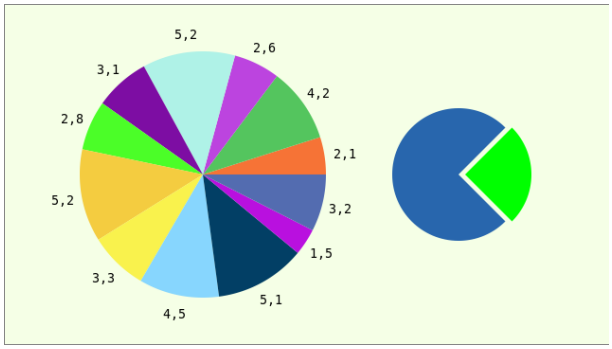

Abbildung 23.3.3.4.1: Kreis-Diagramm mit 12 Werten

Im Kapitel → 23.3.4.1 Projekte1 werden Ihnen in *einem* Projekt 11 weitere Beispiele vorgestellt und ausführlich kommentiert.

23.3.3.5 Standardwerte

Für das Zeichnen mit Paint gibt es festgelegte (Standard-)Startwerte, auf die Sie setzen können:

- Die Linien- oder Füll-Farbe oder die Schriftfarbe ist undurchsichtiges schwarz – ohne Alpha-Kanal in der Farbe. Ist das Paint-Device eine DrawingArea, so ist es ihre Hintergrundfarbe.
- Die Linienstärke beträgt 1.
- Die Kantenglättung (AntiAlias) ist eingeschaltet.
- Die Standardschriftart (Font) in Gambas ist die System-Schriftart. Sie können die Standard-Schriftart in den Gambas-Einstellungen (Schnittstelle → Schriften) ändern.

Bei einem Test ergeben sich diese Werte:

```
Print Paint.Background            ' 0 → schwarz
Print Paint.LineWidth             ' 1
Print Paint.AntiAlias             ' TRUE
Print Paint.Font.ToString()       ' Ubuntu, 11
```

23.3.3.6 Hinweise zur DrawingArea als Zeichenfläche

Mit der DrawingArea steht eine Zeichenfläche zur Verfügung, auf die aus der Anwendung heraus in zwei Modi gezeichnet werden kann. Dreh- und Angelpunkt beim Zeichen auf eine DrawingArea ist das Verständnis des Zusammenhangs zwischen dem Wert der Eigenschaft *DrawingArea.Cached* und dem Einsatz des Events *DrawingArea_Draw()*.

- Wenn *DrawingArea.Cached* = *True* ist, dann wird die DrawingArea in einen 'Leinwand'-Modus geschaltet: Alles, was gezeichnet wird, bleibt auf der DrawingArea sichtbar – solange die Zeichnung nicht mit der Methode *DrawingArea.Clear* gelöscht wird.
- Ist jedoch die Eigenschaft *DrawingArea.Cached* auf den Wert *False* gesetzt, dann bleibt die Zeichnung nur so lange sichtbar, solange die DrawingArea nicht durch eine anderes Fenster verdeckt oder versteckt wurde, weil der (flüchtige) Bildspeicher geleert wurde. Das Bild muss neu gezeichnet werden, um wieder sichtbar zu sein. Genau in diesen Momenten wird das Draw-Event automatisch ausgelöst.
- Das Draw-Event wird in Abhängigkeit vom Wert der Eigenschaft *DrawingArea.Cached* benutzt oder nicht benutzt.

Da die Klasse *Draw* seit der Gambas-Version 3.4 als veraltet gilt, sollten Sie zukünftig nur noch die Klasse *Paint* einsetzen. Deshalb existiert auch die Eigenschaft *DrawingArea.Painted* nicht mehr!

Interessant ist ein Blick in das Modul *Draw.module*, das Sie in den Gambas-Quell-Dateien finden. Er offenbart, dass die Klasse Draw über die Klasse Paint re-definiert wird. Damit wird gesichert, dass gegenwärtig Ihre bestehenden Zeichen-Projekte unter Einsatz der Klasse Draw einsatzfähig bleiben.

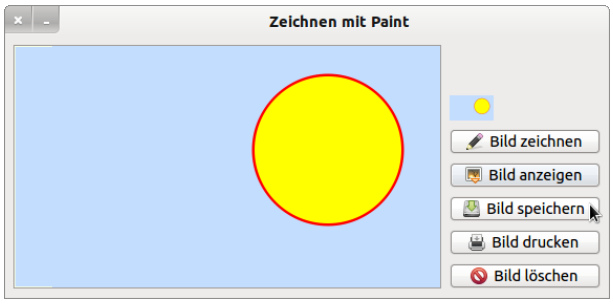

Abbildung 23.3.3.6.1: Start-Projekt

Im Download-Bereich finden Sie ein Projekt, das die o.a. prinzipiellen Ansätze 1:1 umsetzt. Die Sicht auf den vollständigen Quelltext hat vor allem für jene Vorteile, die sich zum ersten mal dem Zeichnen mit Methoden der Klasse Paint zuwenden.

Quelltext:

```
' Gambas class file

Private hPicture As Picture
Private iPrinterDPI As Integer
Private fDesktopresolutionProInch As Float = (Desktop.Resolution / 25.4)
Private bPainted As Boolean = False

Public Sub Form_Open()
  FMain.Center
  FMain.Resizable = False
  pboxPreView.Stretch = True
  daCanvas.Cached = True ' daCanvas ist eine DrawingArea
End ' Form_Open()

Public Sub PaintScriptCircle()
  Try hPicture = New Picture(daCanvas.Width, daCanvas.Height, True) ' Ein Picture-Objekt wird erzeugt
  hPicture.Fill(&HC3DDFF) ' Das Picture bekommt eine hell-blaue Hintergrundfarbe

  Paint.Begin(hPicture)                          ' Auf das Picture wird nun gezeichnet ...
    Paint.Translate(30, 200)                     ' Verschiebung des Koordinatensystems
    Paint.Scale(1, -1)                           ' Invertierung der Richtung der y-Achse → +y ▼
    Paint.Brush = Paint.Color(Color.Red)         ' Farbe Kreisbogen → rot
    Paint.LineWidth = 5                          ' Dicke des Kreisbogens
    Paint.Arc(270, 100, 70)                      ' Kreismittelpunkt M(270|100), Radius 70
    Paint.Stroke(True)                           ' Kreisbogen (nach-)zeichnen → Pfad bleibt erhalten!
    Paint.Brush = Paint.Color(Color.Yellow)      ' Farbe Kreisfläche → gelb
    Paint.Fill()                                 ' Kreisfläche farbig ausfüllen → Pfad wird gelöscht
  Paint.End

End ' PaintScriptCircle()

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

Public Sub myPrinter_Draw()
  Dim iDruckBreite, iDruckHoehe, iDruckrandLinks, iDruckrandOben As Integer
  Dim imgToPrint As Image

  iDruckrandLinks = 25 ' Millimeter – erprobter Wert
  iDruckrandOben = 20  ' Millimeter – erprobter Wert

  If Not hPicture Then
     Return
  Else
   ' Dem Image imgToPrint wird das in ein Image konvertierte hPicture zugewiesen
     imgToPrint = hPicture.Image
  Endif

' Automatische Umschaltung von Querformat auf Hochformat, wenn Bild.W > Bild.H
  If imgToPrint.Width > imgToPrint.Height Then imgToPrint = imgToPrint.Rotate(Pi(0.5))
```

```
' Umstellung von Einheit DA-Punkt auf Einheit Millimeter
  Paint.Scale(Paint.Width / myPrinter.PaperWidth, Paint.Height / myPrinter.PaperHeight)

' Die benutzten Werte 170 (Weite) und 260 (Höhe) sind erprobt für A4
  If PixelToMillimeter(imgToPrint.W) > 170 Then
     iDruckBreite = 170
     iDruckHoehe = CInt(iDruckBreite * (imgToPrint.Height / imgToPrint.Width))
     If iDruckHoehe > 260 Then
        iDruckHoehe = 260
        iDruckBreite = CInt(iDruckHoehe / (imgToPrint.Height / imgToPrint.Width))
     Endif ' iDruckHoehe > 260?
     Paint.DrawImage(imgToPrint, iDruckrandLinks, iDruckrandOben, iDruckBreite, iDruckHoehe)
     Return
  Endif ' PixelToMillimeter(imgToPrint.W) > 170?

  If PixelToMillimeter(imgToPrint.H) > 260 Then
     iDruckHoehe = 260
     iDruckBreite = CInt(iDruckHoehe / (imgToPrint.Height / imgToPrint.Width))
     If iDruckBreite > 170 Then
        iDruckBreite = 170
        iDruckHoehe = CInt(iDruckBreite / (imgToPrint.Height / imgToPrint.Width))
     Endif ' iDruckBreite > 170?
     Paint.DrawImage(imgToPrint, iDruckrandLinks, iDruckrandOben, iDruckBreite, iDruckHoehe)
     Return

  Endif ' PixelToMillimeter(imgToPrint.H) > 260?

  iDruckBreite = PixelToMillimeter(imgToPrint.W)
  iDruckHoehe = CInt(iDruckBreite * (imgToPrint.Height / imgToPrint.Width))

' Das zu druckende Bild – jetzt als Image imgToPrint – wird in den Drucker gedruckt
  Paint.DrawImage(imgToPrint, iDruckrandLinks, iDruckrandOben, iDruckBreite, iDruckHoehe)

End ' myPrinter_Draw()

Public Sub btnDrawPicture_Click()
  PaintScriptCircle() ' Das Bild – Kreis mit rotem Rand und gelber Fläche – wird auf das Picture gezeichnet
  bPainted = True
  pboxPreView.Picture = hPicture ' Mini-Vorschaubild, da das hPicture-Objekt nur im Speicher existiert!
End ' btnDrawPicture_Click()

Public Sub btnShowPicture_Click()
  If bPainted Then
     daCanvas_Draw() ' Das Picture wird in die DrawingArea gezeichnet
  Endif
End ' btnShowPicture_Click()

Public Sub btnSavePicture_Click()
  Dim sPictureFileName As String

  sPictureFileName = Lower("startfrei.png")
  If Exist(Application.Path &/ sPictureFileName) Then Kill Application.Path &/ sPictureFileName
  Wait
  If bPainted Then
     hPicture.Save(Application.Path &/ sPictureFileName) ' Das Picture wird als Bild-Datei gespeichert
     Wait
  Endif
  pboxPreView.Enabled = False
  Wait 0.3
  pboxPreView.Enabled = True
End ' btnSaveCurPicture_Click()

Public Sub btnPrintPicture_Click()
  If bPainted Then
     If myPrinter.Configure() Then Return ' Drucker-Dialog
     Me.Enabled = False          ' Das Formular wird deaktiviert
        Inc Application.Busy     ' Das Programm nimmt keine Eingaben mehr entgegen ...
            myPrinter.Print      ' Der Ausdruck wird gestartet
        Dec Application.Busy     ' Das Programm nimmt wieder Eingaben entgegen...
     Me.Enabled = True           ' Das Formular wird aktiviert
  Endif
End ' btnPrintImage_Click()

Public Sub btnClearPicture_Click()
  hPicture = Null ' Das Picture-Objekt wird gelöscht
  daCanvas.Clear
  pboxPreView.Picture = hPicture
  bPainted = False
End ' btnClearPicture_Click()

Private Function PixelToMillimeter(iPixel As Integer) As Float
  Return iPixel / fDesktopresolutionProInch
End ' PixelToMillimeter(iPixel As Integer) As Float
```