5.8.2 Projekte

In diesem Kapitel werden Ihnen drei Projekte vorgestellt, die auf unterschiedliche Art die Verwendung von Optionen und Argumenten für Gambas-Programme demonstrieren.

Eine weitere interessante Variante wird Ihnen im 'Kapitel 2.6 Weitergabe von Gambas-Programmen – Installationspaket' in einem ausführlich dokumentierten Projekt vorgestellt.

5.8.2.1 Projekt 1 – Gambas-Programm mit 3 Argumenten

In diesem Projekt zur Farbauswahl in einem ColorChooser können Sie dem Programm optional genau 3 Farbwerte (RGB) übergeben, um einen individuellen Startwert für die anzuzeigende Farbe nach dem Programm-Start festzulegen.

- Fehlt ein Argument oder fehlen zwei oder alle 3 Argumente, so wird ein zulässiger Start-Farbwert festgelegt.
- Mit den Werten der 3 Argumente wird – wenn kein Fehler auftrat – ein Farbwert berechnet, der als Vorgabe-Farbe für die eingesetzte Komponente ColorChooser1 dient.
- Die drei Argumente werden in diesem Projekt *nicht* darauf geprüft, ob sie nur valide RGB-Werte [0..255] annehmen.

Der Quelltext des Gambas-Projekts 'ColorSelectA1' ist *analog* zu dem Skript aus dem Kapitel 5.8.0 aufgebaut.

```
' Gambas class file

Public Sub Form_Open()
  Dim sArgument As String
  Dim R, G, B As Integer

  FMain.Center
  FMain.Resizable = False

  FMain.Text = "Programm mit 3 Argumenten (optional)"

  If Application.Args.Count <> 4 Then
     Print "Fehler! Es fehlt mindestens 1 Argument."
     R = 220
     G = 20
     B = 180
  Else
     R = Val(Application.Args[1])
     G = Val(Application.Args[2])
     B = Val(Application.Args[3])
  Endif ' Application.Args.Count <> 4 ?

  ColorChooser1.SelectedColor = Color.RGB(R, G, B)

End ' Form_Open
```

Gambas stellt mit *Application.Args* ein Objekt mit *allen* Argumenten des Gambas-Programms zur Verfügung, dass Sie wie ein Array einsetzen können:

- Das erste Argument Application.Args[0] (≡ $0) ist immer der Name des gestarteten Gambas-Programms.
- Die Eigenschaft *Application.Args.Count* gibt die Anzahl aller Argumente zurück – nicht nur der in der Konsole direkt übergebenen! Daher wird auch oben mit *Application.Args.Count* <> *4* geprüft.

In diesem Punkt unterscheidet sich die Arbeit mit dem Array der Argumente in Gambas von der Ermittlung der Anzahl der Argumente für ein Bash-Skript in der Konsole!

Beim Aufruf in einer Konsole:

```
hans@linux:~$ gbx3 $HOME/ColorSelectA1 -- 225 110 60
```

startet der Gambas-Interpreter das Programm zur Farbauswahl mit der Vorgabe-Farbe *ocker*:

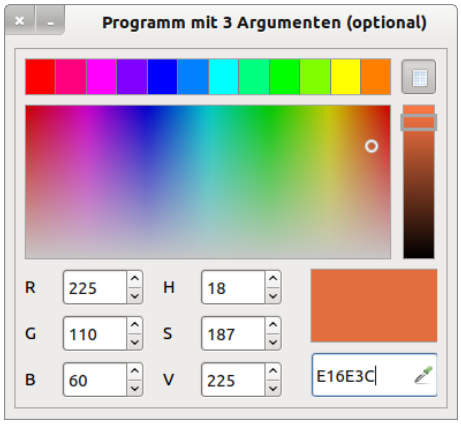

Abbildung 5.8.2.1.1: Programm zur Farbauswahl – Vorgabe-Farbe über 3 Argumente

Bei dem folgenden Aufruf erhalten Sie einen Hinweis und das Programm wird in diesem Fall mit einem Standard-Farbwert (→ Quelltext) gestartet:

```
hans@linux:~$ gbx3 $HOME/E/ColorSelectA1 -- 50 120
Hinweis! Es fehlt mindestens 1 Argument.
```

5.8.2.2 Projekt 2 – Gambas-Programm mit 3 Argumenten

Die Besonderheiten in diesem zweiten Projekt können so beschrieben werden:

- Es werden 2 Eigenschaften der Klasse Args der Komponente *gb* eingesetzt.
- Die dem Programm übergebenen Argumente werden nach unterschiedlichen Kriterien geprüft.
- Tritt ein Fehler ein, dann wird dieser Fehler nur mit einem Hinweis angezeigt und ein zulässiger Start-Farbwert gesetzt, um für den Programm-Start klar definierte Verhältnisse zu schaffen.
- Durch den genutzten Fehler-Vektor besteht die Möglichkeit, eine *differenzierte Fehleranalyse* vornehmen zu können.

Der Quelltext spiegelt die o.a. Besonderheiten wider:

```
' Gambas class file

Public aArgumente As String[]

Public Sub Form_Open()
  FColor.Center
  FColor.Resizable = False
  FColor.Text = "Programm mit 3 Argumenten (optional)"

' String-Array als Kopie des Arrays aller Argumente
  aArgumente = Args.All ' aArgumente[0] = ProgrammName

  If GetErrorVector(aArgumente).ToString(True) = "[0 0 0 0 0 0 0]" Then
     ColorChooser1.Value = Color.RGB(Val(aArgumente[1]), Val(aArgumente[2]), Val(aArgumente[3]))
  Else
     ErrorAnalysis(GetErrorVector(aArgumente))
  Endif ' Ohne Fehler ?

End ' Form_Open

Public Sub ColorChooser1_Change()
  Dim cColor As Integer

  cColor = ColorChooser1.SelectedColor ' Synonym für ColorChooser1.Value, DatenTyp: Integer

End ' ColorChooser1_Change()

Private Function GetErrorVector(aArgs As String[]) As Vector
  Dim vFA As Vector ' vFA → Fehler-Array
  Dim i As Integer = 1

  vFA = New Vector(7, False)
  vFA = [0, 0, 0, 0, 0, 0, 0]
```

```
' (1) Ermittlung Anzahl der Argumente
  If aArgs.Count = 1 Then vFA = [3, 0, 0, 0, 0, 0, 0]
  If aArgs.Count = 2 Then vFA = [2, 0, 0, 0, 0, 0, 0]
  If aArgs.Count = 3 Then vFA = [1, 0, 0, 0, 0, 0, 0]

  If aArgs.Count = 4 Then
     For i = 1 To aArgs.Max
       If Not IsInteger(aArgs[i]) Then ' (2) Prüfung, ob Integer-Zahl
          vFA[i] = 1
       Else
          If Val(aArgs[i]) < 0 Or Val(aArgs[i]) > 255 Then ' (3) Prüfung für das Intervall [0..255]
             vFA[i + 3] = 1
          Endif
       Endif
     Next
  Endif

  Return vFA ' Rückgabe Fehler-Vektor

End ' Function FehlerVektor(..)

Private Sub ErrorAnalysis(vVektor As Vector)
  Dim i, j As Integer

  Print "FEHLER!"
' Differenzierte Auswertung Fehler-Vektor: Element A0 (Anzahl der fehlenden Argumente)
  If vVektor[0] <> 0 Then
     If vVektor[0] = 1 Then
        Print "Es fehlt genau "; vVektor[0]; " Argument!"
        Inc j
     Else
       Print "Es fehlen genau "; vVektor[0]; " Argumente!"
       Inc j
     Endif
  Endif
' Auswertung Fehler-Vektor: A1-A3 (Welches Argument lässt sich nicht in Integer-Zahl konvertieren?)
  For i = 1 To 3
    If vVektor[i] <> 0 Then
       Print "Das "; i; ". Argument ist keine Integer-Zahl."
       Inc j
    Endif
  Next
' Auswertung Fehler-Vektor: A4-A6 (Welches Argument liegt nicht im Intervall [0..255]?)
  For i = 4 To 6
    If vVektor[i] <> 0 Then
       Print "Das "; i - 3; ". Argument liegt nicht im Intervall [0..255]!"
       Inc j
    Endif
  Next
  Print "Deshalb werden 3 Standard-Farbwerte verwendet."
  ColorChooser1.Value = Color.RGB(220, 20, 180) ' Start-Argumente - Farbe: pink

End ' ErrorAnalysis(vVektor As Vector)
```

Hier sehen Sie fünf Aufrufe des Programms zur Farbauswahl in der Konsole und die entsprechenden Programmreaktionen in Abhängigkeit von der Anzahl und dem Typ der übergebenen Argumente:

```
hans@linux:~$ gbx3 $HOME/E/ColorSelectA2 -- 222 133
Hinweis!
Es fehlt genau 1 Argument!
Deshalb werden 3 Standard-Farbwerte verwendet.
hans@linux:~$
```

```
hans@linux:~$ gbx3 $HOME/E/ColorSelectA2 -- 22O, 344 1E2
Hinweis!
Das 1. Argument ist keine Integer-Zahl.
Das 3. Argument ist keine Integer-Zahl.
Das 2. Argument liegt nicht im Intervall [0..255]!
Deshalb werden 3 Standard-Farbwerte verwendet.
hans@linux:~$
```

```
hans@linux:~$ gbx3 $HOME/E/ColorSelectA2 -- "50 120 30" # Es wird genau 1 Zeichenkette übergeben
Hinweis!
Es fehlen genau 2 Argumente!
Deshalb werden 3 Standard-Farbwerte verwendet.
hans@linux:~$
```

Bei diesen beiden Programm-Starts bestimmen die Argumente die gleiche Vorgabe-Farbe:

```
hans@linux:~$ gbx3 $HOME/E/ColorSelectA2 -- "50" "120" "30"
hans@linux:~$ gbx3 $HOME/E/ColorSelectA2 -- 50 120 30
```

5.8.2.3 Projekt 3 – Gambas-Programm mit Option und Argumenten

Das Projekt 3 (→ Kapitel 24.8.1 Projekte zur Demonstration der Komponente gb.map) ermöglicht es Ihnen Karten von *openstreetmap.org* (OSM) anzuzeigen. Sie benötigen für das Anzeigen einer Karte

- den Wert für die geografische Koordinate Breite (Latitude) des Start-Punktes,
- den Wert für die geografische Koordinate Länge (Longitude) des Start-Punktes,
- den Zoom-Wert aus dem Intervall [1..18] und
- eine aktive Internetverbindung, um die (vor-)berechneten Karten vom OSM-Server abzurufen.

Es macht in Abhängigkeit vom verfolgten Zweck durchaus Sinn

- über eine *Option* festzulegen, ob der geografischen Start-Punkt (oder auch ein anderer, besonderer Punkt) in der Karte mit einem Symbol gekennzeichnet wird und
- die *Koordinaten* für den geografischen Start-Punkt in der Karte sowie den Zoom-Faktor interaktiv an das Gambas-Programm als Argument zu übergeben:

```
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52,7904 11,7533 15
hans@linux:~$ gbx3 $HOME/E/OSMapA -- -s n -- 52.7904 11.7533 15
```

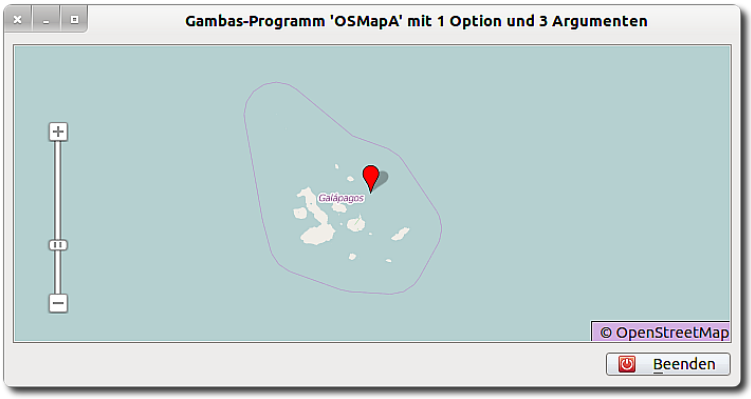

Abbildung 5.8.2.3.1: Galapagos (-- --shape j -- 0.01 -90.002 7)

Das Projekt-Archiv finden Sie im Download-Bereich. Wenn Sie das Archiv in den Pfad $HOME/E entpacken, dann können Sie das Programm sofort mit diesen Aufrufen in einer Konsole testen:

```
hans@linux:~$ gbx3 $HOME/E/OSMapA
hans@linux:~$ gbx3 $HOME/E/OSMapA -- -h
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --help
hans@linux:~$ gbx3 $HOME/E/OSMapA -- -v
hans@linux:~$ gbx3 $HOME/E/OSMapA -- -V
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --version

hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52,7904 1,0 6
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape n -- 52.7904 1.0 6
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 0 0 3

hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- -30,0 -50,9 3
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape y -- 52.7904 1.0 6

hans@linux:~$ gbx3 $HOME/E/OSMapA -- -- 52.7904 1.0 6
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape n
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape -- 52.7904 1.0 6

hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 0.01 -90.002 7

hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52.7904 331.0 6
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52.7904 12

hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52.7904 1.0 1o
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52.7904 1.0 19
hans@linux:~$ gbx3 $HOME/E/OSMapA -- --shape j -- 52.7904 1.0 11.5

hans@linux:~$ gbx3 $HOME/E/OSMapA -- -30,0 -50,9 4
hans@linux:~$ gbx3 $HOME/E/OSMapA -- 30,0 50,9 4
```

Im Projekt-Quelltext sehen Sie, dass die Fehlerfunktionen in der Fehlerbehandlung numerische Werte als Funktionswert liefern, wie es der folgende Quelltext-Ausschnitt zeigt:

```
Private Function IsDecimalDegree(sDegree As String, sFlag As String) As Integer
  If Upper(sFlag) Not Like "{LAT,LON}" Then Return 1

  If Upper(sFlag) = "LAT" Then
     If IsFloat(sDegree) Then
        If (Val(sDegree) <= -90) Or (Val(sDegree) > 90) Then
           Return 2
        Else
           Return 0
        Endif
     Else
        Return 3
     Endif
  Endif

  If Upper(sFlag) = "LON" Then
     If IsFloat(sDegree) Then
        If (Val(sDegree) <= -180) Or (Val(sDegree) > 180) Then
           Return 4
        Else
           Return 0
        Endif
     Else
        Return 5
     Endif
  Endif

End ' Function IsDecimalDegree(..)
```

Der Wert 0 bedeutet, dass kein Fehler auftrat. Mit den Funktionswerten, die größer als 0 sind, könnten Sie die Fehler differenziert ermitteln und anzeigen. Diese Möglichkeit wurde im Projekt nicht umgesetzt. Eine andere Vorgehensweise würde darin bestehen, einen boolschen Wert als Funktionswert zurückzugeben, um so *generell* einen Fehler zu ermitteln, auszuwerten und angemessen darauf zu reagieren.