18.11.1 Projekte – Streifen-Diagramme

Grundlegend für das Zeichnen auf einer Zeichenfläche (Canvas, Device) wie einer DrawingArea sind die Betrachtungen im Buch-Kapitel '3.3.3 Zeichnen mit Paint' sowie die folgenden Kapitel 'Paint-Projekte 1' und 'Paint-Projekte 2'.

18.11.1.0 Projekt 1

Im ersten Projekt wird auf der Basis einer Datenreihe ein Streifen-Diagramm gezeichnet und passend beschriftet. Als Zeichenfläche dient eine DrawingArea:

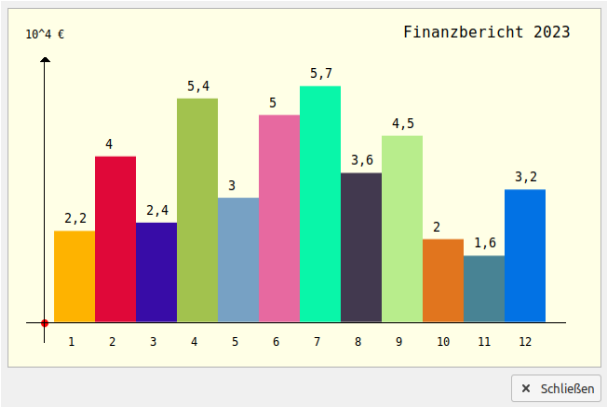

Abbildung 18.11.1.0.1: GUI BarChart – Streifen-Diagramm – 'Finanzbericht 2023'

Der hinreichend kommentierte Quelltext mit allen Zeichnen-Prozeduren wird komplett angegeben:

```
' Gambas class file

Public xTranslate As Float
Public yTranslate As Float
Public xScale As Float = 1
Public yScale As Float = -1


Public Sub Form_Open()

    FMain.Resizable = False
    FMain.Arrangement = Arrange.Vertical
    FMain.Margin = True
    FMain.Spacing = True

    DrawingAreaCanvas.Expand = True

End

Public Sub DrawingAreaCanvas_Draw()

'-- Draw the bar chart. The canvas is a DrawingArea.
    DrawBarChart()

End

Public Sub DrawBarChart()

    Dim i As Integer
    Dim fDeltaX, fDeltaY, fOffsetX, fEndX, fEndY, fMaxValue As Float
    Dim aData, aDataC As Float[]
    Dim sCaption As String
    Dim iBarColor As Integer
    Dim aColorMatrix As New Integer[]

    aColorMatrix = [16691968, 14682169, 3673255, 10666574, 7840196, 15165856, 652969, /
                    4340047, 12119436, 14775582, 4752276, 160484]

'-- Inline array with the barchart values to be displayed
    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()   '-- Copy of the array of original values
    aDataC.Sort(gb.Descent) '-- Descending sorting of the copy elements
    fMaxValue = aDatac[0]   '-- The 1st element is now the largest value in the copied array

    fOffsetX = 10 '-- Definition of the abscissa offset ► +10
```

```
    fEndX = 540  '-- Definition of fixed coordinates for the size
    fEndY = 240
    fDeltaX = Round((fEndX - fOffsetX) / aData.Count, 0) '-- Normalised strip width  (unit)
    fDeltaY = Round(fEndY / fMaxValue, 0)                '-- Normalised strip height (unit)

    xTranslate = 40.0
    yTranslate = 320.0
    Paint.Translate(xTranslate, yTranslate)
    Paint.Scale(1, -1) ' +y ▲

    SetOrigin()
    Draw_X_Axis()
    Draw_Y_Axis()
    Draw_Y_AxisArrow()

'-- Labelling Caption
    Paint.Scale(1, -1) ' +y ▼
    sCaption = ("Financial Report 2023")
    Paint.Font = Font["Monospace, 12"]
    Paint.DrawText(sCaption, 385, -290)
    Paint.Scale(1, -1) ' +y ▲

'-- All 12 financial values are drawn as stripes
    For i = 0 To aData.Max
    '-- Calculation and drawn of all stripes (rectangles) in the diagram with stripe color
        iBarColor = aColorMatrix[i]
        Paint.FillRect(fOffsetX + i * fDeltaX, 1.3, fDeltaX, fDeltaY * aData[i], iBarColor)
        Paint.Scale(1, -1) ' +y ▼
          Paint.Brush = Paint.Color(Color.Black)
          Paint.Font = Font["Monospace, 10"]
          Paint.DrawText(Str(aData[i]), fOffsetX + i * fDeltaX, -fDeltaY * aData[i] - 30, fDeltaX, 30,
Align.Center)
          Paint.Font = Font["Monospace, 10"]
          Paint.DrawText(Str(i + 1), fOffsetX + i * fDeltaX, 1, fDeltaX, 30, Align.Center)
        Paint.Scale(1, -1)
    Next

'-- Labelling the y-axis
    Paint.Scale(1, -1) ' +y ▼
    sCaption = ("Value in 10^4 €")
    Paint.Font = Font["Monospace, 9"]
    If Paint.TextSize(sCaption).W / 2 > xTranslate Then
       Paint.DrawText(sCaption, 0 - xTranslate / 2, -290)
    Else
       Paint.DrawText(sCaption, 0 - Paint.TextSize(sCaption).W / 2, -290)
    Endif
    Paint.Scale(1, -1)

End

Public Sub DrawingAreaCanvas_MouseDown()
'-- Display of the (relative) coordinates - for tests only
    Print Str(Mouse.X - xTranslate) & " | " & Str(-Mouse.Y + yTranslate)
End

'-------------------------------------------------------------------------------------

Private Sub SetOrigin()

'-- Origin of coordinates
    Paint.Brush = Paint.Color(Color.Red)
      Paint.MoveTo(0, 0)
      Paint.Arc(0, 0, 4)
      Paint.Fill()
    Paint.Brush = Paint.Color(Color.Black)

End

Private Sub Draw_X_Axis()

'-- x-axis
    Paint.AntiAlias = False
      Paint.MoveTo(-20, 0)
      Paint.LineTo(560, 0)
      Paint.Stroke()
    Paint.AntiAlias = True

End

Private Sub Draw_Y_Axis()

'-- y-axis
    Paint.AntiAlias = False
      Paint.MoveTo(0, -20)
```

```
      Paint.LineTo(0, 270)
      Paint.MoveTo(0, 0)
      Paint.Stroke()
    Paint.AntiAlias = True

End

Private Sub Draw_Y_AxisArrow()

    Dim i As Integer

'-- y-axis (arrow)
    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

'------------------------------------------------------------------------------

Public Sub ButtonClose_Click()

    FMain.Close()

End
```

Hinweise

- Das Diagramm wird mit festen Werten und Farben für die 12 Streifen gezeichnet.
- Das Zeichnen auf der Zeichenfläche DrawingAreaCanvas erfolgt in der Ereignisbehandlungsroutine DrawingAreaCanvas_Draw().
- Die Hauptlast trägt die Prozedur DrawBarChart(), in die Sie die 4 Prozeduren für das Zeichnen des (verschobenen) Koordinatensystem auch direkt einfügen könnten.
- Beachten Sie, dass die Methode Paint.Scale(1, -1) – je nach Zeichnen-Kontext – genutzt wird, um auch die Richtung für die y-Achse im Koordinatensystem zu ändern, weil das für das Zeichnen geometrischer Objekte und für das Zeichnen von Text unterschiedliche Richtungen in der y-Richtung sind!
- Gezeichnet wird in ein Koordinatensystem, bei dem die y-Achse nach oben zeigt und das über Paint.Translate(xTranslate, yTranslate) einen verschobenen Koordinatenursprung hat.
- Der Offset der Streifen mit fOffsetX = 10 dient nur zur besseren Sichtbarkeit des ersten Streifens.

18.11.1.1 Projekt 2

Das Besondere an diesem Projekt sind die erweiterten Forderungen gegenüber dem ersten Projekt, denn der Inhalt der DrawingArea mit dem Streifen-Diagramm soll

- nicht nur angezeigt werden,
- sondern auch in einer Bild-Datei mit festem Datei-Pfad abgespeichert werden und
- sofort ausgedruckt werden können, wobei in einem Dialog wesentliche Druck-Parameter eingestellt werden. Der Dialog soll auch eine Druck-Vorschau bieten. Das gelingt aber nur dann, wenn Sie die Komponente gb.gtk3 einsetzen, denn beim Einsatz von gb.qt5 fehlt eine Druck-Vorschau.

Auf den Inhalt einer DrawingArea können Sie nicht direkt zugreifen.

- Der Kniff besteht darin, zuerst eine Prozedur zu schreiben, die auf ein Picture als Zeichenfläche genau das zeichnet, was zu zeichnen ist – das Diagramm mit allen Bezeichnungen.
- Dann können Sie das Picture auf eine DrawingArea zeichnen.
- Danach können Sie den Inhalt des Pictures als Bild-Datei speichern.
- Abschließend besteht die Möglichkeit, das in ein Image konvertierte Picture auszudrucken.

Sie können beim Druck in eine Datei im Dialog als Ausgabeformat PDF oder PostScript oder SVG auswählen oder das Image auf einem Drucker wie einem Laserdrucker ausdrucken.

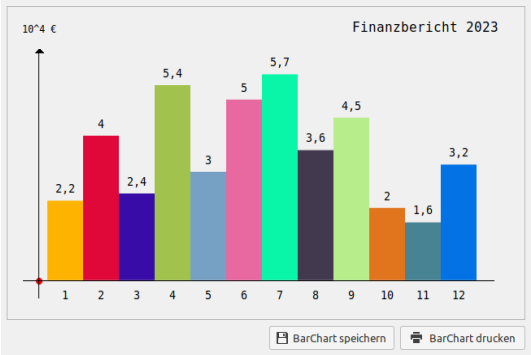

Abbildung 18.11.1.1.1: GUI BarChart – Streifen-Diagramm in der DrawingArea

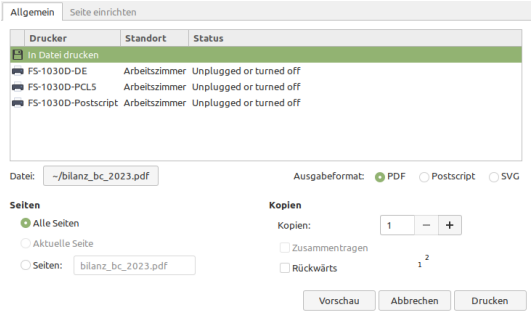

Abbildung 18.11.1.1.2: Ausdruck in eine Datei – 3 Formate

Abbildung 18.11.1.1.3: (Druck-)Vorschau

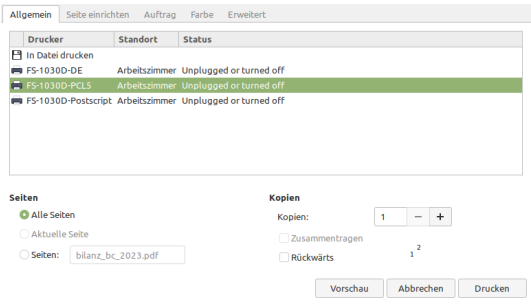

Abbildung 18.11.1.1.4: Ausdruck auf einen Drucker – FS-1030D

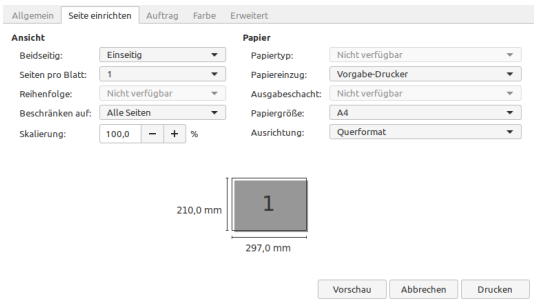

Abbildung 18.11.1.1.5: Drucken – Seite einrichten

Der hinreichend kommentierte Quelltext wird auch für das zweite Projekt komplett angegeben, um die Umsetzung des o.a. Konzeptes zu verdeutlichen. Anschließende Hinweise ergänzen die Kommentare:

```
(1)  ' Gambas class file
(2)
(3)  Public hPicture As Picture
(4)  Public hImage As Image
(5)
(6)  Public xTranslate As Float
(7)  Public yTranslate As Float
(8)  Public xScale As Float = 1
(9)  Public yScale As Float = -1
(10)
(11)
(12) Public Sub Form_Open()
(13)
(14)     FMain.Resizable = False
(15)     FMain.Arrangement = Arrange.Vertical
(16)     FMain.Margin = True
(17)     FMain.Spacing = True
(18)
(19)     DrawingAreaCanvas.Expand = True
(20)
(21) '-- Draw the bar chart. The canvas is a picture (hPicture)!
(22)     DrawBarChart()
(23) '-- Draw the picture in the DrawingArea by raising its _Draw event
(24)     DrawingAreaCanvas.Refresh()
(25)
(26) End
(27)
(28) Public Sub DrawingAreaCanvas_Draw()
(29)
(30)     Paint.DrawPicture(hPicture, 0, 0)
(31)
(32) End
(33)
(34) Public Sub DrawBarChart()
(35)
(36)     Dim i As Integer
(37)     Dim fDeltaX, fDeltaY, fOffsetX, fEndX, fEndY, fMaxValue As Float
(38)     Dim aData, aDataC As Float[]
(39)     Dim sCaption As String
(40)     Dim iBarColor As Integer
(41)     Dim aColorMatrix As Integer[]
(42)
(43)     aColorMatrix = [16691968, 14682169, 3673255, 10666574, 7840196, 15165856, 652969, 4340047,
       12119436, 14775582, 4752276, 160484]
(44)
(45) '-- Creation of a object with defined size of type Picture
(46)     hPicture = New Picture(DrawingAreaCanvas.W, DrawingAreaCanvas.H)
(47)     hPicture.Fill(&HF0F0F0)
(48)
(49)     Paint.Begin(hPicture)
(50)   '-- Inline array with the values to be displayed
(51)       aData = [2.2, 4, 2.4, 5.4, 3, 5.0, 5.7, 3.6, 4.5, 2.0, 1.6, 3.2]
(52)       aDataC = aData.Copy()    '-- Copy of the array of original values
(53)       aDataC.Sort(gb.Descent) '-- Descending sorting of the copy elements
(54)       fMaxValue = aDatac[0]   '-- The 1st element is now the largest value in the copied array
(55)
(56)       fOffsetX = 10 '-- Definition of the abscissa offset ↦ +10
(57)       fEndX = 540   '-- Definition of fixed coordinates for the size
(58)       fEndY = 240
(59)       fDeltaX = Round((fEndX - fOffsetX) / aData.Count, 0) '-- Normalised strip width (unit)
(60)       fDeltaY = Round(fEndY / fMaxValue, 0)                '-- Normalised strip height (unit)
(61)
```

```
(62)        xTranslate = 40.0
(63)        yTranslate = 320.0
(64)        Paint.Translate(xTranslate, yTranslate)
(65)        Paint.Scale(1, -1) ' +y ▲
(66)
(67)        SetOrigin()
(68)        Draw_X_Axis()
(69)        Draw_Y_Axis()
(70)        Draw_Y_AxisArrow()
(71)
(72)    '-- Labelling Caption
(73)        Paint.Scale(1, -1) ' +y ▼
(74)        sCaption = ("Financial Report 2023")
(75)        Paint.Font = Font["Monospace, 12"]
(76)        Paint.DrawText(sCaption, 385, -290)
(77)        Paint.Scale(1, -1) ' +y ▲
(78)
(79)    '-- All 12 financial values are drawn as stripes
(80)        For i = 0 To aData.Max
(81)        '-- Calculation and drawn of all stripes (rectangles) in the diagram with stripe color
(82)            iBarColor = aColorMatrix[i]
(83)            Paint.FillRect(fOffsetX + i * fDeltaX, 1.2, fDeltaX, fDeltaY * aData[i], iBarColor)
(84)            Paint.Scale(1, -1) ' +y ▼
(85)              Paint.Brush = Paint.Color(Color.Black)
(86)              Paint.Font = Font["Monospace, 10"]
(87)              Paint.DrawText(Str(aData[i]), fOffsetX + i * fDeltaX, -fDeltaY * aData[i] - 30, fDeltaX,
      30, Align.Center)
(88)              Paint.Font = Font["Monospace, 10"]
(89)              Paint.DrawText(Str(i + 1), fOffsetX + i * fDeltaX, 1, fDeltaX, 30, Align.Center)
(90)            Paint.Scale(1, -1)
(91)        Next
(92)
(93)    '-- Labelling the y-axis
(94)        Paint.Scale(1, -1) ' +y ▼
(95)        sCaption = ("Value in 10^4 €")
(96)        Paint.Font = Font["Monospace, 9"]
(97)        If Paint.TextSize(sCaption).W / 2 > xTranslate Then
(98)           Paint.DrawText(sCaption, 0 - xTranslate / 2, -290)
(99)        Else
(100)           Paint.DrawText(sCaption, 0 - Paint.TextSize(sCaption).W / 2, -290)
(101)         Endif
(102)         Paint.Scale(1, -1)
(103)       Paint.End()
(104)
(105)   '-- The content of the Picture (current canvas) is assigned to an image that is used for printing
(106)       hImage = hPicture.Image
(107)
(108)   End
(109)
(110)   Public Sub ButtonSaveBarChart_Click()
(111)
(112)       hPicture.Save(Application.Path &/ "chart/barchart.png")
(113)
(114)   End
(115)
(116)   Public Sub ButtonPrintBarChart_Click()
(117)
(118)       Printer1.Paper = Printer1.A4
(119)       Printer1.Orientation = Printer.Landscape
(120)
(121)   '-- Color printing is standard and therefore not required as a specification
(122)       Printer1.GrayScale = False
(123)       Printer1.FullPage = True  '-- Should always be used!
(124)
(125)       Printer1.NumCopies = 1
(126)       If Printer1.NumCopies > 1 Then
(127)          Printer1.ReverseOrder = True
(128)          Printer1.CollateCopies = True
(129)       Else
(130)          Printer1.ReverseOrder = False
(131)          Printer1.CollateCopies = False
(132)       Endif
(133)
(134)   '-- Definition of the 4 margins with fixed margin values
(135)       Printer1.MarginLeft = 30
(136)       Printer1.MarginTop = 30
(137)       Printer1.MarginRight = 100
(138)       Printer1.MarginBottom = 10
(139)
(140)       Printer1.OutputFile = User.Home &/ "bilanz_bc_2023.pdf"
(141)
(142)       If Printer1.Configure() Then Return
(143)       Printer1.Print()
(144)
```

```
(145)  End
(146)
(147)  Public Sub Printer1_Begin()
(148)
(149)      Printer1.Count = 1 '-- This specification ( > 0 ) is required
(150)
(151)  End
(152)
(153)  Public Sub Printer1_Draw()
(154)
(155)      Dim fPrintW, fPrintH As Float
(156)      Dim fRatioImage As Float
(157)
(158)      Paint.Scale(Paint.Width / Printer1.PaperWidth, Paint.Height / Printer1.PaperHeight)
(159)      Paint.Translate(Printer1.MarginLeft, Printer1.MarginTop)
(160)
(161)      fRatioImage = hImage.W / hImage.H
(162)      fPrintW = Printer1.PaperWidth - (Printer1.MarginLeft + Printer1.MarginRight)
(163)      fPrintH = Printer1.PaperHeight - (Printer1.MarginTop + Printer1.MarginBottom)
(164)
(165)  '-- Match diagram rectangle into the target format
(166)      If fRatioImage > (Paint.W / Paint.H) Then
(167)         Paint.DrawImage(hImage, 0, 0, fPrintW, fPrintW / fRatioImage)
(168)      Else
(169)         Paint.DrawImage(hImage, 0, 0, fPrintH / fRatioImage, fPrintH)
(170)      Endif
(171)
(172)  End
(173)
(174)  '-------------------------------------------------------------------------------------
(175)
(176)  Private Sub SetOrigin()
(177)
(178)  '-- Origin of coordinates
(179)      Paint.Brush = Paint.Color(Color.Red)
(180)      Paint.MoveTo(0, 0)
(181)      Paint.Arc(0, 0, 4)
(182)      Paint.Fill()
(183)      Paint.Brush = Paint.Color(Color.Black)
(184)
(185)  End
(186)
(187)  Private Sub Draw_X_Axis()
(188)
(189)  '-- x-axis
(190)      Paint.AntiAlias = False
(191)        Paint.MoveTo(-20, 0)
(192)        Paint.LineTo(560, 0)
(193)        Paint.Stroke()
(194)      Paint.AntiAlias = True
(195)
(196)  End
(197)
(198)  Private Sub Draw_Y_Axis()
(199)
(200)  '-- y-axis
(201)      Paint.AntiAlias = False
(202)        Paint.MoveTo(0, -20)
(203)        Paint.LineTo(0, 270)
(204)        Paint.MoveTo(0, 0)
(205)        Paint.Stroke()
(206)      Paint.AntiAlias = True
(207)
(208)  End
(209)
(210)  Private Sub Draw_Y_AxisArrow()
(211)
(212)      Dim i As Integer
(213)
(214)  '-- y-axis (arrow)
(215)      Paint.AntiAlias = False
(216)        For i = 1 To 5
(217)          Paint.MoveTo(-6 + i, 264 + i)
(218)          Paint.LineTo(7 - i, 264 + i)
(219)          Paint.Stroke()
(220)        Next
(221)      Paint.AntiAlias = True
(222)
(223)  End
(224)
(225)  '-------------------------------------------------------------------------------------
(226)
(227)
(228)  Public Sub Form_Close()
```

```
(229)
(230)      Dim hWindow As Window
(231)
(232)  '-- Close all open windows
(233)      For Each hWindow In Windows
(234)        hWindow.Close()
(235)      Next
(236)
(237)  End
```

Die Hinweise beziehen sich auf die Zeilen 265 bis 270

```
-- Match diagram rectangle into the target format
   If fRatioImage > (Paint.W / Paint.H) Then
      Paint.DrawImage(hImage, 0, 0, fPrintW, fPrintW / fRatioImage)
   Else
      Paint.DrawImage(hImage, 0, 0, fPrintH / fRatioImage, fPrintH)
   Endif
```

Für den ersten Fall gilt: Die Breite des BarChart entspricht der verfügbaren Breite des Druckbereichs. Die Höhe wird entsprechend der Original-Proportionen errechnet:

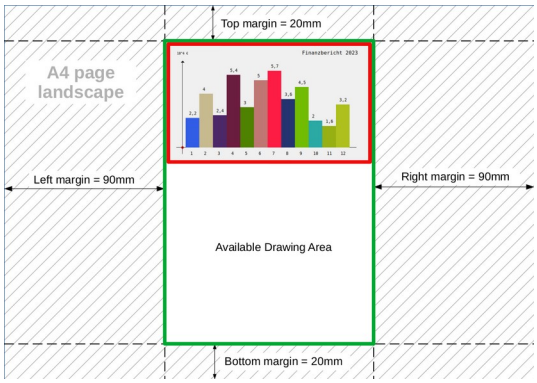

Abbildung 18.11.1.1.6: BarChart – Druckbreite

Für den zweiten Fall gilt: Die Höhe des BarChart entspricht der verfügbaren Höhe des Druckbereichs. Die Breite wird entsprechend der Original-Proportionen errechnet:

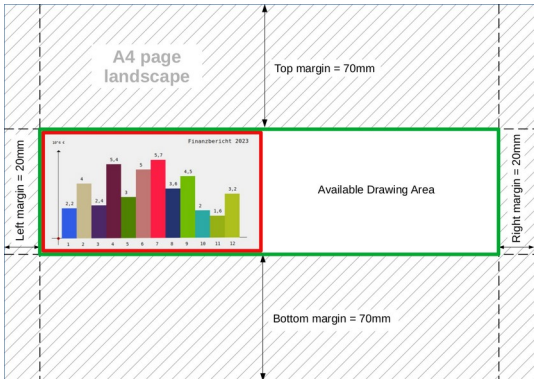

Abbildung 18.11.1.1.7: BarChart – Druckhöhe

Den Quelltext für die beiden Projekte finden Sie in zwei Projekt-Archiven im Download-Bereich.