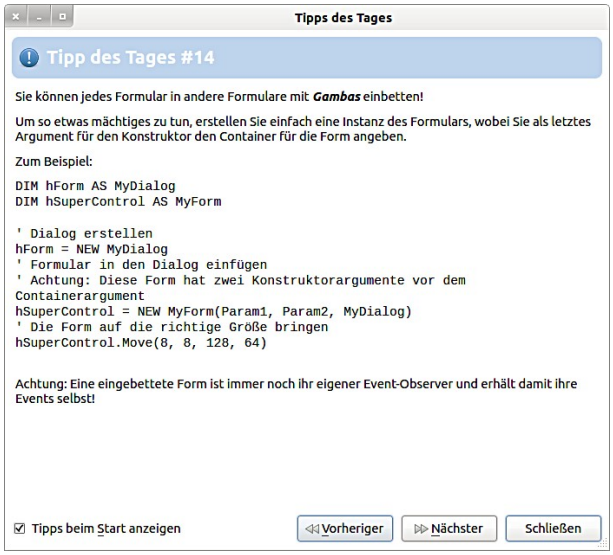
12.2.7 Form – Form in Form

Die Frage "Ist es möglich ein Formular in ein anderes Formular einzubetten (FormInForm), dessen Auswahl beliebig geändert werden kann?" lässt sich generell mit JA beantworten, wobei die Detailantworten unterschiedlich ausfallen:

1. Praktisch einfach zu verwendende Ansätze von 'Formulare in einem anderen Formular' (FormInForm) funktionieren über so genannte 'Multidocument Interfaces' wie den Steuer-Elementen TabStrip (gb.qt4) → Kapitel 12.6 und TabPanel (gb.form) → Kapitel 12.7 sowie Workspace (gb.form.mdi) → Kapitel 12.3.1 und werden in den angegebenen Kapiteln ausführlich beschrieben.
2. Im 'Tipp des Tages Nummer 14' in der Gambas-IDE finden Sie diese Antwort: *Sie können jedes Formular in andere Formulare mit Gambas einbetten! Um so etwas mächtiges zu tun, erstellen Sie einfach eine Instanz des Formulars, wobei Sie als letztes Argument für den Konstruktor den Container für die Form angeben. ... Achtung: Eine eingebettete Form ist immer noch ihr eigener Event-Observer und erhält damit ihre Events selbst!* Ergänzt wird dieser Tipp durch ein Beispiel mit passend gewähltem Quelltext:



3. Ja - Sie können ein Formular erzeugen und in einen beliebigen Container in einem anderen Formular einfügen (und u.U. daraus wieder entfernen) – das ist die flexibelste Lösung.
4. Ja - Sie können durch das Einfügen von Formularen in Formulare *eigene* Steuer-Elemente realisieren. Ein Steuerelement (Control) ist als eine Klasse aufzufassen, die sich in eine GUI integrieren lässt.

In diesem Kapitel werden Ihnen Projekte vorgestellt, in denen die Antworten 2 und 3 praktisch umgesetzt werden. Die Entwicklung eigener Steuer-Elemente wird im → Kapitel 12.2.8 beschrieben. Es werden auch die dazu gehörenden Projekte vorgestellt.

12.2.7.1 Projekt 1

In allen vorgestellten Projekten, auch jenen im Kapitel 12.2.8, wird das folgende Vorgehen umgesetzt:

- Zuerst erzeugen Sie neben dem (Haupt-)Formular des Projektes mindestens ein weiteres Formular – zum Beispiel FEmbed – das Sie in das Haupt-Formular einfügen wollen.
- Dann gestalten Sie das Formular FEmbed mit Gambas-Steuer-Elementen mit der vorgesehenen Funktionalität, die in der Klasse *FEmbed.class* beschrieben wird. Wenn es notwendig ist, können Sie auch ein reguläres Menü oder ein Kontext-Menü in das Formular FEmbed einfügen.
- Abschließend fügen Sie das Formular FEmbed in einen geeigneten *Container* im (Haupt-)Formular ein.

Um das Augenmerk auf das Einfügen eines Formulars in ein (Haupt-)Formular zu legen, fällt die Funktionalität des (Haupt-)Formulars spartanisch aus. Sie beschränkt sich auf das Sichtbarmachen der Hintergrund-Farbe des eingefügten Formulars über einen SwitchButton und einen Button, mit dem das

(Haupt-)Formular geschlossen werden kann. Das eingefügte Formular zeigt entweder einen Text in einer TextArea oder ein Bild in einer PictureBox an. Die Umschaltung erfolgt mit dem Button 'Ansicht wechseln' auf dem *eingefügten* Formular. Ein eingefügtes Formular besitzt keine Titelzeile mit dem Fenster-Titel und den üblichen Schaltflächen, da es kein Top-Level-Fenster ist. Daher verfügt das eingefügte Formular über ein Menü, um hier zu zeigen, dass das eingefügte Formular sein eigener Event-Observer ist und Sie deshalb seine Ereignisse – wie zum Beispiel _Close() – im Quelltext der Klasse selbst managen können.

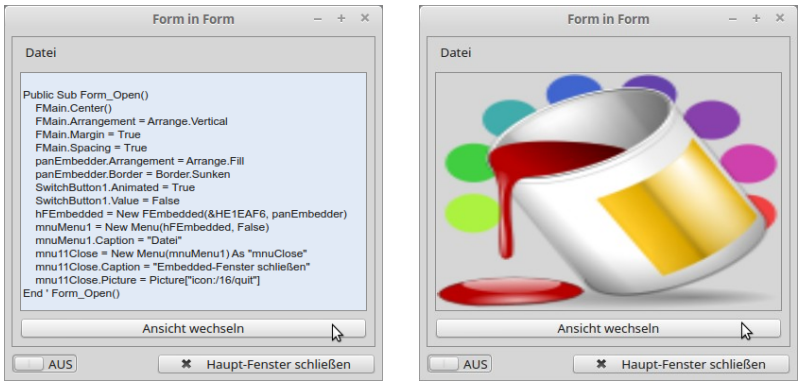


Abbildung 12.2.7.1.1: Haupt-Fenster mit eingefügtem Fenster in Aktion

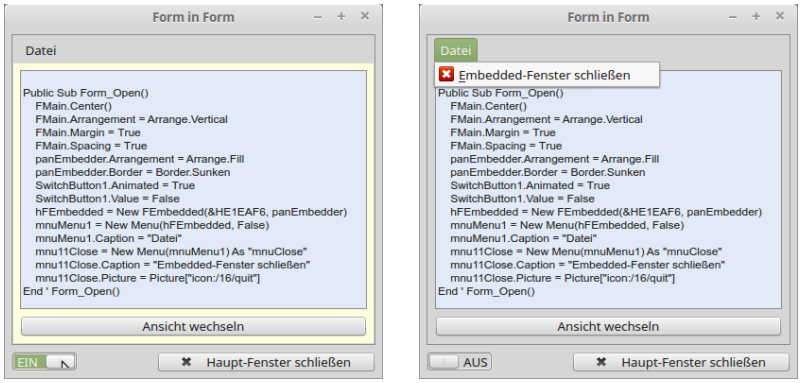


Abbildung 12.2.7.1.2: Haupt-Fenster mit markiertem Fenster (links)

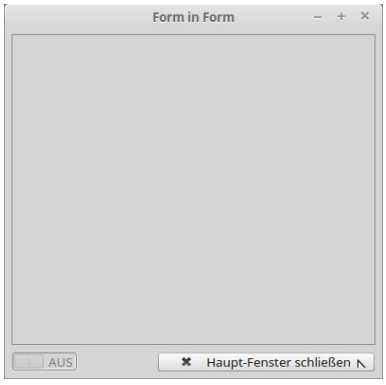


Abbildung 12.2.7.1.3: Haupt-Fenster mit umrandetem, aber leerem Container (Panel)

Im Zusammenhang mit der Entwicklung von Projekten, bei denen Formulare in ein (Haupt-)Formular eingefügt werden, konnten diese Erfahrungen gesammelt werden:

- Planen Sie diese Art von Projekten vor allem unter dem Aspekt, was das einzufügende Formular im Projekt leisten soll.
- Entwickeln und erproben Sie das einzufügende Formular zuvor in einem eigenständigen Projekt.

- Achten Sie darauf, dass Sie später in der Klasse des einzufügenden Formulars den Bezeichner für das Formular durch 'Me' ersetzen!
- Sehen Sie in diesen Test-Projekten die _new-Methode (Konstruktor) vor, ist diese dort aber erst einmal auskommentieren. Diese Prävention ist aber von Vorteil, wenn Sie dem einzufügenden Fenster notwendige Startwerte mitgeben wollen. Zu diesem Zweck können Sie den originalen Konstruktor in der Klasse einfach überschreiben und dort weitere Parameter deklarieren.
- Notieren Sie sich genau, auf welche Steuerelemente der Formulare u.U. wechselseitig zugegriffen werden soll. Bei diesen ausgewählten Steuerelementen müssen Sie die Public-Eigenschaft jeweils permanent oder temporär auf den Wert 'True' setzen.
- Binden Sie in Ihre Überlegungen auch ein Konzept zum Daten-Austausch zwischen dem einzufügenden Formular und dem (Haupt-)Formular ein, wenn das notwendig wird.

Aus dem Quelltext der Klasse für das eingefügte Formular FEmbedded ist nur dieser Ausschnitt für den Konstruktor interessant:

```
Public Sub _new(iColor As Integer)
  TextArea1.Background = iColor
End ' _new(...)
```

So wird das Formular *FEmbedded* in das Formular *FMain* eingefügt:

```
Private hFEmbedded As FEmbedded

Public Sub Form_Open()
  FMain.Center()
' Weitere Initialisierungen ...
  SwitchButton1.Value = False
  hFEmbedded = New FEmbedded(&HE1EAF6, panEmbedder)
End ' Form_Open()
```

Kommentar:

- Man kann zu einer Formular-Klasse – wie zu jeder anderen Klasse auch – einen Konstruktor in der _new(..)-Methode schreiben, der zusätzliche Formular-Parameter (obligatorische oder optionale) definiert. Bei einem Formular zur Passwort-Änderung eines Nutzer-Accounts könnte es zum Beispiel der Name des aktuellen Nutzers sein, der dem Hauptprogramm vorher schon bekannt war.
- Das Beispiel zeigt, dass Konstruktor-Parameter weitervererbt werden. Die Klasse FEmbedded erbt von Form und Form wiederum von Window. Window hat als optionalen Konstruktor-Parameter den *Parent-Container*, Form hat keine weiteren Konstruktor-Parameter.
- Wenn der Klasse FEmbedded ein neuer Parameter im Konstruktor hinzufügt wird, dann wird der neue Konstruktor-Parameter seit Gambas 3 am Ende der Argumentliste angefügt.
- Achtung:Optionale Parameter werden grundsätzlich an das Ende der Argumentliste geschrieben. Da der Parameter *iColor* in der Signatur des Konstruktors ein obligatorischer Parameter und der Parameter *Parent-Container* ein optionaler Parameter ist, wird nun dieser optionale Parameter an das Ende der Argumentliste geschoben. Das wird auch an der folgenden Zeile im o.a. Quelltext-Ausschnitt deutlich:

  ```
  hFEmbedded = New FEmbedded(&HE1EAF6, panEmbedder)
  ```

  Zuerst wird das Argument für die Farbe notiert (obligatorischer Parameter) und dann erst das Argument für den Parent-Container (optionaler Parameter).
- Beachten Sie, dass Sie das Fenster FEmbedded nicht öffnen müssen! Es wird automatisch geöffnet, wenn es in den Container (Panel) auf dem (Haupt-)Formular FMain eingefügt wird.

12.2.7.2 Projekt 2

Im zweiten Projekt erfahren Sie, wie Sie zwei unterschiedliche Formulare in ein (Haupt-)Formular einfügen und sie auch wieder aus ihrem Container auf dem (Haupt-)Formular entfernen.

Zuerst werden drei Top-Level-Fenster erzeugt und nacheinander gruppiert angezeigt. Dann können Sie die beiden Fenster Form1 und Form2 in einen (hier gemeinsamen) Container einfügen. Die beiden eingefügten Fenster können Sie – unabhängig voneinander – wieder als Top-Level-Fenster auf den Desktop setzen. Die Reihenfolge des Einfügens oder Entfernens ist frei wählbar. Als Besonderheit gilt:

Wenn das Haupt-Fenster geschlossen wird, dann werden die beiden Fenster Form1 und Form2 automatisch geschlossen. Über die Eigenschaft *Application.MainWindow* = *FMain* stellen Sie dieses Verhalten ein.

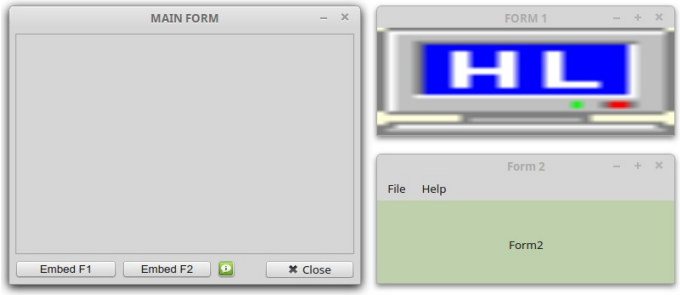


Abbildung 12.2.7.2.1: Drei Top-Level-Fenster

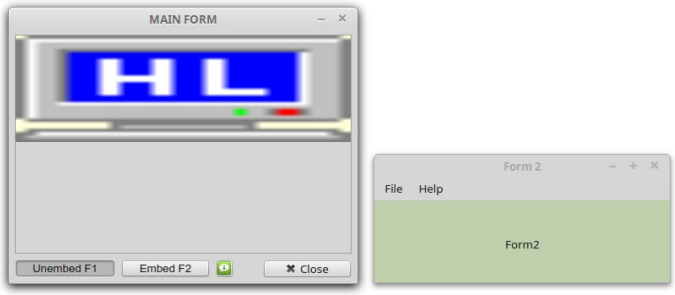


Abbildung 12.2.7.2.2: (Haupt-)Fenster mit eingefügtem Fenster 1

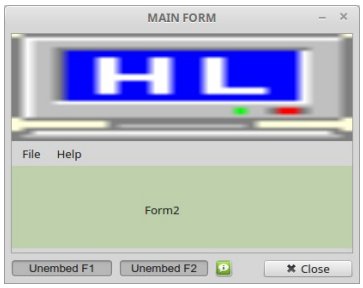


Abbildung 12.2.7.2.3: (Haupt-)Fenster mit eingefügtem Fenster 1 und 2

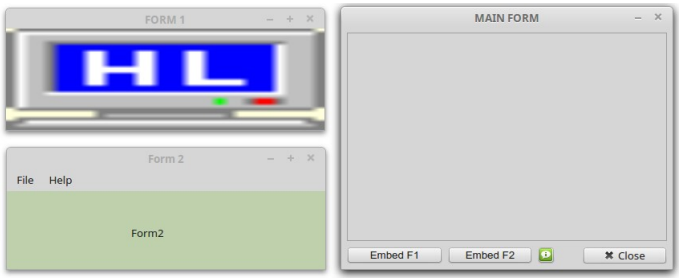


Abbildung 12.2.7.2.4: (Haupt-)Fenster mit den zwei entfernten Fenstern 1 und 2

Für einen Test in der Projekt-Erprobung können Sie sich die Anzahl der Top-Level-Fenster, deren Titel und Fensterhöhe sowie die Namen der Steuer-Elemente auf dem (Haupt-)Formular anzeigen lassen:

```
FMain
Anzahl der geöffneten Fenster = 3

1. Fenster: Caption      = MAIN FORM
1. Fenster: Fenster-Höhe = 301
----------------------------------------
```

```
2. Fenster: Caption      = FORM 1
2. Fenster: Fenster-Höhe = 126
----------------------------------------
3. Fenster: Caption      = Form 2
3. Fenster: Fenster-Höhe = 126
----------------------------------------
panContainer
HBox1
```

Das Einfügen der beiden Fenster im Projekt 2 erfolgt jeweils durch eine der beiden Prozeduren im Event-Handler *Form_Open()* von FMain:

```
Public Sub Form_Open()
' Wenn dieses Fenster geschlossen wird, werden alle Projekt-Fenster automatisch beendet
  Application.MainWindow = FMain
  ...
' Form1 und Form2 in FMain einfügen - Button-Texte setzen
  btnEMUNForm1_Click()
  btnEMUNForm2_Click()
End

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

Public Sub btnEMUNForm2_Click()
  If btnEMUNForm2.Value Then
     Form2.Reparent(panContainer, 0, 0) ' Einfügen
     btnEMUNForm2.Text = ("Unembed F2")
  Else
     Form2.Reparent(Null) ' Entfernen
     btnEMUNForm2.Text = ("Embed F2")
  Endif
End
```