12.2.8 Form – Selbst entwickelte Steuer-Elemente

Im Projekt *RSS-Reader* im → Kapitel 18.12.5 wird gezeigt, wie Sie zusätzlich zum (Haupt-)Formular ein RSS-Feed-Formular in der Gambas-IDE erzeugen (→ Klasse FeedItem.Class) und mehrere Steuerelemente (2 TextAreas und eine PictureBox) in das RSS-Feed-Formular einfügen:

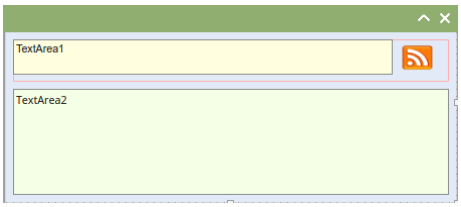

Abbildung 12.2.8.1: RSS-Feed-Formular (IDE)

Dann werden im (Haupt-)Formular entsprechend der *Anzahl der aktuellen Nachrichten* im ausgewählten RSS-Datenstrom RSS-Feed-Formulare erzeugt und sofort in einen ListContainer eingefügt und angezeigt. Unter dem Aspekt der Anwendung können Sie diese in das (Haupt-)Formular eingefügten RSS-Feed-Formulare als eigene, selbst entwickelte Steuerelemente betrachten – als zusammengesetztes Steuer-Element oder 'Compound Control'. Als Container für die RSS-Feed-Formulare dient das inzwischen als veraltet eingestufte Steuerelement *ListContainer*, für das es gegenwärtig mit dem Steuer-Element ScrollView hinreichend gleichwertigen Ersatz gibt.

Abbildung 12.2.8.2: Zwei RSS-Feed-Formulare in einem ListContainer

Interessant ist die Tatsache, dass viele Steuer-Elemente aus gb.form – welche die meisten Anwender vielleicht für "eingebaut" halten – als speziell gestaltete Form-Klassen erzeugt worden sind; ganz so, wie die Klasse FeedItem aus dem o.a. Projekt.

12.2.8.1 Strategie 1

Das Projekt *RSS-Reader* setzt die bereits im Kapitel 12.2.7 genannte Strategie konsequent um:

- Zuerst erzeugen Sie in der Gambas-IDE ein Formular – zum Beispiel FSelfmadeControl.
- Dann realisieren Sie das Formular FSelfmadeControl mit Gambas-Steuer-Elementen als eigenes (zusammengesetztes) Steuer-Element mit der vorgesehenen Funktionalität.
- Danach erproben Sie das Formular FSelfmadeControl als (Top-Level)-Fenster, indem Sie in der IDE das Formular FSelfmadeControl markieren und die Klasse FSelfmadeControl über das Kontext-Menü ausführen. Es spricht nichts dagegen, wenn Sie das Formular FSelfmadeControl in einem eigenen Projekt testen.
- Abschließend fügen Sie das selbst entwickelte Steuer-Element FSelfmadeControl in einen Container im (Haupt-)Formular so ein, als wäre es ein Steuer-Element wie jedes andere:

```
Private hFSelfmadeControl As FSelfmadeControl ' » Compound Control

Public Sub Form_Open()
...
  hFSelfmadeControl = New FSelfmadeControl(Container) As "hFSelfmadeControl"
...
End
```

Die Umsetzung der o.a. Strategie ist ein großartiger Weg, um eigene Steuerelemente auf einfache Art zu entwickeln, weil Sie mit dem Form-Editor in der IDE Formulare entwerfen und Quelltext der Klasse dazu schreiben können. Dieser Entwurfs-Vorgang in der IDE von Gambas erledigt sogar die Projektorganisation in Klassen als Umsetzung des Prinzips der Kapselung in der objekt-orientierten Programmierung (OOP).

12.2.8.2 Projekt 1

Im einem Projekt war es notwendig, eine Farb-Auswahl zu realisieren, die folgende Forderungen erfüllen sollte:

- Die Farbe sollte in einem Steuer-Element schnell auswählbar sein.
- Die Anzahl der Farben sollte einstellbar sein (Option).
- Die gewählte Farbe musste gut von den anderen Farben zu unterscheiden sein.
- Das Steuer-Element zur Farbwahl sollte auf dem (Haupt-)Formular hinreichend klein gemacht werden können.

Bevor Sie das Rad in so einem Fall neu erfinden, lohnt ein Blick in die Sammlung der Steuer-Elemente in der IDE von Gambas. Die relevanten Steuer-Elemente ColorChooser und ColorPalette erfüllten nicht alle gestellten Forderungen in geeigneter Weise. Gegen den ColorButton sprach die Farb-Wahl über einen zusätzlichen Dialog, wie es die beiden folgenden Abbildungen zeigen:

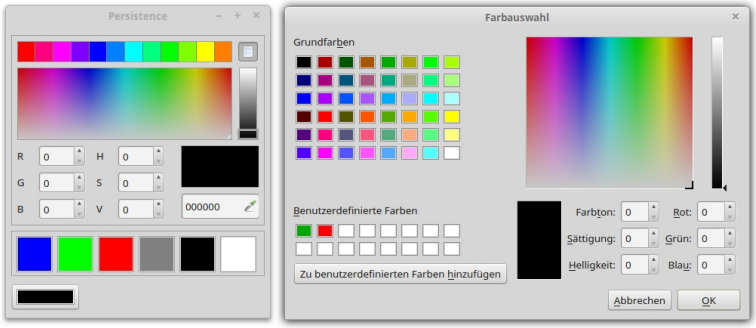

Abbildung 12.2.8.2.1: ColorButton-Dialog (rechts)

Damit stand fest: Das Steuer-Element *FMinimalColorChooser* wird nach der Strategie 1 in der IDE von Gambas selbst entwickelt! Das Steuer-Element verfügt über diese zwei Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
NumberOfColors	Integer	Setzt die Anzahl der Farben [1..12] in der Farb-Palette oder gibt die Anzahl der Farben zurück.
ValueOfColor	Integer	Gibt den Farb-Wert zurück, der in der Farb-Palette ausgewählt wurde.

Tabelle 12.2.8.2.1 : Eigenschaften der Klasse FMCChooser

Das Steuer-Element *FMinimalColorChooser* besitzt nur das Ereignis *Change*. Es wird ausgelöst, wenn eine Farbe in der Farb-Palette mit einem Maus-Klick ausgewählt wurde – sie geändert wurde:

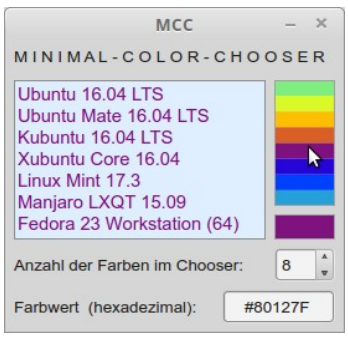

Abbildung 12.2.8.2.2: MiniColorChooser in Aktion

Um das selbst entwickelte Steuer-Element zu testen, wurde ein (Haupt-)Formular entworfen, so dass die gewählte Farbe im Steuer-Element *FMinimalColorChooser* als Vordergrund-Farbe für einen Text in einer TextArea verwendet wird. Zusätzlich wird die Anzahl der Farben in der Farbpalette in einer Spin-Box angezeigt, mit der auch deren Anzahl im vorgegebenen Intervall [1|12] geändert werden kann. Der aktuell ausgewählte Farb-Wert wird in einer TextBox (hexadezimal) angezeigt.

Quelltext:

```
[1]  ' Gambas class file
[2]
[3]  Private hMCChooser As FMinimalColorChooser ' ↘ Compound Control
[4]
[5]  Public Sub Form_Open()
[6]
[7]    FMain.Resizable = False
[8]    panMCC.Arrangement = Arrange.Fill
[9]  ' panMCC.Border = Border.Plain ' Option: Rand um das eingebettete Fenster
[10]   sboxNumberOfColors.MinValue = 1
[11]   sboxNumberOfColors.MaxValue = 12
[12]   txbColorHex.ReadOnly = True
[13]
[14]   hMCChooser = New FMinimalColorChooser(panMCC, 5) As "hMCChooser"
[15]
[16] ' Festlegung ausgewählter Eigenschaften des eingebetteten Fensters von FMinimalColorChooser
[17]   hMCChooser.Arrangement = Arrange.Vertical
[18]   hMCChooser.Spacing = True
[19] ' Festlegung ausgewählter Eigenschaften von Steuer-Elementen auf FMinimalColorChooser,
[20] ' deren Public-Eigenschaft zur Entwicklungszeit auf True gesetzt wurde
[21]   hMCChooser.dwgColors.Expand = True
[22]   hMCChooser.dwgColors.Border = Border.Solid
[23]   hMCChooser.panShowColor.Border = Border.Solid
[24]
[25]   hMCChooser_Change()
[26]
[27] End
[28]
[29] Public Sub hMCChooser_Change()
[30]   SetTAColor(hMCChooser.ValueOfColor)
[31]   txbColorHex.Text = "#" & Hex$(hMCChooser.ValueOfColor, 6)
[32]   sboxNumberOfColors.Value = hMCChooser.NumberOfColors
[33] End
[34]
[35] Public Sub sboxNumberOfColors_Change()
[36]   hMCChooser.NumberOfColors = sboxNumberOfColors.Value
[37] End
[38]
[39] Private Sub SetTAColor(FGColor As Integer)
[40]   TextArea1.Foreground = FGColor
[41]   txbColorHex.SetFocus()
[42] End
```

Kommentar:

- In den Zeilen 3 und 14 wird eine neue Instanz der Klasse FMinimalColorChooser definiert und erzeugt. Als Container wird das Panel *panMCC* angegeben und als 2. Argument die Anzahl der Farben in der Farb-Palette.
- In den Zeilen 29 bis 33 wird festgelegt, welche Reaktionen erfolgen sollen, wenn das Ereignis Change ausgelöst wird → Schriftfarbe in der TextArea ändern, den aktuellen Farb-Wert anzeigen und den Wert für die Anzahl der Farben in der SpinBox aktualisieren.
- Eine Änderung des Wertes in der SpinBox ändert den Wert der NumberOfColors-Eigenschaft für das Farb-Auswahl-Steuer-Element.

Wenig spektakulär sieht das Formular für das Steuer-Element aus:

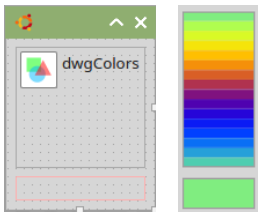

Abbildung 12.2.8.2.3: Formular mit DrawingArea und Panel (IDE und Laufzeit)

Der Quelltext für die Klasse FMinimalColorChooser wird vollständig angegeben:

```
[1]  ' Gambas class file
[2]
[3]  Property NumberOfColors As Integer    '' Anzahl der Farben,  Intervall \[1..12\] (lesen + schreiben)
[4]  Property Read ValueOfColor As Integer '' Farb-Wert (Nur lesen)
[5]
[6]  Private $iNumberOfColors As Integer
[7]  Private $iValueOfColor As Integer
[8]
[9]  Event Change
[10]
[11] Public Sub _new(Optional NumberOfColors As Integer = 6)
[12]   If Not IsMissing(NumberOfColors) Then $iNumberOfColors = NumberOfColors
[13] ' $iValueOfColor = GetColor(0) ' Option: Start-Farbe festlegen
[14] End
[15]
[16] Public Sub dwgColors_Draw()
[17]
[18]   Dim iY As Integer
[19]
[20]   Paint.LineWidth = dwgColors.H / $iNumberOfColors ' Relative Weite - bezogen auf DrawingArea
[21]   Paint.Translate(0, Paint.LineWidth / 2)
[22]   For iY = 0 To $iNumberOfColors - 1
[23]     Paint.Background = GetColor(iY)
[24]     Paint.MoveTo(0, iY * Paint.LineWidth)
[25]     Paint.LineTo(dwgColors.W, iY * Paint.LineWidth) ' Relative Höhe - bezogen auf DrawingArea
[26]     Paint.Stroke()
[27]   Next
[28]
[29]   $iValueOfColor = GetColor(0)
[30]   UpdatePreviewColor()
[31]
[32] End
[33]
[34] Private Function GetColor(Ind As Integer) As Integer ' Farb-Palette erzeugen
[35]
[36]   Dim fR, fG, fB As Float
[37]   Dim fFreq As Float = Pi(2) / $iNumberOfColors
[38]
[39]   fR = Sin(fFreq * Ind) * 127 + 128
[40]   fG = Sin(fFreq * Ind + Pi(1 / 3)) * 127 + 128
[41]   fB = Sin(fFreq * Ind + Pi) * 127 + 128
[42]
[43]   Return Color.RGB(fR, fG, fB)
[44]
[45] End
[46]
[47] Private Sub UpdatePreviewColor()
[48]   panShowColor.Background = $iValueOfColor
[49] End
[50]
[51] Public Sub dwgColors_MouseMove()
[52]   dwgColors_MouseUp()
[53] End
[54]
[55] Public Sub dwgColors_MouseUp()
[56]   $iValueOfColor = GetColor(Mouse.Y * $iNumberOfColors / dwgColors.H)
[57]   UpdatePreviewColor()
[58]   Raise Change ' Das Ereignis Change wird ausgelöst
[59] End
[60]
[61] Private Function NumberOfColors_Read() As Integer
[62]   Return $iNumberOfColors
[63] End
[64]
[65] Private Sub NumberOfColors_Write(Colors As Integer)
[66]   If Colors < 1 Or colors > 12 Then Error.Raise(Subst$(("Invalid number: &1"), Colors))
[67]   $iNumberOfColors = Colors
[68]   dwgColors.Refresh() ' Farb-Palette wird neu gezeichnet
[69] End
[70]
[71] Private Function ValueOfColor_Read() As Integer
[72]   Return $iValueOfColor
[73] End
```

Kommentar:

- Zuerst werden die zwei Eigenschaften NumberOfColors und ValueOfColors in den Zeilen 3 und 4 deklariert. Der Zusatz von 'Read' ist notwendig, wenn die Eigenschaft ValueOfColors nur gelesen werden soll.

- Im Konstruktor _NEW(...) wird als optionales Argument mit der Zahl 8 ein Startwert für die Anzahl der Farben in der Farb-Palette angegeben und es kommt die Funktion IsMissing() zum Einsatz → http://gambaswiki.org/wiki/lang/ismissing.
- In den Zeilen 16 bis 32 wird die Farb-Palette gezeichnet. Die Anzahl der Farben wird durch den aktuellen Wert der lokalen Variablen i$NumberOfColors bestimmt.
- Die Zeilen 51 bis 59 sorgen dafür, dass sowohl mit einem Klick auf das eingebettete Steuer-Element als auch mit der Bewegung der Maus bei gedrückter linker Maustaste über der Farb-Palette eine Farbe ausgewählt werden kann.
- Interessant ist die Fehlerbehandlung in der Zeile 66 innerhalb der Prozedur für das Festlegen des Wertes der Eigenschaft NumberOfColor in den Zeilen 65 bis 69. Es wird ein Fehler ausgelöst, wenn der Wert nicht im Intervall von 1 bis 255 liegt. Im Projekt wird durch die Vorgaben für den minimalen und maximalen Wert der SpinBox präventiv der geforderte Bereich mit Sicherheit eingehalten.

12.2.8.3 Ausblick Strategie 2

So richtig interessant wird es dann, wenn Sie die Klasse FMinimalColorChooser in ein Steuer-Element legen, dass auch in die Sammlung der Steuer-Elemente (Werkzeugsammlung) mit eigenem Symbol integriert werden kann:

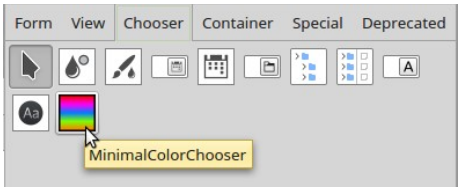

Abbildung 12.2.8.3.1: Werkzeugsammlung (Chooser)

Erst im Kapitel 30 werden Komponenten, Klassen, Module und Steuer-Elemente unter dem Aspekt der objekt-orientierten Programmierung genauer beschrieben. Damit Sie sich einen Einblick in die Strategie zur Entwicklung und Implementation selbst entwickelter Steuer-Elemente für die Werkzeug-Sammlung in der Gambas-IDE verschaffen können, wird Ihnen im Download-Bereich ein geeignetes Projekt von Tobias Boege vorgestellt.

Beachten Sie: Um das Steuer-Element MinimalColorChooser in die Werkzeugsammlung zu integrieren, ist das Verzeichnis MinimalColorChooser in das Projekt-Verzeichnis zu kopieren, welches das Steuer-Element MinimalColorChooser nutzen soll.

Der Quelltext für die entscheidende Klasse MinimalColorChooser sieht so aus:

```
' Gambas class file

Inherits UserControl
Export

Event Change

Public Const _Properties As String = "*,NumberOfColors=8"
Public Const _Group As String = "Chooser"
Public Const _Similar As String = "ColorChooser"
Public Const _DefaultEvent As String = "Change"

Property NumberOfColors As Integer
Property Read ValueOfColor As Integer

Private $hMCChooser As FMinimalColorChooser

Public Sub _new()
  $hMCChooser = New FMinimalColorChooser(Me, 6)
End

Public Sub Raise_ColorChange()
  Raise Change ' To be called from the FMinimalColorChooser
End

Private Function NumberOfColors_Read() As Integer
  Return $hMCChooser.NumberOfColors
End
```

```
Private Sub NumberOfColors_Write(Value As Integer)
  $hMCChooser.NumberOfColors = Value
End

Private Function ValueOfColor_Read() As Integer
  Return $hMCChooser.ValueOfColor
End
```