17.9.1 Projekt – Exkurs TreeView

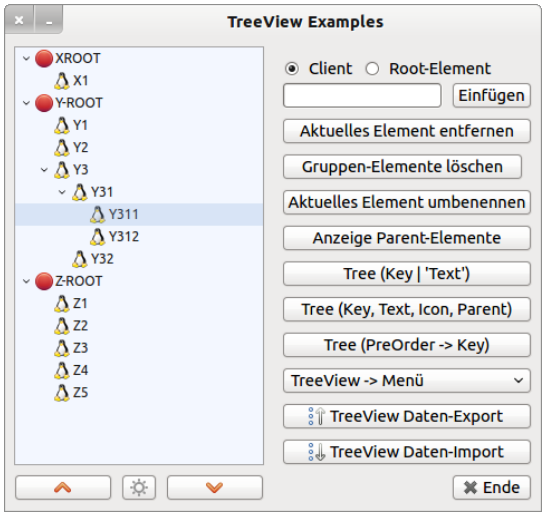

Abbildung 17.9.1.1: TreeView mit markiertem Eintrag

Das Projekt demonstriert, wie Sie mit den Inhalten in einer TreeView arbeiten können:

- Elemente einfügen → Root-Element oder Client-Element,
- Element markieren,
- Aktuelles Element löschen,
- Gruppen-Elemente löschen,
- Aktuelles Element umbenennen,
- Anzeige aller Parent-Elemente einer Gruppe,
- Formatierte Ausgabe aller Elemente einer TreeView (3 Varianten),
- Abbildung der TreeView-Hierachie einer TreeView auf ein Menü für einen Menü-Button,
- Export aller Elemente in der TreeView im JSON-Format→ Kapitel 24.9.0 JSON. Der Text wird in einer Datei *export.dat* im Projekt-Verzeichnis gespeichert,
- Import aller Elemente (JSON-Format) und Anzeige der importierten Elemente in der TreeView in der ursprünglichen Hierachie.
- TreeView einklappen,
- Markierung für alle Elemente entfernen,
- TreeView ausklappen.

Bei einem Klick auf ein Element in der TreeView werden ausgewählte Informationen zum markierten Element in der Konsole der Gambas-IDE ausgegeben:

```
Key = Y | Text = Y-ROOT | Das Element hat kein Parent-Element! | Y-ROOT hat 3 Elemente.
Key = Y3 | Text = Y3 | Parent = Y | Y3 hat 2 Elemente.
Key = Y31 | Text = Y31 | Parent = Y3 | Y31 hat 2 Elemente.
Key = Y312 | Text = Y312 | Parent = Y31 | Y312 hat keine Elemente.
Key = Z | Text = Z-ROOT | Das Element hat kein Parent-Element! | Z-ROOT hat 5 Elemente.
...
```

Den recht umfangreichen Quelltext finden Sie im Projekt 'TreeViewE' im Download-Bereich. Der Quelltext ist hinreichend kommentiert. Für eigene Projekte müssen Sie sich die Teile heraussuchen, die Sie benötigen. Hinweis: Die Eigenschaft *Current* repräsentiert den vom Benutzer markierten Eintrag in einer TreeView, während die Eigenschaft *Item* vom Typ _TreeView_Item ein interner, unsichtbarer und von der Eigenschaft *Current* unabhängiger Zeiger ist. *Current* verwendet man zur Auswertung von Benutzereingaben in einer TreeView und *Item* für alle Algorithmen, welche eine TreeView durchlaufen.

Die dynamische Abbildung der TreeView-Hierachie auf eine Menü-Hierachie gelingt durch diese Prozedur:

```
Private Function TreeViewToMenu(hTree As TreeView) As Menu

  Dim hMenu As New Menu(Me, True)

  hTree.MoveFirst() ' Setzt den internen Zeiger auf das erste Element
```

```
  _TreeViewToMenu(hTree, hMenu)

  Return hMenu

End ' Function TreeViewToMenu(...)

Private Sub _TreeViewToMenu(hTree As TreeView, hParent As Menu)

  Dim hMenu As Menu

  Do
    hMenu = New Menu(hParent) As "mnuLast" ' Ein neues Menü anlegen - Event-Name 'mnuLast'
  ' Der Menü-Text ist der Text des Elements, auf den der interne Zeiger zeigt
    hMenu.Text = hTree.Item.Text
    Print hTree.Item.Text
  ' Der Text des Elements auf den der interne Zeiger zeigt, wird in der Eigenschaft Menu.Tag gespeichert
    hMenu.Tag = hTree.Item.Key
  ' Wenn der interne Zeiger *nicht* weitergesetzt werden kann,
  ' dann wird der interne Zeiger auf die letzte Position (zurück-)gesetzt
  ' sonst
  ' erfolgt ein *rekursiver* Aufruf der aktuellen Prozedur
  ' und dann wird der interne Zeiger auf das übergeordnete Element (Parent) gesetzt
    If hTree.MoveChild() Then
       hTree.MoveBack()
    Else
       _TreeViewToMenu(hTree, hMenu)
       hTree.MoveParent()
    Endif
  Loop Until hTree.MoveNext() ' ... bis kein Element mehr durchlaufen werden kann

  hTree.MoveBack() ' Der interne Zeiger wird auf die letzte Position (zurück-)gesetzt

End ' _TreeViewToMenu(...)
```

So erfolgt der Aufruf des aktuellen Menüs für den Menü-Button:

```
Public Sub MenuButton1_Click()

  mnuMenu = New Menu(Me, True)
  mnuMenu = TreeViewToMenu(TreeView1)
  MenuButton1.Menu = mnuMenu.Name

End ' MenuButton1_Click()
```

und hier sehen Sie einen Teil der Menü-Struktur:

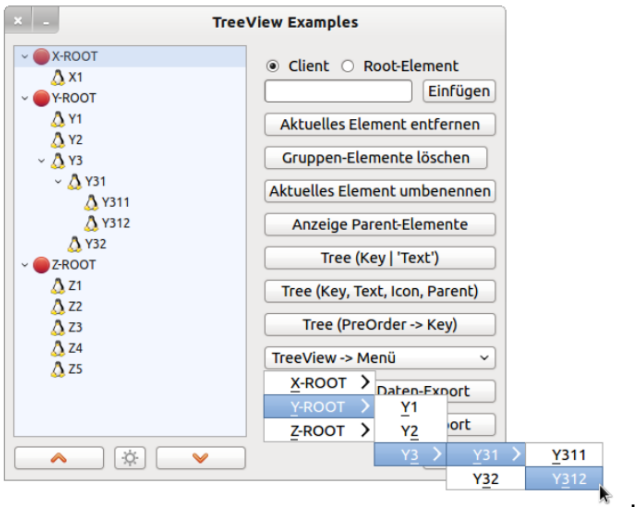

Abbildung 17.9.1.2: Menü-Button mit aktuellem Menü-Ausschnitt