12.2.10 Exotische Programm-Fenster

In diesem Kapitel werden Ihnen einige Programmoberflächen gezeigt, die zum Teil erheblich von der rechteckigen Form der gewöhnlichen Programmfenster abweichen. Es sind

- Programmfenster mit eingebetteten Programmfenstern,
- Programmfenster mit externen Popup-Fenstern,
- transparente Programmfenster,
- Programmfenster ohne Rand und Fensterleiste,
- Programmfenster, die fast nie zu sehen sind,
- runde Programmfenster,
- Programmfenster mit wechselnden Hintergründen,
- rechteckige Programmfenster - jedoch mit abgerundeten Ecken und
- Programmfenster mit ungewöhnlichen, freien Formen.

12.2.10.1 Beispiel 1

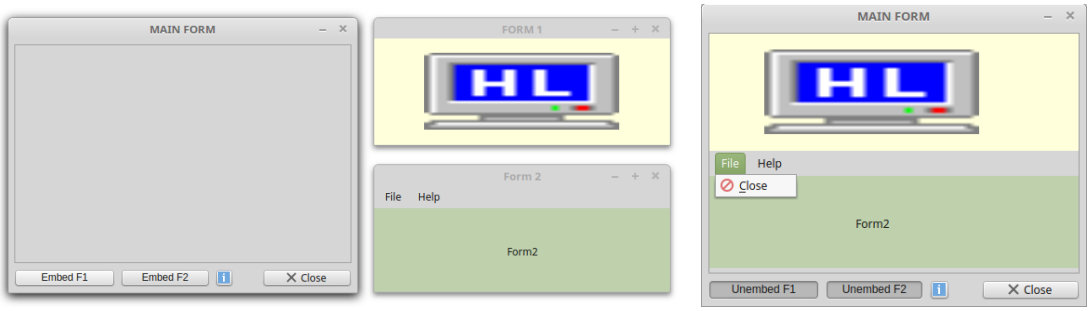

Abbildung 12.2.10.1.1: Zwei Fenster in ein (Haupt-)Fenster einbetten

Im Beispiel 1 existieren 3 Formulare – Main.Form, Form1 und Form2. Es werden zuerst alle drei Fenster geöffnet. Wechselseitig können Form1 und Form2 in das Hauptfenster eingebettet oder freigestellt werden. Durch die Eigenschaft *Application.MainWindow* = *FMain* werden alle Fenster geschlossen, wenn das Haupt-Fenster geschlossen wird.

```
Public Sub Form_Open()

  Application.MainWindow = FMain

  FMain.Resizable = False
  FMain.TopOnly = True
  FMain.Text = "MAIN FORM"

' Positioning and opening of all 3 windows
  Form1.Move(FMain.X + FMain.W + 20, FMain.Y)
  Form1.Show()
  Form2.Move(FMain.X + FMain.W + 20, FMain.Y + Form1.H + 48)
  Form2.Show()

' Vertical placement of child elements in the embedding panel
  panContainer.Arrangement = Arrange.Vertical

' FInsert or remove Form1 and Form2 – set button text
  btnEMUNForm1_Click()
  btnEMUNForm2_Click()
End

Public Sub btnEMUNForm1_Click()
  If btnEMUNForm1.Value Then
     Form1.Reparent(panContainer, 0, 0)
     btnEMUNForm1.Text = ("Unembed F1")
  Else
     Form1.Reparent(Null)
     btnEMUNForm1.Text = ("Embed F1")
  Endif
End

...
```

Das Projekt 2_UnEmbedForms finden Sie im Downloadbereich.

12.2.10.2 Beispiel 2

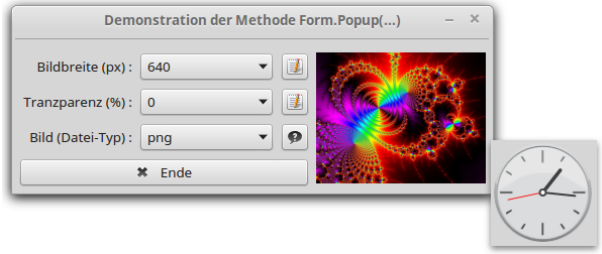

Abbildung 12.2.10.2.1: Popup-Fenster

Fenster als Popup-Fenster (ohne Fensterzeile) können Sie mit der folgenden Methode aufrufen:

```
ShowPopup( [ X As Integer, Y As Integer ] ) As Integer

Public Sub Form_DblClick()
  FClock.ShowPopup()
End
FMain.Transparent = True
FMain.Background = &HB0C3DDFF
```

Beachten Sie, dass die beiden Argumente X und Y optional sind. Fehlen beide Argumente, dann wird das Popup-Fenster an der Stelle im Hauptprogramm angezeigt, auf die mit der Maus geklickt wurde (linke, obere Fensterecke). Das Popup-Fenster muss bereits als Klasse im Projektordner existieren oder wird zur Laufzeit erzeugt. Das Projekt-Archiv und weitere Informationen finden Sie im Kapitel 12.2.2 Form – PopUp-Fenster.

12.2.10.3 Beispiele 3

Transparente Fenster Fenster lassen sich schnell erzeugen. Es reicht die erste Anweisung (A). Mit der Anweisung (B) spendieren Sie dem Fenster einen farbigen Hintergrund. Das erreichen Sie durch die Angabe eines passendes Alpha-Kanals wie B0.

```
(A)           FMain.Transparent = True
(B)           FMain.Background = &HB0C3DDFF
```

Ein volltranparentes Fenster erreichen Sie, wenn Sie die Eigenschaft FMain.Background nicht setzen oder den Alpha-Kanal auf den Wert FF oder 255(dez).

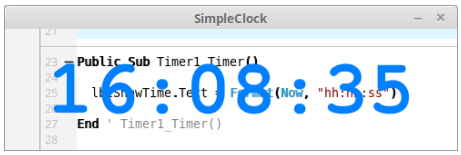

Abbildung 12.2.10.3.1: Volltransparentes Fenster

Für ein Fenster ohne Fensterzeile setzen Sie die Eigenschaft FMain.Border auf False. Vergessen Sie dann aber nicht, dem Fenster zum Beispiel ein Popup-Menü zu spendieren, um es schließen zu können.

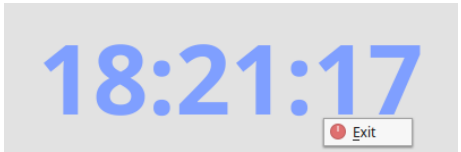

Abbildung 12.2.10.3.2: Fenster ohne Fensterzeile mit Kontext-Menü.

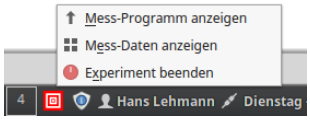

Abbildung 12.2.10.3.3: Tray-Icon-Menü

Es gibt Programme, die ihre Arbeit weitgehend im Verborgenen tun. Nur zu bestimmten Anlässen sollen Sie sich zeigen – dafür benötigen Sie ein Try-Icon-Menü. Viele Informationen und ein Projekt zu diesem Thema finden Sie im Kapitel 13.6 Kontextmenü – TrayIcon.

12.2.10.4 Beispiele 4

Für ein Projekt für eine analoge Uhr passt natürlich ein rundes Programmfenster, ein rechteckiges Programmfenster mit abgerundeten Ecken sieht gefällig aus und bei speziellen Programmen kann es auch ein Programmfenster mit ungewöhnlicher, freier Form oder ein Programmfenster mit wechselnden Hintergründen sein.

Abbildung 12.2.10.4.1: Rundes Programm-Fenster

Beim Uhrenprogramm mit der analogen Uhr kann der Benutzer über ein Kontext-Menü zwischen mehrenen Designs auswählen. Dafür wird die Eigenschaft FMain.Picture über die Methode SetDesign(…) geändert:

```
Public Sub SetDesign(i As Integer)
  iDesign = i

  FMain.Picture = Picture.Load("Images/Gambas_Clock_" & Format(iDesign, "0") & ".png")

  PaintClock(Now)
  Module_Settings.StoreClockSettings()

End

...
Public Sub mnuDesign1_Click()
  SetDesign(1)
End

...
Public Sub mnuDesign7_Click()
  SetDesign(7)
End
```

Mit weiteren Methoden werden anschließend die drei Zeiger und die Zeigerwelle der Uhr auf diesem Hintergrund gezeichnet:

Abbildung 12.2.10.4.2: Uhren-Designs (Auswahl)

Abbildung 12.2.10.4.3: Fenster mit abgerundeten Ecken

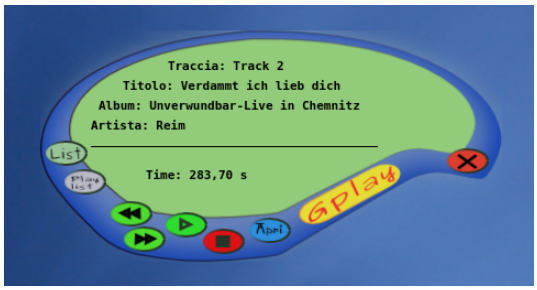

Abbildung 12.2.10.4.4: Freie Fenster-Form

Das folgende Fenster sieht schon sehr edel aus – oder? Das Projekt Gambas-Web-Radio hat der Autor Ingo Beckert entworfen und erprobt.

Abbildung 12.2.10.4.5: Fenster mit abgerundeten Ecken

So unterschiedlich die Programmoberflächen auch sind, hinter allen Varianten steckt das gleiche Vorgehen. Um Fenster wie die oben gezeigten zu erzeugen, benötigen Sie (mindestens) eine Bild-Datei mit dem Hintergrund für das Programm-Fenster und optional mehrere Bild-Dateien für die verschiedenen Buttons.

An einem Beispiel wird das oben skizzierte Vorgehen demonstriert, der relevante Quelltext angegeben und abschließend kommentiert:

```
[1]  Public hPicture As Picture
[2]
[3]  Public Sub Form_Open()
[4]
[5]  ...
[6]
[7]  ' Setting the size of the clock (and the Form)
[8]    FMain.W = 199
[9]    FMain.H = 199
[10]
[11] ' Setting the size of the picture used for drawing the hands
[12]   PictureBox1.W = FMain.W
[13]   PictureBox1.H = FMain.H
[14]
[15] '-------------------------------------------------------------------------------------
[16]   FMain.Border = False
[17]   FMain.Transparent = True
[18] ' Loading the clock face into the form
[19]   FMain.Picture = Picture.Load("Images/Gambas_Clock_" & Format(iDesign, "0") & ".png")
[20]   FMain.Mask = True
[21] '-------------------------------------------------------------------------------------
[22]
[23] ' Updating the time display at program start
[24]   Timer1.Trigger()
[25]
[26] End
```

Kommentar

- Zuerst setzen Sie in der Zeile 16 die Eigenschaft *FMain.Border* = *False*. Sie erhalten damit ein Fenster ohne Fensterzeile. Das macht es u.U. erforderlich ein Kontext-Menü zu entwerfen, über das Sie das Programm sicher beenden können.
- Die Transparenz des Programm-Fensters wird in der Zeile 17 festgelegt. Folglich sehen Sie

später nur noch das Hintergrund-Bild – auf dem gezeichnet werden kann – sowie weitere Elemente. Beachten Sie, dass das Programm-Fenster nach wie vor rechteckig ist – nur sehen Sie das nicht mehr!

- In der Zeile 19 erhält das Fenster ein Picture als Hintergrund.
- Setzen Sie die Eigenschaft Form.Picture, dann ist der Hintergrund des Fensters nicht mehr einfarbig, sondern stellt das zugewiesene Picture dar. Die Mask-Eigenschaft wird in der Zeile 20 gesetzt und geht einen Schritt weiter und wendet, wenn sie auf True geschaltet ist, auch den Alpha-Kanal des Bildes auf das Fenster an. Somit werden Regionen, die im zugewiesenen Picture transparent sind, aus dem Fenster "ausgeschnitten", gleichsam "maskiert" – so, als wäre das Picture eine Schablone. Über die Mask-Eigenschaft des Haupt-Fensters und einem geeigneten Picture lassen sich nicht-rechteckige Fensterformen realisieren oder Fenster mit Löchern.

Die folgenden Quelltext-Abschnitte zeigen das oben skizzierte Vorgehen in anderen Projekten:

```
' Autor:                   Sergio <fsurfing@tutto-opensource.org>
' Projekt:                 GPlay
' Beschreibung:            Lettore musicale opensource (Opensource-Musikplayer)

Public Sub Timer1_Timer()

  Dim x As Integer
  Dim y As Integer

  x = Mouse.ScreenX - FMain.X
  y = Mouse.ScreenY - FMain.Y

'-------------------------------------------------------------------------------------
  FMain.Border = False
  FMain.Transparent = True
  FMain.Picture = MDisegna.GUI(tag_titolo, tag_traccia, tag_album, tag_artista, x, y)
  FMain.Mask = True
'-------------------------------------------------------------------------------------

  If riproduci = True Then Play.Play()

End
```

```
' Autor:                   Ingo Beckert (Co-Autor Gambas-Buch | www.gambas-buch.de)
' Projekt:                 Gambas Web-Radio
' Beschreibung:            GUI für die Nutzung von Internetradio mit der Komponente gb.media.

Public Sub Form_Open()

...
'-------------------------------------------------------
  WebRadio.Border = False
  WebRadio.Transparent = True
  WebRadio.Picture = Picture["icons/Background2.png"]
  WebRadio.Mask = True

  Buttons_Load()
'-------------------------------------------------------
...

End
```

Ein interessantes Detail zu diesem Programm ist die Verwendung des gleichen Vorgehens auch beim (Dialog-)Fenster für das Einstellen der einzelnen Kanäle im Equalizer:

```
Public formEqualizer As Form
Private $x As Integer
Private $y As Integer

Event Close

Public Sub _new(x As Integer, y As Integer)
  $x = x
  $y = y
  FormInit()
End

Private Sub FormInit()

...
'-------------------------------------------------------------
  formEqualizer = New Form As "Equalizer"
  formEqualizer.Border = False
```

```
  formEqualizer.Transparent = True
  formEqualizer.Picture = Picture["icons/background_eq.png"]
  formEqualizer.Mask = True
  formEqualizer.Height = formEqualizer.Picture.H
  formEqualizer.Width = formEqualizer.Picture.W
'------------------------------------------------------------
...

End
```

Auf dem (originalen) Bild *background_eq.png* werden anschließend die einzelnen Schieberegler (interaktiv) gezeichnet:

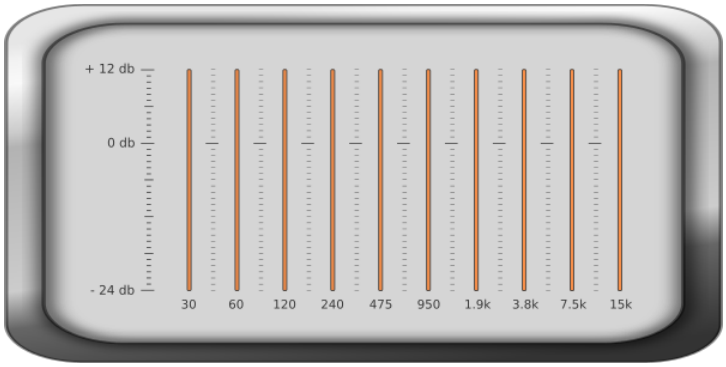

Abbildung 12.2.10.4.6: Equalizer (Original-Bild)

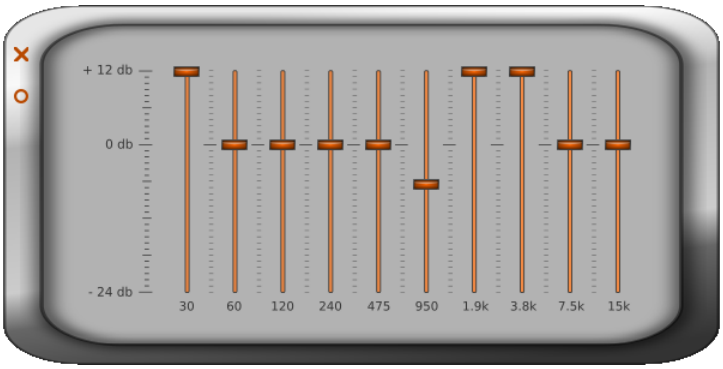

Abbildung 12.2.10.4.7: Equalizer mit Schiebereglern und 2 Steuerelementen (Bilder!)