17.1 ProgressBar – Fortschrittsbalken

Als visuelle Anzeigen für den Status einer Aufgabe, die von einem Programm ausgeführt wird, werden neben unterschiedlichen Mauszeigern oft Throbber eingesetzt. Throbber sind kleine animierte Grafiken in unterschiedlichen Größen und Ausführungen. Eine bekannte Throbber-Grafik ist das drehende Rad (spinning wheel). Throbber kennzeichnen den Beginn und das Ende einer aktuellen Operation (Beispiele: Download, Upload oder Dekompression einer Datei, Wartezeit).

Abbildung 17.1.1: Vier Throbber

Oft interessieren nicht nur der *Status*, sondern auch Informationen zum *Fortschritt* einer Aufgabe. Genau für diesen Einsatzfall werden *Fortschrittsbalken* verwendet.

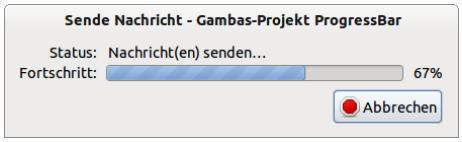

Abbildung 17.1.1: Status-Anzeige und Fortschrittsbalken beim EMail-Client Thunderbird

Mit der Komponente ProgressBar (gb.qt4) stellt Gambas einen Fortschrittsbalken zur Verfügung. In einer ProgressBar kann der Fortschritt nicht nur optisch über die Länge des angezeigten Balken signalisiert werden, sondern auch optional Text (Angabe in Prozent) auf dem Balken angezeigt werden.

Für eine ProgressBar haben sich drei typische Einsatzfälle bewährt:

- (Relative) Anzeige, wie lange ein Vorgang noch andauert
- (Relative) Anzeige, wie weit ein Vorgang bereits abgeschlossen ist
- Alternierende Anzeige als Statusanzeige: Vorgang nicht abgeschlossen

17.1.1 Eigenschaften und Methoden ProgressBar

Diese Klasse ProgressBar (gb.qt4) implementiert einen Fortschrittsbalken.

Es gibt für eine ProgressBar nur 2 ausgewählte Eigenschaften und eine spezielle Methode:

- Eigenschaft *Label* As Boolean
  → Zeigt an, ob ein Prozentwert angezeigt wird oder nicht.
- Eigenschaft Property *Value* Als Float
  → Ermittelt oder legt den Wert der ProgressBar (relative Länge des angezeigten Fortschrittsbalken fest . Dieser Wert muss zwischen 0 und 1 liegen.
- Methode Reset
  → Die Methode *ProgressBar.Reset()* setzt den Wert auf 0 zurück.

Hinweise:

- Wenn Sie den Wert der Progressbar gesetzt oder geändert haben, dann sollten Sie ein *Wait* folgen lassen, damit die Änderung *sofort* sichtbar wird.
- Da die ProgressBar als Eingabewerte nur Zahlen (Datentyp Float) zwischen 0 und 1 akzeptiert, müssen Sie die zu verarbeitenden Werte auf dieses Einheitsintervall [0|1] transformieren. Eine *Transformationsfunktion* leistet dann gute Dienste.
- Obgleich der Eigenschaftsname *Label* anderes vermuten lässt, können Sie keinen *eigenen* Text auf der Progressbar anbringen. Der Datentyp von Label ist Boolean!

17.1.2 Projekt 1 – Progressbar als Throbber

Im vorgestellten Projekt *ThrobberPB* werden bekannte Throbber eingesetzt und eine Progressbar als Throbber konfiguriert. Außerdem wird gezeigt, wie Sie Zahlenwerte in einer Progressbar (statisch und relativ) anzeigen können – sie visualisieren. In diesem Zusammenhang wird eine Funktion vorgestellt, mit der Sie Zahlen aus einem Intervall $[x_0|x_1]$ linear auf ein Intervall $[y_0|y_1]$ abbilden können.

```
Public Function Transform(fValue As Float, x0 As Float, x1 As Float, y0 As Float, y1 As Float) As Float
' Lineare Abbildung: Intervall [x0|x1] ---> [y0|y1]
' 2-Punkte-Gleichung mit P1(x0|y0) und P2(x1|y2) im Definitionsbereich [x0|x1]
' (y-y0)*(x1-x0) = (y1-y0)*(x-x0)
' y = ((y1-y0)/(x1-x0)*(x-x0)) + y0 → y=g(x)=m*x+n

  If Abs(x1 - x0) > 0.0001 Then
     Return (((y1 - y0) / (x1 - x0)) * (fValue - x0)) + y0
  Else
     Message.Error("FEHLER!")
  Endif ' Abs(x1 - x0) > 0.0001 ?

End ' Function  Transform(..)
```

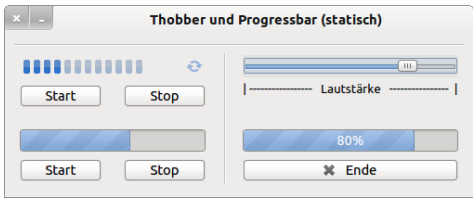

Abbildung 17.1.2.1: Drei Throbber und Relativ-Anzeige Zahlenintervall [0|60] → [0|1]

Der Quelltextausschnitt zeigt die wesentlichen Prozeduren:

```
[1]  ' Gambas class file
[2]
[3]  Private bSuccess As Boolean = False
[4]
[5]  Public Sub _new()
[6]    Slider1.MinValue = 0
[7]    Slider1.MaxValue = 60
[8]    Slider1.Value = 20
[9]    ...
[10] End ' _new()
[11]
[12] Public Function Transformation(...)
[13] ...
[14] Public Sub Slider1_Change()
[15]   Progressbar1.Value = Transformation(Slider1.Value, Slider1.MinValue, Slider1.MaxValue, 0, 1)
[16]   Wait
[17] End ' hSlider1_Change
[18]
[19] Public Sub btnWaitOutStart_Click()
[20]   WaitOut(0.05)
[21]  End ' btnWaitOutStart_Click()
[22]
[23] Public Sub btnWaitOutStop_Click()
[24]   bSuccess = True
[25] End ' btnWaitOutStop_Click()
[26]
[27] Public Sub WaitOut(iTime As Float)
[28]   Dim iCount As Integer
[29]
[30]   If iTime < 0.01 Or iTime > 0.2 Then iTime = 0.05
[31]   pbWait.Visible = True
[32]   pbWait.Label = False
[33]   Repeat
[34]      For iCount = 0 To 10
[35]        pbWait.Value = iCount / 10
[36]        Wait iTime
[37]      Next
[38]      For iCount = 0 To 10
[39]        pbWait.Value = 1 - (iCount / 10)
[40]        Wait iTime
[41]      Next
[42]   Until bSuccess = True
[43]   bSuccess = False
[44]
[45] End ' WaitOut(iTime As Integer)
[46]
[47] Public Sub btnStart_Click()
[48]  movieboxWaitOut.Path = "WaitOutIcons/time.bar.a.gif"
[49]  movieboxWaitOut.Playing = True ' Animierte Grafik 1 zeigen
[50]  movieboxWaitOut2.Path = "WaitOutIcons/arrows_a.gif"
[51]  movieboxWaitOut2.Playing = True ' Animierte  Grafik 2 zeigen
[52] End ' btnStart_Click()
[53]
[54] Public Sub btnStop_Click()
[55]  movieboxWaitOut.Playing = False
[56]  movieboxWaitOut.Path = "WaitOutIcons/time.bar.0.gif"
```

```
[57]  movieboxWaitOut.Playing = True ' Standbild 1 zeigen
[58]  movieboxWaitOut2.Playing = False
[59]  movieboxWaitOut2.Path = "WaitOutIcons/arrows_0.gif"
[60]  movieboxWaitOut2.Playing = True ' Standbild 2 zeigen
[61] End ' btnStop_Click()
[62]
[63] Public Sub btnClose_Click()
[64]   FMain.Close
[65] End ' btnClose_Click()
```

Wenn Sie das Programm starten, dann werden Sie sicher auch Möglichkeiten erkennen, diese und ähnliche Throbber sowie die Relativ-Anzeige von Zahlenintervallen in eigenen Programmen einzusetzen.

17.1.3 Projekt 2 – Progressbar als Fortschrittsanzeige bei einem Datei-Download

In diesem Projekt besteht die Herausforderung darin, die Gesamtgröße der Download-Datei vor dem Download zu bestimmen als auch aktuelle Dateigröße der Download-Datei lokal im Download-Ordner während des Downloads. Deshalb wird mit *wget* ein externes (Konsolen-)Programm in einem Prozess eingesetzt, so dass man die Ausgaben des Programms erfassen, filtern und auswerten kann. Als Besonderheit zeigte sich, dass das Programm *wget* alle Ausgaben von der Standardausgabe (intern) auf die Standard-Fehlerausgabe umlenkt! Im Projekt wurden die Hinweise – vor allem aus dem Kapitel 21.3.3 Projekt – Prozess-Steuerung und Prozess-Daten – erfolgreich umgesetzt. Deshalb wird Ihnen nur der Quelltext der zwei wichtigsten Prozeduren vorgestellt.

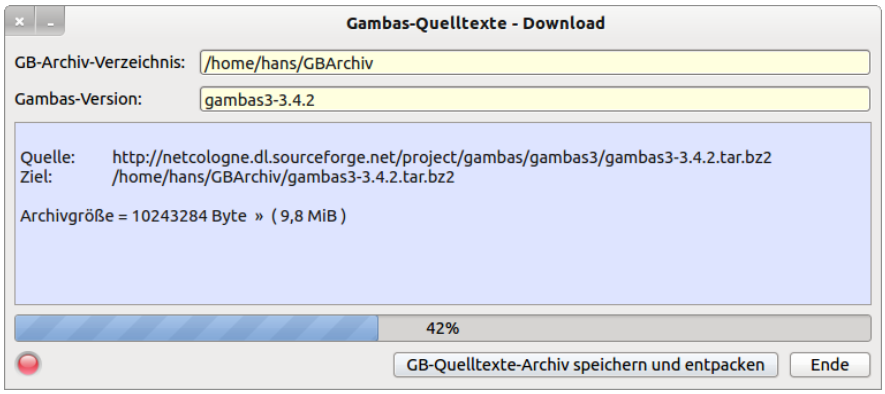

Abbildung 17.1.3.1: Der Download ist aktiv

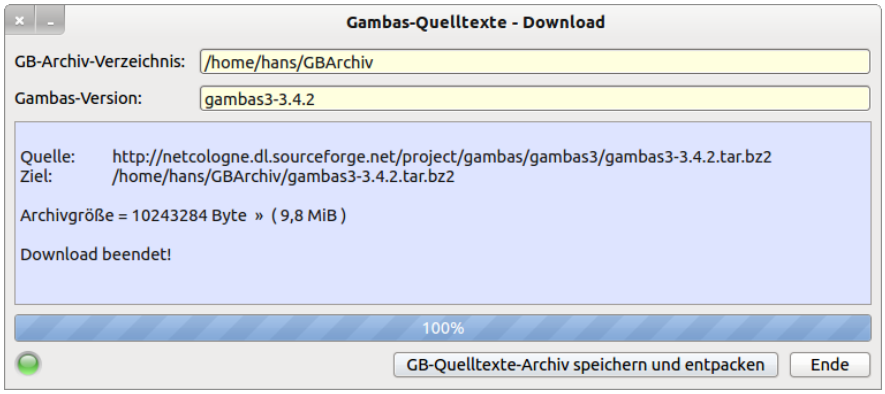

Abbildung 17.1.3.2: Der Download ist beendet

```
[1]  Public Sub Timer1_Timer()
[2]    Dim iCurrentFileSize As Integer
[3]
[4]    iCurrentFileSize = Stat($sArchivPath).Size ' Aktuelle Dateigröße beim Download
[5]    ProgressBar1.Value = iCurrentFileSize / $iArchivSize ' Wert im Intervall [0-1]
[6]    Wait
[7]  End ' Timer1_Timer()
[8]
[9]  Public Sub myWGETProcess_Error(sOutput As String)
[10] ' Alle Ausgaben kommen über die Standard-Fehler-Ausgabe
[11]   Dim aMatrix, aListe As String[]
```

```
[12]   Dim sElement As String
[13]   Dim iSizeMB As Float
[14]
[15]   aMatrix = Split(sOutput, gb.newline)
[16]   For Each sElement In aMatrix
[17]       If sElement Begins "Länge:" Or sElement Begins "Length:" Then
[18]          aListe = Split(sElement, " ")
[19]          $iArchivSize = CInteger(aListe[1]) ' 0,9765625 (Byte), 0,953674316 (MB)
[20]          iSizeMB = Round(($iArchivSize / 1000000) * 0.953674316, -1)
[21]          txaContent.Insert(gb.NewLine)
[22]          txaContent.Insert("Archiv = " & Str($iArchivSize) & " Byte  »  ( " & Str(iSizeMB) & " MiB )")
[23]          txaContent.Insert(gb.NewLine & gb.NewLine)
[24]          Timer1.Start
[25]       Endif ' sElement Begins "Länge:" ?
[26]       If sElement Ends "200 OK" Then
[27]          SetLEDColor("red")
[28]       Endif ' Ends "200 OK" ?
[29]       If sElement Like "* Server existiert." Or sElement Like "*Remote file exists*" Then
[30]          SetLEDColor("green")
[31]          txaContent.Insert("Dateigrößenübermittlung abgeschlossen!" & gb.NewLine)
[32]       Endif ' LIKE ... ?
[33]       If sElement Like "*gespeichert*" Or sElement Like "*saved*" Then
[34]          SetLEDColor("green")
[35]          txaContent.Insert("Download beendet!" & gb.NewLine)
[36]          Wait 2
[37]          Timer1.Stop
[38]          ProgressBar1.Value = 1
[39]          Wait
[40]          $bFlag = True
[41]       Endif ' LIKE ... ?
[42]
[43]   Next ' sElement
[44] End ' myWGETProcess_Error(..)
```

Das Entpacken der Archiv-Datei wird mit einem Throbber signalisiert. Die Throbber-Grafik *LevelBar* zeigt sich nur während des Entpackens in einer MovieBox:

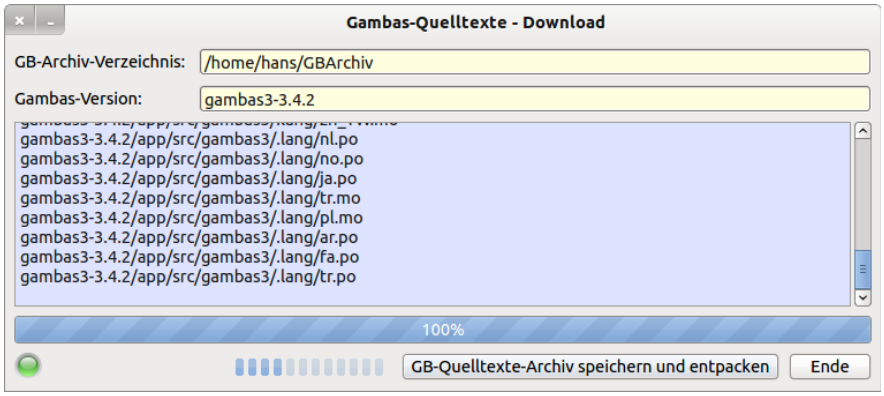

Abbildung 17.1.3.3: Das Download-Archiv wird entpackt – LevelBar als Throbber

Das Projekt 3 finden Sie nur im Downloadbereich. Es nutzt einerseits eine ProgressBar zur Anzeige, wie viel Dateien noch zu bearbeiten sind (*Time-remaining Progress Indicator*) und andererseits einen Fortschrittsbalken, um den Fortschritt bei einem Datei-Download von k Dateien anzuzeigen:

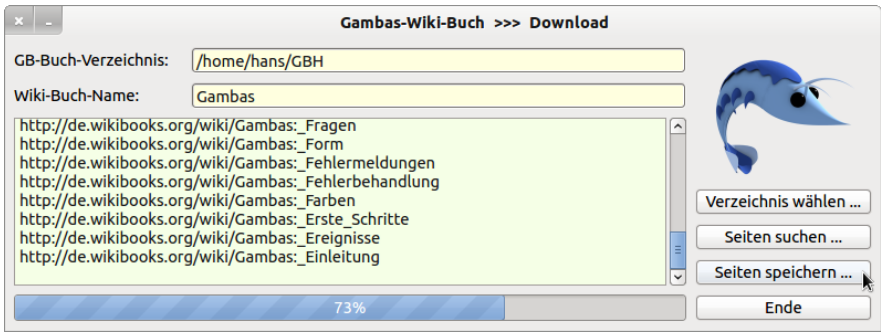

Abbildung 17.1.3.4: Download-Fortschritt