RapaGUI 1.2

Rapid Cross-Platform GUI Development On All Islands

Andreas Falkenhahn

Inhaltsverzeichnis

1 Allgemeine Informationen

1.1 Einführung

RapaGUI ist ein Plugin für Hollywood, mit dem Sie ganz einfach GUIs mit Hollywood erstellen können. Sie müssen lediglich die Gestaltung der GUI in einer XML-Datei definieren, die von RapaGUI automatisch in eine vollwertige GUI umgewandelt wird. Einfacher geht es nicht!

RapaGUI's beste Eigenschaft ist definitiv das plattformübergreifende Design. RapaGUI wurde wie Hollywood als plattformunabhängiges Programm konzipiert. Daher läuft es auf einer Vielzahl von Plattformen, aber dennoch verwendet es OS-native Widgets (das ist ein Kofferwort aus "window" und "gadget" und ist ein sichtbares Bedienelement) auf allen unterstützten Plattformen, um Ihren Programmen ein authentisches Aussehen zu verleihen. Derzeit ist RapaGUI in nativen Versionen für Windows, Linux (GTK), Mac OS X, AmigaOS, MorphOS und AROS verfügbar. Mit RapaGUI müssen Sie Ihre Programme nur einmal schreiben und sie laufen automatisch auch auf vielen anderen Plattformen! Dies ermöglicht es Ihnen, Ihre Zeit auf die wirklich wichtigen Dinge, d.h. Programm-Abläufe zu verwenden, statt Anpassungen für viele verschiedene Plattformen schreiben zu müssen.

RapaGUI verwendet ein objektorientiertes Design, das aus über 40 MOAI-Klassen (Magic Omnigui Architecture Interface) besteht. Diese MOAI-Klassen bilden das Herzstück von RapaGUI. Alle von RapaGUI unterstützten GUI-Elemente (Fenster, Widgets, Menüleisten...) sind einfach Objekte, die von diesen MOAI-Klassen abgeleitet sind. Indem diese Klassen die verschiedenen nativen OS-GUI-APIs in plattformunabhängige MOAI-Klassen umwandeln, reduzieren sie die vielen Designs der verschiedenen OS-GUI-APIs in nur ein einziges MOAI API-Design, welches von RapaGUI in Stein gemeißelt wurde!

RapaGUI unterstützt alle Widgets, die Sie zur Erstellung moderner GUI-Programmen benötigen, einschließlich mehrspaltige Listenansicht, Baumansicht, Registerkarten, Symbolleiste, Texteditor, Menüleisten, HTML-Ansichten und vieles mehr. Das Highlight von RapaGUI ist jedoch sicherlich die eingebaute Hollywood-MOAI-Klasse. Diese Klasse ermöglicht die dynamische Einbettung kompletter Hollywood-Displays in GUIs, mit denen Hollywoods leistungsstarke Multimedia-Funktionalität mit RapaGUIs GUI-Fähigkeiten in einem einzigen leistungsstarken Programm kombiniert werden kann.

RapaGUI kommt mit einer umfangreichen Dokumentation in verschiedenen Formaten wie PDF, HTML, AmigaGuide und CHM daher, die die Grundlagen der GUI-Programmierung detailliert beschreibt und eine komfortable MOAI-Funktion und Klassenreferenz bietet. Eine Schritt-für-Schritt-Anleitung, die Sie zu Ihrem ersten RapaGUI-Programm führt, ist ebenfalls enthalten. Darüber hinaus sind viele Beispielskripte im RapaGUI-Archiv enthalten, darunter auch fortgeschrittene Skripte wie ein kompletter Videoplayer, die die Leistungsfähigkeit von Hollywood und RapaGUI bei der Zusammenarbeit unter Beweis stellen.

All dies macht RapaGUI zum ultimativen plattformübergreifenden GUI-Werkzeug, das sorgfältig für Sie als Segler der sieben GUI-Seen entwickelt wurde! Nur RapaGUI ermöglicht eine schnelle plattformübergreifende GUI-Entwicklung auf allen Inseln - es ist die ultimative Verschmelzung der verschiedenen OS-GUI-Werkzeugen zu einem MOAI-Werkzeug, das für die Ewigkeit und darüber hinaus in Stein gemeißelt ist.

1.2 Lizenzvereinbarung

BEIM ANWENDER. SOLLTE SICH DAS PROGRAMM ALS DEFEKT HERAUSSTELLEN, LIEGEN ALLE KOSTEN FÜR SERVICE, INSTANDSETZUNG ODER NACHBESSERUNG BEIM ANWENDER.

KEIN COPYRIGHT-INHABER ODER DRITTER, DER DAS PROGRAMM WIE OBEN ERLAUBT WEITERVERKAUFT, KANN FÜR SCHÄDEN IRGENDWELCHER ART HAFTBAR GEMACHT WERDEN (DIES BEINHALTET, IST ABER NICHT BESCHRÄNKT AUF, DATENVERLUST INFOLGE UNFÄHIGKEIT DES PROGRAMMS, MIT ANDEREN PROGRAMMEN ZUSAMMENZUARBEITEN), SELBST WENN EIN SOLCHER INHABER ODER DRITTER AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE, AUSSER ES BESTEHT EINE SCHRIFTLICHE EINWILLIGUNG ODER WIRD VOM GESETZ VERLANGT.

1.3 Anforderungen

RapaGUI benötigt mindestens Hollywood 6.1. Je nach Plattform ergeben sich folgende zusätzliche Anforderungen:

Win32 Version:

- mindestens Windows 2000

Mac OS Version:

- Intel Macs: mindestens Mac OS 10.7
- PowerPC Macs: mindestens Mac OS 10.5

Linux version:

- erfordert GTK+ 2, das für 32-Bit-Architekturen entwickelt wurde (oft müssen Sie den 32-Bit-Build von GTK+ 2 manuell installieren, da 64-Bit-Linux-Distributionen standardmäßig nicht mehr mit 32-Bit-Shared Objects ausgeliefert werden!
- optional: WebKitGTK+ wird für die HTMLview-Klasse benötigt.

AmigaOS version:

- erfordert MUI 3.8 oder besser; die Verwendung von MUI 4 wird jedoch dringend empfohlen!
- 68020+ oder PowerPC-Prozessor
- CyberGraphX oder Picasso96
- codesets.library für die UTF-8-Unterstützung
- Toolbar-Klasse erfordert TheBar.mcc
- extEditor-Klasse erfordert TextEditor.mcc
- HTMLview-Klasse erfordert HTMLview.mcc

2 Über RapaGUI

2.1 Geschichte

Für ein vollständiges Änderungsprotokoll von RapaGUI beachten Sie bitte die auf englisch verfasste Datei `history.txt`.

2.2 Zukunft

Hier sind einige Punkte, die auf meiner Aufgabenliste stehen:

- Unterstützung für Scintilla für eine erweiterte plattformübergreifende Texteditor-Komponente
- Unterstützung für erweiterte Benutzeroberflächen mit wxAUI (Docking etc.)
- Unterstützung für Ziehen und Ablegen
- Unterstützung für mehr Widgets
- Unterstützung für GTK+ 3

2.3 Häufig gestellte Fragen

Dieser Abschnitt behandelt einige häufig gestellte Fragen. Bitte lesen Sie diese zuerst, bevor Sie auf der Mailingliste oder im Forum nachfragen, da Ihr Problem möglicherweise hier behandelt wurde.

F: Meine GUI ist nicht größenveränderbar. Was mache ich falsch?

A: Es gibt einige Punkte, die Sie beachten müssen, damit die Fähigkeit zum Ändern der Größe korrekt funktioniert. Wenn es ein Widget in Ihrer GUI gibt, welches eine feste Größe hat, müssen Sie es mit Hilfe von größenveränderbaren `<rectangle>`-Objekten (Rechteck-Objekte) an den Seiten auffüllen. Dann wird Ihre GUI wieder größenveränderbar sein. Stellen Sie sich zum Beispiel vor, Sie haben ein 64x64 `<image>`-Objekt (Bild-Objekt) in einer horizontalen Gruppe in Ihrem Fenster. RapaGUI wird nicht in der Lage sein, die Größe dieses Fensters zu ändern, es sei denn, Sie fügen Rechteck-Objekte an den Seiten Ihres Bild-Objekts hinzu, weil RapaGUI ein Objekt finden muss, das in der Größe verändert werden kann. So müssen Sie darauf achten, dass nicht größenveränderbare Objekte in Ihrer Benutzeroberfläche immer mit größenveränderbaren Objekten aufgefüllt werden. Übrigens, seien Sie vorsichtig mit dem Tag `<label>`: Widget der Label-Klasse (Beschriftungen) sind eigentlich nicht größenveränderbar, weil größenveränderbare Beschriftungen ziemlich unangenehm aussehen!

F: RapaGUI läuft nicht auf meiner Linux-Distribution, obwohl ich GTK+ installiert habe. Was könnte der Grund dafür sein?

A: Stellen Sie zunächst sicher, dass Sie GTK+ 2 installiert haben. Aktuelle Linux-Distributionen werden oft mit GTK+ 3 ausgeliefert, welche inkompatibel zu Version 2 ist. Daher reicht es nicht aus, GTK+ 3 zu haben. Sie benötigen definitiv die Version 2. Denken Sie außerdem daran, dass Hollywood und RapaGUI 32-Bit-Anwendungen sind. Daher benötigen Sie auch die 32-Bit-Builds von GTK+ 2. Viele Linux-Distributionen werden

heutzutage nur noch mit den 64-Bit Shared Objects ausgeliefert. Diese funktionieren jedoch nicht mit 32-Bit-Anwendungen wie Hollywood und RapaGUI.

2.4 Danksagungen

RapaGUI wurde von Andreas Falkenhahn geschrieben. Das Design wurde von meinem MUI Royale Plugin inspiriert, welches nur auf AmigaOS und Kompatibeln läuft und Ende 2012 erstmals veröffentlicht wurde. Erste Experimente mit einem wxWidgets-basierten GUI-Toolkit für Hollywood wurden bereits 2013 gestartet. Ursprünglich wollte ich nur ein Wrapper-Plugin entwickeln, welches es Hollywood-Skripten erlaubt, wxWidgets zu verwenden. Aber dann wurde mir klar, dass die wxWidgets-API ziemlich kompliziert in der Anwendung sein kann und ich dachte, dass ein MUI Royale-basierter Ansatz viel benutzerfreundlicher und bequemer für GUI-Skripting sein würde. Durch die Nachahmung der MUI-Programmierparadigmen in RapaGUI konnte ich auch Versionen für AmigaOS und Kompatible veröffentlichen, was RapaGUI zum ersten plattformübergreifenden GUI-Toolkit macht, welches das native MUI-Toolkit des Amiga unterstützt. Damit hat RapaGUI das Potenzial, den Traum vieler Amiga-Anwender wahr werden zu lassen: Ein plattformübergreifendes GUI-Toolkit, das native Widgets auf allen Plattformen verwendet! RapaGUI hat diese Mission endlich erfüllt. Vielen Dank an das wxWidgets-Team und Stefan Stuntz für ihre wunderbaren GUI-Toolkits. Ein besonderer Dank geht an Vadim Zeitlin und Eric Jensen für ihre wertvolle Hilfe zu wxWidgets sowie Thore Böckelmann für seine sofortigen MUI-Fixes und seine Offenheit für nützliche MUI-Erweiterungen, die er ebenfalls sehr schnell implementiert hat.

Ein weiterer Dank geht an Alfonso Ranieri für TheBar.mcc und codesets.library, Allan Odgaard für TextEditor.mcc, HTMLview.mcc und den Open Source-Teams TheBar.mcc, HTMLview.mcc sowie TextEditor.mcc für die Pflege dieser Klassen und das Beheben alter Fehler.

Der nächste besondere Dank geht an Dominic Widmer und Helmut Haake für die Übersetzung des Handbuchs ins Deutsche. Fehler oder Verbesserungsvorschläge bzgl. des deutschen Hollywood-Handbuchs bitte an das Übersetzungsteam richten, welches unter handbuch@gmx.ch erreicht werden kann.

Wenn Sie mich kontaktieren möchten, senden Sie bitte eine E-Mail an andreas@airsoftsoftwair . de oder nutzen Sie das Kontaktformular unter http://www.hollywood-mal.com.

3 Konzeptioneller Überblick

3.1 Anwendungsbaum

Jede RapaGUI-Anwendung ist im Grunde genommen ein Baum, der viele MOAI-Objekte enthält, aus denen sich die Anwendung zusammensetzt. Typischerweise sind MOAI-Objekte nur Widgets wie Schaltflächen, Listenansichten, Textlisten etc, können aber auch separate Fenster oder abstrakte Objekte wie Gruppenobjekte sein, die zur Berechnung des GUI-Layouts verwendet werden.

Das Wurzelelement jedes Anwendungsbaum muss immer eine Instanz von der Applications-Klasse sein. Außerhalb des Applikations-Objekts darf kein MOAI-Objekt existieren. Applikations-Objekte werden durch den Aufruf von `moai.CreateApp()` erzeugt und es kann in jedem Programm nur ein Applikations-Objekt geben. Das Applikations-Objekt ist für die Abfertigung aller Ereignisse und Nachrichten zuständig, die Ihr Programm benötigt. Siehe Abschnitt 9.1 [Application-Klasse], Seite 57, für Details.

Die wichtigsten Elemente des Applikations-Objekt sind die separaten im Vordergrund befindlichen Fenster Ihrer Anwendung. Dies können normale Top-Level-Fenster oder modale Dialoge sein, die den Rest Ihrer Anwendung blockieren, während sie geöffnet sind. Um diese Objekte in XML zu erzeugen, müssen Sie sie nur mit den Tags `<window>` und `<dialog>` erzeugen. Siehe Abschnitt 58.1 [Window-Klasse], Seite 263, für Details. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details.

Jedes Fenster oder jeder Dialog muss immer genau ein Wurzelobjekt haben, das von der Group-Klasse abgeleitet sein muss. Mit der Group-Klasse können Sie ein oder mehrere Widgets in einem horizontalen, vertikalen oder Raster-basierten Layout kombinieren. Dies ist eine der wichtigsten Klassen, auf die mit den Tags `<vgroup>`, `<hgroup>` und `<colgroup>` aus XML zugegriffen werden kann. Wenn sich die Fenstergröße ändert, wird das an alle Gruppen automatisch weitergeleitet, was bedeutet, dass Sie sich nicht um die komplexe Aufgabe kümmern müssen, Ihr GUI-Layout auf der Grundlage der aktuellen Fenstergröße neu zu berechnen. RapaGUI macht das alles automatisch für Sie. Siehe Abschnitt 17.1 [Group-Klasse], Seite 87, für Details.

Eine weitere sehr wichtige Klasse ist die Area-Klasse (Bereichs-Klasse). Alle Widgets sind untergeordnete Elemente der Area-Klasse, weil Area-Klasse im Grunde nur einen rechteckigen Bereich innerhalb des GUI-Layouts beschreibt, auf dem Widgets-abhängige Grafiken gezeichnet werden. Dies bedeutet, dass Sie alle Attribute und Methoden dieser Klasse für alle MOAI-Objekte, die Widgets sind, verwenden können. Sie können zum Beispiel die Dimensionen aller Ihrer Widgets festlegen, indem Sie nur die Attribute `Area.Width` und `Area.Height` verwenden. Dies ist jedoch normalerweise nicht notwendig, da RapaGUI automatisch die Dimensionen für alle Widgets auswählt. Wenn Sie mit der Wahl von RapaGUI nicht zufrieden sind, können Sie sie jedoch mit diesen Attributen außer Kraft setzen. Siehe Abschnitt 10.1 [Area-Klasse], Seite 61, für Details.

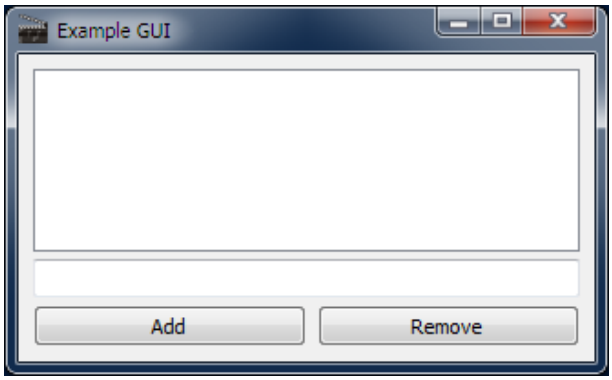
3.2 GUI-Layout

Wie Sie bereits oben gesehen haben, werden in RapaGUI GUIs vollständig mit der XML-Sprache definiert. Auf diese Weise können Sie schnell GUI-Layouts erstellen, indem Sie

einfach eine Baumhierarchie aus mehreren Fenstern, Gruppen und Widgets zusammensetzen. Hier ist ein Beispiel, wie ein Applikations-Baum in der XML-Sprache aussehen könnte:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<application>
   <window title="Example GUI">
      <vgroup>
         <listview>
            <column/>
         </listview>
         <textentry/>
         <hgroup>
            <button id="add">Add</button>
            <button id="rem">Remove</button>
         </hgroup>
      </vgroup>
   </window>
</application>
```

Und hier sehen Sie, wie diese GUI unter Windows 7 aussieht:



Wie Sie im XML sehen können, werden keine Positionskoordinaten oder -größen angegeben. Dies alles wird von RapaGUI automatisch berechnet, um ein sauberes Erscheinungsbild auf allen Systemen zu gewährleisten. Denken Sie daran, dass RapaGUI-Applikationen auf einer Vielzahl von Systemen laufen können: Auf Windows, GTK, Mac OS und sogar auf AmigaOS. Durch Weglassen von fest programmierten Fenster- und Widget-größen sowie -positionen kann RapaGUI automatisch die besten Größen und Positionen in Abhängigkeit von der aktuellen Bildschirm- und Schriftgröße auswählen. Natürlich können Sie RapaGUI auch zwingen, fest codierte Größen zu verwenden, indem Sie die Attribute `Area.Width` und `Area.Height` verwenden. Um eine Listenansichts-Größe von mindestens 600x400 Pixel zu erzwingen, könnte man einfach schreiben:

```
<listview width="600" height="400">
   <column/>
</listview>
```

Das Einzige, was Sie nicht ändern können, ist die Position des Widgets. Da die Position der Widgets stark von der aktuellen Schriftgröße der Benutzeroberfläche abhängt, macht es wirklich keinen Sinn, hartcodierte Widget-Positionen in einem plattformübergreifenden

GUI-Werkzeug zuzulassen, da es massive Unterschiede zwischen den verschiedenen Betriebssystemen geben wird. RapaGUI erlaubt es Ihnen also nicht, Widget-Positionen fest zu programmieren.

3.3 Ausführen von GUIs

Alles, was Sie tun müssen, um mit XML definierten GUIs zu erstellen und auszuführen, ist ein paar Zeilen Hollywood-Code zu schreiben. Nehmen wir an, Sie haben die XML-Beispieldefinition von oben in einer Datei mit dem Namen `GUI.xml` gespeichert. Zum Erstellen und Anzeigen dieser GUI aus einem Hollywood-Skript müssen Sie ein Hollywood-Skript mit den folgenden Zeilen schreiben:

```
@REQUIRE "RapaGUI"
moai.CreateApp(FileToString("GUI.xml"))
Repeat
   WaitEvent
Forever
```

Das ist alles! Wenn Sie das Hollywood-Skript von oben ausführen, wird Ihre GUI automatisch erstellt und angezeigt, unabhängig davon, ob Ihr Skript unter Windows, Linux, Mac OS oder AmigaOS läuft.

3.4 Umgang mit Objekten

Wie Sie bereits gesehen haben, verwendet RapaGUI XML-Dateien, um MOAI-Objekte zu erstellen. Beim Anlegen von Objekten können Sie alle mit dem Buchstaben `I` gekennzeichneten Attribute im Abschnitt Anwendbarkeit der zugehörigen Attributdokumentation verwenden. Um beispielsweise ein Texteingabe-Objekt mit einer maximalen Länge von 80 Zeichen zu erstellen, verwenden Sie einfach das Attribut `TextEntry.MaxLen` und nehmen es in Ihre XML-Definition auf.

```
<textentry id="mystring" maxlen="80"/>
```

Sobald Ihr Objekt fertig ist, können Sie mit der Interaktion beginnen, indem Sie seine Attribute festlegen oder abrufen oder Sie ihre Methoden ausführen. Zu diesem Zweck ist es wichtig, dass Sie Ihrem Objekt einen Identifikator (ID, Identität, Kennung) mit dem Attribut `id` geben, damit die Befehle `moai.Set()`, `moai.Get()` und `moai.DoMethod()` Ihr Objekt finden kann. Im obigen Code haben wir unserem TextEntry-Widget (Texteingabe) die ID `mystring` zugewiesen. Hier ist ein Beispiel, wie wir jetzt mit diesem Widget interagieren können:

```
moai.Set("mystring", "text", "look")
s$ = moai.Get("mystring", "text")
DebugPrint("Always " .. s$ .. " on the bright side of life.")
```

Wie bereits oben erwähnt, sind alle Attribute und Methoden in der Dokumentation von RapaGUI vollständig beschrieben.

Das nächste, was Sie erledigen müssen, ist die Verarbeitung von Ereignissen, die von Ihrer GUI ausgelöst werden. Dies wird im nächsten Abschnitt erklärt.

3.5 Ereignisse verarbeiten

RapaGUI-Ereignisse werden genauso verarbeitet wie normale Hollywood-Ereignisse. Das bedeutet, dass Sie lediglich eine Callback-Funktion an den Befehl `InstallEventHandler()` von Hollywood übergeben müssen, und jedes Mal, wenn ein RapaGUI-Ereignis auftritt, wird Ihre Callback-Funktion aufgerufen.

Hier ist ein Beispiel, wie Sie eine Callback-Funktion für ein RapaGUI-Ereignis installieren:

```
InstallEventHandler({RapaGUI = p_EventHandler})
```

Immer wenn ein Ereignis eintritt, ruft RapaGUI die Funktion `p_EventHandler()` mit einer Tabelle als Parameter auf. In dieser Tabelle werden die folgenden Felder initialisiert:

`Action:` Enthält "RapaGUI".

`Class:` Enthält den Namen der MOAI-Klasse, von der dieses Ereignis stammt, z.B. "Choice".

`Attribute:`
Enthält den Namen des Klassenattributs, welches das Ereignis ausgelöst hat, z.B. "Active".

`ID:` Enthält die ID des MOAI-Objekts, welches dieses Ereignis ausgelöst hat, z.B. "meineauswahl".

`TriggerValue:`
Enthält den aktuellen Wert des Attributs, welches dieses Ereignis ausgelöst hat. Sie können dies auch herausfinden, indem Sie den Befehl `moai.Get()` auf dem Objekt anwenden, aber es ist bequemer, den aktuellen Wert direkt mit ihrer Callback-Funktion auslesen.

`MOAIUserData:`
Wenn dem Objekt bestimmte Benutzerdaten zugewiesen wurden, wird es in diesem Tag an die Callback-Funktion übergeben. Siehe Abschnitt 30.7 [MOAI.UserData], Seite 155, für Details.

`NotifyData:`
Wenn dem Objekt bestimmte Benachrichtigungsdaten zugewiesen wurden, wird es in diesem Tag an die Callback-Funktion übergeben.

Darüber hinaus ist es auch notwendig, RapaGUI mitzuteilen, welche Ereignisse Sie erhalten möchten. Nur Ereignisse, die Sie explizit anfordern, werden an Ihre Callback-Funktion übergeben, um die Verwaltungsdaten (Overhead) zu minimieren. Eine Ausnahme bilden Schaltflächen, Symbolleistenschaltflächen und Menüeinträge. Sie werden an Ihre Callback-Funktion weitergeleitet, auch wenn Sie sie nicht angefordert haben. Der Grund für diese Auswahl von Verwaltungsdaten liegt darin, dass normalerweise alle diese Ereignisse individuell behandelt werden müssen, da der Benutzer erwartet, dass auf Knopfdruck etwas passiert.

Alle RapaGUI-Ereignisse sind mit Attributen von MOAI-Objekten gekoppelt. Der nächste Abschnitt beschreibt das im Detail.

3.6 Benachrichtigungen der Attribute

Der Umgang mit Ereignissen von RapaGUI basiert auf den Werten bestimmter Attribute der verschiedenen MOAI-Klassen. Sie können die Werte aller Attribute überwachen, die eine Anwendbarkeit von `N` haben. Wenn Sie einen Benachrichtigung für einen Attributwert einrichten, führt RapaGUI Ihre Callback-Funktion aus, wenn sich dieser Attributwert ändert.

Sie können z.B. eine Benachrichtigung für das Attribut `Listview.Active` einrichten, welches das aktive Element eines Listview-Widgets (Listenansicht) ist. In diesem Fall wird Ihre Callback-Funktion immer dann aufgerufen, wenn sich das aktive Element des jeweiligen Listview-Widget ändert. Um eine Überwachung auf einem bestimmten Attribut zu installieren, müssen Sie beim Erstellen Ihres Objekts eine Benachrichtigung für dieses Attribut einrichten. Dies geschieht direkt in der XML-Datei unter Verwendung des Attributs `notify`, das von allen MOAI-Klassen akzeptiert wird:

```
<listview id="lv" notify="active">
   <column>
      <item>One</item>
      <item>Two</item>
      <item>Three</item>
   </column>
</listview>
```

Der obige Code führt Ihre Callback-Funktion immer dann aus, wenn sich das aktive Element des Listview-Objekts mit der ID `lv` ändert, weil Sie den Wert des Attributs `Listview.Active` überwachen.

Wenn Sie mehrere verschiedene Benachritigungen vom demselben MOAI-Objekt einrichten möchten, müssen Sie diese durch Semikolons (;) trennen, z.B:

```
<listview id="lv" notify="active; doubleclick">
...
</listview>
```

Wie Sie sehen können, installiert der obige Code die Überwachung für die beiden Attributen `Listview.Active` und `Listview.DoubleClick`, so dass Ihre Callback-Funktion auch dann aufgerufen wird, wenn der Benutzer auf einen Listeneintrag doppelklickt.

Alternativ können Sie Benachrichtigungen zur Laufzeit auch mit dem Befehl `moai.Notify()` aus Ihrem Code einrichten oder entfernen.

Um das gerade aktive Listenelement in Ihrer Callback-Funktion auszulesen, können Sie den folgenden Code verwenden:

```
Function p_EventHandler(msg)
    Switch msg.action
    Case "RapaGUI":
       Switch msg.attribute
       Case "Active":
          Switch msg.id
          Case "lv":
             DebugPrint("Active listview item:", msg.triggervalue)
          EndSwitch
```

```
        EndSwitch
    EndSwitch
EndFunction
```

Beachten Sie jedoch, dass Attributwerte auch manuell geändert werden können. Beispielsweise könnte Ihr Programm ein bestimmtes Element der Listenansicht manuell aktivieren wollen, indem es z.B. folgenden Befehl ausführt:

```
moai.Set("lv", "active", 5)  ; aktiviert Element Nummer 6
```

Da der obige Code auch den Wert des Attributs `Listview.Active` ändert, wird auch Ihre Callback-Funktion durch diesen Aufruf ausgelöst. Wenn Sie dies nicht wollen, können Sie das spezielle Attribut `MOAI.NoNotify` im Befehl `moai.Set()` verwenden. Immer wenn `MOAI.NoNotify` auf `True` gesetzt ist, wird der Wert des Attributs geändert, ohne dass irgendwelche Callback-Funktionen aufgerufen werden. Um also das aktive Element der Listenansicht ohne Callback-Funktionen zu ändern, müssen Sie einfach folgendes schreiben:

```
moai.Set("lv", "active", 5, "nonotify", True)
```

Schließlich gibt es, wie bereits im vorherigen Abschnitt erwähnt, einige Attribute, die RapaGUI immer überwacht. Dies sind: `Button.Pressed`, `Button.Selected`, `Toolbarbutton.Pressed`, `Toolbarbutton.Selected` und `Menuitem.Selected`. Da diese so häufig und weit verbreitet sind und Sie normalerweise immer Attribute wie `Button.Pressed` überwachen möchten, weil das Drücken einer Taste immer eine Reaktion hervorruft, benachrichtigt Sie RapaGUI automatisch. Daher müssen Sie nicht explizit eine Benachrichtigung von diesen Attributen anfordern, d.h. das Schreiben des Folgenden ist überflüssig:

```
<button notify="pressed">Click me</button>
```

Stattdessen können Sie einfach schreiben:

```
<button>Click me</button>
```

Dasselbe gilt für Menüeinträge und Schaltflächen in der Symbolleiste (Toolbar).

3.7 Dynamische Objekte

Die meisten Programme erstellen ihre GUI durch einen einfachen Aufruf von `moai.CreateApp()`, dem eine XML-Definition übergeben wird, welche die komplette Benutzeroberfläche des gesamten Programms enthält. In manchen Situationen kann es jedoch notwendig sein, MOAI-Objekte nach dem Aufruf von `moai.CreateApp()` zu erzeugen und in die bestehende Anwendung einzufügen.

Dies ist möglich, indem Sie den Befehl `moai.CreateObject()` verwenden. Dieser Befehl ermöglicht es Ihnen, ein MOAI-Objekt aus einer XML-Definition zu erstellen, ähnlich wie `moai.CreateApp()`, außer dass `moai.CreateObject()` Ihnen nicht erlaubt ein GUI-Application-Objekt (Anwendung) zu erstellen, da es nur ein GUI-Application-Objekt geben kann.

Zum Beispiel können Sie eine Schaltfläche erstellen, indem Sie `moai.CreateObject()` wie folgt verwenden:

```
moai.CreateObject([[<button id="mybutton">Click me</button>]])
```

Es ist sehr wichtig, eine ID für das MOAI-Objekt zu setzen, welches Sie mit `moai.CreateObject()` erstellen, da der Identifikator später für das Objekt benötigt

wird. Nachdem wir die Schaltfläche erstellt haben, können wir sie in ein bestehendes Fensterlayout einfügen. Nehmen wir an, unser Fensterlayout sieht momentan so aus:

```
<window>
   <hgroup id="mygroup">
      <button id="ok">OK</button>
      <button id="cancel">Cancel</button>
   </hgroup>
</window>
```

Wir könnten nun die Methode `Group.Insert` verwenden, um die neu erstellte Schaltfläche nach der Schaltfläche "OK" einzufügen. Dazu müssten wir den folgenden Code verwenden:

```
moai.DoMethod("mygroup", "initchange")
moai.DoMethod("mygroup", "insert", "mybutton", "ok")
moai.DoMethod("mygroup", "exitchange")
```

Sie sehen, dass wir auch die Methoden `Group.InitChange` und `Group.ExitChange` aufrufen. Das ist sehr wichtig. Diese Methoden müssen immer dann ausgeführt werden, wenn Sie die Elemente einer Gruppe ändern, d.h. wenn Sie Elemente entfernen oder hinzufügen. RapaGUI benötigt den Aufruf dieser Methoden, da nach dem Entfernen oder Hinzufügen von Gruppenelementen ein Neuzeichnen des Fensterlayout notwendig wird. Deshalb reicht es nicht aus, einfach `Group.Insert` oder eine andere Methode aufzurufen, die die Anzahl der Gruppenelemente ändert.

Natürlich können wir auch Elemente aus Gruppen entfernen. Dies ist mit `Group.Remove` möglich. Der folgende Code entfernt die Schaltfläche "Abbrechen" aus dem obigen Fensterlayout:

```
moai.DoMethod("mygroup", "initchange")
moai.DoMethod("mygroup", "remove", "cancel")
moai.DoMethod("mygroup", "exitchange")
```

Nach diesem Aufruf ist das MOAI-Objekt mit der ID "cancel" ein losgelöstes Objekt geworden. Das bedeutet, dass Sie es jetzt auch wieder an eine Gruppe anhängen können, indem Sie `Group.Insert` oder eine ähnliche Methode verwenden. Beachten Sie, dass alle Methoden, die MOAI-Objekte in bestehende Layouts einfügen, nur losgelöste Objekte akzeptieren. Es ist nicht möglich, dasselbe MOAI-Objekt in mehrere Gruppen einzufügen. Sobald Sie ein MOAI-Objekt in eine Gruppe (oder einen Menübaum) einfügen, ändert es seinen Zustand von losgelöst in angehängt. Sobald ein MOAI-Objekt im angehängten Zustand ist, kann es nicht mehr an Methoden übergeben werden, die ein losgelöstes Objekt erwarten. Um ein MOAI-Objekt vom angefügten in den gelösten Zustand zu versetzen, müssen Sie eine Methode wie `Group.Remove` verwenden, um ein MOAI-Objekt von einer Gruppe zu trennen.

Als letzte Anmerkung: Der XML-Code, den Sie an `moai.CreateObject()` übergeben, kann auch ein vollständiger Baum sein, z.B. könnten Sie ein vollständiges Fenster mit nur einem Aufruf mit dem Befehl `moai.CreateObject()` erstellen. Das ist alles möglich. `moai.CreateObject()` ist nicht auf das Erzeugen einzelner MOAI-Objekte beschränkt, sondern kann auch mehrere Objekte für Sie erstellen. Wenn Sie jedoch ein Fenster oder ein Dialog mit `moai.CreateObject()` erstellen, vergessen Sie nicht, es zum Applications-Objekt hinzuzufügen, indem Sie `Application.AddWindow` aufrufen. Fensterobjekte sind standardmäßig im gelösten Zustand, sofern sie keinem Application-Objekt hinzugefügt

wurden. Für Dialoge können Sie einfach `moai.CreateDialog()` verwenden, welches `Application.AddWindow` einschließlich aufruft.

3.8 Grössenveränderung

Eine der wichtigsten Eigenschaften von RapaGUI ist die Fähigkeit, das gesamte GUI-Layout automatisch neu zu berechnen, wenn die Größe des Fensters mit den Widgets geändert wird. Dies ist jedoch nur möglich, wenn alle Gruppen in einem Fenster in der Größe geändert werden können, da sonst die Größe des Fensters unverändert bleibt und nicht skalierbar ist. Um frei skalierbare Fenster zu erstellen, müssen Sie wissen, welche Widgets in der Größe geändert werden können und welche nicht. Beachten Sie die folgende GUI-Definition:

```
<window>
   <vgroup>
      <button>Hello World!</button>
   </vgroup>
</window>
```

Dieses Fenster ist nur horizontal skalierbar, da Schaltflächen-Widgets standardmäßig nur horizontal gedehnt/gestaucht werden können. Sie sind standardmäßig nicht vertikal skalierbar, da Schaltflächen, die ihre vertikale Größe ändern, ziemlich hässlich aussehen. Normalerweise ist die Höhe einer Schaltfläche an die Standardschaltflächenhöhe gebunden, die derzeit auf dem Betriebssystem benutzt wird.

Wenn Sie möchten, dass dieses Fenster in der Größe veränderbar ist, haben Sie mehrere Möglichkeiten: Die meistgebrauchte Möglichkeit ist das Einfügen von leerem Raum mit dem Rechteck-Objekt Rectangle-Klasse. Diese sind in alle Richtungen veränderbar. Wenn Sie also ein Objekt `<rectangle>` einfügen, wird Ihr Fenster plötzlich in alle Richtungen skalierbar:

```
<window>
   <vgroup>
      <button>Hello World!</button>
      <rectangle/>
   </vgroup>
</window>
```

Alternativ können Sie auch ein anderes Widget einfügen, das vertikal skalierbar ist, z.B. eine Listenansicht (Listview). Oder Sie können die Schaltfläche sogar vertikal skalierbar machen, indem Sie das Attribut `Area.FixHeight` auf `False` setzen:

```
<window>
   <vgroup>
      <button fixheight="false">Hello World!</button>
   </vgroup>
</window>
```

Aber das sieht nicht so schön aus, weil der Benutzer jetzt eine Schaltfläche mit einer Höhe von einigen hundert Pixeln haben könnte. Deshalb ist der Ansatz mit `<rectangle>` die am häufigsten verwendete Methode, um Füllbereiche für nicht veränderbare Objekte einzufügen. Deshalb hier eine Übersicht über die Größe der einzelnen Widgets, die von RapaGUI unterstützt werden:

`Busybar-Klasse (Beschäftigung):`
: Horizontal veränderbar.

`Button-Klasse (Schaltflächen):`
: Horizontal veränderbar.

`Checkbox-Klasse (Auswahlkästchen):`
: Nicht veränderbar.

`Choice-Klasse (Auswahl):`
: Horizontal veränderbar.

`Combobox-Klasse (Textlisten):`
: Horizontal veränderbar.

`Hollywood-Klasse:`
: Nicht veränderbar.

`HLine-Klasse (HorizontalLinie):`
: Horizontal veränderbar.

`HSpace-Klasse (HorizontalAbstand):`
: Vertikal veränderbar.

`HTMLview-Klasse (HTMLAnsicht):`
: In alle Richtungen veränderbar.

`Image-Klasse (Bild):`
: Nicht veränderbar.

`Label-Klasse (Beschriftung):`
: Nicht veränderbar.

`Listview-Klasse (Listenansicht):`
: In alle Richtungen veränderbar.

`Pageview-Klasse (Seitenansicht):`
: In alle Richtungen veränderbar.

`Popcolor-Klasse (Farbdialog):`
: Horizontal veränderbar.

`Popfile-Klasse (Dateidialog):`
: Horizontal veränderbar.

`Popfont-Klasse (Schriftdialog):`
: Horizontal veränderbar.

`Poppath-Klasse (Verzeichnisdialog):`
: Horizontal veränderbar.

`Progressbar-Klasse (Fortschrittsbalken):`
: Horizontal skalierbar für horizontale Fortschrittsbalken. Vertikal skalierbar für vertikale Fortschrittsbalken.

`Radio-Klasse:`
: Nicht veränderbar.

`Rectangle-Klasse (Rechteck):`
: In alle Richtungen veränderbar.

`Scrollbar-Klasse (Bildlaufleisten):`
: Horizontal skalierbar für horizontale Bildlaufleisten. Vertikal skalierbar für vertikale Bildlaufleisten.

`Scrollcanvas-Klasse (Bildlaufleinwand):`
: In alle Richtungen veränderbar.

`Scrollgroup-Klasse (Bildlaufgruppen):`
: In alle Richtungen veränderbar.

`Slider-Klasse(Schieberegler):`
: Horizontal skalierbar für horizontale Schieberegler. Vertikal skalierbar für vertikale Schieberegler.

`Text-Klasse:`
: Horizontal veränderbar.

`Texteditor-Klasse:`
: In alle Richtungen veränderbar.

`Textentry-Klasse (Texteingabe):`
: Horizontal veränderbar.

`Textview-Klasse (Textanzeige):`
: In alle Richtungen veränderbar.

`Treeview-Klasse (Baumansicht):`
: In alle Richtungen veränderbar.

`VLine-Klasse (VertikalLinie):`
: Vertikal veränderbar.

`VSpace-Klasse (VertikalAbstand):`
: Horizontal veränderbar.

Natürlich können Sie diese Standardeinstellungen ändern, indem Sie die Attribute `Area.FixWidth` und `Area.FixHeight` entsprechend setzen, um bestimmte Größeneinstellungen zu aktivieren oder zu deaktivieren. Aber das ist oft nicht empfehlenswert, da es ziemlich hässlich aussieht, wenn z.B. ein Texteingabe-Widget plötzlich vertikal skalierbar ist. Dies würde den Benutzer unnötig verwirren, da er am Ende ein Texteintrags-Widget mit einer Höhe von 300 Pixeln haben könnte, welches aber nur die Eingabe einer Textzeile erlaubt. Daher ist es die beste Idee, die Standardeinstellungen des Betriebssystems für die Standard-Widgets zu respektieren.

Natürlich gibt es auch Widgets, bei denen es völlig akzeptabel ist, die Standardeinstellungen von oben zu überschreiben. Zum Beispiel sind Widgets, die von der Hollywood-Klasse abgeleitet sind, standardmäßig nicht skalierbar. Aber natürlich sollten Sie `Area.FixWidth` und `Area.FixHeight` verwenden, um die Größe des Widgets an Ihre persönlichen Bedürfnisse anzupassen.

3.9 Anwendbarkeit

In der Dokumentation jedes Objektattributs finden Sie Informationen über die Anwendbarkeit dieses Attributs. Die Attribut-Anwendbarkeit wird in Form einer Kombination der vier Buchstaben I, S, G und N beschrieben. Dies teilt Ihnen die verschiedenen Möglichkeiten mit, in denen das Attribut verwendet werden kann.

Hier ist eine Erklärung der verschiedenen Möglichkeiten der Anwendbarkeit:

`I` Das Attribut kann beim Anlegen des Objekts in der XML-Datei verwendet werden. (Initialisierungszeit)

`S` Das Attribut kann zur Laufzeit mit `moai.Set()` verwendet werden.

`G` Das Attribut kann mit `moai.Get()` zur Laufzeit verwendet werden.

`N` Benachrichtigungen zu diesem Attribut sind entweder über das Attribut "Notify" in der XML-Definition oder durch Aufruf von `moai.Notify()` zur Laufzeit möglich.

Wenn zum Beispiel ein Attribut eine Anwendbarkeit von nur "I" hat, dann kann dieses Attribut nur während der Initialisierung des Objekts verwendet und kann später nicht mit `moai.Set()` geändert werden. Wenn ein Attribut eine Anwendbarkeit von nur "S" hat, ist es andererseits nicht möglich, das Attribut bereits zur Initialisierungszeit in der XML-Datei anzugeben. Attribute, die eine Anwendbarkeit von "ISGN" haben, können mit allen Möglichkeiten verwendet werden.

3.10 Tastaturkürzel

RapaGUI ermöglicht es Ihnen, Tastenkombinationen ganz leicht einzurichten, indem Sie einfach ein Unterstreichungszeichen ("_") vor dem Zeichen verwenden, das Sie als Tastenkombination einrichten möchten. Sie sollten Tastaturkürzel immer unterstützen, weil viele Benutzer sie lieber als die Maus verwenden. Besonders wenn es um Aktionen geht, die Dutzende Male wiederholt werden müssen, ist es viel einfacher, die Tastatur zu verwenden. Mit der Maus müssen Sie ständig zu bestimmten Untermenüs navigieren, die tief in der Menühierarchie versteckt sind. Die folgenden Klassen unterstützen Tastaturkürzel, die durch den Unterstrich definiert werden:

- Button-Klasse (Schaltflächen)
- Checkbox-Klasse (Auswahlkästchen)
- Label-Klasse (Beschriftungen)
- Menu-Klasse
- Menuitem-Klasse
- Radio-Klasse
- Toolbarbutton-Klasse (Symbolleistenschaltflächen)

Hier ist ein Beispiel, wie Sie zwei Tastenkombinationen für Widgets einrichten, die von Button-Klasse abgeleitet sind:

```
<hgroup>
   <button>_OK</button>
   <button>_Cancel</button>
```

```
</hgroup>
```

Der obige XML-Code wird "O" und "C" als Tastaturkürzel einrichten. Unter Windows muss der Benutzer mit `ALT+O` die Schaltfläche "OK" und mit `ALT+C` die Schaltfläche "Abbrechen" auswählen. Auf anderen Plattformen sind die Kombinationen manchmal unterschiedlich, z.B. unter Mac OS müssen Sie die Taste `COMMAND` anstelle der Taste `ALT` verwenden.

Da die Label-Klasse keine Widgets erstellt, die vom Benutzer gesteuert werden können, aktiviert ein Tastaturkürzel für Beschriftungen einfach das nächste Widget im GUI-Layout, das normalerweise das Widget ist, das durch das Label beschrieben wird. Beispielsweise:

```
<hgroup>
   <label>_Name</label>
   <textentry/>
</hgroup>
```

Wie Sie sehen, wurde im obigen XML-Code "N" als Tastenkombination eingerichtet, indem ein Unterstrich verwendet wurde. Wenn Sie `ALT+N` drücken, wird das Texteingabe-Widget aktiviert, da Beschriftungen den Fensterfokus nicht übernehmen können.

In dem seltenen Fall, dass Sie den Unterstrich als Beschriftung für eines Ihrer Widgets verwenden und nicht automatisch in ein Tastaturkürzel konvertieren möchten, können Sie einfach das Attribut `Area.NoAutoKey` auf `True` setzen . Wenn dies eingestellt ist, zeigt RapaGUI einfach den Unterstrich an und behandelt ihn nicht als Sonderzeichen für ein Tastaturkürzel. Hinweis: Da Menü- und Menüobjekte nicht von der Area-Klasse abgeleitet sind, müssen Sie `Menu.NoAutoKey` und `Menuitem.NoAutoKey` verwenden, wenn Sie die automatische Verknüpfungserzeugung für Menü- oder Menüobjekte deaktivieren möchten.

Wenn Sie komplexere Tastenkombinationen (z. B. bestimmte Funktions- oder Steuertasten, Tastenkombinationen) verwenden wollen, müssen Sie eine Tastaturkürzel-Tabelle mit der Accelerator-Klasse für Ihr Fenster einrichten. Die Accelerator-Klasse ermöglicht Ihen die Definition von erweiterten Tastaturkürzeln, die auch unabhängig von Widgets in Ihrem Fenster sind. Wenn Ihre komplexen Tastenkombinationen jedoch mit Menü-Elementen verknüpft sind, müssen Sie nicht die Accelerator-Klasse verwenden, sondern Sie können das Attribut `Menuitem.Shortcut` einfach auf die gewünschte Verknüpfung setzen. Dadurch wird automatisch ein Tastaturkürzel eingerichtet. Siehe Abschnitt 29.6 [Menuitem.Shortcut], Seite 149, für Details. Siehe Abschnitt 7.1 [Accelerator-Klasse], Seite 53, für Details.

3.11 Textformatierungscodes

Widgets vom Typ Textview-Klasse (Textanzeige) und Texteditor-Klasse unterstützen die Textformatierung, wenn das Attribut `Styled` dieser Klassen auf `True` gesetzt ist. RapaGUI unterstützt die Textformatierung durch spezielle Steuercodes, die in diesem Abschnitt beschrieben werden.

Formatierungscodes beginnen immer mit einem Escape-Zeichen, gefolgt von einer Zeichenfolge, die den Formatierungscode beschreibt. In der Dezimalschreibweise entspricht das Escape-Zeichen dem ASCII-Code 27, der in Oktalzahl 33 und in Hexadezimalschreibweise $1B ist. Bei der Verwendung von Formatierungscodes ist Vorsicht geboten, da sie in XML-Dateien und Hollywood-Quelldateien unterschiedlich verwendet werden. In XML-Dateien wird die Oktalnotation verwendet, d.h. sie starten eine Escape-Sequenz mit einem Backslash und der Oktalzahl 33 ('\33'). In Hollywood-Quellcodes werden Oktalzahlen nach einem umgekehrten Schrägstrich jedoch nicht unterstützt. Hollywood erwartet immer nach einem

umgekehrten Schrägstrich den ASCII-Code in Dezimalschreibweise. Deshalb müssen Sie '\27' verwenden, um eine Escape-Sequenz im Hollywood-Code zu initiieren.

Um diesen Unterschied etwas besser zu verdeutlichen, lassen Sie uns zwei Beispiele anschauen. Hier ist ein Beispiel für die Erstellung eines fettgedruckten Textobjekts in einer XML-Datei. Fettgedruckter Text wird aktiviert, indem nach dem Escape-Zeichen das Zeichen 'b' verwendet wird:

```
<textview styled="true" id="mytext">\33bBold text</textview>
```

Sie sehen, dass hier die Oktalschreibweise verwendet wird, da XML-Dateien nach einem umgekehrten Schrägstrich eine Oktalzahl erwarten. In Hollywood ist das anders, denn Hollywood erwartet ein Dezimalzeichen nach einem umgekehrten Schrägstrich. Beim nächsten Beispiel sehen Sie, wie Sie Escape-Codes angeben müssen, wenn Sie sie aus einer Hollywood-Quelldatei verwenden:

```
moai.Set("mytext", "text", "\27bBold text")
```

Sie können sehen, dass der Code derselbe ist, außer dass wir \27b anstelle von \33b verwenden, da Hollywood nach einem umgekehrten Schrägstrich immer Dezimalzahlen anstelle von Oktalzahlen verwendet.

Die folgenden Formatierungscodes werden derzeit unterstützt:

`\33u` Setzt den Textstil auf Unterstrichen.

`\33b` Setzt den Textstil auf Fett.

`\33i` Setzt den Textstil auf Kursiv.

`\33n` Setzt den Textstil zurück auf Normal.

`\33P[RRGGBB]`
: Ändern Sie die Frontfarbe in die angegebene RGB-Farbe. Die RGB-Farbe muss in Form von sechs Hexadezimalziffern RRGGBB angegeben werden. Auf AmigaOS und Kompatible erfordert dies MUI 4.0.

3.12 Implementierung von Hilfetexten

RapaGUI unterstützt mehrere Möglichkeiten, kontextsensitive Hilfetexte in Ihrem Programm zu implementieren. Zuerst einmal kann jedes Widgets Tooltips haben, die auftauchen, nachdem die Maus für einige Zeit über einem Widget schwebt. Dies geschieht über das Attribut `Area.Tooltip`. Da Area-Klasse (Bereich) die Oberklasse für alle Widgets ist, können Sie dieses Attribut verwenden, um Tooltip-Hilfe zu all Ihren Widgets hinzuzufügen. Hier ist ein Beispiel:

```
<vgroup>
   <listview tooltip="List of loaded video files">
      <column/>
   </listview>
   <button tooltip="Plays a video stream">Play</button>
   <button tooltip="Stops a video stream">Stop</button>
   <button tooltip="Exits the program.">Quit</button>
</vgroup>
```

Darüber hinaus unterstützen Menüeinträge und Symbolleistenschaltflächen die Attribute `Menuitem.Help` und `Toolbarbutton.Help`. Wenn in Ihrem Fenster eine Statusleiste angebracht ist, wird der Text, den Sie in diesen Attributen angeben, automatisch in der Statusleiste angezeigt, wenn sich der Mauszeiger über dem entsprechenden Menüeintrag oder der entsprechenden Schaltfläche der Symbolleiste befindet. Diese Art von visuellem Feedback ist sehr hilfreich, da er sofort in der Statusleiste angezeigt wird und der Benutzer nicht wie bei Tooltips einige Sekunden warten muss.

3.13 Kontextmenüs

RapaGUI unterstützt Kontextmenüs für jedes seiner Widget-Klassen. Kontextmenüs werden den einzelnen Widgets mithilfe des Attributs `Area.ContextMenu` zugewiesen. Da die Area-Klasse (Bereich) die Oberklasse für alle Widgets ist, können Sie dieses Attribut für jedes MOAI-Objekt verwenden, das ein Widget erstellt. Wenn Sie `Area.ContextMenu` für ein Widget definiert haben, zeigt RapaGUI automatisch das Kontextmenü an, wenn der Benutzer mit der rechten Maustaste klickt, während sich der Mauszeiger über einem Widget befindet, an das ein Kontextmenü angehängt ist.

`Area.ContextMenu` erwartet ein Objekt, das von der Menu-Klasse als Argument abgeleitet ist, so dass Sie zuerst ein solches MOAI-Objekt für Ihr Kontextmenü in XML erstellen müssen. Es ist sehr wichtig zu beachten, dass Sie Ihre Menüs im Bereich `<application>` definieren müssen, da Menüs globale Objekte sind und später nur an Fenster oder Widgets angehängt werden. Aus diesem Grund ist es nicht erlaubt, Menüs in einem Abschnitt `<window>` zu definieren.

Hier ist ein Beispiel, in dem wir ein Kontextmenü zum Ausschneiden, Kopieren und Einfügen in ein Objekt vom Typ Texteditor-Klasse einfügen:

```
<menu title="Context menu" id="ctxtmenu">
   <item>Cut</item>
   <item>Copy</item>
   <item>Paste</item>
</menu>

<window>
...
   <texteditor contextmenu="ctxtmenu"/>
...
</window>
```

Beachten Sie, dass bei der Verwendung der Menu-Klasse zur Erstellung von Kontextmenüs das Attribut `Menu.Title` nur unter AmigaOS und kompatiblen Betriebssystemen verwendet wird. Kontextmenüs unter Windows, Linux und Mac OS zeigen keinen Titel an.

Beachten Sie auch, dass Kontextmenü-Ereignisse über den standardmäßigen Mechanismus `Menuitem.Selected` und nicht über eine speziellen Kontextmenü-Callback-Funktion ausgeliefert werden. Da Sie dasselbe Menüobjekt als Kontextmenü für mehrere Widgets verwenden können, benötigen Sie eine Möglichkeit, das Widget herauszufinden, dessen Kontextmenü das Ereignis ausgelöst hat. Um Ihnen diese Informationen zu geben, enthält die Ereignismeldung einen zusätzlichen Eintrag mit dem Namen `"Parent"`, der die ID des Wid-

gets des Kontextmenüs enthält. Auf diese Weise können Sie dasselbe Menüobjekt als Kontextmenü für mehrere übergeordnete Widgets wiederverwenden.

AmigaOS-Benutzer beachten bitte, dass MUI keine Kontextmenü-Objekte in Fenstern freigeben kann. Sie dürfen nicht dasselbe Menüobjekt mit Widgets verwenden, die sich in verschiedenen Fenstern befinden. Ein Menüobjekt für mehrere Widgets im selben Fenster zu verwenden, funktioniert, aber nicht für Widgets in einem anderen Fenster. Alle anderen Plattformen haben diese Einschränkung nicht, nur AmigaOS ist hiervon betroffen.

3.14 Zeichenkodierung

RapaGUI unterstützt zwei Zeichenkodierungen in den XML-Dateien, die zur Beschreibung des GUI-Layouts verwendet werden: `iso-8859-1` und `utf-8`. Wenn Sie die Codierung `utf-8` unter AmigaOS und kompatiblen Betriebssystemen verwenden, benötigt RapaGUI die `codesets.library`, um Zeichenkonvertierungen von UTF-8 in den Standardzeichensatz des Systems durchzuführen.

Amiga-Anwender beachten bitte, dass MUI UTF-8 nicht unterstützt. MUI verwendet immer den Standardzeichensatz des Systems. Wenn Sie also UTF-8-Codierung in Ihren XML-Dateien verwenden, wird RapaGUI versuchen, diese Zeichenfolgen dem Standardzeichensatz des Systems zuzuordnen. Das wird nicht immer gelingen. Wenn beispielsweise der Standardzeichensatz des Systems ISO-8859-1 ist und die UTF-8 XML-Datei einige osteuropäische Zeichen verwendet, die in ISO-8859-1 nicht vorhanden sind, werden sie nicht korrekt dargestellt. Sie werden nur dann korrekt angezeigt, wenn sie auch im Standardzeichensatz des Systems enthalten sind.

3.15 Verbindung zu Hollywood

Ein mächtige Fähigkeit von RapaGUI ist die Möglichkeit, komplette Hollywood-Displays mit der Hollywood-Klasse in Ihre GUIs einzubetten. Immer wenn Sie etwas auf ein Hollywood-Display zeichnen, das mit der Hollywood-Klasse verbunden ist, wird es automatisch auch auf Ihrem GUI-Widget gezeichnet. Sie können sogar das Hollywood-Display ausblenden, so dass der Benutzer nicht einmal merkt, dass Hollywood im Hintergrund läuft. Darüber hinaus werden alle Mausklicks und Tastenanschläge, die innerhalb der Hollywood-Klasse stattfinden, als normale Hollywood-Ereignisse an das entsprechende Hollywood-Display weitergeleitet. So können Sie mit der Hollywood-Klasse fast alle leistungsstarken Funktionen von Hollywood auch in einem GUI-Widget nutzen.

Schauen wir uns ein Beispiel an. Der folgende Code verwendet die Hollywood-Klasse zum Einbetten von einer spielenden Animation der Größe 320x200 in einem GUI-Widget. Dafür wird die Größe des Hollywood-Displays auf 320x200 geändert und dann in die GUI als ein MOAI-Objekt vom Typ Hollywood-Klasse eingebettet. Hier zuerst die Definition der XML-GUI:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<application>
   <window title="Hollywood bridge">
      <vgroup>
         <hgroup>
            <rectangle/>
            <hollywood display="1"/>
```

```
            <rectangle/>
         </hgroup>
         <hgroup>
            <button id="play">Play</button>
            <button id="stop">Stop</button>
         </hgroup>
      </vgroup>
   </window>
</application>
```

Beachten Sie, dass hier MOAI-Objekte der Rectangle-Klasse (Rechteck) verwendet wurde, um das nicht größenveränderliche Hollywood-Objekt aufzufüllen. Dies ist notwendig, damit die GUI in der Größe veränderbar bleibt. Hier ist der Code, der Ihnen zeigt, wie Sie Ihr GUI-Widget mit Hollywood verbinden können:

```
@REQUIRE "RapaGUI"
@ANIM 1, "amy_walks.anim"
@DISPLAY {Width = 320, Height = 200}

Function p_AnimFunc()
   Local numframes = GetAttribute(#ANIM, 1, #ATTRNUMFRAMES)
   curframe = Wrap(curframe + 1, 1, numframes + 1)
   DisplayAnimFrame(1, 0, 0, curframe)
EndFunction

Function p_EventFunc(msg)
   Switch msg.Class
   Case "Button":
      Switch msg.Attribute
      Case "Pressed":
         Switch msg.ID
         Case "play":
            SetInterval(1, p_AnimFunc, 50)
         Case "stop":
            ClearInterval(1)
         EndSwitch
      EndSwitch
   EndSwitch
EndFunction

InstallEventHandler({RapaGUI = p_EventFunc})
moai.CreateApp(FileToString("GUI.xml"))

Repeat
   WaitEvent
Forever
```

Mit etwas mehr Arbeit können wir die Animation auch mit der Maus bewegen. Der Benutzer kann dann in das Hollywood-Objekt klicken und die Animation mit der Maus umherziehen. Hier ist der Code dafür:

```
@REQUIRE "RapaGUI"
@ANIM 1, "amy_walks.anim"
@DISPLAY {Width = 320, Height = 200}

Function p_AnimFunc()
   Local numframes = GetAttribute(#ANIM, 1, #ATTRNUMFRAMES)
   curframe = Wrap(curframe + 1, 1, numframes + 1)
   SelectBrush(1)
   Cls
   DisplayAnimFrame(1, offx, offy, curframe)
   EndSelect
   DisplayBrush(1, 0, 0)
EndFunction

Function p_MouseFunc()
   If IsLeftMouse() = True
     Local mx, my = MouseX(), MouseY()
     If (grabx = -1) And (graby = -1) Then
       grabx, graby = mx - offx, my - offy
     offx = mx - grabx
     offy = my - graby
   Else
     grabx, graby = -1, -1
   EndIf
EndFunction

Function p_EventFunc(msg)
   Switch msg.Class
   Case "Button":
      Switch msg.Attribute
      Case "Pressed":
         Switch msg.ID
         Case "play":
            SetInterval(1, p_AnimFunc, 50)
            SetInterval(2, p_MouseFunc, 20)
         Case "stop":
            ClearInterval(1)
            ClearInterval(2)
         EndSwitch
      EndSwitch
   EndSwitch
EndFunction
```

```
CreateBrush(1, 320, 200)

InstallEventHandler({RapaGUI = p_EventFunc})
moai.CreateApp(FileToString("GUI.xml"))

Repeat
   WaitEvent
Forever
```

Schließlich ist es auch möglich, das Hollywood-Objekt in der Größe zu verändern, indem man die Attribute `Area.FixWidth` und `Area.FixHeight` auf `False` setzt. Immer wenn der Benutzer die Fenstergröße ändert, erhält Ihr Hollywood-Display ein `SizeWindow`-Ereignis, das Sie mit dem Hollywood-Befehl `InstallEventHandler()` überwachen können. Wenn wir ein in der Größe veränderbares Hollywood-Objekt verwenden, können wir die beiden Objekte `<rectangle>` entfernen, die ausschließlich als Füllraum verwendet werden. Der XML-Code sieht dann so aus:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<application>
   <window title="Hollywood bridge">
      <vgroup>
         <hollywood display="1" fixwidth="false" fixheight="false"/>
         <hgroup>
            <button id="play">Play</button>
            <button id="stop">Stop</button>
         </hgroup>
      </vgroup>
   </window>
</application>
```

Unser Code ist so ziemlich derselbe wie vorher, mit der Ausnahme, dass wir jetzt das Ereignis `SizeWindow` überwachen müssen, um auf die Größenänderungen der GUI zu reagieren. Hier ist der angepasste Code von oben:

```
@REQUIRE "RapaGUI"
@ANIM 1, "amy_walks.anim"
@DISPLAY {Width = 320, Height = 200}

Function p_AnimFunc()
   Local numframes = GetAttribute(#ANIM, 1, #ATTRNUMFRAMES)
   curframe = Wrap(curframe + 1, 1, numframes + 1)
   SelectBrush(1)
   Cls
   DisplayAnimFrame(1, offx, offy, curframe,
     {Width = swidth, Height = sheight})
   EndSelect
   DisplayBrush(1, 0, 0)
EndFunction

Function p_MouseFunc()
```

```
   If IsLeftMouse() = True
     Local mx, my = MouseX(), MouseY()
     If (grabx = -1) And (graby = -1) Then
       grabx, graby = mx - offx, my - offy
     offx = mx - grabx
     offy = my - graby
   Else
     grabx, graby = -1, -1
   EndIf
EndFunction

Function p_EventFunc(msg)
   If msg.Action = "SizeWindow"
     swidth = msg.Width
     sheight = msg.Height
     CreateBrush(1, swidth, sheight)
     Return
   EndIf

   Switch msg.Class
   Case "Button":
      Switch msg.Attribute
      Case "Pressed":
         Switch msg.ID
         Case "play":
            SetInterval(1, p_AnimFunc, 50)
            SetInterval(2, p_MouseFunc, 20)
         Case "stop":
            ClearInterval(1)
            ClearInterval(2)
         EndSwitch
      EndSwitch
   EndSwitch
EndFunction

swidth, sheight = 320, 200
CreateBrush(1, swidth, sheight)

InstallEventHandler({RapaGUI = p_EventFunc, SizeWindow = p_EventFunc})
moai.CreateApp(FileToString("GUI.xml"))

Repeat
   WaitEvent
Forever
```

Bei diesem Beispiel können Sie sehen, dass die Hollywood-Klasse wirklich mächtig ist und dazu verwendet werden kann, viele innovative GUI-Ideen zu verwirklichen, die nur durch Ihre Kreativität begrenzt sind!

3.16 Bilder-Cache

Viele MOAI-Klassen erlauben es Ihnen, Symbole mit den von ihnen erstellten Widgets zu verwenden. Z.B. können Sie ein Symbol zu einem Schaltflächen-Widget hinzufügen, indem Sie einfach das Attribut `Button.Icon` verwenden. Alle MOAI-Klassen, die mit Symbolen umgehen können, benötigen einen Hollywood-Pinsel, der als Symbol für die Klasse verwendet werden soll.

Wichtig ist zu wissen, dass RapaGUI diese Symbole aus Performance- und Wirtschaftlichkeitsgründen zwischenspeichert. Stellen Sie sich eine Listenansicht mit Tausenden von Zeilen und einem Symbol in jeder Zeile vor. Es wäre ein absoluter Performance-Killer, wenn RapaGUI diese Symbole für jede einzelne Zeile von einem Hollywood-Pinsel in ein RapaGUI-Symbol umwandeln müsste. Deshalb werden die Symbole zwischengespeichert.

Schauen Sie sich dieses Beispiel an:

```
<button icon="1">OK</button>
```

In der obigen Definition steht Hollywood-Pinsel Nummer 1 neben dem Text "OK" für die Schaltfläche. RapaGUI wird nun die Bilddaten des Hollywood-Pinsels Nummer 1 intern zwischenspeichern und jedes Mal, wenn es wieder einen Verweis auf den Hollywood-Pinsel Nummer 1 gibt, wird einfach die Kopie aus seinem Cache genommen, anstatt den Originalpinsel. Das bedeutet, dass alle Änderungen, die Sie an Hollywood-Pinsel Nummer 1 nach der obigen Definition vornehmen, aufgrund des Bild-Caches keinerlei Auswirkungen auf RapaGUI haben werden! Bitte denken Sie daran. Sie müssen eine andere Pinselnummer verwenden, wenn Sie ein aktualisiertes Symbol anzeigen wollen, um RapaGUI's Bilder-Cache zu umgehen.

Es gibt nur eine Ausnahme von der Regel: Widgets, die von der Image-Klasse (Bild) abgeleitet sind, speichern keine Bilddaten, obwohl sie auch mit Hollywood-Pinseln arbeiten. Der Grund dafür ist, dass Bild-Widgets oft ziemlich große Bilder anzeigen, die auch häufig animiert oder geändert werden können. Aus diesem Grund ist es nicht sinnvoll, Bilddaten für Widgets der Image-Klasse zu cachen. In anderen Widgets verwendete Symbole, z.B. Schaltflächen, Symbolleistenschaltflächen, Listen-, Baum-, Seitenansichten sind normalerweise sehr klein und es gibt im normalen Fall nur eine feste Anzahl von Symbolen. Obendrein werden sie vielleicht sehr oft benötigt. Man denke nur an eine Listenansicht mit Tausenden von Zeilen und einem Symbol in jeder dieser Zeilen. Deshalb ist hier wirklich ein Bilder-Cache notwendig, um eine gute Performance zu garantieren.

3.17 Plattformabhängige Fähigkeiten

Generell versucht RapaGUI so wenig plattformabhängige Besonderheiten wie möglich einzubauen, damit die gleiche Anwendung auf einer Vielzahl von Plattformen ohne Anpassungen laufen kann. Es gibt jedoch sehr wenige Attribute und Methoden, die nicht auf allen Plattformen verfügbar sind, aber dennoch enthalten sind. Dies weil sie als so wichtig erachtet wurden, dass sie aufgenommen wurden, obwohl sie in gewissem Widerspruch zu RapaGUIs plattformblindem GUI-Toolkit-Ansatz stehen.

Zum Beispiel ist das Attribut `Window.HideFromTaskBar` nur unter Windows und GTK+ verfügbar, da es unter AmigaOS und Mac OS keine Taskleiste gibt. Ein weiteres Beispiel ist `Application.OpenConfigWindow`, das nur unter AmigaOS und kompatiblen Betriebssystemen verfügbar ist.

Plattformspezifische Fähigkeiten werden immer als solche dokumentiert. Sie müssen nur in der Dokumentation nachsehen, ob eine bestimmte Fähigkeit plattformspezifisch ist. Um plattformspezifischen Code zu minimieren, können alle plattformabhängigen Fähigkeiten auch auf allen von RapaGUI unterstützten Plattformen spezifiziert werden. RapaGUI ignoriert dann einfach die Attribute und Methoden, anstatt eine Fehlermeldung zu melden. So könnte man `Window.HideFromTaskBar` auch auf AmigaOS und Mac OS setzen. RapaGUI meldet keine Fehlermeldung. Der Aufruf wird einfach ignoriert.

3.18 Kompatibilität mit MUI Royale

RapaGUI begann als eine Abzweigung des populären MUI Royale Plugins für Hollywood. Leute, die mit MUI Royale vertraut sind, werden ohne Zweifel sehen, dass RapaGUI das gleiche Design benutzt, so dass die Portierung von Programmen von MUI Royale nach RapaGUI nicht besonders schwierig ist. Obwohl RapaGUI aufgrund der AmigaOS-basierten Terminologie viele semantische Änderungen erfahren hat, sind die Designunterschiede tatsächlich sehr gering. Wenn Sie also Programme von MUI Royale nach RapaGUI portieren, werden Sie sich nur mit Namensänderungen beschäftigen. Die meisten Leute werden wahrscheinlich nicht einmal auf einen der wenigen Design-Unterschiede zwischen MUI Royale und RapaGUI treffen. Dennoch ist hier eine nicht vollständige Liste von Designunterschieden zwischen MUI Royale und RapaGUI, um Ihre Portierungsbemühungen zu vereinfachen:

- MUI Royale verlangte von Ihnen, dass Sie `Window.CloseRequest` für alle Ihre Fenster verwenden, sonst würden sie offen bleiben. RapaGUI schließt automatisch Fenster, die keine Überwachung für `Window.CloseRequest` installiert haben.
- MUI Royale verlangte auch, dass Sie die Ereignisse "HideWindow" und "ShowWindow" überwachten, um die Minimierung von Fenstern zu ermöglichen. RapaGUI übernimmt dies nun automatisch.
- RapaGUI richtet automatisch für Schaltflächen, Symbolleistenschaltflächen und Menüeinträge Ereignisüberwachung ein. MUI Royale verlangte von Ihnen, dass Sie diese explizit anfordern, aber dies führte zu unnötig wortreichen XML-Dateien, in denen Sie so etwas wie `notify="pressed"` für jede einzelne Schaltfläche einfügen mussten, weil Sie unter normalen Umständen über jeden Schaltflächen-Klick benachrichtigt werden möchten. RapaGUI ruft automatisch Ihre Callback-Funktion für alle Schaltflächen- und Symbolleisten-Schaltflächen-Klicks (Toolbarbutton) und Menüeintrag-Ereignisse auf, um den XML-Code besser lesbar zu halten.
- RapaGUI blendet automatisch alle Hollywood-Displays beim Start aus, während dies bei MUI Royale nicht der Fall ist. Bei MUI Royale müssen Sie die Standard-Displays manuell als verborgen definieren.
- MUI Royale ermöglicht es Ihnen, fast überall Textformatierungscodes zu verwenden. Da es sich hierbei um eine hochgradig MUI-zentrierte Fähigkeit handelt, ist es unmöglich, diese plattformunabhängig zu implementieren. RapaGUI unterstützt Textformatierung nur in den Klassen Textview (Textanzeige) sowie Texteditor und nur dann, wenn `Textview.Styled` sowie `Texteditor.Styled` auf `True` gesetzt wurden.

- MUI Royale erlaubte es Ihnen auch, Symbole in fast allen Widgets über Textformatierungscodes einzubinden. Dies ist auch nicht plattformübergreifend realisierbar. Dennoch hat RapaGUI Symbol-Unterstützung in vielen Klassen, aber die Art und Weise, wie dies implementiert wird, ist immer klassenabhängig. Mit RapaGUI können Sie Symbole mit folgenden Klassen verwenden: Button-Klasse, Listview-Klasse, Pageview-Klasse und Treeview-Klasse (Schaltflächen, Listenansicht, Seitenansicht und Baumansicht). Schauen Sie sich die Dokumentation dieser Klassen an, um mehr darüber zu erfahren, wie man Symbole mit ihnen verwendet.
- MUI Royale verlangt, dass Sie ein von der Klasse `<virtgroup>` abgeleitetes Objekt als Element für `<scrollgroup>`-Objekte verwenden. Mit RapaGUI können Sie ganz normale Gruppen verwenden.
- MUI Royale sendet alle Kontextmenüs-Ereignisse immer über das Attribut `Area.ContextMenuTrigger`. RapaGUI sendet Kontextmenü-Ereignisse einfach als normale Menüpunkt-Ereignisse. Aber es enthält ein Feld `Parent` in der Ereignismeldung, um Sie über das Widget zu informieren, dessen Kontextmenü das Ereignis ausgelöst hat.
- RapaGUI's Checkbox-Klasse erzeugt ein Auswahlkästchen mit einer Textbeschriftung (Label). MUI Royale's Checkmark-Klasse erstellt einfach ein Auswahlkästchen-Bild und Sie müssen die Beschriftung (Label) selbst erstellen.
- RapaGUI hat keine Unterstützung für das Attribut `Listview.InsertPosition`, aber die Methode `Listview.Insert` gibt einfach die Position des neu eingefügten Eintrags zurück.
- In MUI Royale wurden Texteditor-Bereiche durch Blockkoordinaten angegeben, die aus vier verschiedenen Werten (x1, y1, x2, y2) bestehen. RapaGUI verwendet jetzt nur noch Start- und Stoppkoordinaten, so dass alle Methoden und Attribute, die sich mit Textbereichen befassen, jetzt nur noch zwei Argumente benötigen/zurückgeben.
- RapaGUIs Texteditor-Klasse, Textview-Klasse (Textanzeige) und Textentry-Klasse (Texteingabe) verwenden den Inhalt zwischen den öffnenden und schließenden XML-Tags als initialen Widget-Inhalt, während Sie mit MUI Royale immer ein Attribut verwenden müssen, um den ursprünglichen Inhalt festzulegen.
- RapaGUI verlangt, dass Sie ein übergeordnetes Objekt angeben, wenn Sie Objekte dynamisch mit `moai.CreateObject()` erstellen. Das war bei MUI Royale nicht nötig.
- Da RapaGUI's Treeview-Klasse (Baumansicht) mehrere Spalten unterstützt, müssen Sie bei der Definition Ihrer Baumansicht in XML mindestens einen `<column>` erstellen. Dies war bei MUI Royale nicht nötig, da die Treeview-Klasse nur einzelne Spaltenbäume unterstützt.
- MUI Royale hat kein Konzept mit Dialogen, da AmigaOS auch keines hat. Modale Dialoge wurden nur mit normalen Fenstern emuliert und alle anderen Fenster wurden in den Ruhezustand versetzt. RapaGUI führt jetzt echte Dialoge ein. Während Sie bei MUI Royale normalerweise alle diese Pseudo-Dialoge beim Start erstellen, wird dies mit RapaGUI nicht mehr empfohlen. Da Fenster auf manchen Betriebssystemen eine recht begrenzte Ressource ist, sollten Sie Dialoge nur dann erstellen, wenn Sie sie benötigen und sie nachher wieder sofort löschen. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details. Wenn Sie nur AmigaOS verwenden möchten, dann können Sie natürlich alle Ihre Fenster mit nur einem einzigen Aufruf von `moai.CreateApp()` erstellen, aber

wenn Sie andere Platformen auch berücksichtigen möchten, sollten Sie die empfohlene Programmierrichtlinie befolgen und Dialoge nur dann erstellen, wenn Sie sie benötigen und diese sofort nach dem Schliessen wieder löschen.

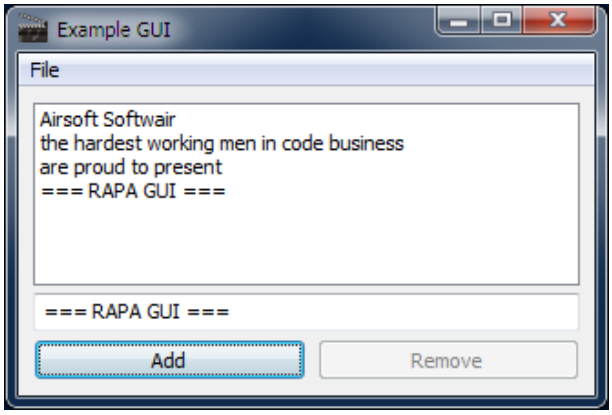
- Objekte der Radio-Klasse werden immer in RapaGUI gerahmt. In MUI Royale werden sie standardmäßig nicht gerahmt.
- Wenn Sie einen etwas komplexere Tastaturkürzel-Zeichenfolge wie `CTRL+V`, `Alt+X` oder `F5` mit `Menuitem.CommandString` verwenden, überwacht diese MUI Royale nicht automatisch. Dies muss manuell durchgeführt werden. Nur Standardabkürzungen in Form einer einzelnen alphabetischen Taste, kombiniert mit der Taste `CMD`, werden von MUI Royale automatisch überwacht. RapaGUI verarbeitet jedoch auch komplexe Verknüpfungen automatisch.
- Wenn `Listview.Active` abgerufen wird und kein Eintrag aktiv ist, gibt MUI Royale die spezielle Zeichenfolge `Off` zurück, während RapaGUI -1 zurückgibt.

Natürlich gibt es auch viele Attribute und Methoden, die in MUI Royale vorhanden sind, aber in RapaGUI fehlen. Der Grund dafür ist natürlich, dass RapaGUI aufgrund seines plattformübergreifenden Designs den kleinsten gemeinsamen Nenner zwischen Windows, Linux, Mac OS und AmigaOS darstellt. Deshalb konnten nicht alle Fähigkeiten von MUI Royale auf RapaGUI übertragen werden.

4 Tutorial

4.1 Tutorial

Willkommen zum RapaGUI-Tutorial! Dieses kleine Schritt-für-Schritt-Dokument führt Sie in wenigen Schritten durch den Prozess der Erstellung Ihrer ersten GUI mit RapaGUI. Wir werden eine kleines GUI-Programm erstellen, welches aus einer Listenansicht, einem Texteingabe-Widget und zwei Schaltflächen besteht (Listview, Textentry, Button). Die beiden Schaltflächen sollten so programmiert werden, dass sie das Hinzufügen und Entfernen von Einträgen aus der Listenansicht ermöglichen. Die Daten, die in die Listenansicht eingefügt werden sollen, sind dem Texteingabe-Widget zu entnehmen. So sieht dieses kleine Programm unter Windows aus:



Beginnen wir mit den Grundlagen: In RapaGUI werden GUIs mit Hilfe von XML-Dateien erstellt, die eine Beschreibung einer Reihe von Fenstern enthalten, die eine Vielzahl von GUI-Widgets beinhalten. Hier ist eine minimale GUI-Beschreibung für eine RapaGUI-GUI, die ein Fenster mit einer Listenansicht, einem Texteingabe-Widget und zwei Schaltflächen im XML-Format enthält:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<application>
   <window title="Example GUI">
      <vgroup>
         <listview>
            <column/>
         </listview>
         <textentry/>
         <hgroup>
            <button>Add</button>
            <button>Remove</button>
         </hgroup>
      </vgroup>
   </window>
</application>
```

Das Objekt `<application>` ist das Haupt-MOAI-Objekt für jede Anwendung und darf nur einmal pro Anwendung verwendet werden. Alle anderen MOAI-Objekte sind untergeordnete Elemente der Application-Klasse. Das Wurzelelement jedes `<window>`-Objekts muss von der Group-Klasse (Gruppen) abgeleitet sein, d.h. es muss entweder `<vgroup>`, `<hgroup>` oder `<colgroup>` sein. Beachten Sie auch, dass pro Fenster nur ein Wurzelelement vorhanden sein muss.

Um zu sehen, wie unsere obige XML-Definition als GUI aussieht, müssen wir sie in einer Datei namens `GUI.xml` speichern und dann den folgenden Code verwenden, damit RapaGUI sie in eine native GUI für das Betriebssystem konvertiert, auf dem Hollywood läuft:

```
@REQUIRE "RapaGUI"

moai.CreateApp(FileToString("GUI.xml"))

Repeat
   WaitEvent
Forever
```

Als Nächstes sollten Sie einige Informationen über Ihr Programm hinzufügen, indem Sie die Präprozessor-Anweisungen `@APPAUTHOR`, `@APPCOPYRIGHT`, `@APPDESCRIPTION`, `@APPTITLE` und `@APPVERSION` verwenden. RapaGUI benötigt diese Informationen für verschiedene Zwecke, z.B. werden die Informationen, die in `@APPTITLE` übergeben werden, auch vom MUI-Einstellungsfenster auf AmigaOS und kompatiblen Systemen verwendet. Hier ist eine Beispieldefinition dieser Präprozessor-Anweisungen:

```
@APPTITLE "Tutorial"
@APPVERSION "$VER: Tutorial 1.0 (29.12.15)"
@APPCOPYRIGHT "Copyright ©2015, Andreas Falkenhahn"
@APPAUTHOR "Andreas Falkenhahn"
@APPDESCRIPTION "The tutorial app from the RapaGUI guide"
```

Als nächstes müssen wir eine Callback-Funktion mit Hilfe des Hollywood-Befehls `InstallEventHandler()` installieren, da unser Hollywood-Skript jedes Mal informiert werden muss, wenn ein RapaGUI-Ereignis eintrifft. Daher müssen wir unseren Code wie folgt modifizieren:

```
@REQUIRE "RapaGUI"

Function p_EventFunc(msg)
   ; contents follow below
EndFunction

InstallEventHandler({RapaGUI = p_EventFunc})
moai.CreateApp(FileToString("GUI.xml"))

Repeat
   WaitEvent
Forever
```

Was wir hier getan haben, ist die Installation der Funktion `p_EventFunc` als Callback-Funktion, die immer dann ausgeführt wird, wenn ein RapaGUI-Ereignis eintritt. Wenn wir

ein solches Ereignis erhalten, müssen wir dann prüfen, welche MOAI-Klasse und welches Attribut es ausgelöst hat. Dies geschieht, indem man die Felder `msg.Class` und `msg.Attribute` der Ereignismeldung überprüft, die unsere Callback-Funktion als ersten Parameter erhält.

Als nächstes wollen wir die Funktionalität hinzufügen, dass der Text im Texteingabe-Widget immer dann als letzter Eintrag in die Listenansicht eingefügt wird, wenn der Benutzer auf die Schaltfläche `"Add"` (Hinzufügen) klickt. Dazu müssen wir zunächst einen Weg finden, unsere Widgets aus dem Hollywood-Skript zu identifizieren. Dies geschieht durch die Angabe von IDs in der XML-Definition. IDs sind einfache Textzeichenfolgen, die für die Kommunikation mit MOAI-Objekten aus Hollywood-Skripten verwendet werden. Fügen wir also jetzt einige IDs für alle Widgets hinzu, mit denen wir kommunizieren wollen. Wir müssen unsere XML-Definition so modifizieren:

```
...
         <listview id="mylistview">
            <column/>
         </listview>
         <textentry id="mystring"/>
         <hgroup>
            <button id="mybt1">Add</button>
            <button id="mybt2">Remove</button>
         </hgroup>
...
```

Nun, da wir dies erledigt haben, können wir etwas Code zu unserer Callback-Funktion hinzufügen, die den Inhalt des Texteintrag-Widgets erfasst und am Ende der Liste in unserem Listview-Objekt (Listenansicht) hinzufügt. Dies geschieht, indem man zuerst `moai.Get()` für das Attribut `Textentry.Text` aufruft, um den Inhalt des Texteintrag-Widgets zu erhalten und dann die Methode `Listview.Insert` ausführt. Fügen Sie mit `moai.DoMethod()` den Eintrag in die Listenansicht ein. Hier ist der Code, der zu diesem Zweck in die Funktion `p_EventFunc` eingefügt werden muss:

```
Switch msg.Class
...
Case "Button":
   Switch msg.Attribute
   Case "Pressed":
      Switch msg.ID
      Case "mybt1":    ; "Add" button was pressed
         Local s$ = moai.Get("mystring", "text")
         moai.DoMethod("mylistview", "insert", "bottom", s$)
      EndSwitch
   EndSwitch
EndSwitch
```

Als nächstes wollen wir die Funktionalität unserer Schaltfläche `"Remove"` (Entfernen) implementieren. Immer wenn diese Schaltfläche gedrückt wird, soll der aktive Eintrag aus der Listenansicht entfernt werden. Wir können dies tun, indem wir die Methode `Listview.Remove` in der Listenansicht ausführen. Daher müssen wir unseren Code so modifizieren:

```
         Switch msg.ID
```

```
            ...
            Case "mybt2":    ; "Remove" button was pressed
               moai.DoMethod("mylistview", "remove", "active")
            EndSwitch
```

Nun wollen wir, dass der aktive Eintrag der Listenansicht automatisch im Texteingabe-Element angezeigt wird. Dazu müssen wir eine Benachrichtigung für das Attribut `Listview.Active` einrichten, die immer dann ausgelöst wird, wenn sich der aktive Eintrag der Listenansicht ändert. Daher müssen wir unsere XML-Datei so modifizieren:

```
...
            <listview id="mylistview" notify="active">
               <column/>
            </listview>
...
```

In unserer Callback-Funktion können wir diese Funktionalität ganz einfach implementieren, indem wir die Methode `Listview.GetEntry` ausführen und dann den Inhalt des Texteingabe-Widgets mit dem Attribut `Textentry.Text` setzen. Hier ist der Code dafür:

```
Switch msg.Class
...
Case "Listview":
   Switch msg.Attribute
   Case "Active":
      Local s$ = moai.DoMethod("mylistview", "getentry", "active")
      moai.Set("mystring", "text", s$)
   EndSwitch
EndSwitch
```

Wenn Sie diesen Code ausprobieren, werden Sie feststellen, dass das Attribut `Listview.Active` nicht nur dann ausgelöst wird, wenn der Benutzer mit der Maus einen neuen Listeneintrag auswählt, sondern auch dann, wenn Einträge aus der Listenansicht entfernt werden und somit ein neuer Eintrag aktiv wird.

Als nächstes wollen wir die Schaltfläche "Remove" deaktivieren, wenn kein Eintrag in der Listenansicht aktiv ist. Wir können Widgets deaktivieren, indem wir das Attribut `Area.Disabled` auf `True` setzen. Da es zunächst keine Einträge in der Listenansicht gibt, müssen wir `Area.Disabled` bereits beim Start unseres Programms auf `True` setzen. Also müssen Sie diesen Code eingeben:

```
...
moai.CreateApp(FileToString("GUI.xml"))
moai.Set("mybt2", "disabled", True)
...
```

Nun müssen wir einige Änderungen an unserer Callback-Funktion vornehmen. Wann immer wir die Benachrichtigung mit `Listview.Active` erhalten, müssen wir prüfen, ob sie sich von dem speziellen Wert "Off" unterscheidet. Wenn dies der Fall ist, aktivieren wir die Schaltfläche "Remove". Der spezielle Wert "Off" (-1) wird von `Listview.Active` zurückgegeben, wenn in der Listenansicht kein aktiver Eintrag vorhanden ist. Wir müssen unseren Code folgendermaßen anpassen:

```
Switch msg.Class
```

```
...
Case "Listview":
   Switch msg.Attribute
   Case "Active":
      Local s$ = moai.DoMethod("mylistview", "getentry", "active")
      moai.Set("mystring", "text", s$)
      moai.Set("mybt2", "disabled", IIf(msg.triggervalue = -1,
          True, False))
   EndSwitch
EndSwitch
```

Wir verwenden hier das Feld `msg.TriggerValue`. Dieser enthält immer den aktuellen Wert des Attributs, welches das Ereignis ausgelöst hat, d.h. in unserem Fall den aktuellen Wert des Attributs `Listview.Active`. Wir könnten auch `moai.Get()` zuerst manuell auf `Listview.Active` aufrufen, aber das ist nicht wirklich notwendig, da wir einfach `msg.TriggerValue` verwenden können.

Als Letztes fügen wir ein Menü zu unserer GUI hinzu. Alle RapaGUI-Programme sollten über einen Menüpunkt verfügen, der den Benutzer darüber informiert, dass dieses Programm mit RapaGUI erstellt wurde. Sie können diesen Dialog öffnen, indem Sie die Methode `Application.AboutRapaGUI` ausführen. Um dieses Menü unserem Programm hinzuzufügen, verwenden wir die Menubar-Klasse (Menüleisten). Hier ist der XML-Code, den Sie vor Ihrer Fensterdefinition hinzufügen müssen:

```
...
<application>
   <menubar id="mymenubar">
      <menu title="File">
         <item id="menabout">About...</item>
         <item id="menaboutrapagui">About RapaGUI...</item>
         <item/>
         <item id="menquit">Quit</item>
      </menu>
   </menubar>
   ...
</application>
```

Nachdem wir unser Menüleisten-Objekt mit Hilfe des obigen XML-Codes erstellt haben, müssen wir diese Menüleiste an unserem Fenster anhängen. Dies geschieht durch das Setzen des Attributs `Window.Menubar` auf unser Menüleisten-Objekt. Hier ist der XML-Code dafür:

```
<window title="Example GUI" menubar="mymenubar">
```

Nun müssen wir unserer Callback-Funktion Code hinzufügen, der die entsprechende Aktion ausführt, wenn ein Menüpunkt ausgewählt wird. Bevor wir das tun können, müssen wir jedoch unserem Applications-Objekt eine ID zuweisen, damit wir `moai.DoMethod()` verwenden können. Hier sehen Sie, wie der XML-Code angepasst werden muss:

```
<application id="app">
```

Jetzt können wir den Code für unsere Callback-Funktion schreiben, welche die Menüeinträge berücksitigt:

```
   Switch msg.Class
```

```
...
Case "Menuitem":
   Switch msg.Attribute
   Case "Selected":
      Switch msg.id
      Case "menabout":
         moai.Request("Test", "Test program\n" ..
             "© 2015 by Andreas Falkenhahn", "OK")
      Case "menaboutrapagui":
         moai.DoMethod("app", "aboutrapagui")
      Case "menquit":
         End
      EndSwitch
   EndSwitch
EndSwitch
```

Einige letzte Änderungen an unserem Programm könnten darin bestehen, Online-Hilfe zu unseren Widgets hinzuzufügen, indem man die Attribute `Area.Tooltip` und `Menuitem.Help` verwendet, eine Statusleiste und Tastaturkürzel für die Menüeinträge mit dem Unterstrichzeichen hinzufügt. Diese Änderungen werden dem Leser als Übung überlassen.

So sieht unser endgültiges Programm aus. Zuerst die XML-Datei:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<application id="app">
   <menubar id="mymenubar">
      <menu title="File">
         <item id="menabout">About...</item>
         <item id="menaboutrapagui">
            About RapaGUI...</item>
         <item/>
         <item id="menquit">Quit</item>
      </menu>
   </menubar>
   <window title="Example GUI" menubar="mymenubar">
      <vgroup>
        <listview id="mylistview" notify="active">
            <column/>
         </listview>
         <textentry id="mystring"/>
         <hgroup>
            <button id="mybt1">Add</button>
            <button id="mybt2">Remove</button>
         </hgroup>
      </vgroup>
   </window>
</application>
```

Und hier ist der Code für die Programmlogik:

```
@REQUIRE "RapaGUI"

@APPTITLE "Tutorial"
@APPVERSION "$VER: Tutorial 1.0 (29.12.15)"
@APPCOPYRIGHT "Copyright ©2015, Andreas Falkenhahn"
@APPAUTHOR "Andreas Falkenhahn"
@APPDESCRIPTION "The tutorial app from the RapaGUI guide"

Function p_EventFunc(msg)

   Switch msg.Class
   Case "Button":
      Switch msg.Attribute
      Case "Pressed":
         Switch msg.ID
         Case "mybt1":    ; "Add" button was pressed
            Local s$ = moai.Get("mystring", "text")
            moai.DoMethod("mylistview", "insert", "bottom", s$)
         Case "mybt2":    ; "Remove" button was pressed
            moai.DoMethod("mylistview", "remove", "active")
         EndSwitch
      EndSwitch

   Case "Listview":
      Switch msg.Attribute
      Case "Active":
         Local s$ = moai.DoMethod("mylistview", "getentry", "active")
         moai.Set("mystring", "text", s$)
         moai.Set("mybt2", "disabled", IIf(msg.triggervalue = -1,
             True, False))
      EndSwitch

   Case "Menuitem":
      Switch msg.Attribute
      Case "Selected":
         Switch msg.id
         Case "menabout":
            moai.Request("Test", "Test program\n" ..
                "© 2015 by Andreas Falkenhahn", "OK")
         Case "menaboutrapagui":
            moai.DoMethod("app", "aboutrapagui")
         Case "menquit":
            End
         EndSwitch
      EndSwitch
   EndSwitch
```

```
EndFunction

InstallEventHandler({RapaGUI = p_EventFunc})
moai.CreateApp(FileToString("GUI.xml"))
moai.Set("mybt2", "disabled", True)

Repeat
   WaitEvent
Forever
```

Das ist es! Jetzt sollten Sie in der Lage sein, fantastische neue GUI-Programme mit Hollywood und RapaGUI zu erstellen. Vielen Dank für das Lesen dieses Tutorials und genießen Sie die Leistungsfähigkeit moderner GUI-Programmierung mit Hollywood und RapaGUI in Ihren Händen!

5 Beispiele

5.1 Beispiele

RapaGUI enthält eine Reihe von Beispielen, die bestimmte Fähigkeiten demonstrieren und Ihnen einen schnellen Einstieg ermöglichen sollen. Hier ist eine Liste von Beispielen, die mit RapaGUI mitgeliefert werden:

Accelerators
: Zeigt, wie man eine Tastaturkürzel-Tabelle für die erweiterte Tastatursteuerung der GUI einrichtet. Siehe Abschnitt 7.1 [Accelerator-Klasse], Seite 53, für Details.

Demo
: Eine umfangreiche Präsentation der meisten von RapaGUI unterstützten Widgets. Schauen Sie sich dieses Beispiel an, um zu sehen, wie die verschiedenen MOAI-Klassen von RapaGUI als Widgets erscheinen.

Dialogs
: Zeigt, wie man modale Dialoge mit RapaGUI verwendet. Dieses Beispiel demonstriert sowohl funktionsgesteuerte Dialoge als auch normale modale Dialoge. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details.

DragNDrop
: Zeigt, wie man Ziehen und Fallenlassen mit RapaGUI implementiert.

Dynamic1
: Demonstriert, wie zur Laufzeit neue Fenster erstellt werden.

Dynamic2
: Demonstriert, wie zur Laufzeit neue Schaltflächen erstellt werden.

Dynamic3
: Zeigt, wie man zur Laufzeit neue Registerkarten für die Pageview-Klasse (Seitenansicht) erstellt. Siehe Abschnitt 31.1 [Pageview-Klasse], Seite 157, für Details.

Dynamic4
: Zeigt, wie man zur Laufzeit Menüs und Menüeinträge erstellt.

HTMLView
: Dies ist ein rudimentärer Internet-Browser, der in RapaGUI und Hollywood geschrieben wurde. Er demonstriert die HTMLview-Klasse. (HTMLAnsicht) Siehe Abschnitt 21.1 [HTMLview-Klasse], Seite 105, für Details.

Image
: Dieses Beispiel zeigt, wie man einen Hollywood-Pinsel mit der Image-Klasse (Bild) in die GUI einbetten kann. Siehe Abschnitt 22.1 [Image-Klasse], Seite 109, für Details.

MultiDisplays
: Demonstriert, wie man vier verschiedene Hollywood-Displays in ein einziges RapaGUI-Fenster einbetten kann. Dies zeigt die Magie, die durch die Kombination von Hollywood und RapaGUI möglich ist. Siehe Abschnitt 19.1 [Hollywood-Klasse], Seite 101, für Details.

Pages
: Zeigt, wie man ein Fenster mit Registergruppen mit dem MOAI-Objekt Pageview-Klasse erstellt. Siehe Abschnitt 31.1 [Pageview-Klasse], Seite 157, für Details.

Scrollbar Demonstriert, wie eine mit der Scrollbar-Klasse erstellte Bildlaufleiste an ein benutzerdefiniertes Widget angehängt wird. Siehe Abschnitt 39.1 [Scrollbar-Klasse], Seite 179, für Details. Alternativ können Sie auch Scrollcanvas-Klasse (Bildlaufleinwand) verwenden, um dasselbe zu erreichen. Siehe unten für ein Beispiel.

Scrollcanvas
Dieses Beispiel demonstriert eine andere Art der Verwendung von Bildlaufleisten und Widgets mit benutzerdefinierten Grafiken in RapaGUI. Diesmal wird die Scrollcanvas-Klasse verwendet, um zu sehen, dass das Gleiche möglich wie mit der Scrollbar-Klasse ist. Siehe Abschnitt 40.1 [Scrollcanvas-Klasse], Seite 183, für Details. Siehe oben für ein Beispiel.

ShowHide Zeigt, wie MOAI-Objekte zur Laufzeit ein- und ausgeblendet werden können, indem das Attribut `Area.Hide` gesetzt wird.

SongPlayer
Dies ist ein Audiodatei-Player, der mit RapaGUI geschrieben wurde.

TextEditor
Ein kleiner Texteditor, der in Hollywood geschrieben wurde. Außerdem wird gezeigt, wie man eine Symbolleiste und eine Statusleiste verwendet.

Treeview Zeigt, wie man in RapaGUI mit der Treeview-Klasse eine mehrspaltige Baumansicht erstellt. Siehe Abschnitt 19.1 [Hollywood-Klasse], Seite 101, für Details.

VideoPlayer
Dies ist ein mit RapaGUI geschriebener Video-Player. Er zeigt, wie ein Hollywood-Display mit der Hollywood-Klasse in ein RapaGUI-Fenster eingebettet wird. Siehe Abschnitt 19.1 [Hollywood-Klasse], Seite 101, für Details. Darüber hinaus wird gezeigt, wie Sie Bildschaltflächen, eine Listenansicht mit Bildern und Schieberegler-Objekte verwenden.

6 Befehlsreferenz

6.1 moai.CreateApp

BEZEICHNUNG

moai.CreateApp – erstellt ein Application-Objekt aus einer XML-Quelle

ÜBERSICHT

`moai.CreateApp(xml$)`

BESCHREIBUNG

Dieser Befehl erstellt eine GUI aus der XML-Beschreibung, die im Argument `xml$` übergeben wird. Bitte beachten Sie, dass `xml$` eine Zeichenkette sein muss, die die XML-GUI-Definition und nicht einen Dateinamen enthält. Wenn Sie eine XML-GUI-Definition aus einer externen Datei verwenden wollen, müssen Sie diese Datei zuerst in eine Zeichenkette konvertieren, z.B. mit dem Hollywood-Befehl `FileToString()`.

Sobald dieser Befehl die GUI erstellt hat, können Sie auf allen MOAI-Objekten, die Sie in der XML-GUI-Definition beschrieben haben, die Befehle `moai.Set()`, `moai.Get()` und `moai.DoMethod()` anwenden.

Beachten Sie, dass es in einem Programm nur ein GUI-Application-Objekt geben und somit dieser Befehl nur einmal aufgerufen werden kann. Wenn Sie `moai.CreateApp()` ein zweites Mal aufrufen wollen, müssen Sie zuerst die alte GUI mit dem Befehl `moai.FreeApp()` löschen.

Wenn Sie Objekte wie Fenster oder Schaltflächen dynamisch zu Ihrem GUI-Application-Objekt hinzufügen müssen, können Sie dazu den Befehl `moai.CreateObject()` verwenden. Für Dialoge können Sie den Befehl `moai.CreateDialog()` benutzen.

EINGABEN

`xml$` eine Zeichenkette, die eine XML-GUI-Definition enthält

BEISPIEL

```
moai.CreateApp([[
<?xml version="1.0" encoding="iso-8859-1"?>
<application>
   <window title="Test program">
      <vgroup>
         <button>Hello World!</button>
      </vgroup>
   </window>
</application>
]])

InstallEventHandler({RapaGUI = Function(msg)
      If msg.attribute = "Pressed" Then DebugPrint("Button pressed!")
   EndFunction})

Repeat
```

```
   WaitEvent
Forever
```

Der obige Code erzeugt eine minimale GUI mit nur einem Fenster und einer einzigen Schaltfläche.

6.2 moai.CreateDialog

BEZEICHNUNG

moai.CreateDialog – erstellt ein Dialogobjekt aus einer XML-Quelle

ÜBERSICHT

`moai.CreateDialog(xml$[, parent$])`

BESCHREIBUNG

Mit diesem Befehl können Sie dynamisch ein Dialogobjekt aus einer XML-Quelle erzeugen. Das neu erstellte Dialogobjekt wird ebenfalls automatisch an Ihr Programm angehängt, so dass es sofort einsatzbereit ist. Sie sollten auch ein übergeordnetes Fenster für Ihren Dialog mit dem optionalen Argument `parent$` angeben.

`moai.CreateDialog()` ist sehr wichtig, da Sie Dialoge nur dann erstellen sollten, wenn Sie sie brauchen und sie aus dem Speicher löschen, sobald Sie den Dialog beendet haben. Es wird nicht empfohlen, alle Ihre Dialoge beim Start mit `moai.CreateApp()` zu erstellen und sie ständig im Speicher zu behalten. Stattdessen sollten Sie einen Dialog nur dann erstellen, wenn Sie ihn benötigen und ihn dann löschen, sobald Sie ihn beendet haben. Der Grund dafür ist, dass Fenster auf einigen von RapaGUI unterstützten Betriebssystemen eine begrenzte Ressource ist. Zum Beispiel gibt es unter Windows ein Limit von ungefähr 10.000 Fenstern pro Prozess. Das klingt vielleicht ausreichend genug, aber bedenken Sie, dass unter Windows jedes GUI-Widget für das Betriebssystem ein "Fenster" ist. Z.B. jede Beschriftung, jede Schaltfläche, jeder Rahmen, jedes Auswahlkästchen, jede Gruppe usw. ist ein Fenster in Ihrer Anwendung (Application), so dass Sie darauf achten sollten, dass Sie Ihre Dialoge nur mit `moai.CreateDialog()` erstellen und anschließend direkt wieder löschen.

In der Praxis ist es ratsam, für jeden Ihrer Dialoge eine eigene XML-Datei zu erstellen und mit `moai.CreateDialog()` diese XML-Datei zur Laufzeit in einen Dialog zu konvertieren. Sobald der Benutzer ihn schließt, lässt man ihn automatisch durch RapaGUI wieder aus dem Speicher löschen.

Technisch gesehen handelt es sich bei diesem Befehl lediglich um eine Komfortfunktion, die intern `moai.CreateObject()` aufruft und dann das neu erstellte Objekt durch Aufruf von `Application.AddWindow` dem Applications-Objekt hinzufügt. `moai.CreateDialog()` kombiniert diese beiden Schritte einfach in einem.

Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für mehr Informationen über Dialoge.

EINGABEN

`xml$`	eine Zeichenkette, welche eine XML-MOAI-Dialogbeschreibung enthält
`parent$`	optional: übergeordnetes Fenster für das Dialogobjekt; Details siehe oben

BEISPIEL

```
moai.CreateDialog([[
```

```
<dialog id="dlg" title="Question">
  <vgroup>
      <text>What is your name?</text>
      <textentry/>
      <hgroup>
         <button>OK</button>
         <button>Cancel</button>
      </hgroup>
   </vgroup>
</window>
]])

mui.DoMethod("dlg", "showmodal")
```

Der obige Code erzeugt einen neuen Dialog und zeigt ihn an.

6.3 moai.CreateObject

BEZEICHNUNG

moai.CreateObject – erstellt ein MOAI-Objekt aus einer XML-Quelle

ÜBERSICHT

`moai.CreateObject(xml$[, parent$])`

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein MOAI-Objekt dynamisch aus einer XML-Quelle zu erstellen. Wenn `moai.CreateObject()` beendet ist, wird das neu erstellte MOAI-Objekt nicht an ein übergeordnetes Objekt angefügt und befindet sich losgelöst (isoliert) von Ihrem Application-Objekt, das mit `moai.CreateApp()` erstellt wurde. Um diesen lösgelösten Zustand aufzuheben, müssen Sie das Objekt zuerst an ein übergeordnetes Objekt anhängen, bei dem es sich entweder um ein Gruppen-, ein Menü- oder ein Application-Objekt handeln kann. Wenn das neu zugewiesene MOAI-Objekt ein Fensterobjekt ist, müssen Sie es an das Application-Objekt anfügen, indem Sie die Methode `Application.AddWindow` verwenden. Um Menüobjekte anzuhängen, müssen Sie die Methoden `Menubar.Prepend`, `Menubar.Append`, `Menubar.Insert`, `Menu.Prepend`, `Menu.Append` oder `Menu.Insert` verwenden. Alle anderen Objekte können mit den Gruppenmethoden `Group.Prepend`, `Group.Append` und `Group.Insert` angefügt werden.

Für die meisten MOAI-Klassen müssen Sie im zweiten Parameter `parent$` ein übergeordnetes Objekt angeben. RapaGUI wird das neue Objekt nicht an das übergeordnete Objekt anhängen, aber es muss immer noch das übergeordnete Objekt kennen, um bestimmte Einstellungen festlegen zu können. Die einzigen MOAI-Klassen, für die Sie kein übergeordnetes Objekt angeben müssen, sind die folgenden:

- Window-Klasse (Fenster)
- Dialog-Klasse
- Menubar-Klasse (Menüleisten)
- Menu-Klasse

– Menuitem-Klasse (Menüelement)

Bei allen anderen MOAI-Klassen müssen Sie den Identifikator des übergeordneten Objekts angeben.

Normalerweise müssen Sie nur den Identifikator des Fensters übergeben, an das Sie das MOAI-Objekt als übergeordnetes Objekt anhängen möchten. Es gibt eine Ausnahme: Wenn Sie planen, das MOAI-Objekt einer Gruppe mit einem Rahmen hinzuzufügen, d.h. eine Instanz von der Group-Klasse (Gruppen) mit `Group.Frame` auf `True` gesetzt, dann müssen Sie dieses Gruppenobjekt als übergeordnetes Objekt übergeben. Beachten Sie, dass dies nur für Gruppen mit einem Rahmen gilt. Wenn Sie vorhaben, das MOAI-Objekt einer normalen Gruppe hinzuzufügen, die keinen Rahmen hat, müssen Sie nur den Identifikator des Fensters an `moai.CreateObject()` übergeben. Beachten Sie jedoch, dass für den Fall, dass die normale Gruppe selbst nur ein Element ist, das irgendwo in der Hierarchie einer gerahmten Gruppe angehört, Sie den Identifikator dieser gerahmten Gruppe übergeben müssen.

Um es abstrakt auszudrücken: Das übergeordnete Objekt muss auf das nächste Objekt in der Layout-Hierarchie gesetzt werden, das eine visuelle Darstellung hat. Normale Gruppen sind nur Layout-Werkzeuge. Sie existieren nicht als Widget. Gerahmte Gruppen hingegen sind zwar Layout-Werkzeuge, aber sie sind auch Widgets, da sie eine visuelle Darstellung haben. Wenn Ihr Objekt also irgendwo in eine gerahmte Gruppe eingebettet werden soll, muss die gerahmte Gruppe als übergeordnetes Objekt übergeben werden. Wenn es in eine normale Gruppe eingebettet ist, dann muss das Fenster der obersten Ebene das übergeordnete Fenster sein. Vergessen Sie nicht, in Hierarchien zu denken: Auch wenn Sie Ihr Objekt an eine normale Gruppe anhängen wollen, kann es sein, dass die normale Gruppe selbst in eine gerahmte Gruppe eingebettet ist, was bedeutet, dass Sie die gerahmte Gruppe trotzdem als übergeordnete Gruppe übergeben müssen.

Im Gegensatz zu `moai.CreateApp()` können Sie `moai.CreateObject()` beliebig oft aufrufen, da es kein Application-Objekt für Sie erstellt, sondern nur losgelöste MOAI-Objekte, von denen Sie beliebig viele haben können.

Es ist wichtig, dass Sie in der XML-Definition eine ID für Ihr MOAI-Objekt angeben, da Sie diese ID verwenden müssen, um auf dieses Objekt zu verweisen, wenn Sie es zu einem Application-, Menü- oder Gruppenobjekt hinzufügen möchten.

Sobald dieser Befehl beendet ist, können Sie mit dem neu erstellten MOAI-Objekt (und allen seinen untergeordneten) die Befehle `moai.Set()`, `moai.Get()` und `moai.DoMethod()` benutzen.

Losgelöste MOAI-Objekte können mit dem Befehl `moai.FreeObject()` gelöscht werden. Dies müssen Sie aber nur in bestimmten Fällen aufrufen, z.B. wenn Sie es mit vielen dynamisch zugewiesenen MOAI-Objekten zu tun haben und Sie etwas Speicher und Ressourcen sparen wollen.

EINGABEN

`xml$`	eine Zeichenkette, die eine XML MOAI-Objektbeschreibung enthält
`parent$`	optional: gewünschtes übergeordnetes Objekt für das Objekt; Details siehe oben

BEISPIEL

```
moai.CreateObject([[
```

```
<window id="newwindow" title="A new window">
  <vgroup>
      <button>Hello World!</button>
   </vgroup>
</window>
]])

mui.DoMethod("app", "addwindow", "newwindow")
mui.Set("newwindow", "open", True)
```

Der obige Code erzeugt ein neues Fenster, fügt es dem bestehenden Application-Objekt hinzu und öffnet es.

```
moai.CreateObject([[
<button id="newbutton">Dynamically created button!</button>
]], "mywindow")

mui.DoMethod("mygroup", "initchange")
mui.DoMethod("mygroup", "append", "newbutton")
mui.DoMethod("mygroup", "exitchange", false)
```

Der obige Code erzeugt dynamisch ein neues Schaltflächen-Objekt und fügt es als letztes Element der Gruppe mit der ID "mygroup" hinzu.

6.4 moai.DoMethod

BEZEICHNUNG

moai.DoMethod – wendet eine Methode auf ein MOAI-Objekt an

ÜBERSICHT

```
r = moai.DoMethod(id$, method$, ...)
```

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine Methode für das angegebene MOAI-Objekt auszuführen. Sie müssen den Identifikator des MOAI-Objekts im ersten Argument `id$` und den Namen der Methode im zweiten Argument `method$` übergeben. Bei Methoden- und Objekt-IDs wird nicht zwischen Groß- und Kleinschreibung unterschieden. Es spielt also keine Rolle, ob Sie Groß- oder Kleinbuchstaben verwenden.

Welche Methoden Sie mit diesem Befehl verwenden können, hängt von der Klasse des angegebenen MOAI-Objekts ab. Schauen Sie sich die Klassenreferenz an, um zu sehen, welche Methoden von den verschiedenen MOAI-Klassen unterstützt werden.

Auch die Argumente, die Sie diesem Befehl nach dem Methodennamen übergeben müssen, hängen von der Methode ab. Sie sind für jede Methode unterschiedlich. Dasselbe gilt für die Rückgabewerte. Einige Methoden geben Werte zurück, andere nicht. Bitte beachten Sie die Klassenreferenz, um zu sehen, welche Argumente Ihre Methode benötigt und ob es Rückgabewerte für sie gibt.

EINGABEN

`id$` Identifikator des MOAI-Objekt, auf dem die Methode angewendet wird

`method$` Methodenname als Zeichenkette

`...` weitere Argumente sind methodenabhängig (Details siehe Klassenreferenz)

RÜCKGABEWERTE

`r` der Rückgabewert hängt vom Methodentyp ab

BEISPIEL

```
moai.DoMethod("my_listview", "insert", "bottom", "Last entry")
```

Der obige Code fügt einen neuen Eintrag mit dem Namen "Last entry" am unteren Rand der Listenansicht unter Verwendung des Identifikators "my_listview" hinzu. Dies geschieht durch die Ausführen der Methode `Listview.Insert` auf dem Objekt Listview.

6.5 moai.FreeApp

BEZEICHNUNG

moai.FreeApp – löscht das gesamte Application-Objekt

ÜBERSICHT

```
moai.FreeApp()
```

BESCHREIBUNG

Verwenden Sie diesen Befehl, um das gesamte Application-Objekt zu löschen, das mit dem letzten Befehl `moai.CreateApp()` erstellt wurde. `moai.FreeApp()` löscht das gesamte Application-Objekt und alle daran angehängten Elementen. Nachdem dieser Befehl beendet wurde, können Sie, wenn Sie möchten, eine neues Application-Objekt mit dem Befehl `moai.CreateApp()` erstellen.

Beachten Sie, dass `moai.FreeApp()` keine MOAI-Objekte löscht, die sich momentan in einem losgelösten Zustand befinden. Diese müssen mit `moai.FreeObject()` gelöscht werden.

EINGABEN

keine

6.6 moai.FreeDialog

BEZEICHNUNG

moai.FreeDialog – löscht ein Dialogobjekt

ÜBERSICHT

```
moai.FreeDialog(id$)
```

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein mit `moai.CreateDialog()` erzeugtes Dialogobjekt zu löschen. Normalerweise ist es nicht notwendig, diesen Befehl aufzurufen, da Dialoge automatisch durch `Dialog.EndModal` gelöscht werden oder wenn der Dialogbefehl beendet wird. Wenn Sie eine fein abgestimmte Kontrolle über die Löschung von Dialogen benötigen, können Sie diesen Befehl jedoch verwenden.

`moai.FreeDialog()` löst den Dialog zunächst vom Application-Objekt und löscht es dann. Technisch gesehen ist dieser Befehl also nur ein Komfortbefehl, der das Dialogobjekt intern aus dem Application-Objekt entfernt, indem sie `Application.RemoveWindow` und dann `moai.FreeObject()` aufruft, um das Dialogobjekt zu löschen. `moai.FreeDialog()` kombiniert diese beiden Schritte einfach in einem.

Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für mehr Informationen über Dialoge.

EINGABEN

`id$` Identifikator des Dialogobjekts, welches gelöscht wird

6.7 moai.FreeImage

BEZEICHNUNG

moai.FreeImage – löscht den Pinsel im Bilder-Cache (V1.2)

ÜBERSICHT

```
moai.FreeImage(id)
```

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Pinsel im internen Bilder-Cache von RapaGUI zu löschen. Sie müssen den Identifikator des Pinsels übergeben, der aus dem Speicher gelöscht werden soll. Alternativ können Sie auch -1 an diesen Befehl übergeben, um alle Pinsel im internen Bilder-Cache von RapaGUI aus dem Speicher zu löschen.

Sie müssen sicherstellen, dass der angegebene Pinsel nicht mehr von irgendwelchen Widgets in Ihrer GUI verwendet wird, bevor Sie diesen Befehl aufrufen. Beachten Sie, dass es unter normalen Bedingungen nicht notwendig ist, diesen Befehl aufzurufen, da normalerweise alle Pinsel automatisch von RapaGUI aus dem Speicher gelöscht werden. Unter bestimmten Umständen kann es jedoch sinnvoll sein, diesen Befehl aufzurufen.

RapaGUI speichert alle Hollywood-Pinsel, die Sie als Bilder in Ihrer GUI verwenden. Wenn Sie also versuchen, einen Hollywood-Pinsel in Ihrer GUI ein zweites Mal zu verwenden, wird er aus Performance-Gründen nur aus dem internen Bilder-Cache von RapaGUI geladen, unabhängig vom aktuellen Inhalt des Pinsels in Hollywood. Dies kann zu unerwünschtem Verhalten führen, falls Sie die Grafiken Ihres Pinsels in der Zwischenzeit aktualisiert haben und RapaGUI die aktualisierten Grafiken verwenden soll. In diesem Fall müssen Sie zuerst den Pinsel im internen Bilder-Cache von RapaGUI löschen, indem Sie diesen Befehl verwenden. Wenn Sie den Pinsel erneut an RapaGUI übergeben, wird er aus dem aktuellen Zustand des Pinsels neu erstellt und erneut zwischengespeichert.

EINGABEN

`id` Identifikator des Pinsels, der aus dem Bilder-Cache gelöscht werden soll oder -1 für das Löschen aller Pinsel

6.8 moai.FreeObject

BEZEICHNUNG

moai.FreeObject – löscht ein losgelöstes MOAI-Objekt

ÜBERSICHT

```
moai.FreeObject(id$)
```

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein losgelöstes MOAI-Objekt zu löschen, das entweder von `moai.CreateObject()` oder `moai.CreateApp()` erstellt wurde. Das MOAI-Objekt, das Sie hier angeben, darf nicht länger an ein Application-, Gruppen- oder Menüobjekt angehängt sein, da angehängte MOAI-Objekte mit ihren übergeordneten Objekten freigegeben werden. Stellen Sie daher sicher, dass Sie diesen Befehl nur für MOAI-Objekte verwenden, die von ihren übergeordneten Objekten losgelöst wurden und nicht mehr an ein übergeordnetes Objekt gebunden sind.

Um MOAI-Objekte von ihren jeweiligen übergeordneten Objekten loszulösen, müssen Sie eine der folgenden Methoden verwenden: `Application.RemoveWindow`, `Menubar.Remove`, `Menu.Remove` oder `Group.Remove`.

Wenn RapaGUI beendet wird, werden alle losgelösten MOAI-Objekte automatisch aus dem Speicher gelöscht, so dass Sie diesen Befehl normalerweise überhaupt nicht aufrufen müssen, wenn Ihr Programm nicht ständig MOAI-Objekte zur Laufzeit hinzufügt und entfernt.

EINGABEN

`id$`	Identifikator des MOAI-Objekts, welches gelöscht werden soll

6.9 moai.Get

BEZEICHNUNG

moai.Get – gibt den Wert eines MOAI-Objekt-Attributs zurück

ÜBERSICHT

```
r = moai.Get(id$, attr$)
```

BESCHREIBUNG

Mit diesem Befehl kann der aktuelle Wert des Attributs `attr$` im in `id$` angegebenen MOAI-Objekt ermittelt werden. Attributnamen und Objekt-IDs sind unabhängig von Groß- und Kleinschreibung.

Die Attribute, die Sie mit diesem Befehl verwenden können, hängen von der Klasse des angegebenen MOAI-Objekts ab. Schauen Sie sich die Klassenreferenz an, um zu sehen, welche Attribute von den verschiedenen MOAI-Klassen unterstützt werden. Um ein Attribut mit diesem Befehl zu verwenden, muss es eine Anwendbarkeit von "G" haben. Attribute von Area-Klasse (Bereich) und MOAI-Klasse können für fast alle anderen Klassen verwendet werden, da die Area- und MOAI-Klassen für die meisten anderen Klassen als Oberklasse fungieren.

EINGABEN

`id$`	Identifikator des abzufragenden MOAI-Objekts
`attr$`	Attribut, dessen Wert abgerufen werden soll

RÜCKGABEWERTE

`r`	aktueller Attributwert

BEISPIEL

```
DebugPrint(moai.Get("my_listview", "active"))
```

Der obige Code gibt den Index des aktuell aktiven Eintrags in der Listenansicht zurück, der den Bezeichner "my_listview" hat, indem er das Attribut `Listview.Active` abfragt.

6.10 moai.HaveObject

BEZEICHNUNG

moai.HaveObject – prüft, ob ein MOAI-Objekt vorhanden ist

ÜBERSICHT

```
r = moai.HaveObject(id$)
```

BESCHREIBUNG

Dieser Befehl prüft lediglich, ob das MOAI-Objekt im angegebenen Attribut `id$` existiert oder nicht. Falls vorhanden, wird `True` zurückgegeben, ansonsten `False`.

EINGABEN

`id$`	Identifikator des zu prüfenden MOAI-Objekts

RÜCKGABEWERTE

`r`	`True` oder `False`, je nachdem, ob das Objekt existiert oder nicht.

6.11 moai.Notify

BEZEICHNUNG

moai.Notify – fügt eine Benachrichtigung einem MOAI-Objekt hinzu oder enternt sie

ÜBERSICHT

```
moai.Notify(id$, attr$, enable)
```

BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine Benachrichtigung für das Attribut `attr$` in dem in `id$` angegebenen MOAI-Objekt hinzuzufügen oder zu entfernen. Sie müssen den Attributnamen und ein boolesches Flag übergeben, das angibt, ob Sie Benachrichtigungen für dieses Attribut aktivieren oder deaktivieren möchten. Attributnamen und Objekt-IDs unterscheiden nicht zwischen Groß- und Kleinschreibung, d.h. es spielt keine Rolle, ob Sie für sie Groß- oder Kleinbuchstaben verwenden.

Welche Attribute Sie mit diesem Befehl verwenden können, hängt von der Klasse des angegebenen MOAI-Objekts ab. Welche Attribute von den verschiedenen MOAI-Klassen unterstützt werden, sehen Sie in der Klassenreferenz. Um ein Attribut mit diesem Befehl verwenden zu können, muss es eine Anwendbarkeit von "N" haben. Attribute von Area-Klasse (Bereich) und MOAI-Klasse können auf fast allen anderen Klassen verwendet werden, da die Area- und MOAI-Klassen als Oberklassen für die meisten anderen Klassen dienen.

Sobald Sie eine Benachrichtigung über ein bestimmtes Objektattribut eingerichtet haben, können Sie diese Ereignisse überwachen, indem Sie eine RapaGUI-Callback-Funktion

mit dem Hollywood-Befehl `InstallEventHandler()` installieren. Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

Beachten Sie, dass Benachrichtigungen auch in der XML-GUI-Definition mithilfe des Attributs `Notify` eingerichtet werden können. Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

EINGABEN

`id$` ID des MOAI-Objekts

`attr$` Attribut, bei dem Sie eine Benachrichtigung hinzufügen oder entfernen wollen

`enable` `True`, um eine Benachrichtigung hinzuzufügen oder `False`, um eine Benachrichtigung aus diesem Objekt zu entfernen

BEISPIEL

```
moai.Notify("my_listview", "active", True)
```

Der obige Code installiert eine Benachrichtigung, die ausgelöst wird, wenn sich das Attribut `Listview.Active` in der Listenansicht mit dem Identifikator `"my_listview"` ändert.

6.12 moai.Request

BEZEICHNUNG

moai.Request – öffnet ein Auswahldialogfenster

ÜBERSICHT

```
r = moai.Request(title$, body$, buts$[, icon$])
```

BESCHREIBUNG

Dieser Befehl öffnet einen Standard-Auswahldialogfenster, das die in `body$` angegebene Nachricht anzeigt und dem Benutzer außerdem erlaubt, eine Auswahl unter Verwendung einer der durch `buts$` spezifizierten Schaltflächen zu treffen. Optional können Sie auch ein Symbol in `icon$` angeben, das im Dialogfenster angezeigt werden soll. Wenn Sie in `title$` eine leere Zeichenfolge angeben, wird automatisch der in `@APPTITLE` angegebene Titel verwendet.

Die in `buts$` angegebenen Schaltflächen trennen Sie mit einem `"|"`. Der Rückgabewert `r` gibt an, welche Taste der Benutzer gedrückt hat. Bitte beachten Sie, dass die rechte Schaltfläche immer den Wert `False` (0) hat, da er typischerweise als `"Abbrechen"`-Schaltfläche verwendet wird. Wenn Sie zum Beispiel drei Schaltflächen `"Eins|Zwei|Drei"` gewählt haben, hat die Schaltfläche `"Drei"` den Rückgabewert 0, `"Zwei"` gibt 2 und `"Eins"` gibt 1 zurück.

Das optionale Argument `icon$` kann auf einen der folgenden vordefinierten Werte gesetzt werden:

`"None"`: Kein Symbol

`"Information"`:
Ein Informationszeichen

`"Error"`: Ein Fehlerzeichen

`"Warning"`:
Ein Warnzeichen

`"Question"`:
Ein Fragezeichen

EINGABEN

`title$` Title für das Dialogfenster; übergeben Sie eine leere Zeichenfolge (`""`), wird der `@APPTITLE` verwendet

`body$` Text, der als Nachricht im Dialogfenster erscheint

`buts$` eine oder mehrere Schaltflächen, die der Benutzer drücken kann

`icon$` optional: Symbol, welches im Dialogfenster angezeigt werden kann (Standardeinstellung ist `"Information"`)

RÜCKGABEWERTE

`r` die Schaltfläche, die durch den Benutzer gedrückt wurde

BEISPIEL

```
moai.Request("RapaGUI", "Hello!\n\n" ..
    "Do you like RapaGUI!", "Yes|No")
```

Der obige Code demonstriert die Verwendung des Befehls `moai.Request()`.

6.13 moai.Set

BEZEICHNUNG

moai.Set – setzt den Wert eines MOAI-Objekt-Attributs

ÜBERSICHT

```
moai.Set(id$, attr1$, val1$, ...)
```

BESCHREIBUNG

Mit diesem Befehl kann der aktuelle Wert eines oder mehrerer Attribute in dem in `id$` angegebenen MOAI-Objekt gesetzt werden. Für jedes zu ändernde Attribut müssen Sie den Attributnamen und den gewünschten neuen Wert übergeben. Sie können diese Attribut/Wertpaare beliebig oft wiederholen, um mehrere Attribute mit nur einem einzigen Aufruf von `moai.Set()` zu ändern. Attributnamen und Objekt-IDs sind unabhängig von Groß- und Kleinschreibung.

Welche Attribute Sie mit diesem Befehl verwenden können, hängt von der Klasse des angegebenen MOAI-Objekts ab. Welche Attribute von den verschiedenen MOAI-Klassen unterstützt werden, sehen Sie in der Klassenreferenz. Um ein Attribut mit diesem Befehl verwenden zu können, muss es eine Anwendbarkeit von "S" haben. Attribute der Area-Klasse (Bereich) und MOAI-Klasse können auf fast allen anderen Klassen verwendet werden, da die Area- und MOAI-Klassen als Oberklassen für die meisten anderen Klassen dienen.

Wenn Sie mit diesem Befehl eine Benachrichtigung für das Attribut erstellt haben, das Sie ändern möchten, wird die Benachrichtigung ausgelöst, sobald Sie `moai.Set()` für dieses Attribut aufrufen. Wenn Sie dieses Verhalten nicht möchten, können Sie das Attribut

`MOAI.NoNotify` verwenden, um zu verhindern, dass eine Benachrichtigung ausgegeben wird.

EINGABEN

`id$`	Identifikator des zu ändernden MOAI-Objekts
`attr1$`	Attribut, dessen Wert geändert werden soll
`val1$`	neuer Wert für das Attribut
`...`	Sie können Attribut/Wertpaare beliebig oft wiederholen

BEISPIEL

```
moai.Set("my_listview", "active", 15)
```

Der obige Code setzt die Eintragsnummer 15 als derzeit aktiven Eintrag in der Listenansicht mit dem Identifikator "my_listview", indem das Attribut `Listview.Active` gesetzt wird.

```
moai.Set("my_listview", "nonotify", True, "active", 15)
```

Dieser Code macht dasselbe wie der obige Code, verhindert aber, dass Benachrichtigungen ausgegeben werden, indem das Attribut `MOAI.NoNotify` auf `True` gesetzt wird. Dies ist nützlich, wenn Sie zwischen Benutzerselektionen in der Listenansicht und programmgesteuerten Selektionen mit `moai.Set()` unterscheiden müssen.

7 Accelerator-Klasse (Tastaturkürzel)

7.1 Übersicht

Die Accelerator-Klasse kann verwendet werden, um globale Tastaturkürzel für ein Fenster einzurichten. Normale Tastaturkürzel, die mit dem Unterstrichzeichen definiert wurden, werden von RapaGUI automatisch überwacht. Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details. Manchmal möchten Sie jedoch komplexere Tastaturkürzel verwenden, z.B. die Escape-Taste, Funktionstasten, bestimmte Steuertasten oder Tastenkombinationen (z.B. `CTRL+V` zum Einfügen). Dies ist durch die Verwendung der Accelerator-Klasse möglich.

Mit der Accelerator-Klasse können Sie eine Tabelle von Tastaturkürzeln einrichten, die dann einem Fenster zugewiesen werden können, indem Sie das Attribut `Window.Accelerator` verwenden, um diese Tastaturkürzel im jeweiligen Fenster global verfügbar zu machen. Jede Tastaturkürzel-Tabelle muss mindestens ein Element vom Typ Acceleratoritem-Klasse enthalten, das eine einzige Tastaturkürzel definiert.

Hier ist ein Beispiel, wie man eine Tastaturkürzel-Tabelle in XML einrichtet:

```
<accelerator>
   <item id="ac_cut" mod="ctrl">X</item>
   <item id="ac_copy" mod="ctrl">C</item>
   <item id="ac_paste" mod="ctrl">V</item>
   <item id="ac_f1">F1</item>
</accelerator>
```

Die oben angegebene Tastaturkürzel-Tabelle legt vier verschiedene Tastaturkürzel fest: `CTRL+X`, `CTRL+C`, `CTRL+V` und `F1`. Wenn der Benutzer eine solche Taste oder Tastenkombination drückt, wird die Benachrichtigung `Acceleratoritem.Pressed` ausgelöst und Ihr Programm kann darauf reagieren.

Beachten Sie, dass Sie, wenn Sie beabsichtigen, Ihre Tastaturkürzel-Einträge mit Menüeinträgen zu verknüpfen, überhaupt keine Accelerator-Klasse verwenden müssen. In diesem Fall können Sie einfach das Attribut `Menuitem.Shortcut` verwenden, um Verknüpfungen für Ihre Menüeinträge einzurichten. Accelerator-Klasse ist wirklich nur notwendig, wenn Sie Tastatur-Ereignisse überwachen müssen, ohne irgendwelche Menüeinträge zu haben.

Die Accelerator-Klasse definiert selbst keine Attribute oder Methoden. Siehe Abschnitt 8.1 [Acceleratoritem-Klasse], Seite 55, für alle notwendigen Informationen.

8 Acceleratoritem-Klasse (Tastaturkürzelelement)

8.1 Übersicht

Die Acceleratoritem-Klasse wird verwendet, um eine einzelne Tastenkombination als Teil einer Tastaturkürzel-Tabelle für die Accelerator-Klasse zu erstellen. Sie können keine unabhängigen Instanzen der Acceleratoritem-Klasse erstellen. Sie müssen immer in die Accelerator-Klasse eingebettet werden. Siehe Abschnitt 7.1 [Accelerator-Klasse], Seite 53, für Details.

Die folgenden Tasten können derzeit beim Erstellen eines Tastaturkürzel-Eintrags mit dem XML-Tag `<item>` angegeben werden:

`UP`	Pfeil nach oben
`DOWN`	Pfeil nach unten
`RIGHT`	Pfeil nach rechts
`LEFT`	Pfeil nach links
`HELP`	Help-Taste
`DEL`	Delete-Taste
`BACKSPACE`	Rückschritt-Taste
`TAB`	Tabulator-Taste
`RETURN`	Return-Taste
`ENTER`	Enter-Taste
`ESC`	Escape-Taste
`SPACE`	Leertaste
`F1 - F16`	Funktionstasten
`INSERT`	Einfüge-Taste
`HOME`	Anfangs-Taste
`END`	Ende-Taste
`PAGEUP`	Seite rauf
`PAGEDOWN`	Seite runter
`PRINT`	Drucken-Taste
`PAUSE`	Pause-Taste

Darüber hinaus unterstützt die Acceleratoritem-Klasse die englischen Buchstaben von A bis Z sowie die Zahlen 0 bis 9 als Tastendefinition. Dabei spielt es keine Rolle, ob Sie alle diese Tasten in Groß-, Klein- oder Mischschreibung angeben. Siehe Abschnitt 7.1 [Accelerator-Klasse], Seite 53, für ein Beispiel.

8.2 Acceleratoritem.Mod

BEZEICHNUNG

Acceleratoritem.Mod – setzt die Umschalttaste(n)

BESCHREIBUNG

Mit diesem Attribut können Sie den oder die Umschalttasten für dieses Tastaturkürzel-Element angeben. Dies kann eine Kombination der folgenden vordefinierten Werte sein:

`None`	Verwendet keine Umschalttasten. Dies ist die Voreinstellung.
`Ctrl`	Fügt die Control-Taste zu den Umschalttasten hinzu.
`Alt`	Fügt die Alt-Taste zu den Umschalttasten hinzu.
`Cmd`	Fügt die Steuer-Taste zu den Umschalttasten hinzu. Dies wird nur auf Mac OS und AmigaOS unterstützt.
`Shift`	Fügt die Shift-Taste zu den Umschalttasten hinzu.

Wenn Sie mehr als eine Umschalttaste angeben, müssen Sie die einzelnen Tasten durch ein Semikolon (;) trennen, z.B. `"ctrl;shift"`.

Bei Verwendung von Umschalttasten wird die Tastenkombination nur ausgelöst, wenn die Umschalttaste(n) und die Haupttaste(n) gedrückt sind. Dies ist nützlich für die Überwachung von Tastenkombinationen wie `CTRL+V` zum Einfügen von Daten, etc.

TYP

Zeichenkette

ANWENDBARKEIT

I

8.3 Acceleratoritem.Pressed

BEZEICHNUNG

Acceleratoritem.Pressed – benachrichtigt, wenn das Tastaturkürzel gedrückt wird

BESCHREIBUNG

Dieses Attribut wird ausgelöst, wenn der Benutzer die Taste(n) drückt, die durch dieses Tastaturkürzel-Element definiert sind. RapaGUI überwacht automatisch dieses Attribut, so dass Sie nicht explizit eine Benachrichtigung mit dem Attribut MOAI.Notify anfordern müssen.

TYP

Boolesch

ANWENDBARKEIT

N

9 Application-Klasse (Anwendung)

9.1 Übersicht

Die Application-Klasse (Anwendung) ist die Hauptklasse, die ein Programm verwaltet. Daher kann es in jedem Programm nur eine Instanz dieser Klasse geben. Diese Instanz wird durch einen Aufruf von `moai.CreateApp()` erzeugt. Alle Fenster, die ein Programm anzeigt, sind Elemente der Application-Klasse. Jede GUI-Definition muss also mit der Application-Klasse beginnen.

9.2 Application.AboutMUI

BEZEICHNUNG

Application.AboutMUI – zeigt das Über-MUI-Fenster an

ÜBERSICHT

```
mui.DoMethod(id, "AboutMUI")
```

PLATTFORMEN

Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG

Zeigt das Über-MUI-Fenster an. Aus der MUI-Styleguide ist zu entnehmen, dass alle MUI-Programme einen Menüpunkt namens "Über MUI ..." enthalten sollten.

EINGABEN

`id`	ID des Applications-Objekts

9.3 Application.AboutRapaGUI

BEZEICHNUNG

Application.AboutRapaGUI – zeigt das Über-RapaGUI-Fenster an

ÜBERSICHT

```
moai.DoMethod(id, "AboutRapaGUI")
```

BESCHREIBUNG

Diese Methode zeigt das Über-RapaGUI-Fenster an. Sie sollten den Menüpunkt "Über RapaGUI...." in alle Ihre Programme aufnehmen.

EINGABEN

`id`	ID des Applications-Objekts

9.4 Application.AddWindow

BEZEICHNUNG

Application.AddWindow – fügt ein losgelöstes Fensterobjekt zum Applications-Objekt hinzu

ÜBERSICHT

```
moai.DoMethod(id, "AddWindow", window)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um dem Applications-Objekt ein losgelöstes Fensterobjekt hinzuzufügen. Nachdem das Fensterobjekt an das Applications-Objekt angehängt wurde, können Sie das Fenster öffnen, indem Sie sein Attribut `Window.Open` setzen.

Losgelöste Fensterobjekte können entweder durch Aufruf des Befehls `moai.CreateObject()` erzeugt werden oder Sie können mit der Methode `Application.RemoveWindow` ein Fensterobjekt explizite von ihrem übergeordneten Objekt loslösen.

Bitte beachten Sie, dass Sie die Methode `Application.AddWindow` nicht für Dialoge aufrufen müssen, die mit dem Befehl `moai.CreateDialog()` erstellt wurden, da `moai.CreateDialog()` bereits intern `Application.AddWindow` aufruft.

EINGABEN

`id`	ID des Applications-Objekts
`window`	ID des hinzuzufügenden Fensterobjekts

BEISPIEL

Siehe Abschnitt 6.3 [moai.CreateObject], Seite 43.

9.5 Application.Icon

BEZEICHNUNG

Application.Icon – setzt das Programm-Icon

PLATTFORMEN

Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG

Mit diesem Tag können Sie das Icon festlegen, das auf dem Bildschirm der Workbench angezeigt wird, wenn das Programm ikonisiert wird. Sie müssen diesen Tag auf den Dateinamen einer Amiga-Icon-Datei `.info` setzen.

Normalerweise sollten Sie die Icons für Ihre Anwendung mit der Präprozessor-Anweisung `@APPICON` von Hollywood setzen. Dieser Tag ist nur hier, um Amiga-Programme auf der Workbench zu unterstützen.

TYP

Zeichenkette

ANWENDBARKEIT

I

9.6 Application.OpenConfigWindow

BEZEICHNUNG

Application.OpenConfigWindow – zeigt das MUI-Einstellungsfenster

ÜBERSICHT

```
mui.DoMethod(id, "OpenConfigWindow")
```

PLATTFORMEN

Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG

Mit dieser Methode können Sie das Einstellungsfenster von MUI für das Programm öffnen. MUI unterstützt individuelle Benutzeroberflächeneinstellungen für jedes Programm, weshalb Sie jedem MUI-Programm einen Menüpunkt wie "Einstellungen/MUI...." hinzufügen sollten, damit der Benutzer diese Einstellungen vornehmen kann.

EINGABEN

`id` ID des Applications-Objekts

9.7 Application.RemoveWindow

BEZEICHNUNG

Application.RemoveWindow – löst ein Fensterobjekt vom Applications-Objekt

ÜBERSICHT

```
moai.DoMethod(id, "RemoveWindow", window)
```

BESCHREIBUNG

Diese Methode entfernt das angegebene Fensterobjekt aus dem Applications-Objekt und versetzt das Fensterobjekt in den Zustand "losgelöst". Fensterobjekte im losgelösten Zustand können entweder durch Ausführen der Methode `Application.AddWindow` wieder angefügt oder durch den Befehl `moai.FreeObject()` aus dem Speicher gelöscht werden.

Bitte beachten Sie, dass Sie `Application.RemoveWindow` nicht für Dialoge aufrufen müssen, die mit `moai.FreeDialog()` freigegeben wurden, da `moai.FreeDialog()` bereits intern `Application.RemoveWindow` aufruft.

EINGABEN

`id` ID des Applications-Objekts

`window` ID des zu entfernenden Fensterobjekts

9.8 Application.Sleep

BEZEICHNUNG

Application.Sleep – versetzt das Programm in den Ruhezustand

BESCHREIBUNG

Dieses Attribut versetzt das Programm in den Ruhezustand. Alle geöffneten Fenster werden deaktiviert und sie akzeptieren keine Benutzereingaben mehr. Dies kann nützlich sein, wenn Ihr Programm gerade Dateien lädt oder speichert, wodurch Benutzereingaben nicht möglich sind.

TYP

Boolesch

ANWENDBARKEIT

S

10 Area-Klasse (Bereich)

10.1 Übersicht

Area-Klasse (Bereich) ist die Oberklasse jeder MOAI-Klasse, die Objekte erzeugt, die eine visuelle Darstellung innerhalb eines Fensters haben. Als solches erlaubt es Ihnen, verschiedene generische Attribute wie Sichtbarkeit von Objekten (ein- oder ausgeblendet), Status (aktiviert oder deaktiviert), Tooltips und Kontextmenüs zu verwalten. Elemente, die von der Area-Klasse abgeleitet sind, werden gewöhnlich Widgets genannt (ein Kofferwort aus "window" und "gadget"). Die Area-Klasse kann Ihnen auch Informationen über die Position und Größe eines Widgets sowie andere Details liefern, die zu seiner visuellen Darstellung innerhalb eines Fensters gehören.

Beachten Sie, dass Objekte, die von der Group-Klasse (Group) abgeleitet werden, keine Elemente der Area-Klasse sind, da die Group-Klasse nur Gruppen von Widgets erstellt, aber kein Widget selbst ist, sondern nur ein Layout-Werkzeug.

10.2 Area.ContextMenu

BEZEICHNUNG

Area.ContextMenu – setzt das Kontextmenü für ein Objekt

BESCHREIBUNG

Setzen Sie dieses Attribut auf den Identifikator eines MOAI-Objekts, welches von der Menu-Klasse abgeleitet ist und somit erhält das entsprechende Objekt ein Kontextmenü. Wenn der Benutzer die rechte Maustaste auf das jeweilige Objekt drückt, wird das Kontextmenü angezeigt.

Beachten Sie, dass Kontextmenü-Ereignisse dann über den Standardmechanismus `Menuitem.Selected` und nicht über einen speziellen Kontextmenü-Ereignishandler geliefert werden. Da Sie dasselbe Menüobjekt für mehrere Widgets verwenden können, benötigen Sie eine Möglichkeit, das Widget herauszufinden, dessen Kontextmenü das Ereignis ausgelöst hat. Um Ihnen diese Informationen zu liefern, enthält die Ereignismeldung einen zusätzlichen Eintrag namens "Parent", der die ID des Widgets enthält. Dadurch können Sie dasselbe Menüobjekt als Kontextmenü für mehrere übergeordnete Widgets verwenden.

AmigaOS-Benutzer beachten bitte, dass MUI die Freigabe von Kontextmenü-Objekten über Fenster hinweg nicht erlaubt. Es ist nicht möglich, dasselbe Menüobjekt mit Widgets in verschiedenen Fenstern zu verwenden. Es ist in Ordnung, ein Menüobjekt für mehrere Widgets im selben Fenster zu verwenden, aber nicht für Widgets in einem anderen Fenster. Alle anderen Plattformen haben diese Einschränkung nicht, nur AmigaOS ist hier betroffen.

Um ein Kontextmenü vollständig zu entfernen, übergeben Sie diesem Attribut den speziellen String "(none)".

Siehe Abschnitt 3.13 [Kontextmenüs], Seite 20, für ein Beispiel.

TYP

MOAI-Objekt

ANWENDBARKEIT
IS

10.3 Area.Disabled

BEZEICHNUNG
Area.Disabled – setzt oder gibt den Deaktivierstatus zurück

BESCHREIBUNG
Setzen Sie dieses Attribut, um ein Widget zu deaktivieren oder zu aktivieren. Deaktivierte Widgets akzeptieren keine Benutzereingaben mehr und sind optisch als deaktivierte Objekte erkennbar.

TYP
Boolesch

ANWENDBARKEIT
ISG

10.4 Area.FixHeight

BEZEICHNUNG
Area.FixHeight – fixiert die Höhe eines Objektes

BESCHREIBUNG
Setzen Sie dies auf `True`, um die vertikale Größenanpassung für dieses Objekt zu deaktivieren.

TYP
Boolesch

ANWENDBARKEIT
I

10.5 Area.FixWidth

BEZEICHNUNG
Area.FixWidth – fixiert die Breite eines Objektes

BESCHREIBUNG
Setzen Sie dies auf `True`, um die horizontale Größenänderung für dieses Objekt zu deaktivieren.

TYP
Boolesch

ANWENDBARKEIT
I

10.6 Area.Height

BEZEICHNUNG
Area.Height – setzt/ermittelt die Höhe eines Objekts

BESCHREIBUNG
Setzen Sie dieses Attribut auf die gewünschte Objekthöhe in Pixel. Dies ist normalerweise nicht notwendig, da RapaGUI automatisch eine geeignete Größe für seine Objekte wählt. In manchen Fällen kann es jedoch sinnvoll sein, die Objektgröße fein abzustimmen.

Wenn das übergeordnete Fenster des Objekts geöffnet ist, können Sie auch die Höhe des Objekts über dieses Attribut auslesen.

TYP
Zahl

ANWENDBARKEIT
IG

10.7 Area.Hide

BEZEICHNUNG
Area.Hide – blendet ein Objekt ein oder aus

BESCHREIBUNG
Setzen Sie dieses Attribut, um ein Objekt ein- oder auszublenden.

TYP
Boolesch

ANWENDBARKEIT
ISG

10.8 Area.Left

BEZEICHNUNG
Area.Left – gibt die linke Kante des Objekts zurück

BESCHREIBUNG
Dieses Attribut gibt die x-Position eines Objekts innerhalb seines übergeordneten Fensters zurück. Dies gilt nur, wenn das Objekt aktuell sichtbar ist.

TYP
Zahl

ANWENDBARKEIT
G

10.9 Area.NoAutoKey

BEZEICHNUNG
Area.NoAutoKey – deaktiviert die automatische Tastaturkürzel-Generierung

BESCHREIBUNG
Setzen Sie dieses Attribut, um die automatische Tastaturkürzel-Generierung zu deaktivieren. Standardmäßig wird das Zeichen nach einem Unterstrich in einem Label oder einem Menüeintrag als Tastaturkürzel verwendet, mit der Sie über die Tastatur auf das Widget zugreifen können. Wenn Sie dieses Attribut setzen, werden Unterstrichzeichen niemals als Markierung für Tastaturkürzel verwendet und werden als normaler Text bei Beschriftungen (Label) und in Menüeinträgen angezeigt.

Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details.

TYP
Boolesch

ANWENDBARKEIT
I

10.10 Area.Redraw

BEZEICHNUNG
Area.Redraw – erzwingt die vollständige Neuzeichnung eines Widget

ÜBERSICHT
```
moai.DoMethod(id, "Redraw")
```

BESCHREIBUNG
Diese Methode zeichnet vollständig das Widget neu. Normalerweise ist dies nicht notwendig, da Widgets selbst entscheiden, wann sie neu gezeichnet werden müssen. Dennoch könnte es zu Testzwecken oder bei Verwendung eines von Scrollcanvas-Klasse (Bildlaufleinwand) abgeleiteten Widget nützlich sein.

EINGABEN

`id`	ID des Widget

10.11 Area.Tooltip

BEZEICHNUNG
Area.Tooltip – setzt/ermittelt die Zeichenkette des Tooltip

BESCHREIBUNG
Legen Sie eine Zeichenkette fest, die in einem Tooltip angezeigt werden soll, wenn Sie mit der Maus über das Objekt fahren.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

10.12 Area.Top

BEZEICHNUNG
Area.Top – gibt die obere Kante des Objekts zurück

BESCHREIBUNG
Dieses Attribut gibt die y-Position eines Objekts innerhalb seines übergeordneten Fensters zurück. Dies gilt nur, wenn das Objekt aktuell sichtbar ist.

TYP
Zahl

ANWENDBARKEIT
G

10.13 Area.Weight

BEZEICHNUNG
Area.Weight – setzt die Objektgröße

BESCHREIBUNG
Dieses Attribut setzt die Größe eines Objekts innerhalb einer Gruppe von Objekten. Standardmäßig haben alle Objekte einer Gruppe eine Größe von 100. Wenn Sie ein Objekt haben wollen, das doppelt so groß erscheint, müssen Sie ihm eine Größe von 200 geben. Um ein Objekt eineinhalb Mal so groß erscheinen zu lassen, verwenden Sie einfach 150 und so weiter.

Natürlich ist dieses Attribut nur für Objekte sinnvoll, die in der Größe veränderbar sind.

TYP
Zahl

ANWENDBARKEIT
I

10.14 Area.Width

BEZEICHNUNG
Area.Width – setzt/ermittelt die Breite des Objekts

BESCHREIBUNG
Setzen Sie dieses Attribut auf die gewünschte Objektbreite in Pixel. Dies ist normalerweise nicht notwendig, da RapaGUI automatisch eine geeignete Größe für seine Objekte wählt. In manchen Fällen kann es jedoch sinnvoll sein, die Objektgröße fein abzustimmen.

Wenn das übergeordnete Fenster des Objekts geöffnet ist, können Sie auch die Breite des Objekts über dieses Attribut auslesen.

TYP
Zahl

ANWENDBARKEIT
IG

11 Busybar-Klasse (Beschäftigung)

11.1 Übersicht

Die Busybar-Klasse (Beschäftigung) erstellt ein Widget, welches eine Animation zeigt, womit der Benutzer erkennt, dass das Programm gerade beschäftigt ist. Diese Animation zeigt jedoch nur Aktivität, aber keinen Fortschritt. Wenn Sie den Fortschritt visualisieren möchten, verwenden Sie stattdessen die Progressbar-Klasse (Fortschrittsbalken). Siehe Abschnitt 36.1 [Progressbar-Klasse], Seite 173, für Details.

Sobald Sie ein Objekt der Busybar-Klasse erstellt haben, müssen Sie `Busybar.Move` wiederholt aufrufen, um die Animation zum Leben zu erwecken. Dies kann durch die Einrichtung eines Intervall-Timers mit Hilfe des Hollywood-Befehls `SetInterval()` erreicht werden.

11.2 Busybar.Move

BEZEICHNUNG

Busybar.Move – zeichnet das nächste Animationseinzelbild

ÜBERSICHT

```
moai.DoMethod(id, "Move")
```

BESCHREIBUNG

Sie müssen diese Methode wiederholt aufrufen, um das Widget zu animieren. Eine gute Idee ist es, einen Hollywood-Intervall-Timer mit `SetInterval()` zu installieren und dann diese Methode etwa 10 mal pro Sekunde aufzurufen.

EINGABEN

`id`	ID des Busybar-Objekts

11.3 Busybar.Reset

BEZEICHNUNG

Busybar.Reset – stoppt die Animation

ÜBERSICHT

```
moai.DoMethod(id, "Reset")
```

BESCHREIBUNG

Diese Methode stoppt die Animation und setzt sie auf den meisten Plattformen zurück. Sukzessive Aufrufe von `Busybar.Move` beginnen normalerweise wieder ab dem ersten Einzelbild zu zeichnen.

EINGABEN

`id`	ID des Busybar-Objekts

12 Button-Klasse (Schaltflächen)

12.1 Übersicht

Mit der Button-Klasse können Sie sehr einfach Schaltflächen (Knöpfe, Tasten, Schalter) erstellen. Der Inhalt des XML-Tags wird als Schaltflächentext verwendet. Wenn der Schaltflächentext einen Unterstrich enthält, richtet RapaGUI automatisch das auf diesen Unterstrich folgende Zeichen als Tastaturkürzel ein. Wenn Sie dieses Verhalten nicht wünschen, setzen Sie `Area.NoAutoKey` auf `True`. Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details.

Beispiel:

```
<button id="ok">OK</button>
```

12.2 Button.Icon

BEZEICHNUNG

Button.Icon – zeigt ein Symbol in der Schaltfläche an

BESCHREIBUNG

Setzen Sie dieses Attribut auf die ID eines Hollywood-Pinsels, damit dieser Pinsel als Symbol in der Schaltfläche angezeigt wird. Sie können die Position des Symbols in Bezug auf die Beschriftung der Schaltfläche steuern, indem Sie das Attribut `Button.IconPos` festlegen.

Bitte lesen Sie auch den Abschnitt Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Beachten Sie, dass unter AmigaOS und kompatiblen die Unterstützung von Symbolen nur mit MUI 4.0 oder höher verfügbar ist.

TYP

Zahl

ANWENDBARKEIT

I

12.3 Button.IconPos

BEZEICHNUNG

Button.IconPos – definiert die Position des Schaltflächen-Symbols

BESCHREIBUNG

Stellen Sie die gewünschte Position eines mit `Button.Icon` angegebenen Symbols ein. Dies kann einer der folgenden Werte sein:

`Left` Zeigt das Symbol links neben dem Text der Schaltfläche. Dies ist die Voreinstellung.

`Right` Zeigt das Symbol rechts neben dem Text der Schaltfläche.

Beachten Sie, dass die Unterstützung von Symbolen auf AmigaOS- und kompatiblen nur mit MUI 4.0 oder besser verfügbar ist.

TYP
Zeichenkette (mögliche Werte siehe oben)

ANWENDBARKEIT
I

12.4 Button.Pressed

BEZEICHNUNG
Button.Pressed – benachrichtigt, ob eine Schaltfläche gedrückt wird

BESCHREIBUNG
Dieses Attribut wird ausgelöst, wenn der Benutzer die Schaltfläche drückt. RapaGUI überwacht automatisch dieses Attribut für alle Schaltflächen, so dass Sie nicht explizit eine Benachrichtigung mit dem Attribut MOAI.Notify anfordern müssen.

TYP
Boolesch

ANWENDBARKEIT
N

12.5 Button.Selected

BEZEICHNUNG
Button.Selected – Auswahlzustand erfahren/umschalten/benachrichtigen

BESCHREIBUNG
Benutzen Sie diese Option, um den Auswahlzustand eines Umschaltknopfes zu erfahren, umzuschalten oder sich über ein Benutzer-Umschalt-Ereignis benachrichtigen zu lassen. Wenn Sie dieses Attribut verwenden wollen, müssen Sie zuerst `Button.Toggle` auf `True` setzen.

RapaGUI überwacht dieses Attribut automatisch für alle Schaltflächen, sodass Sie nicht explizit eine Benachrichtigung mit dem Attribut MOAI.Notify anfordern müssen.

TYP
Boolesch

ANWENDBARKEIT
ISGN

12.6 Button.Text

BEZEICHNUNG
Button.Text – setzt/ermittelt den Schaltflächentext

BESCHREIBUNG
Mit diesem Attribut setzen oder holen Sie die Beschriftung der Schaltfläche.

TYP
Zeichenkette

ANWENDBARKEIT
SG

12.7 Button.Toggle

BEZEICHNUNG
Button.Toggle – erstellt einen Umschaltknopf

BESCHREIBUNG
Wenn Sie diesen Wert auf `True` setzen, wird ein Umschaltknopf erstellt. Umschaltknöpfe haben zwei Zustände: Ein und Aus. Sie können das Attribut `Button.Selected` verwenden, um den Zustand umzuschalten und sich über Zustandswechsel-Ereignisse informieren zu lassen.

Standardmäßig ist die Schaltfläche eine normale Schaltfläche.

TYP
Boolesch

ANWENDBARKEIT
I

13 Checkbox-Klasse (Auswahlkästchen)

13.1 Übersicht

Eine Checkbox (Auswahlkästchen) ist eine beschriftete Box, die standardmäßig entweder aktiviert (Häkchen ist sichtbar) oder deaktiviert ist (kein Häkchen). Sie wird am häufigsten verwendet, um dem Benutzer das Umschalten zwischen Optionen zu ermöglichen. Der Inhalt des XML-Tags wird als Text für das Auswahlkästchen verwendet.

Hier ist ein Beispiel, wie man ein Auswahlkästchen in XML erstellt:

```
<checkbox>Yes, I'm over 18 years old</checkbox>
```

13.2 Checkbox.Right

BEZEICHNUNG
Checkbox.Right – legt die Position des Auswahlkästchen-Textes fest

BESCHREIBUNG
Setzen Sie dies auf `True`, damit das Auswahlkästchen rechts neben dem zugehörigen Text erscheint. Standardmäßig wird das Auswahlkästchen links neben dem Text angezeigt.

TYP
Boolesch

ANWENDBARKEIT
I

13.3 Checkbox.Selected

BEZEICHNUNG
Checkbox.Selected – setzt oder ermittelt den das Auswahlkästchen-Status

BESCHREIBUNG
Setzt oder ermittelt den ausgewählten Zustand eines Checkbox-Objekts.

Sie können auch eine Benachrichtigung über dieses Attribut einrichten, um zu erfahren, dass der Benutzer den Status des Auswahlkästchen umschaltet.

TYP
Boolesch

ANWENDBARKEIT
ISGN

14 Choice-Klasse (Auswahl)

14.1 Übersicht

Mit dem Auswahl-Widgets (Choice) kann der Benutzer einen Eintrag aus einer vordefinierten Liste von Elementen auswählen. Nur das ausgewählte Element ist sichtbar, bis der Benutzer die Auswahlliste herunterzieht.

Wenn Sie ein Auswahl-Widget definieren, können Sie den Tag `<item>` verwenden, um das Auswahl-Widget mit Elementen zu füllen. Hier ist ein XML-Beispielauszug zur Erstellung eines Auswahl-Widget mit drei Einträgen:

```
<choice id="printer">
    <item>HP Deskjet</item>
    <item>NEC P6</item>
    <item>Okimate 20</item>
</choice>
```

Alternativ können Sie auch ein leeres Auswahl-Widgets erstellen und später mit der Methode `Choice.Insert` mit Einträgen füllen. So erstellen Sie ein leeres Auswahlwidget:

```
<choice/>
```

14.2 Choice.Active

BEZEICHNUNG

Choice.Active – setzt oder ermittelt die aktive Auswahl

BESCHREIBUNG

Setzt oder ermittelt den aktive Eintrag im Auswahl-Widget von Index 0 für den ersten Eintrag bis zur Anzahl der Einträge -1 für den letzten Eintrag.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, um zu erfahren, wenn der Benutzer die aktive Auswahlmöglichkeit ändert.

Sie können auch die speziellen Werte `Next` oder `Prev` übergeben, um durch die Einträge des Auswahl-Widget zu blättern.

TYP

Zahl oder Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

ISGN

14.3 Choice.Clear

BEZEICHNUNG

Choice.Clear – löscht alle Einträge (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Clear")
```

BESCHREIBUNG
Entfernt alle Einträge aus dem vom Auswahl-Widgets.

Beachten Sie, dass diese Methode unter AmigaOS und kompatiblen Betriebssystemen nur unter MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id` ID des Auswahl-Widgets

14.4 Choice.Count

BEZEICHNUNG
Choice.Count – ruft die Anzahl der Einträge im Auswahl-Widget ab (V1.1)

BESCHREIBUNG
Gibt die aktuelle Anzahl der Einträge im Auswahl-Widget zurück.

TYP
Zahl

ANWENDBARKEIT
G

14.5 Choice.GetEntry

BEZEICHNUNG
Choice.GetEntry – gibt einen Eintrag zurück (V1.1)

ÜBERSICHT
```
e$ = moai.DoMethod(id, "GetEntry", pos)
```

BESCHREIBUNG
Diese Methode gibt Ihnen den in `pos` angegebenen Eintrag von einem Auswahl-Widget zurück. Sie können entweder einen absoluten Index oder den speziellen Wert `Active` in `pos` angeben, um den aktiven Eintrag zu erhalten. `Choice.GetEntry` gibt dann den Eintrag als Zeichenkette zurück.

EINGABEN

`id` ID des Auswahl-Widget

`pos` Index des Eintrags oder `"Aktiv"`

RÜCKGABEWERTE

`e$` Eintrag vom angegebenen Index

14.6 Choice.Insert

BEZEICHNUNG

Choice.Insert – fügt neuen Eintrag ein (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Insert", pos, e$)
```

BESCHREIBUNG

Diese Methode fügt den in `e$` angegebenen Eintrag in das Auswahl-Widget ein. Die Einfügeposition wird im Argument `pos` angegeben und der neue Eintrag wird vor dem in `pos` angegebenen Eintrag eingefügt. Dies kann eine absolute Indexposition ab 0 für den ersten Eintrag oder einer der folgenden Sonderwerte sein:

`Top` Als ersten Eintrag einfügen.

`Active` Wird vor dem aktiven Eintrag eingefügt. Wenn es keinen aktiven Eintrag gibt, wird der Eintrag ganz oben in die Eintragsliste eingefügt.

`Bottom` Als letzten Eintrag einfügen.

Wenn `pos` größer oder gleich der Anzahl der Einträge im Auswahl-Widget ist, wird der Eintrag als letzter Eintrag eingefügt.

Beachten Sie, dass diese Methode unter AmigaOS und kompatiblen Betriebssystemen nur unter MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id` ID des Auswahl-Widget

`pos` Position als Absolutwert oder Sonderwert einfügen (siehe oben)

`e$` Eintrag, der eingefügt werden soll

14.7 Choice.Remove

BEZEICHNUNG

Choice.Remove – entfernt einen Eintrag aus dem Auswahl-Widget (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Remove", pos)
```

BESCHREIBUNG

Entfernt einen Eintrag aus einem Auswahl-Widget. Die Position kann als absoluter Indexwert oder als einer der folgenden Sonderwerte angegeben werden:

`First` Entfernt den ersten Eintrag.

`Active` Entfernt den aktiven Eintrag.

`Last` Entfernt den letzten Eintrag.

Wenn der aktive Eintrag entfernt wird, wird der folgende Eintrag aktiv.

Beachten Sie, dass diese Methode unter AmigaOS und kompatiblen Betriebssystemen nur unter MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id` ID des Auswahl-Widget

`pos` Index des zu entfernenden Eintrags oder einer der speziellen Werte (siehe oben)

14.8 Choice.Rename

BEZEICHNUNG

Choice.Rename – benennt einen Eintrag um (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Rename", pos, newname$)
```

BESCHREIBUNG

Benennt den Auswahleintrag an der in `pos` angegebenen Position in den in `newname$` genannten Namen um. `pos` kann eine absolute Indexposition ab 0 für den ersten Eintrag oder einer der folgenden Sonderwerte sein:

`Active` Benennt den aktiven Eintrag um.

Beachten Sie, dass diese Methode unter AmigaOS und kompatiblen Betriebssystemen nur unter MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id` ID des Auswahl-Widget

`pos` Index des Eintrags, der umbenannt wird oder einer der speziellen Werte (siehe oben)

`newname$` Neuer Name des Eintrags

15 Combobox-Klasse (Textlisten)

15.1 Übersicht

Die Combobox-Klasse (Textlisten) erzeugt ein Widget, das eine Kombination aus einer Liste und einem Text-Widget ist. Der Benutzer kann entweder einen Eintrag aus einer vordefinierten Liste von Einträgen auswählen oder einzelne Daten in das Text-Widget eingeben.

Wenn Sie ein Combobox-Objekt erstellen, können Sie den Tag `<item>` verwenden, um es mit Einträgen zu füllen. Hier ist ein Beispiel:

```
<combobox>
   <item>The</item>
   <item>quick</item>
   <item>brown</item>
   <item>fox</item>
   <item>jumps</item>
   <item>over</item>
   <item>the</item>
   <item>lazy</item>
   <item>dog</item>
</combobox>
```

Sie können auch eine leere Textliste erstellen und diese später mit Einträgen füllen, indem Sie die Methode `Combobox.Insert` verwenden. So erstellen Sie eine leere Textliste:

```
<combobox/>
```

15.2 Combobox.Clear

BEZEICHNUNG

Combobox.Clear – entfernt alle Textlisten-Einträge (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Clear")
```

BESCHREIBUNG

Entfernt alle Einträge aus der in `id` angegebene Textliste.

EINGABEN

`id`	ID des Combobox-Objekts

15.3 Combobox.Count

BEZEICHNUNG

Combobox.Count – gibt die Anzahl der Einträge in der Textliste zurück (V1.1)

BESCHREIBUNG

Gibt die aktuelle Anzahl der Einträge in der Textliste zurück.

TYP

Zahl

ANWENDBARKEIT
G

15.4 Combobox.GetEntry

BEZEICHNUNG
Combobox.GetEntry – gibt den Textlisten-Eintrag zurück (V1.1)

ÜBERSICHT
```
e$ = moai.DoMethod(id, "GetEntry", pos)
```

BESCHREIBUNG
Diese Methode gibt den in Position `pos` angegebenen Eintrag aus dem Textlisten-Widget zurück.

EINGABEN

`id`	ID des Combobox-Objekts
`pos`	Index des Eintrags

RÜCKGABEWERTE

`e$`	Eintrag vom angegebenen Index

15.5 Combobox.Insert

BEZEICHNUNG
Combobox.Insert – fügt einen neuen Eintrag ein (V1.1)

ÜBERSICHT
```
moai.DoMethod(id, "Insert", pos, e$)
```

BESCHREIBUNG
Mit dieser Methode fügen Sie den in `e$` angegebenen Eintrag in die Textliste ein. Die Einfügeposition wird im Argument `pos` angegeben und der neue Eintrag wird vor dem in `pos` angegebenen Eintrag eingefügt. Dies kann eine absolute Indexposition ab 0 für den ersten Eintrag oder einer der folgenden Sonderwerte sein:

`Top`	Als ersten Eintrag einfügen.
`Bottom`	Als letzten Eintrag einfügen.

Wenn `pos` größer oder gleich der Anzahl der Einträge in der Textliste ist, wird der Eintrag als letzter Eintrag eingefügt.

EINGABEN

`id`	ID des Combobox-Objekts
`pos`	fügt die Position als Absolut- oder Sonderwert ein (siehe oben)
`e$`	Eintrag, der eingefügt werden soll

15.6 Combobox.Remove

BEZEICHNUNG

Combobox.Remove – entfernt einen Eintrag aus der Textliste (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Remove", pos)
```

BESCHREIBUNG

Diese Methode entfernt der an der in `pos` angegebenen Eintrag aus der Textliste.

EINGABEN

`id`	ID des Combobox-Objekts
`pos`	Index des Eintrags, der entfernt wird

15.7 Combobox.Rename

BEZEICHNUNG

Combobox.Rename – benennt einen Eintrag um (V1.1)

ÜBERSICHT

```
moai.DoMethod(id, "Rename", pos, newname$)
```

BESCHREIBUNG

Hiermit benennen Sie den Textlisten-Eintrag an der angegebenen Position in den in `newname$` angegebenen Namen um. Die Eingabeposition wird im Argument `pos` angegeben. Dies muss eine absolute Indexposition sein, die beim ersten Eintrag mit 0 beginnt.

EINGABEN

`id`	ID des Combobox-Objekts
`pos`	Eingabeposition als absolute Zahl
`newname$`	Neuer Name für den Textlisten-Eintrag

15.8 Combobox.Value

BEZEICHNUNG

Combobox.Value – setzt oder ermittelt den aktuellen Inhalt der Textliste

BESCHREIBUNG

Holt oder setzt den aktuellen Inhalt der Textliste.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, die ausgelöst wird, sobald sich der Inhalt des Combobox-Objekts ändert.

TYP

Zeichenkette

ANWENDBARKEIT

ISGN

16 Dialog-Klasse

16.1 Übersicht

Die Dialogklasse ist eine Unterklasse der Window-Klasse (Fenster) und erzeugt modale Dialoge. Modale Dialoge sind spezielle Top-Level-Fenster, die den Rest des Programmes blockieren, bis sie geschlossen werden. Dialoge werden typischerweise verwendet, wenn Benutzeraktionen erforderlich sind, um mit dem Programm fortzufahren oder um anzuzeigen, dass das Programm gerade beschäftigt ist. Ein Dialog könnte dann z.B. einen Fortschrittsbalken anzeigen.

Wie bei Fenstern muss das Wurzelelement eines Dialogs immer ein einzelnes Gruppenobjekt sein, d.h. eine Instanz der Group-Klasse (Gruppen). Siehe Abschnitt 17.1 [Group-Klasse], Seite 87, für Details. Darüber hinaus ist es nicht erlaubt, mehrere Elemente auf der Stammebene des Dialogs zu haben. Sie müssen nur ein einzelnes Gruppenobjekt als Wurzel-Element verwenden. Hier ein Beispiel für einen einfachen Fortschrittsdialog in XML:

```
<dialog id="mydlg" title="Working...">
   <vgroup>
      <progressbar id="prg"/>
      <button>Cancel</button>
   </vgroup>
</dialog>
```

Sie sollten Dialoge nur dann erstellen, wenn Sie sie brauchen, und sie aus dem Speicher löschen, sobald Sie damit fertig sind. Es wird nicht empfohlen, alle Ihre Dialoge beim Start mit `moai.CreateApp()` zu erstellen und sie ständig im Speicher zu behalten. Stattdessen sollten Sie `moai.CreateDialog()` verwenden, um einen Dialog zu erstellen, wenn Sie ihn benötigen und ihn dann anschliessend aus dem Speicher löschen. Der Grund dafür ist, dass Fenster auf einigen von RapaGUI unterstützten Betriebssystemen eine begrenzte Ressource sind. Zum Beispiel gibt es unter Windows ein Limit von ungefähr 10.000 Fenstern pro Prozess. Das klingt vielleicht ausreichend genug, aber bedenken Sie, dass unter Windows jedes Widget ein "Fenster" für das Betriebssystem ist, z.B. alle Beschriftungen, Schaltflächen, Rahmen, Auswahlkästchen, Gruppe, etc. in Ihrem Programm ist ein Fenster. So sollten Sie darauf achten, dass Sie Ihre Dialoge nur bei Bedarf mit `moai.CreateDialog()` erstellen und anschließend direkt wieder löschen.

In der Praxis empfiehlt es sich, für jeden Ihrer Dialoge eine eigene XML-Datei zu erstellen und dann `moai.CreateDialog()` zu verwenden, um die XML-Datei zur Laufzeit in einen Dialog zu konvertieren und anschliessend den Dialog automatisch von RapaGUI löschen zu lassen, sobald der Benutzer ihn schließt.

Als Unterklasse der Window-Klasse können die meisten Attribute und Methoden der Window-Klasse auch mit der Dialog-Klasse verwendet werden. Siehe Abschnitt 58.1 [Window-Klasse], Seite 263, für Details. Beachten Sie jedoch, dass Dialoge nicht durch Setzen von `Window.Open` geöffnet werden, sondern durch Ausführen der Methode `Dialog.ShowModal`. Um einen Dialog zu schließen, rufen Sie einfach die Methode `Dialog.EndModal` auf. Dies löscht auch automatisch den Dialog aus dem Speicher, d.h. es wird implizit `moai.FreeDialog()` auf den Dialog aufgerufen, es sei denn, Sie fordern ausdrücklich, dass der Dialog durch Setzen eines optionalen Arguments auf `True` nicht gelöscht werden soll.

In ähnlicher Weise wird der Dialog auch automatisch gelöscht, wenn der Benutzer auf das Schließsymbol eines Dialogs klickt. Wenn Sie dies nicht möchten oder wenn Sie das Verhalten des Programmes für das Schließsymbol anpassen müssen, brauchen Sie eine Überwachung mit dem Attribut `Window.CloseRequest`. Wenn für dieses Attribut keine Überwachung vorhanden ist, ruft RapaGUI einfach `Dialog.EndModal` mit dem Parameter 0 auf, sobald auf das Schließsymbol des Dialogs geklickt wird.

Beachten Sie, dass es zwei verschiedene Möglichkeiten gibt, einen Dialog zu verwalten: Mit oder ohne Dialogfunktion. Siehe Abschnitt 16.3 [Dialog.ShowModal], Seite 85, für Details.

16.2 Dialog.EndModal

BEZEICHNUNG

Dialog.EndModal – schließt den Dialog und beendet die Modal-Schleife

ÜBERSICHT

```
moai.DoMethod(id, "EndModal", retval[, nodestroy])
```

BESCHREIBUNG

Diese Methode schließt den angegebenen Dialog, der mit `Dialog.ShowModal` geöffnet wurde und bricht seine Modalschleife ab. Der Wert, den Sie in `retval` übergeben, ist dann der Rückgabewert des Aufrufs der Methode `Dialog.ShowModal` dieses Dialogs. Dieser Wert wird oft verwendet, um Erfolg oder Misserfolg anzuzeigen, d.h. Sie können `True` zurückgeben, wenn der Benutzer die Taste `"OK"` gedrückt hat, andernfalls `False`.

Zusätzlich löscht automatisch diese Methode den Dialog aus dem Speicher, d.h. sie ruft implizit den Befehl `moai.FreeDialog()` für dieses Dialogobjekt auf. Wenn Sie dies nicht wollen, müssen Sie das optionale Argument `nodestroy` auf `True` setzen. Es wird jedoch empfohlen und es ist eine gute Programmierpraxis, jeden Dialog aus dem Speicher zu löschen, sobald Sie damit fertig sind. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details.

Bitte beachten Sie, dass Sie `Dialog.EndModal` nur für Dialoge aufrufen sollten, die keine Dialogfunktion verwenden. Für dialogfunktionsbasierte Dialoge ist es besser, einfach die beiden Werte `retval` und `nodestroy` aus Ihrer Dialogfunktion zurückzugeben. Wenn Sie `Dialog.EndModal` aus einer Dialogfunktion aufrufen, springt RapaGUI sofort zu dem Punkt zurück, an dem `Dialog.ShowModal` aufgerufen wurde, was verwirrend sein kann. Deshalb ist das Beenden von Dialogen mit Rückgabewerten eine viel sauberere Lösung und der empfohlene Weg, Dialoge mit Dialogfunktion zu schließen. Siehe Abschnitt 16.3 [Dialog.ShowModal], Seite 85, für Details.

EINGABEN

`id` ID des Dialogobjekts

`retval` gewünschter Rückgabewert für `Dialog.ShowModal`

`nodestroy`
optional: `True`, wenn diese Methode den Dialog nicht automatisch löschen soll (Standard ist `False`, was bedeutet, dass der Dialog nach dem Schließen aus dem Speicher gelöscht wird).

16.3 Dialog.ShowModal

BEZEICHNUNG

Dialog.ShowModal – öffnet einen Dialog und startet die Modal-Schleife

ÜBERSICHT

```
retval = moai.DoMethod(id, "ShowModal"[, dlgfunc, userdata])
```

BESCHREIBUNG

Diese Methode öffnet das Dialogfenster und startet eine modale Schleife, die den Rest der Anwendung blockiert und darauf wartet, dass der Dialog geschlossen wird, bevor die Kontrolle an das Skript zurückgegeben wird. Es gibt zwei verschiedene Möglichkeiten, einen Dialog zu verwalten: Mit oder ohne Dialogfunktion. Wenn Sie eine Hollywood-Funktion in `dlgfunc` übergeben, wird RapaGUI diese Funktion direkt nach dem Öffnen des Dialogs ausführen und `Dialog.ShowModal` wird beendet, sobald `dlgfunc` abgeschlossen ist oder der Benutzer den Dialog schließt. Wenn Sie keine Hollywood-Funktion übergeben, wird `Dialog.ShowModal` beendet, wenn `Dialog.EndModal` aufgerufen wird oder der Benutzer den Dialog schließt.

Typischerweise verwenden Sie eine Dialogfunktion für Dialoge, die dazu dienen, den Fortschritt zu visualisieren, während die Anwendung im Hintergrund arbeitet. In diesem Fall würde die Dialogfunktion von Zeit zu Zeit einen Fortschrittsbalken aktualisieren, um eine visuelle Rückmeldung über den erzielten Fortschritt zu geben. Dialoge, die den Benutzer lediglich auffordern, bestimmte Informationen anzugeben, brauchen in der Regel keine Dialogfunktion zu verwenden, da keine Arbeit im Hintergrund ausgeführt wird, während der Dialog geöffnet ist.

Wenn Sie eine Dialogfunktion verwenden, ist es sehr wichtig, dass Sie den Befehl `CheckEvent()` von Hollywood regelmäßig aufrufen, um die GUI ansprechbar zu halten und sicherzustellen, dass auf Widget-Ereignisse reagiert wird. Zum Beispiel könnte es eine Schaltfläche "Abbrechen" in Ihrem Fortschrittsbalken-Dialog geben. Wenn Sie den Befehl `CheckEvent()` nicht regelmäßig aufrufen, kann auf das Drücken der Schaltfläche "Abbrechen" nicht reagiert werden. Stellen Sie daher sicher, dass Sie `CheckEvent()` mehrmals pro Sekunde aufrufen.

Bei Verwendung einer Dialogfunktion können Sie auch den optionalen Parameter `userdata` verwenden, um Ihrer Dialogfunktion einen beliebigen Benutzerdatenwert zu übergeben.

Die in `dlgfunc` angegebene Funktion wird mit einem einzigen Tabellenargument aufgerufen, das die folgenden Felder initialisiert hat:

`Action:` Enthält "RapaGUI".

`Class:` Enthält "Dialog".

`Attribute:`
Enthält "ShowModal".

`ID:` Enthält die ID des Dialogobjekts.

`UserData:`
Enthält den Benutzerdatenwert, den Sie an `Dialog.ShowModal` übergeben haben. Wenn Sie keine Benutzerdaten übergeben, wird dieses Feld überhaupt nicht initialisiert.

Wenn Ihre Dialogfunktion abgeschlossen ist, wird der Dialog geschlossen und `Dialog.ShowModal` wird automatisch beendet. Der Rückgabewert von `Dialog.ShowModal` ist der erste Wert, den Ihre Dialogfunktion zurückgegeben hat. Der zweite Rückgabewert heißt `nodestroy`. Es zeigt an, ob der Dialog automatisch aus dem Speicher gelöscht werden soll oder nicht. Wenn Sie hier `False` zurückgeben, wird Ihr Dialog automatisch nach dem Schließen aus dem Speicher gelöscht (dies ist der Standardwert) und wenn Sie `True` zurückgeben, wird er nicht gelöscht.

Hier ist ein Beispiel:

```
Function p_DialogFunc(msg)
   For Local k = 0 To 100
      ...do some work...
      moai.Set("prgbar", "level", k)
      CheckEvent() ; Sehr wichtig! Siehe oben!
      If abortpressed=True Then Return(False,False)
      Wait(1)      ; verhindert 100% CPU-Leistung fürs Skript
   Next
   Return(True,False)
EndFunction
```

Die obige Funktion gibt `True` zurück, wenn der Dialog seine Arbeit beendet hat und `False`, wenn der Benutzer die Schaltfläche "Abbrechen" gedrückt hat. Damit dies funktioniert, müssen Sie auch eine Schaltflächen-Callback-Funktion einrichten, die die Variable `abortpressed` auf `True` setzt, wenn der Benutzer die Schaltfläche "Abort" drückt. Außerdem wird `False` zurückgegeben, damit RapaGUI den Dialog nach dem Schließen automatisch löscht. Beachten Sie, dass die beiden Werte, die Ihre Dialogfunktion zurückgeben soll, einfach die beiden Parameter widerspiegeln, die von der Methode `Dialog.EndModal` akzeptiert werden, die zum Schließen von Dialogen ohne Dialogfunktion verwendet wird.

Beachten Sie, dass es eine gute Programmierpraxis ist, Dialoge nach Bedarf zu erstellen und zu löschen. Sie sollten sie nicht für die gesamte Lebensdauer Ihres Programmes im Speicher behalten, sondern nur bei Bedarf erstellen und löschen, sobald Sie damit fertig sind. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details.

Wenn Sie benachrichtigt werden möchten, wenn der Benutzer auf das Schließsymbol eines Dialogs klickt, müssen Sie eine Überwachung mit dem Attribut `Window.CloseRequest` installieren. Wenn es keine Überwachung auf diesem Attribut gibt, ruft RapaGUI beim Klick auf das Schließsymbol einfach `Dialog.EndModal` mit 0 als Parameter auf.

EINGABEN

`id`	ID des Dialogobjekts
`dlgfunc`	optional: Funktion, die den Dialog verwaltet
`userdata`	optional: Benutzerdaten, welche an die Dialogfunktion übergeben werden

RÜCKGABEWERTE

`retval`	Rückgabewert des Dialogs

BEISPIEL

`Siehe @command{moai.CreateObject()}`

17 Group-Klasse (Gruppen)

17.1 Übersicht

Die Group-Klasse (Gruppen) kann verwendet werden, um eine Reihe von Elementen auf verschiedene Weise zu gestalten. Wenn sich die Größe des übergeordneten Objekts ändert, passen sich dementsprechend die Größe ihrer Elementen an, was sehr nützlich sein kann, um in alle Richtungen frei skalierbare Fenster zu erstellen.

Die folgenden verschiedenen Gruppentypen werden von RapaGUI unterstützt:

`<hgroup>` Gruppenelemente werden in einer Reihe (horizontal, waagrecht) angeordnet.

`<vgroup>` Gruppenelemente werden in einer Spalte (vertikal, senkrecht) angeordnet.

`<colgroup>`
: Gruppenelemente werden in mehreren Spalten angeordnet.

`<scrollgroup>`
: Eine Gruppe mit Bildlaufleiste. Dies ist eine spezielle Gruppe, die eine eingebettete Gruppe mit Bildlaufleiste anzeigt. Siehe Abschnitt 41.1 [Scrollgroup-Klasse], Seite 189, für Details.

Spaltengruppen sind nützlich, wenn Sie identische Widget-Größen für alle Ihre Elemente in einer Gruppe benötigen, um ein angenehmeres Erscheinungsbild zu erhalten. Stellen Sie sich zum Beispiel ein Formular vor, das aus Texteingabe-Widgets und Textobjekten besteht. Es wird empfohlen, hier ein `<colgroup>` zu verwenden, da es zu einem klaren und geordneten Erscheinungsbild führt. Hier ist ein Beispiel:

```
<colgroup columns="2">
   <label>Name</label>
   <textentry/>
   <label>Street</label>
   <textentry/>
   <label>City</label>
   <textentry/>
   <label>Zip code</label>
   <textentry/>
   <label>Country</label>
   <textentry/>
   <label>Telephone</label>
   <textentry/>
   <label>Email</label>
   <textentry/>
</colgroup>
```

Wenn wir `<vgroup>` mit `<hgroup>` pro Zeile verwenden würden, wäre die Darstellung ziemlich schlecht, da die Breite des Texteingabe-Widgets für jede Zeile unterschiedlich ist, was ziemlich hässlich aussieht.

Beachten Sie, dass die Gruppenklasse nicht von der Area-Klasse (Bereich) abgeleitet wird, da Gruppen nicht als physische Widgets existieren, sondern nur als Layoutwerkzeuge für

ihre untergeordneten Widgets oder Gruppen. Aus diesem Grund können Sie keine Attribute und Methoden der Area-Klasse für Gruppenobjekte verwenden.

17.2 Group.Append

BEZEICHNUNG

Group.Append – fügt das losgelöste Objekt als letztes Gruppenelement hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Append", obj)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das in `obj` angegebene losgelöste Objekt dem durch `id` angegebenen Gruppenobjekt hinzuzufügen. Das losgelöste Objekt wird als letztes untergeordnetes Objekt der Gruppe hinzugefügt. Nachdem diese Methode abgeschlossen ist, ändert das angegebene Objekt seinen Status von gelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte mit diesem Objekt verwenden.

Bevor Sie diese Methode aufrufen können, müssen Sie die Gruppe in einen speziellen Zustand versetzen, in dem Elemente hinzugefügt und entfernt werden können. Dies kann durch Ausführen der Methoden `Group.InitChange` und `Group.ExitChange` für das jeweilige Gruppenobjekt erfolgen.

Losgelöste MOAI-Objekte können entweder durch Aufrufen des Befehls `moai.CreateObject()` oder durch explizites Trennen ihrer MOAI-Objekte mithilfe der Methode `Group.Remove` erstellt werden.

EINGABEN

`id`	ID des Gruppenobjekts
`obj`	ID des anzuhängenden Objekts

BEISPIEL

```
Siehe @command{moai.CreateObject()}
```

17.3 Group.Color

BEZEICHNUNG

Group.Color – setzt die Hintergrundfarbe der Gruppe

BESCHREIBUNG

Legen Sie die Hintergrundfarbe für die Gruppe fest. Dies darf nur mit Objekten wie `<hgroup>` oder `<vgroup>` verwendet werden. Es wird nicht beim Objekt `<colgroup>` unterstützt. Außerdem dürfen Sie `Group.Frame` nicht auf `True` setzen, wenn Sie dieses Attribut verwenden. Wenn Sie die Hintergrundfarbe einer gerahmten Gruppe einstellen möchten, müssen Sie eine Hilfsgruppe um die Gruppe herum anlegen, deren Hintergrundfarbe Sie dann einstellen, z.B. so:

```
<vgroup frame="true" padding="0">
    <vgroup color="#ffffff">
```

```
            ...
        </vgroup>
    </vgroup>
```

Wenn Sie die Hintergrundfarbe der Gruppe `<colgroup>` setzen wollen, können Sie auch einfach eine Hilfsgruppe um die Spaltengruppe herum anlegen (siehe oben).

TYP
Zahl

ANWENDBARKEIT
I

17.4 Group.Columns

BEZEICHNUNG
Group.Columns – definiert die Anzahl Gruppenspalten

BESCHREIBUNG
Legen Sie die Anzahl der Spalten für eine zweidimensionale Gittergruppe fest. Wenn Sie diesen Tag angeben, stellen Sie sicher, dass das Gitter ausgeglichen ist, d.h. es gibt keinen Rest bei der Division der Gesamtzahl der Elemente durch die Anzahl der Spalten.

Dieses Attribut darf nur mit dem Tag `<colgroup>` verwendet werden.

TYP
Zahl

ANWENDBARKEIT
I

17.5 Group.ExitChange

BEZEICHNUNG
Group.ExitChange – beendet den Gruppenänderungsmodus

ÜBERSICHT
`moai.DoMethod(id, "ExitChange"[, force])`

BESCHREIBUNG
Diese Methode beendet den von `Group.InitChange` festgelegten Modus. Wenn Elemente hinzugefügt oder entfernt wurden, aktualisiert RapaGUI die Gruppe und macht die Änderungen für den Benutzer sichtbar. Sie können RapaGUI zwingen, diese Aktualisierung durchzuführen, indem Sie das Argument `force` auf `True` setzen. In diesem Fall aktualisiert RapaGUI immer die gesamte Gruppe, unabhängig davon, ob Objekte hinzugefügt oder entfernt wurden. Das Erzwingen einer Aktualisierung ist nützlich, wenn sich in Ihrer Gruppe ein Objekt befindet, für das eine Aktualisierung erzwungen werden soll.

EINGABEN

`id`	ID des Gruppenobjekts

`force` optional: Geben Sie hier `True` an, um eine vollständige Aktualisierung zu erzwingen. Andernfalls aktualisiert RapaGUI die Gruppe nur, wenn Objekte hinzugefügt oder entfernt wurden

BEISPIEL

`Siehe @command{moai.CreateObject()}`

17.6 Group.Frame

BEZEICHNUNG

Group.Frame – erstellt eine gerahmte Gruppe

BESCHREIBUNG

Setzen Sie dieses Attribut auf `True`, um einen Rahmen um diese Gruppe hinzuzufügen. Sie können auch einen Titel für den Rahmen hinzufügen, indem Sie das Attribut `Group.FrameTitle` verwenden.

TYP

Boolesch

ANWENDBARKEIT

I

17.7 Group.FrameTitle

BEZEICHNUNG

Group.FrameTitle – setzt den Titel für die gerahmte Gruppe

BESCHREIBUNG

Wenn Sie eine gerahmte Gruppe erstellen, indem Sie `Group.Frame` auf `True` setzen, können Sie mit diesem Attribut für den Rahmen einen Titeltext hinzufügen. Dieses Attribut darf nur verwendet werden, wenn `Group.Frame` auf `True` gesetzt wurde.

TYP

Zeichenkette

ANWENDBARKEIT

I

17.8 Group.HAlign

BEZEICHNUNG

Group.HAlign – setzt die Standard-Horizontalausrichtung

BESCHREIBUNG

Setzen Sie dieses Attribut, um festzulegen, wie nicht veränderbare Elemente ausgerichtet werden sollen, wenn sie kleiner als die ihnen zugewiesene Größe sind. Die folgenden Optionen stehen zur Verfügung:

`Left` Links ausgerichtet.

`Right` Rechts ausgerichtet.

`Center` Zentriert ausgerichtet. Dies ist auch voreingestellt.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
I

17.9 Group.Hide

BEZEICHNUNG
Group.Hide – blendet die Gruppe ein/aus

BESCHREIBUNG
Setzen Sie dieses Attribut, um eine Gruppe ein- oder auszublenden. Beachten Sie, dass Gruppen ihren privaten Sichtbarkeitsstatus beibehalten, so dass das Setzen von diesem Attribut nicht bedeutet, dass die Gruppe einfach `Area.Hide` auf alle ihre Elemente setzt. Sowohl Widgets als auch Gruppen behalten ihren eigenen Sichtbarkeitsstatus.

TYP
Boolesch

ANWENDBARKEIT
ISG

17.10 Group.HorizSpacing

BEZEICHNUNG
Group.HorizSpacing – setzt den horizontalen Abstand

BESCHREIBUNG
Legen Sie die Anzahl der Pixel fest, die als horizontaler Abstand zwischen den Elementen der Gruppe verwendet werden soll.

TYP
Zahl

ANWENDBARKEIT
I

17.11 Group.Icon

BEZEICHNUNG
Group.Icon – setzt das Gruppen-Symbol für die Seitenansicht

BESCHREIBUNG

Wenn Sie Gruppen für eine Seitenansicht definieren, können Sie mit diesem Attribut ein Symbol für eine Gruppe angeben, das in dem Widget zum Durchblättern der einzelnen Seiten angezeigt werden soll. Siehe Abschnitt 31.1 [Pageview-Klasse (Seitenansicht)], Seite 157, für Details. Setzen Sie dieses Attribut einfach auf den Identifikator eines Hollywood-Pinsels, der in der Seitenansicht als Symbol dieser Gruppe angezeigt werden soll. Um das Symbol von einer Gruppenseite zu entfernen, setzen Sie dieses Attribut auf -1.

Bitte lesen Sie auch den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Ab RapaGUI 1.1 hat dieses Attribut eine Anwendbarkeit von `ISG`. Beachten Sie, dass unter AmigaOS und Kompatiblen das Ändern des Gruppensymbols zur Laufzeit für alle Seitenansichtsmodi (außer dem Listenmodus) nur unter MUI 4.0 oder höher verfügbar ist.

TYP

Zahl

ANWENDBARKEIT

ISG

17.12 Group.InitChange

BEZEICHNUNG

Group.InitChange – bereitet die Gruppe für das Hinzufügen/Entfernen von Elementen vor

ÜBERSICHT

```
moai.DoMethod(id, "InitChange")
```

BESCHREIBUNG

Diese Methode bereitet eine Gruppe zum Hinzufügen oder Entfernen von Elementen vor. Wann immer Sie Methoden wie `Group.Append`, `Group.Prepend`, `Group.Insert` oder `Group.Remove` verwenden, müssen Sie zuerst diese Methode aufrufen, um die Gruppe in diesen speziellen Austauschmodus zu versetzen. Wenn Sie fertig sind, verwenden Sie `Group.ExitChange`, um die Gruppe neu zu gestalten.

EINGABEN

`id` ID des Gruppenobjekts

BEISPIEL

```
Siehe @command{moai.CreateObject()}
```

17.13 Group.Insert

BEZEICHNUNG

Group.Insert – fügt ein losgelöstes Objekt nach dem angegebenen Element ein

ÜBERSICHT

```
moai.DoMethod(id, "Insert", obj, pred)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` spezifizierte losgelöste Objekt in das durch `id` spezifizierte Gruppenobjekt einzufügen. Das losgelöste Objekt wird nach dem in `pred` angegebenen Element eingefügt. Nachdem diese Methode beendet ist, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Bevor Sie diese Methode aufrufen können, müssen Sie die Gruppe in einen speziellen Zustand versetzen, in dem Elemente hinzugefügt und entfernt werden können. Dies kann durch Ausführen der Methoden `Group.InitChange` und `Group.ExitChange` für das jeweilige Gruppenobjekt erfolgen.

Losgelöste MOAI-Objekte können entweder durch Aufrufen des Befehls `moai.CreateObject()` oder durch explizites Trennen ihrer MOAI-Objekte mithilfe der Methode `Group.Remove` erstellt werden.

EINGABEN

`id`	ID des Gruppenobjekts
`obj`	ID des einzufügenden Objekts
`pred`	Das Objekt wird nach diesem Objekt eingefügt

BEISPIEL

```
Siehe @command{moai.CreateObject()}
```

17.14 Group.Padding

BEZEICHNUNG

Group.Padding – legt den Abstand zwischen Rahmen und Gruppe fest

BESCHREIBUNG

Wenn Sie eine gerahmte Gruppe erstellen, können Sie mit diesem Attribut die Anzahl der Füll-Pixel zwischen dem Rahmen und den Elementen der Gruppe festlegen.

Unter Windows, Linux und Mac OS X definiert dieses Attribut auch die Anzahl der Füll-Pixel um die Gruppe herum, wenn die Gruppe Teil eines Pageview-Objekts (Seitenansicht) ist. Siehe Abschnitt 31.1 [Pageview-Klasse], Seite 157, für Details.

TYP

Zahl

ANWENDBARKEIT

I

17.15 Group.Paint

BEZEICHNUNG

Group.Paint – fordert eine Zeichnungsbenachrichtigung an

BESCHREIBUNG

Richten Sie eine Überwachung für dieses Attribut ein, damit Ihre Callback-Funktion immer dann aufgerufen wird, wenn der Gruppenhintergrund gezeichnet werden muss. Ihre Callback-Funktion kann dann eigene Grafiken in den Gruppenhintergrund zeichnen.

RapaGUI übergibt den Identifikator eines Hollywood-Pinsels, dessen Größe genau so groß ist wie der Gruppenhintergrund. Sie müssen dann die gewünschten Hintergrundgrafiken zu diesem Pinsel zeichnen. Genau, Sie müssen nur in dem Rechteck zeichnen, das durch die vier Koordinaten `X`, `Y`, `Width` und `Height` definiert ist, die auch an Ihre Callback-Funktion übergeben werden. Diese vier Koordinaten beschreiben einen rechteckigen Bereich innerhalb der Dimensionen des Pinsels, der an Ihre Callback-Funktion übergeben wird. Wenn ein vollständiges Neuzeichnen benötigt wird, sind `X` sowie `Y` 0 und `Width` sowie `Height` stimmen mit den Abmessungen des Pinsels überein. Meistens wird jedoch nur ein partielles Neuzeichnen benötigt und dann müssen Sie nur auf den Teil des Pinsels zeichnen, der durch diese Koordinaten definiert ist.

Die folgenden zusätzlichen Argumente werden an Ihre Callback-Funktion übergeben:

`Brush:` Enthält den Identifikator eines Pinsels, auf den Sie zeichnen müssen. Verwenden Sie den Befehl `SelectBrush()` von Hollywood, um diesen Pinsel als Ausgabegerät in Ihrer Callback-Funktion auszuwählen. Vergessen Sie nicht, `EndSelect()` aufzurufen, wenn Sie fertig sind!

`ViewWidth:` Enthält die gesamte Gruppenbreite. Diese ist auch identisch mit der Breite des Pinsels, der an Ihre Callback-Funktion übergeben wird.

`ViewHeight:` Enthält die gesamte Gruppenhöhe. Diese ist auch identisch mit der Höhe des Pinsels, der an Ihre Callback-Funktion übergeben wird.

`X:` Enthält die x-Position innerhalb des Pinsels, an der Sie mit dem Zeichnen beginnen sollen. Siehe oben für Details.

`Y:` Enthält die y-Position innerhalb des Pinsels, an der Sie mit dem Zeichnen beginnen sollen. Siehe oben für Details.

`Width:` Enthält die Anzahl der Spalten, die Sie auf den Pinsel malen sollten (ab `X`). Siehe oben für Details.

`Height:` Enthält die Anzahl der Zeilen, die Sie auf den Pinsel malen sollten (ab `Y`). Siehe oben für Details.

Beachten Sie, dass `Group.Paint` nur mit Objekten wie `<hgroup>` oder `<vgroup>` verwendet werden darf. `<colgroup>`-Objekte werden nicht unterstützt. Außerdem dürfen Sie `Group.Frame` nicht auf `True` setzen, wenn Sie dieses Attribut verwenden. Wenn Sie den Hintergrund einer gerahmten Gruppe zeichnen möchten, müssen Sie eine Hilfsgruppe um die Gruppe herum erstellen, auf deren Hintergrund Sie zeichnen möchten, z.B. so:

```
<vgroup frame="true" padding="0">
```

```
        <vgroup notify="paint">
            ...
        </vgroup>
    </vgroup>
```

Wenn Sie den Hintergrund eines `<colgroup>` zeichnen möchten, können Sie auch einfach eine Hilfsgruppe um die Spaltengruppe herum erstellen (siehe oben).

Beachten Sie, dass dieses Attribut bei AmigaOS und Kompatiblen nur mit MUI 4.0 oder höher verfügbar ist.

TYP

Boolesch

ANWENDBARKEIT

N

17.16 Group.Prepend

BEZEICHNUNG

Group.Prepend – fügt ein losgelöstes Objekt als erstes Gruppenelement hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Prepend", obj)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` angegebene losgelöste Objekt dem durch `id` angegebene Gruppenobjekt hinzuzufügen. Das losgelöste Objekt wird als erstes Element der Gruppe hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Bevor Sie diese Methode aufrufen können, müssen Sie die Gruppe in einen speziellen Zustand versetzen, in dem Elemente hinzugefügt und entfernt werden können. Dies kann durch Ausführen der Methoden `Group.InitChange` und `Group.ExitChange` für das jeweilige Gruppenobjekt erfolgen.

Losgelöste MOAI-Objekte können entweder durch Aufrufen des Befehls `moai.CreateObject()` oder durch explizites Trennen ihrer MOAI-Objekte mithilfe der Methode `Group.Remove` erstellt werden.

EINGABEN

`id`	ID des Gruppenobjekts
`obj`	ID des einzufügenden Objekts

BEISPIEL

```
Siehe @command{moai.CreateObject()}
```

17.17 Group.Remove

BEZEICHNUNG

Group.Remove – löst das Objekt von der Gruppe

ÜBERSICHT

```
moai.DoMethod(id, "Remove", obj)
```

BESCHREIBUNG

Mit dieser Methode kann das angegebene Objekt von der angegebenen Gruppe gelöst werden. Nachdem diese Methode ausgeführt wurde, wird das angegebene Objekt seinen Zustand von angehängt auf losgelöst ändern. Das bedeutet, dass Sie es nun mit der Methode `Group.Insert` an eine andere Gruppe anhängen oder mit mit dem Befehl `moai.FreeObject()` aus dem Speicher löschen können.

Bevor Sie diese Methode aufrufen können, müssen Sie die Gruppe in einen speziellen Zustand versetzen, in dem Elemente hinzugefügt und entfernt werden können. Dies kann durch Ausführen der Methoden `Group.InitChange` und `Group.ExitChange` für das jeweilige Gruppenobjekt erfolgen.

EINGABEN

`id`	ID des Gruppenobjekts
`obj`	ID des zu loslösenden Objekts

BEISPIEL

```
moai.DoMethod("mygroup", "initchange")
moai.DoMethod("mygroup", "remove", "mychild")
moai.DoMethod("mygroup", "exitchange", false)
```

Der obige Code entfernt das Element "mychild" aus der Gruppe "mygroup". Sie können dann "mychild" an eine andere Gruppe anhängen oder aus dem Speicher löschen.

17.18 Group.SameSize

BEZEICHNUNG

Group.SameSize – stellt die gleiche Größe für alle Elemente ein

BESCHREIBUNG

Setzen Sie dieses Attribut auf `True`, um die Gruppe zu zwingen, die gleiche Größe für alle Elemente zu verwenden. Dadurch entsteht oft ein einheitlicheres und besseres Erscheinungsbild. RapaGUI setzt dieses Attribut automatisch für horizontale Gruppen, die eine Reihe von Schaltflächen enthalten.

TYP

Boolesch

ANWENDBARKEIT

I

17.19 Group.Spacing

BEZEICHNUNG
Group.Spacing – setzt den Abstand

BESCHREIBUNG
Stellen Sie die Anzahl der Pixel ein, die als horizontaler und vertikaler Abstand zwischen den Elementen der Gruppe verwendet werden soll. Das Setzen dieses Attributs hat den gleichen Effekt wie das Setzen von `Group.HorizSpacing` und `Group.VertSpacing`.

TYP
Zahl

ANWENDBARKEIT
I

17.20 Group.Title

BEZEICHNUNG
Group.Title – setzt den Gruppentitel für eine Seitenansicht

BESCHREIBUNG
Wenn Sie Gruppen für eine Pageview-Klasse (Seitenansicht) definieren, können Sie mit diesem Attribut einen Titel für diese Gruppe angeben, der in dem Widget angezeigt werden soll, das zum Durchblättern der einzelnen Seiten verwendet wird.

Ab RapaGUI 1.1 hat dieses Attribut eine Anwendbarkeit von `ISG`. Zuvor war seine Anwendbarkeit nur `I`. Beachten Sie, dass bei AmigaOS und Kompatiblen der Gruppentitel zur Laufzeit für alle Seitenansichtsmodi (außer dem Listenmodus) nur bei MUI 4.0 oder höher geändert werden kann.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

17.21 Group.VAlign

BEZEICHNUNG
Group.VAlign – setzt die Standard-Vertikalausrichtung

BESCHREIBUNG
Setzen Sie dieses Attribut, um festzulegen, wie nicht veränderbare Elemente ausgerichtet werden sollen, wenn sie kleiner als die ihnen zugewiesene Größe sind. Die folgenden Optionen stehen zur Verfügung:

`Top`	Oben ausgerichtet.
`Bottom`	Unten ausgerichtet.
`Center`	Zentriert ausgerichtet. Dies ist auch voreingestellt.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
I

17.22 Group.VertSpacing

BEZEICHNUNG
Group.VertSpacing – setzt den vertikalen Abstand

BESCHREIBUNG
Legen Sie die Anzahl der Pixel fest, die als vertikaler Abstand zwischen den Elementen der Gruppe verwendet werden soll.

TYP
Zahl

ANWENDBARKEIT
I

17.23 Group.Weight

BEZEICHNUNG
Group.Weight – setzt die Gruppengröße

BESCHREIBUNG
Dieses Attribut setzt die Größe eines Gruppenobjekts innerhalb einer anderen Gruppe von Objekten. Standardmäßig haben alle Objekte einer Gruppe eine Größe von 100. Wenn Sie ein Objekt haben wollen, das doppelt so groß erscheint, müssen Sie ihm eine Größe von 200 geben. Um eine Gruppe eineinhalb Mal so groß erscheinen zu lassen, verwenden Sie einfach 150 und so weiter.

Natürlich ist dieses Attribut nur für Gruppen sinnvoll, die in der Größe veränderbar sind.

TYP
Zahl

ANWENDBARKEIT
I

18 HLine-Klasse (HorizontalLinie)

18.1 Übersicht

Die HLine-Klasse (HorizontalLinie) erzeugt einfach eine horizontale Trennlinie, die als separate Gruppe von Widgets verwendet werden kann. Es gibt auch eine VLine-Klasse, welche vertikale Trennlinien erzeugt. Siehe Abschnitt 56.1 [VLine-Klasse], Seite 259, für Details.

Die HLine-Klasse hat keine Attribute.

19 Hollywood-Klasse

19.1 Übersicht

Die Hollywood-Klasse ist eine mächtige Klasse, die es Ihnen erlaubt, ein komplettes Hollywood-Display als Widget in Ihre GUI einzubetten. Immer wenn Sie etwas auf ein Hollywood-Display zeichnen, das an ein Widget angehängt ist, wird es automatisch auch in Ihr Widget gezeichnet. Sie können sogar das Hollywood-Display ausblenden und es wird immer noch funktionieren. Außerdem werden alle Mausklicks und Tastaturanschläge, die innerhalb der Hollywood-Klasse stattfinden, als normale Hollywood-Ereignisse an das entsprechende Widget weitergeleitet. So erlaubt Ihnen die Hollywood-Klasse, fast alle leistungsstarken Befehle von Hollywood in einem Widget transparent zu nutzen.

Hier ist ein Beispiel, wie Sie das Hollywood-Display 1 in Ihre GUI einbinden können:

```
<hollywood display="1"/>
```

Beachten Sie, dass Hollywood-Widgets standardmäßig nicht skalierbar sind. Sie können dies ändern, indem Sie die Attribute `Area.FixWidth` und `Area.FixHeight` entsprechend setzen. Wenn Sie eines dieser Attribute auf `False` setzen, erhalten Sie ein skalierbares Hollywood-Widget und wie bei normalen Hollywood-Displays wird Ihrem Hollywood-Widget auch eine `SizeWindow`-Benachrichtigung gesendet, wenn sich die Größe des Widgets ändert. Sie können dann den Inhalt Ihres Widgets an die neue Größe anpassen.

Siehe Abschnitt 3.15 [Hollywood-Brücke], Seite 21, für Details.

19.2 Hollywood.Display

BEZEICHNUNG

Hollywood.Display – integriert das Hollywood-Display in einem Objekt

BESCHREIBUNG

Integriert ein Hollywood-Display in einem Objekt. Hier müssen Sie den Identifikator eines Hollywood-Displays angeben.

Dieses Attribut ist obligatorisch und muss immer dann angegeben werden, wenn Sie ein Hollywood-Objekt erstellen. Sie können kein leeres Hollywood-Objekt ohne angehängtes Display erstellen.

Standardmäßig ist das Widget nicht in der Größe veränderbar. Sie können dies ändern, indem Sie die Attribute `Area.FixWidth` und `Area.FixHeight` auf `False` setzen. In diesem Fall wird Ihr Hollywood-Widget größenveränderbar und wenn der Benutzer die Fenstergröße ändert, erhält Ihre Hollywood-Display ein `"SizeWindow"`-Benachrichtigung, das Sie mit dem Befehl `InstallEventHandler()` überwachen können. Sie können dann auf dieses Ereignis entsprechend reagieren und Ihr Display neu zeichnen, etc.

TYP

Zahl

ANWENDBARKEIT

IS

19.3 Hollywood.DropFile

BEZEICHNUNG

Hollywood.DropFile – benachrichtigt, wenn Datei(en) über das Widget gezogen wird (V1.1)

BESCHREIBUNG

Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, sobald eine oder mehrere Dateien auf das Hollywood-Widget gezogen wurden. Das Feld `TriggerValue` der Nachrichtentabelle wird auf eine Tabelle gesetzt, die eine Liste aller auf das Widget gezogenen Dateien enthält.

Zusätzlich enthält diese Tabelle die folgenden zwei weiteren Felder:

`X:` Die x-Position, an der der Benutzer die Datei(en) abgelegt hat. Dies ist relativ zur linken Ecke des Widgets.

`Y:` Die y-Position, an der der Benutzer die Datei(en) abgelegt hat. Dies ist relativ zur oberen Ecke des Widgets.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

Beachten Sie, dass `Hollywood.DropFile` nur dann ausgelöst wird, wenn `Hollywood.DropTarget` zuerst auf `True` gesetzt wurde.

TYP

Boolesch

ANWENDBARKEIT

N

19.4 Hollywood.DropTarget

BEZEICHNUNG

Hollywood.DropTarget – lässt das Ablegen von Dateien überm Widget zu oder nicht (V1.1)

BESCHREIBUNG

Setzen Sie dies auf `True`, wenn Dateien auf diesem Hollywood-Widget abgelegt werden können. Sie können das Attribut `Hollywood.DropFile` überwachen, um zu erfahren, wenn der Benutzer eine oder mehrere Dateien auf dem Hollywood-Widget ablegt.

TYP

Boolesch

ANWENDBARKEIT

ISG

20 HSpace-Klasse (HorizontalAbstand)

20.1 Übersicht

HSpace-Klasse (HorizontalAbstand) erzeugt einfach Objekte mit einer festen Pixelgröße. Dies wird typischerweise zur Feinabstimmung des GUI-Layouts verwendet. Um HSpace-Objekte zu erstellen, die frei skalierbar sind, können Sie stattdessen Rectangle-Klasse (Rechteck) verwenden. Siehe Abschnitt 38.1 [Rectangle-Klasse], Seite 177, für Details.

20.2 HSpace.Width

BEZEICHNUNG
HSpace.Width – setzt den horizontalen Abstand

BESCHREIBUNG
Legt den gewünschten horizontalen Abstand für dieses Objekt in Pixeln fest.

TYP
Zahl

ANWENDBARKEIT
I

21 HTMLview-Klasse (HTMLAnsicht)

21.1 Übersicht

HTMLview-Klasse (HTMLAnsicht) erstellt ein Widget, das eine HTML-Seite entweder als Rohdaten, als lokal oder auf einem entfernten Server gespeicherte Dateiquelle anzeigt. Sie unterstützt auch einige Browser-Fähigkeiten wie eine Liste der besuchten Seiten und die Möglichkeit, durch diese Liste zu blättern.

Unter Windows und Mac OS X verwendet diese Klasse ein vom Betriebssystem bereitgestellte HTML-Steuerung. Unter Linux erfordert diese Klasse die Installation von WebKitGTK+. Da dies nicht auf jedem System verfügbar ist, gibt es auch eine separate RapaGUI-Version ohne diese Klasse und damit ohne die Abhängigkeit von WebKitGTK+. Auf AmigaOS und kompatiblen Systemen benötigt diese Klasse die Erweiterung `HTMLview.mcc`.

21.2 HTMLview.CanGoBack

BEZEICHNUNG
HTMLview.CanGoBack – ermittelt, ob der Browser zurück gehen kann

BESCHREIBUNG
Dieses Attribut wird auf `True` gesetzt, wenn es eine Seite im Verlauf gibt, zu der der Browser zurückkehren kann. Sie können eine Benachrichtigung für dieses Attribut einrichten und die Schaltfläche "Zurück" Ihrer Symbolleiste je nach Zustand dieses Attributs aktivieren oder deaktivieren.

TYP
Boolesch

ANWENDBARKEIT
GN

21.3 HTMLview.CanGoForward

BEZEICHNUNG
HTMLview.CanGoForward – ermittelt, ob der Browser vorwärts gehen kann

BESCHREIBUNG
Dieses Attribut wird auf `True` gesetzt, wenn es eine Seite im Verlauf gibt, zu der der Browser vorwärts gehen kann. Sie können eine Benachrichtigung für dieses Attribut einrichten und die Schaltfläche "Vorwärts" Ihrer Symbolleiste je nach Zustand dieses Attributs deaktivieren oder aktivieren.

TYP
Boolesch

ANWENDBARKEIT
GN

21.4 HTMLview.ClearHistory

BEZEICHNUNG
HTMLview.ClearHistory – löscht den Browserverlauf

ÜBERSICHT
`moai.DoMethod(id, "ClearHistory")`

BESCHREIBUNG
Diese Methode löscht den gesamten Browserverlauf.

EINGABEN

`id`	ID des HTMLview-Objekts

21.5 HTMLview.Contents

BEZEICHNUNG
HTMLview.Contents – setzt oder ermittelt den Widget-Inhalt

BESCHREIBUNG
Setzt oder ermittelt die vom Widget angezeigten HTML-Daten. Wenn Sie dieses Attribut setzen, müssen Sie dem Widget gültigen HTML-formatierten Code übergeben. Mit diesem Attribut erhalten Sie den Quellcode der aktuell angezeigten Seite.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

21.6 HTMLview.File

BEZEICHNUNG
HTMLview.File – setzt/ermittelt die Datei zum Anzeigen

BESCHREIBUNG
Legen Sie die Datei fest, die im Widget angezeigt werden soll. Dies ist normalerweise eine HTML-Seite, aber Sie können auch einen Pfad zu einem Bild übergeben.

Beachten Sie, dass dieses Attribut im Gegensatz zu `HTMLview.URL` nur einen normalen Pfad zu einer Datei erwartet, d.h. verwenden Sie hier nicht den Protokoll-Präfix `file://`.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

21.7 HTMLview.GoBack

BEZEICHNUNG
HTMLview.GoBack – geht im Browserverlauf zurück

ÜBERSICHT
```
moai.DoMethod(id, "GoBack")
```

BESCHREIBUNG
Navigiert im Verlauf der besuchten Seiten zurück.

EINGABEN

`id`	ID des HTMLview-Objekts

21.8 HTMLview.GoForward

BEZEICHNUNG
HTMLview.GoForward – geht im Browserverlauf vorwärts

ÜBERSICHT
```
moai.DoMethod(id, "GoForward")
```

BESCHREIBUNG
Navigiert im Verlauf der besuchten Seiten vorwärts.

EINGABEN

`id`	ID des HTMLview-Objekts

21.9 HTMLview.Reload

BEZEICHNUNG
HTMLview.Reload – lädt die aktuelle Seite neu

ÜBERSICHT
```
moai.DoMethod(id, "Reload")
```

BESCHREIBUNG
Die aktuelle Seite wird neu geladen.

EINGABEN

`id`	ID des HTMLview-Objekts

21.10 HTMLview.Search

BEZEICHNUNG
HTMLview.Search – durchsucht die aktuelle Seite

ÜBERSICHT
```
moai.DoMethod(id, "Search", t$, flags$)
```

BESCHREIBUNG

Diese Methode sucht auf der aktuellen Seite nach der Zeichenkette `t$` und scrollt das Ergebnis in die Ansicht und wählt es aus.

`flags$` kann eine Kombination der folgenden Flags sein:

`CaseSensitive`
: Groß- und Kleinschreibung wird beachtet.

`Backwards`
: Sucht rückwärts.

Wenn Sie mehrere Optionen in `flags$` angeben, trennen Sie sie mit einem Semikolon/Strichpunkt (;).

EINGABEN

`id`	ID des HTMLview-Objekts
`t$`	Zeichenkette, die gesucht wird
`flags$`	Kombination von Flags oder leere Zeichenfolge für Standardoptionen

21.11 HTMLview.Title

BEZEICHNUNG

HTMLview.Title – ermittelt den Titel der aktuellen Seite

BESCHREIBUNG

Ermittelt den Titel der aktuell angezeigten Seite, d.h. den Wert des Tags `<title>`.

TYP

Zeichenkette

ANWENDBARKEIT

G

21.12 HTMLview.URL

BEZEICHNUNG

HTMLview.URL – setzt/ermittelt die aktuelle URL

BESCHREIBUNG

Setzt oder ermittelt die URL, die im Widget angezeigt werden soll oder wird. Diese URL muss ein Protokoll-Präfix enthalten, typischerweise `http://`. Beim Öffnen lokaler Dateien müssen Sie `file://` als Protokoll-Präfix verwenden. Aber Sie können auch einfach `HTMLview.File` benutzen, was in diesem Fall bequemer ist.

TYP

Zeichenkette

ANWENDBARKEIT

ISG

22 Image-Klasse (Bild)

22.1 Übersicht

Die Image-Klasse kann verwendet werden, um einfach ein Bild anzuzeigen. Die Bilddaten stammen von einem Hollywood-Pinsel.

Hier ist eine Beispiel für eine XML-Definition:

```
<image brush="1"/>
```

Der obige XML-Code erzeugt ein Bildobjekt aus dem Hollywood-Pinsel Nummer 1. Die Image-Klasse unterstützt in Hollywood-Pinseln Masken- und Alphakanaltransparenz, so dass Sie auch Bilder mit transparenten Bereichen einbetten können.

22.2 Image.Brush

BEZEICHNUNG

Image.Brush – zeigt ein Bild an

BESCHREIBUNG

Setzen Sie dieses Attribut, um den Pinsel zu definieren, der vom Image-Objekt angezeigt werden soll. Übergeben Sie einfach den Identifikator des Hollywood-Pinsels, den Sie anzeigen lassen möchten.

Die Image-Klasse unterstützt Masken- und Alphakanaltransparenz in Hollywood-Pinseln, so dass Sie auch Bilder mit transparenten Bereichen einbetten können.

TYP

Zahl

ANWENDBARKEIT

IS

23 Label-Klasse (Beschriftung)

23.1 Übersicht

Die Label-Klasse kann verwendet werden, um komfortabel Beschriftungen für Ihre Widgets zu erstellen. Beschriftungen enthalten typischerweise ein Unterstrichzeichen, welches das Tastaturkürzel für den Zugriff auf das beschreibende Widget angibt.

Hier ist ein Beispiel für die Verwendung von `<label>`:

```
<hgroup>
    <label>_Name</label>
    <textentry/>
</hgroup>
```

Der obige Code richtet ein Texteingabe-Widget (Textentry) ein, das mit "Name" beschriftet ist. Außerdem wird eine Verknüpfung eingerichtet: Wenn der Benutzer `ALT+N` drückt, wird das Texteingabe-Widget automatisch aktiviert. Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details.

Bitte beachten Sie, dass die Beschriftungen horizontal nicht skalierbar sind. Das ist der Grund, warum sie die Größenanpassung der gesamten GUI leicht blockieren können. Um dieses Problem zu umgehen, können Sie das Label einfach in einer `<hgroup>` zusammen mit einem Rectangle-Objekt `<rectangle>` (Rechteck) einfügen. Siehe Abschnitt 38.1 [Rectangle-Klasse], Seite 177, für Details.

23.2 Label.Align

BEZEICHNUNG
Label.Align – setzt die Ausrichtung der Beschriftung

BESCHREIBUNG
Stellen Sie die gewünschte Ausrichtung für das Label ein. Dies kann einer der folgenden Werte sein:

`Left`	Links ausgerichtet.
`Right`	Rechts ausgerichtet. Dies ist auch voreingestellt.
`Center`	Zentriert ausgerichtet.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
I

23.3 Label.Text

BEZEICHNUNG
Label.Text – setzt den Text der Beschriftung oder gibt ihn zurück

BESCHREIBUNG

Setzen oder ermitteln Sie den Text der Beschriftung. Wenn der Text einen Unterstrich enthält, wird das Zeichen hinter dem Unterstrich automatisch als Tastaturkürzel hervorgehoben. Sie können dieses Verhalten deaktivieren, indem Sie das Attribut `Area.NoAutoKey` auf `True` setzen.

TYP

Zeichenkette

ANWENDBARKEIT

SG

24 Listview-Klasse (Listenansicht)

24.1 Übersicht

Die Listview-Klasse (Listenansicht) ist eines der am häufigsten verwendeten Widgets. Sie zeigt einen Container, der mit Datenelementen gefüllt werden kann. Die Listview-Klasse von RapaGUI ist sehr mächtig und unterstützt mehrspaltige Listen (siehe weiter unten), Auswahlkästchen (Checkboxen), editierbare Listenelemente, Datensortierung über benutzerdefinierte Callback-Funktionen und Symbole für die einzelnen Listenelemente.

Wenn Sie eine Listenansicht im XML-Code erstellen, müssen Sie immer mindestens eine Spalte hinzufügen. Dies geschieht über die Listviewcolumn-Klasse (Listenansichtsspalten). Hier ist ein Beispiel für eine minimale Listenansicht-Definition mit nur einer einzigen Spalte:

```
<listview>
   <column/>
</listview>
```

Es ist auch möglich, einige Einträge direkt während der Definition in die Listenansicht einzufügen. Dies kann mit dem Tag `<item>` geschehen:

```
<listview>
   <column>
      <item>Entry 1</item>
      <item>Entry 2</item>
      <item>Entry 3</item>
   </column>
</listview>
```

Wenn Sie eine mehrspaltige Liste haben wollen, müssen Sie den Tag `<column>` mehrmals verwenden. Hier ist ein Beispiel:

```
<listview>
   <column title="Column 1">
      <item>Entry 1</item>
      <item>Entry 2</item>
      <item>Entry 3</item>
   </column>
   <column title="Column 2">
      <item>Entry 1</item>
      <item>Entry 2</item>
      <item>Entry 3</item>
   </column>
   <column title="Column 3">
      <item>Entry 1</item>
      <item>Entry 2</item>
      <item>Entry 3</item>
   </column>
</listview>
```

In diesem Beispiel haben wir auch das Attribut `Listviewcolumn.Title` verwendet, um jeder unserer Spalten eine Titelleiste hinzuzufügen. Es gibt noch einige weitere Attribute, mit

denen Sie das Aussehen Ihrer Spalten anpassen können. Beispielsweise können Sie Ihren Spalten Auswahlkästchen (Checkboxen) hinzufügen und die Bearbeitung von Spaltenelementen erlauben. Siehe Abschnitt 25.1 [Listviewcolumn-Klasse], Seite 133, für Details.

Beachten Sie, dass RapaGUI für diese Klasse bis zu drei verschiedene Arten von Widgets verwenden kann: Wenn Sie eine Listenansicht erstellen, die keine erweiterten Eigenschaften verwendet (z.B. mehrere Spalten, Symbole, Sortierung), erstellt RapaGUI möglicherweise ein einfacheres Listenansichts-Widget für Sie, weil einige Betriebssysteme verschiedene Arten von listenbasierten Widgets anbieten. So gibt es z.B. unter Windows ein Listbox-Widget und ein Listenansichts-Widget. RapaGUI verwendet das Listbox-Widget, falls Ihre Listenansicht keine der erweiterten Eigenschaften verwendet, da Listbox-Widgets normalerweise schneller sind als Listenansichts-Widgets. Wenn Sie das nicht möchten, können Sie RapaGUI erzwingen, Ihnen immer eine vollständige Listenansicht zu geben, indem Sie das Attribut `Listview.ForceMode` auf den entsprechenden Tag setzen. Siehe Abschnitt 24.16 [Listview.ForceMode], Seite 120, für Details.

24.2 Listview.AbortEditing

BEZEICHNUNG

Listview.AbortEditing – benachrichtigt, wenn der Benutzer die Bearbeitung abbricht (V1.1)

BESCHREIBUNG

Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, wenn der Benutzer einen Bearbeitungsvorgang an einem Element abbricht, z.B. durch Drücken der Escape-Taste oder durch Klicken außerhalb des gerade bearbeitenden Element-Widgets.

Beachten Sie, dass Sie `Listviewcolumn.Editable` auf `True` setzen müssen, bevor Sie dieses Attribut verwenden können.

Ihre Callback-Funktion wird mit den folgenden zusätzlichen Argumenten aufgerufen:

`Row:` Zeilenindex des Eintrags, den der Benutzer bearbeitet hat, als er den Vorgang abbrach.

`Column:` Spaltenindex des Eintrags, den der Benutzer bearbeitet hat, als er den Vorgang abbrach.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP

Boolesch

ANWENDBARKEIT

N

24.3 Listview.Active

BEZEICHNUNG

Listview.Active – setzt oder ermittelt den aktiven Listeneintrag

BESCHREIBUNG

Mit diesem Attribut kann der aktive Listeneintrag gesetzt oder ermittelt werden. Dieser liegt immer zwischen 0 und `Listview.Entries`-1 oder dem speziellen Wert -1, falls es momentan keinen aktiven Eintrag gibt.

Wenn Sie dieses Attribut setzen, scrollt die Listenansicht automatisch die Position des angegebenen Eintrags in die Ansicht.

Neben einem absoluten Index können Sie hier auch die folgenden speziellen Werte übergeben:

`Off`	Löscht die Auswahl.
`Top`	Markiert den ersten Eintrag.
`Bottom`	Markiert den letzten Eintrag.
`Up`	Markiert den vorherigen Eintrag.
`Down`	Markiert den nächsten Eintrag.
`PageUp`	Bewegt den Listencursor eine Seite nach oben.
`PageDown`	Bewegt den Listencursor eine Seite nach unten.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, um bei jeder Änderung des aktiven Eintrags benachrichtigt zu werden.

TYP

Zahl oder Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

ISGN

24.4 Listview.Alternate

BEZEICHNUNG

Listview.Alternate – verwendet wechselnde Zeilenfarben

PLATTFORMEN

Windows, Linux, Mac OS

BESCHREIBUNG

Setzen Sie dieses Attribut auf `True`, damit die Listenansicht mit wechselnden Zeilenfarben erscheint.

TYP

Boolesch

ANWENDBARKEIT

I

24.5 Listview.Clear

BEZEICHNUNG
Listview.Clear – löscht alle Listenansichts-Einträge

ÜBERSICHT
```
moai.DoMethod(id, "Clear")
```

BESCHREIBUNG
Diese Methode entfernt alle Einträge aus der Listenansicht.

EINGABEN

`id` ID des Listenansichts-Objekts

24.6 Listview.ClickColumn

BEZEICHNUNG
Listview.ClickColumn – ermittelt, auf welche Spalte zuletzt geklickt wurde

BESCHREIBUNG
In mehrspaltigen Listenansichten ermittelt dieses Attribut die Nummer der Spalte, auf die der Benutzer zuletzt geklickt hat.

TYP
Zahl

ANWENDBARKEIT
GN

24.7 Listview.CompareItems

BEZEICHNUNG
Listview.CompareItems – legt fest, wie die Einträge sortiert werden sollen

BESCHREIBUNG
Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI die Callback-Funktion aus, wenn die Listeneinträge sortiert werden müssen. Die Callback-Funktion erhält zwei Einträge als Argumente und muss bestimmen, welcher Eintrag zuerst eingefügt werden soll.

Ihre Callback-Funktion wird mit den folgenden zusätzlichen Argumenten aufgerufen:

`Entry1:` Den ersten Eintrag.

`Entry2:` Den zweiten Eintrag.

Ihre Callback-Funktion muss dann einen Wert zurückgeben, der angibt, wie die beiden Einträge in der Listenansicht ausgerichtet werden sollen. Soll der Eintrag 1 vor 2 gesetzt werden, muss die Callback-Funktion -1 zurückgegeben. Soll Eintrag 1 nach Eintrag 2 gesetzt werden, muss von Ihrer Callback-Funktion 1 zurückkommen. Wenn die beiden Einträge gleich sind, gibt die Callback-Funktion 0 zurück.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Boolesch

ANWENDBARKEIT
N

24.8 Listview.DefClickColumn

BEZEICHNUNG
Listview.DefClickColumn – setzt die Standardspalte

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Wenn Sie die Listenansicht über die Tastatur steuern und RETURN drücken, wird die hier eingestellte Spaltennummer als Standard für `Listview.ClickColumn` verwendet.

TYP
Number

ANWENDBARKEIT
ISG

24.9 Listview.DoubleClick

BEZEICHNUNG
Listview.DoubleClick – benachrichtigt, wenn auf Listenansichts-Einträge Doppelgeklickt wird

BESCHREIBUNG
Richten Sie eine Benachrichtigung für dieses Attribut ein, um bei Doppelklicks auf Listeneinträge zu reagieren.

TYP
Boolesch

ANWENDBARKEIT
N

24.10 Listview.DropFile

BEZEICHNUNG
Listview.DropFile – benachrichtig, wenn Dateien auf die Listenansicht gezogen werden (V1.1)

BESCHREIBUNG
Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, sobald eine oder mehrere Dateien auf das Listenansichts-Widget

gezogen wurden. Das Feld `TriggerValue` der Nachrichtentabelle wird auf eine Tabelle gesetzt, die eine Liste aller Dateien enthält, die auf das Widget gezogen wurden.

Zusätzlich enthält die Nachrichtentabelle die folgenden zwei Felder:

`X:` Die x-Position, an der der Benutzer die Datei(en) abgelegt hat. Dies ist relativ zur linken Ecke des Widgets.

`Y:` Die y-Position, an der der Benutzer die Datei(en) abgelegt hat. Dies ist relativ zur oberen Ecke des Widgets.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

Beachten Sie, dass `Listview.DropFile` nur dann ausgelöst wird, wenn `Listview.DropTarget` zuerst auf `True` gesetzt wurde.

TYP
Boolesch

ANWENDBARKEIT
N

24.11 Listview.DropTarget

BEZEICHNUNG
Listview.DropTarget – stellt das Ablegen von Dateien ein (V1.1)

BESCHREIBUNG
Setzen Sie dies auf `True`, wenn Dateien auf diesem Listenansichts-Widget abgelegt werden können. Sie können das `Listview.DropFile` Attribut überwachen, um zu erfahren, wenn der Benutzer eine oder mehrere Dateien auf dem Listenansicht-Widget ablegt.

TYP
Boolesch

ANWENDBARKEIT
ISG

24.12 Listview.Edit

BEZEICHNUNG
Listview.Edit – fordert den Benutzer auf, einen Eintrag zu bearbeiten

ÜBERSICHT
```
moai.DoMethod(id, "Edit", row, column)
```

BESCHREIBUNG
Diese Methode kann verwendet werden, um die Bearbeitung von Einträgen vom Programm aus zu starten. Normalerweise wird die Bearbeitung eines Eintrags vom Benutzer durch einen langsamen Doppelklick auf einen Eintrag gestartet. Diese Methode stellt eine Alternative zu diesem Benutzermechanismus dar.

Dies funktioniert nur, wenn `Listviewcolumn.Editable` für die jeweilige Listenansichts-Spalte auf `True` gesetzt wurde.

Wenn der Benutzer die Bearbeitung beendet hat, wird das Attribut `Listview.ValueChange` ausgelöst.

Bitte beachten Sie, dass wenn Sie eine Überwachung auf dem Attribut `Listview.StartEditing` installiert haben, dann wird diese Callback-Funktion zuerst um Erlaubnis gefragt, bevor die Bearbeitung tatsächlich gestartet wird.

Um zu erfahren, wenn Bearbeitungsvorgänge abgebrochen werden, können Sie das Attribut `Listview.AbortEditing` überwachen.

Beachten Sie auch, dass diese Methode unter AmigaOS und kompatiblen Betriebssystemen nur unter MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id`	ID des Listview-Objekts
`row`	Zeilenindex des zu bearbeitenden Eintrags
`column`	Spaltenindex des zu bearbeitenden Eintrags

24.13 Listview.Entries

BEZEICHNUNG

Listview.Entries – ermittelt die Anzahl der Listeneinträge

BESCHREIBUNG

Gibt die aktuelle Anzahl der Einträge in der Listenansicht zurück.

TYP

Zahl

ANWENDBARKEIT

G

24.14 Listview.Exchange

BEZEICHNUNG

Listview.Exchange – tauscht zwei Einträge aus

ÜBERSICHT

```
moai.DoMethod(id, "Exchange", pos1, pos2)
```

BESCHREIBUNG

Diese Methode tauscht zwei Einträge in einer Listenansicht aus. Die Positionen können entweder als absolute Werte von 0 bis `Listview.Entries`-1 oder als einer der folgenden Sonderwerte übergeben werden:

`Top`	Verwendet den ersten Eintrag.
`Active`	Verwendet den aktiven Eintrag.

`Bottom` Verwendet den letzten Eintrag.

`Next` Verwendet den nächsten Eintrag. Dies geht nur für den Parameter `pos2`.

`Previous` Verwendet den vorherigen Eintrag. Dies geht nur für den Parameter `pos2`.

EINGABEN

`id` ID des Listview-Objekts

`pos1` Nummer des ersten Eintrags

`pos2` Nummer des zweiten Eintrags

24.15 Listview.First

BEZEICHNUNG

Listview.First – ermittelt den ersten sichtbaren Eintrag

BESCHREIBUNG

Liefert den ersten sichtbaren Listeneintrag.

TYP

Zahl

ANWENDBARKEIT

G

24.16 Listview.ForceMode

BEZEICHNUNG

Listview.ForceMode – überschreibt den Standard-Listenansichtmodus

BESCHREIBUNG

RapaGUI kann je nach Einstellung bis zu drei verschiedene Widgets für die Listview-Klasse verwenden. Wenn beispielsweise eine einspaltige Liste ohne Symbole und Überschriften verwendet wird, kann RapaGUI aus Gründen der Effizienz ein anderes Widget verwenden, wenn das Host-Betriebssystem ein solches Widget bereitstellt. Beispielsweise verwendet RapaGUI unter Windows in diesen Fällen das Listbox-Widget anstelle einer voll ausgestatteten Listenansicht. Wenn Sie das nicht wollen, setzen Sie dieses Attribut auf das gewünschte Widget und RapaGUI wird versuchen, es zu verwenden.

Folgende Modi werden derzeit erkannt:

`Normal` Wählt automatisch das Widget aus, das am besten passt. Dies ist die Voreinstellung.

`Listbox` Verwendet ein Listbox-Widget. Listbox-Widgets unterstützen nur eine einzelne Spalte, keine Symbole, keine Überschriften, keine Auswahlkästchen (Checkboxen), keine bearbeitbaren Einträge und keine ausgeblendeten Spalten.

`Listview` Verwendet ein Listenansichts-Widget. Listenansichts-Widgets unterstützen alles wie Auswahlkästchen, editierbare Einträge, der rechten und zentriete Ausrichtung und dem Ausblenden von Spalten.

`Dataview` Verwendet ein Dataview-Widget. Unterstützt alles, verwendet aber derzeit eine generische Implementierung unter Windows.

Beachten Sie, dass dieses Attribut auf AmigaOS und Kompatiblen keinen Effekt hat, da RapaGUI auf diesen Plattformen immer das gleiche Widget verwendet.

TYP

Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

I

24.17 Listview.GetDisabled

BEZEICHNUNG

Listview.GetDisabled – ermittelt, ob das Auswahlkästchen deaktiviert ist

ÜBERSICHT

```
state = moai.DoMethod(id, "GetDisabled", row, column)
```

BESCHREIBUNG

Gibt den deaktivierten Zustand des Auswahlkästchen (Checkbox) in der angegebenen Zeile `row` und Spalte `column` zurück. Dies ist entweder `True` (deaktiviert) oder `False` (aktiviert).

EINGABEN

`id` ID des Listview-Objekts

`row` Zeilenindex des Auswahlkästchen

`column` Spaltenindex des Auswahlkästchen

RÜCKGABEWERTE

`state` `True` wenn das Auswahlkästchen deaktiviert ist, sonst `False`

24.18 Listview.GetEntry

BEZEICHNUNG

Listview.GetEntry – gibt den Listeneintrag zurück

ÜBERSICHT

```
column1$, ... = moai.DoMethod(id, "GetEntry", pos)
```

BESCHREIBUNG

Holt einen Eintrag aus der in `id` angegebenen Listenansicht. Sie können entweder einen absoluten Index in `pos` oder den speziellen Wert `Active` übergeben, um den aktiven Eintrag zu erhalten. `Listview.GetEntry` gibt dann die Einträge aller Spalten der durch

`pos` angegebenen Zeile zurück. Sie erhalten so viele Rückgabewerte, wie es Spalten in der Listenansicht gibt.

EINGABEN

`id`	ID des Listview-Objekts
`pos`	Index der Listenansichts-Zeile oder "Active"

RÜCKGABEWERTE

`column1$`	Daten der ersten Spalte
`...`	Weitere Daten bei mehrspaltiger Listenansicht

24.19 Listview.GetSelection

BEZEICHNUNG

Listview.GetSelection – ermittelt die ausgewählten Einträge

ÜBERSICHT

```
t = moai.DoMethod(id, "GetSelection")
```

BESCHREIBUNG

Gibt eine Tabelle mit allen markierten Einträgen einer Mehrfachauswahl-Listenansicht zurück. Beachten Sie, dass dies nur bei Mehrfachauswahllisten verwendet werden sollte. Für einfach ausgewählte Listenansichten können Sie `Listview.Active` verwenden, um den ausgewählten Eintrag zu erhalten.

EINGABEN

`id`	ID des Listview-Objekts

RÜCKGABEWERTE

`t`	Tabelle mit den ausgewählten Einträgen

24.20 Listview.GetState

BEZEICHNUNG

Listview.GetState – fragt den Umschaltstatus des Auswahlkästchen ab

ÜBERSICHT

```
state = moai.DoMethod(id, "GetState", row, column)
```

BESCHREIBUNG

Gibt den Umschaltstatus des Auswahlkästchen (Checkbox) in der angegebenen Zeile `row` und Spalte `column` zurück. Dies ist entweder `True`, wenn das Auswahlkästchen aktiviert ist, oder `False`.

EINGABEN

`id`	ID des Listview-Objekts
`row`	Zeilenindex des Auswahlkästchen

`column` Spaltenindex des Auswahlkästchen

RÜCKGABEWERTE

`state` `True`, wenn das Auswahlkästchen markiert ist, ansonsten `False`

24.21 Listview.HRules

BEZEICHNUNG

Listview.HRules – zeichnet horizontale Rasterlinien zwischen den Zeilen

PLATTFORMEN

Windows, Linux, Mac OS

BESCHREIBUNG

Setzen Sie dies auf `True`, um horizontale Rasterlinien zwischen den Zeilen der Listenansicht zu aktivieren.

TYP

Boolesch

ANWENDBARKEIT

I

24.22 Listview.Insert

BEZEICHNUNG

Listview.Insert – fügt einen neuen Eintrag ein

ÜBERSICHT

```
pos = moai.DoMethod(id, "Insert", pos, [icon1,] column1$, ...)
```

BESCHREIBUNG

Fügen Sie einen neuen Eintrag in die Listenansicht ein. Wenn die Listenansicht mehrere Spalten hat, müssen Sie für alle Spalten der Listenansicht individuelle Eingabedaten übergeben.

Die Eingabedaten bestehen aus einer Textzeichenkette und, wenn die Spalte das Attribut `Listviewcolumn.Icon` hat, einem Symbol für jede Spalte. Das Symbol muss vor der Textzeichenkette übergeben werden und es muss ein Identifikator eines Hollywood-Pinsels sein, der als Symbol für den Eintrag verwendet werden soll. Wenn `Listviewcolumn.Icon` nicht gesetzt ist, müssen Sie den Parameter `Icon` weglassen und nur Textdaten für den Listenansichts-Eintrag übergeben. Wenn Sie `Listviewcolumn.Icon` auf `True` gesetzt haben und kein Symbol in dieser bestimmten Zeile und Spalte anzeigen möchten, können Sie auch den speziellen Wert -1 übergeben. In diesem Fall zeigt RapaGUI kein Symbol an, obwohl `Listviewcolumn.Icon` auf `True` gesetzt wurde. Bitte lesen Sie auch den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Falls eine Spalte ein Auswahlkästchen (Checkbox) zeigt, müssen Sie "On", "True" oder "1" übergeben, um das Auswahlkästchen und einen beliebigen anderen Text zu markieren.

Die Einfügeposition wird im Argument `pos` angegeben. Der neue Eintrag wird vor dem in `pos` angegebenen Eintrag eingefügt. Dies kann eine absolute Indexposition ab 0 für den ersten Eintrag oder einer der folgenden Sonderwerte sein:

`Top` Als ersten Eintrag einfügen.

`Active` Vor dem aktiven Eintrag einfügen. Wenn es keinen aktiven Eintrag gibt, wird der Eintrag ganz oben in die Liste eingefügt.

`Sorted` Sortiert den Eintrag in der Listenansicht. Bevor Sie diesen Modus verwenden können, müssen Sie zunächst die Referenzsortierspalte definieren, indem Sie das Attribut `Listviewcolumn.Sortable` setzen.

`Bottom` Als letzten Eintrag einfügen.

Wenn `pos` größer oder gleich der Anzahl der Einträge in der Listenansicht ist, wird der Eintrag als letzter Eintrag eingefügt.

`Listview.Insert` gibt die Position des neu eingefügten Eintrags zurück. Dies ist besonders nützlich bei der Angabe von "Sorted" in `pos`, da man in diesem Fall nicht einfach berechnen kann, wo der Eintrag in der Listenansicht landen wird.

Beachten Sie, dass unter AmigaOS und kompatiblen die Unterstützung von Symbolen nur mit MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id` ID des Listview-Objekts

`pos` Position als Absolut- oder Sonderwert einfügen (siehe oben)

`icon1` optional: Symbol für die erste Spalte; diese muss nur übergeben werden, wenn die Spalte das Icon-Attribut gesetzt hat.

`column1$` Eintrag zum Einfügen in die erste Spalte

`...` weitere Einträge bei mehrspaltiger Listenansicht

RÜCKGABEWERTE

`pos` Position des neu eingefügten Eintrags (ab 0)

24.23 Listview.Jump

BEZEICHNUNG

Listview.Jump – scrollt zu einem Eintrag

ÜBERSICHT

```
moai.DoMethod(id, "Jump", pos)
```

BESCHREIBUNG

Scrollt den angegebenen Eintrag in den sichtbaren Teil der Listenansicht. Dabei kann `pos` ein absoluter Index oder einer der folgenden Sonderwerte sein:

`Top` Scrollt den obersten Eintrag in den sichtbaren Teil.

`Active` Scrollt den aktiven Eintrag in den sichtbaren Teil.

`Bottom` Scrollt den letzten Eintrag in den sichtbaren Teil.

EINGABEN

`id` ID des Listview-Objekts

`pos` Nummer des Eintrags, der sichtbar gemacht werden soll oder spezieller Wert (siehe oben)

24.24 Listview.Move

BEZEICHNUNG

Listview.Move – verschiebt einen Eintrag an die neue Position

ÜBERSICHT

```
moai.DoMethod(id, "Move", from, to)
```

BESCHREIBUNG

Verschiebt den in `from` angegebenen Eintrag an die neue Position `to`. Positionen müssen als absolute Werte von 0 bis `Listview.Entries`-1 oder einen der folgenden speziellen Werte übergeben werden:

`Top` Verwendet den ersten Eintrag.

`Active` Verwendet den aktiven Eintrag.

`Bottom` Verwendet den letzten Eintrag.

`Next` Verwendet den nächsten Eintrag. Dies geht nur für den Parameter `to`.

`Previous` Verwendet den vorherigen Eintrag. Dies geht nur für den Parameter `to`.

EINGABEN

`id` ID des Listview-Objekts

`from` Nummer des ersten Eintrags

`to` Nummer des zweiten Eintrags

24.25 Listview.MultiSelect

BEZEICHNUNG

Listview.MultiSelect – aktiviert den Mehrfachauswahl-Modus

BESCHREIBUNG

Setzen Sie dieses Attribut auf `True`, um die Auswahl mehrerer Einträge in dieser Listenansicht zu ermöglichen. Standardmäßig kann immer nur ein Eintrag ausgewählt werden.

TYP

Boolesch

ANWENDBARKEIT

I

24.26 Listview.Quiet

BEZEICHNUNG
Listview.Quiet – deaktiviert die Aktualisierung der Listenansicht

BESCHREIBUNG
Das Hinzufügen oder Entfernen vieler Einträge auf einmal ist in der Regel recht aufwendig, da das Widget nach jedem Hinzufügen und Entfernen aktualisiert wird. Wenn Sie `Listview.Quiet` auf `True` setzen, wird die Aktualisierung der Listenansicht vorübergehend deaktiviert, bis Sie das Attribut wieder auf `False` setzen. Dies kann das Einfügen und Entfernen von Operationen erheblich beschleunigen, wenn viele Einträge betroffen sind.

TYP
Boolesch

ANWENDBARKEIT
S

24.27 Listview.Remove

BEZEICHNUNG
Listview.Remove – entfernt einen Eintrag aus der Listenansicht

ÜBERSICHT
```
moai.DoMethod(id, "Remove", pos)
```

BESCHREIBUNG
Entfernt einen Eintrag aus einer Listenansicht. Die Position kann als absoluter Indexwert oder als einer der folgenden Sonderwerte angegeben werden:

`First`	Entfernt den ersten Eintrag.
`Active`	Entfernt den aktiven Eintrag.
`Last`	Entfernt den letzten Eintrag.

Wenn der aktive Eintrag entfernt ist, wird der folgende Eintrag aktiv.

EINGABEN

`id`	ID des Listview-Objekts
`pos`	Index des zu entfernenden Eintrags oder einer der speziellen Werte (siehe oben)

24.28 Listview.Rename

BEZEICHNUNG
Listview.Rename – benennt einen Eintrag um

ÜBERSICHT
```
moai.DoMethod(id, "Rename", pos, [icon1,] column1$, ...)
```

BESCHREIBUNG

Benennt den Listeneintrag an der angegebenen Position `pos` um. Wenn die Listenansicht mehrere Spalten hat, müssen Sie für jede Spalte einen neuen Namen übergeben. Es ist nicht möglich, nur einen einzelnen Spalteneintrag umzubenennen; diese Methode wirkt sich immer auf die gesamte Zeile aus, so dass Sie so viele Zeichenketten übergeben müssen, wie es Spalten in Ihrer Listenansicht gibt.

Für alle Spalten, die das Attribut `Listviewcolumn.Icon` gesetzt haben, müssen Sie auch ein Symbol vor dem eigentlichen Text übergeben, welches ein Identifikator eines Hollywood-Pinsels sein muss. Wenn `Listviewcolumn.Icon` nicht gesetzt ist, müssen Sie den Parameter `icon` weglassen und nur Textdaten für den Listenansichts-Eintrag übergeben. Wenn Sie `Listviewcolumn.Icon` auf `True` gesetzt haben und kein Symbol in dieser bestimmten Zeile und Spalte anzeigen möchten, können Sie auch den speziellen Wert -1 übergeben. In diesem Fall zeigt RapaGUI kein Symbol an, obwohl `Listviewcolumn.Icon` auf `True` gesetzt wurde. Bitte lesen Sie auch den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Falls eine Spalte ein Auswahlkästchen (Checkbox) zeigt, müssen Sie `"On"`, `"True"` oder `"1"` übergeben, um das Auswahlkästchen und einen beliebigen anderen Text zu markieren.

Die Einfügeposition wird im Argument `pos` angegeben. Dies kann eine absolute Indexposition ab 0 für den ersten Eintrag oder einer der folgenden Sonderwerte sein:

`Active`	Benennt den aktiven Eintrag um.

Beachten Sie, dass unter AmigaOS und kompatiblen die Unterstützung von Symbolen nur mit MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id`	ID des Listview-Objekts
`pos`	Eingabeposition als absolute Zahl oder Sonderwert (siehe oben)
`icon1`	optional: Symbol für die erste Spalte; diese muss nur übergeben werden, wenn die Spalte das Icon-Attribut gesetzt hat.
`column1$`	neuer Text für den Eintrag in der ersten Spalte
`...`	weitere Einträge bei mehrspaltiger Listenansicht

24.29 Listview.Select

BEZEICHNUNG

Listview.Select – markiert einen Listeneintrag oder wählt/fragt ihn ab

ÜBERSICHT

```
state = moai.DoMethod(id, "Select", pos, seltype)
```

BESCHREIBUNG

Markiert einen Listeneintrag, wählt ihn ab oder Sie können den Auswahlzustand eines Eintrags abfragen.

Entweder kann `pos` die Nummer des Eintrags oder einer der folgenden Sonderwerte sein:

`Active` Verwendet den aktiven Eintrag.

`All` Verwendet alle Einträge.

Einer der folgenden Werte kann `seltype` sein:

`Off` Wählt den Eintrag ab.

`On` Markiert den Eintrag.

`Toggle` Wechselt den Auswahlzustand für den Eintrag.

`Ask` Ermittelt den Auswahlzustand des angegebenen Eintrags. Wenn dies festgelegt ist, gibt Listview.Select den Auswahlstatus des angegebenen Eintrags (1 oder 0) zurück.

EINGABEN

`id` ID des Listview-Objekts

`pos` Eingabeposition als absolute Zahl oder Sonderwert (siehe oben)

`seltype` Selektionsart (siehe oben)

RÜCKGABEWERTE

`state` Auswahlzustand des Eintrags; dies gilt nur bei Verwendung von `"Ask"` für `seltype` (siehe oben)

24.30 Listview.SetDisabled

BEZEICHNUNG

Listview.SetDisabled – setzt den Deaktivierungsstatus des Auswahlkästchen

ÜBERSICHT

```
moai.DoMethod(id, "SetDisabled", row, column, state)
```

BESCHREIBUNG

Setzt den deaktivierten Zustand des Auswahlkästchen (Checkbox) in der angegebenen Zeile `row` und Spalte `column`. Übergeben Sie `True`, um das Auswahlkästchen zu deaktivieren, oder `False`, um sie zu aktivieren.

EINGABEN

`id` ID des Listview-Objekts

`row` Zeilenindex des Auswahlkästchen

`column` Spaltenindex des Auswahlkästchen

`state` `True`, wenn das Auswahlkästchen deaktiviert werden soll, andernfalls `False`.

24.31 Listview.SetState

BEZEICHNUNG

Listview.SetState – setzt den Umschaltstatus des Auswahlkästchen

ÜBERSICHT

```
moai.DoMethod(id, "SetState", row, column, state)
```

BESCHREIBUNG

Setzt den Umschaltstatus des Auswahlkästchen (Checkbox) in der angegebenen Zeile `row` und Spalte `column`. Übergeben Sie `True`, um das Auswahlkästchen zu markieren, oder `False`, um sie abzuwählen.

EINGABEN

`id`	ID des Listview-Objekts
`row`	Zeilenindex des Auswahlkästchen
`column`	Spaltenindex des Auswahlkästchen
`state`	`True` wenn das Auswahlkästchen markiert werden soll, andernfalls `False`

24.32 Listview.Sort

BEZEICHNUNG

Listview.Sort – sortiert die Einträge

ÜBERSICHT

```
moai.DoMethod(id, "Sort")
```

BESCHREIBUNG

Diese Methode sortiert alle Einträge der Listenansicht. Bevor Sie diese Methode verwenden können, müssen Sie zunächst die Spalte definieren, deren Einträge sortiert werden sollen, indem Sie das Attribut `Listviewcolumn.Sortable` setzen.

Standardmäßig werden die Einträge in aufsteigender alphabetischer Reihenfolge sortiert. Sie können die Sortierung anpassen, indem Sie mit dem Attribut `Listview.CompareItems` eine Benachrichtigung anfordern, wenn Elemente sortiert werden müssen.

EINGABEN

`id`	ID des Listview-Objekts

24.33 Listview.StartEditing

BEZEICHNUNG

Listview.StartEditing – benachrichtigt, wenn ein Eintrag bearbeitet wird

BESCHREIBUNG

Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, wenn der Benutzer ein Listenelement durch langsames Doppelklicken bearbeiten möchte oder wenn Ihr Skript die Methode `Listview.Edit` ausführt. Ihre

Callback-Funktion kann dann den Bearbeitungsauftrag des Benutzers erlauben oder verbieten. Um die Anfrage zu verbieten, muss Ihre Callback-Funktion `False` oder um die Anfrage zu erlauben, muss sie `True` zurückgeben.

Beachten Sie, dass Sie `Listviewcolumn.Editable` auf `True` setzen müssen, bevor Sie dieses Attribut verwenden können.

Ihre Callback-Funktion wird mit den folgenden zusätzlichen Argumenten aufgerufen:

`Row:` Zeilenindex des Eintrags, den der Benutzer bearbeiten möchte.

`Column:` Spaltenindex des Eintrags, den der Benutzer bearbeiten möchte.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Boolesch

ANWENDBARKEIT
N

24.34 Listview.TitleClick

BEZEICHNUNG
Listview.TitleClick – legt die Spaltenindexnummer ab

BESCHREIBUNG
Dieses Attribut wird auf die Spaltenindexnummer gesetzt, wenn der Benutzer auf eine Spaltentitel-Schaltfläche klickt.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert dieses Attribut mindestens MUI 4.0.

TYP
Zahl

ANWENDBARKEIT
N

24.35 Listview.ValueChange

BEZEICHNUNG
Listview.ValueChange – benachrichtigt, wenn sich der Wert des Listeneintrags ändert

BESCHREIBUNG
Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, wenn sich der Wert eines Listeneintrags geändert hat, weil der Benutzer das Auswahlkästchen (Combobox) umgeschaltet oder das Element bearbeitet hat. Beachten Sie, dass `Listview.ValueChange` nicht ausgelöst wird, wenn der Wert des Elements mit der Methode `Listview.Rename` geändert wurde.

Für Elemente in Auswahlkästchen-Spalten wird `TriggerValue` entweder auf `True` oder `False` gesetzt, was den neuen Auswahlkästchen-Status widerspiegelt. Für Elemente in Textspalten enthält `TriggerValue` den neuen Eintragstext.

Zusätzlich wird Ihre Callback-Funktion mit den folgenden zusätzlichen Argumenten aufgerufen:

`Row:` Zeilenindex des Eintrags, dessen Wert sich geändert hat.

`Column:` Spaltenindex des Eintrags, dessen Wert sich geändert hat.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Boolesch oder Zeichenkette (abhängig vom Spaltentyp)

ANWENDBARKEIT
N

24.36 Listview.Visible

BEZEICHNUNG
Listview.Visible – ermittelt die Anzahl der sichtbaren Einträge

BESCHREIBUNG
Ermittelt die Anzahl der Einträge in der Listenansicht, die gerade sichtbar sind.

TYP
Zahl

ANWENDBARKEIT
G

24.37 Listview.VRules

BEZEICHNUNG
Listview.VRules – zeichnet vertikale Rasterlinien zwischen den Spalten

BESCHREIBUNG
Setzen Sie dies auf `True`, um vertikale Rasterlinien zwischen den Spalten der Listenansicht zu aktivieren.

TYP
Boolesch

ANWENDBARKEIT
I

25 Listviewcolumn-Klasse (Listenansichtsspalten)

25.1 Übersicht

Die Listviewcolumn-Klasse (Listenansichtsspalten) wird beim Erstellen von Listenansichten benötigt. Es erlaubt Ihnen, verschiedene Attribute für die Spalten Ihrer Listenansichten anzugeben.

Die Listviewcolumn-Klasse muss immer in eine `<listview>`-Definition eingebettet werden. Der XML-Tag ist `<column>`. Siehe Abschnitt 24.1 [Listview-Klasse], Seite 113, für Details.

Beachten Sie, dass Sie keine Instanzen dieser Klasse mit `moai.CreateObject()` erzeugen können. Die Spaltennummern der Listenansicht sind derzeit statisch, d.h. Sie können zur Laufzeit keine Spalten hinzufügen oder entfernen.

25.2 Listviewcolumn.Align

BEZEICHNUNG
Listviewcolumn.Align – setzt/ermittelt die Spaltenausrichtung

BESCHREIBUNG
Setzt oder ermittelt die Spaltenausrichtung. Dies kann einer der folgenden Werte sein:

`Left` Links ausgerichtet. Dies ist auch voreingestellt.

`Right` Rechts ausgerichtet.

`Center` Zentriert ausgerichtet.

Auf Nicht-AmigaOS-Systemen wird dieses Attribut nur für das DataView-Widget unterstützt. Wenn Sie eine Listenansicht erstellen und `Listviewcolumn.Align` auf einen anderen Wert als `Left` gesetzt ist, wechselt RapaGUI automatisch zum DataView-Widget. Wenn Sie dieses Attribut nicht zum Zeitpunkt der Erstellung angeben, sondern es später mit `moai.Set()` festlegen möchten, müssen Sie explizit ein DataView-Widget anfordern, indem Sie das Attribut `Listview.ForceMode` setzen.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
ISG

25.3 Listviewcolumn.Checkbox

BEZEICHNUNG
Listviewcolumn.Checkbox – setzt die Spalte in den Auswahlkästchen-Modus

BESCHREIBUNG
Setzen Sie dies auf `True`, um diese Spalte als Auswahlkästchen-Spalte (Checkbox) zu benutzen. Auswahlkästchen-Spalten zeigen Auswahlkästchen anstelle von Text an. Immer wenn der Text eines Eintrags in einer Auswahlkästchen-Spalte auf `"On"`, `"True"`

oder "1" gesetzt ist, wird das Auswahlkästchen ausgewählt. Alle anderen Texte führen zu einem nicht ausgewähltem Auswahlkästchen.

Sie können die Zustände der Auswahlkästchen entweder mit der Methode `Listview.Rename` oder mit der dedizierten Methode `Listview.SetState` ändern. Ebenso ist es möglich, den Status eines Auswahlkästchen über die Methoden `Listview.GetEntry` oder `Listview.GetState` abzurufen.

Um benachrichtigt zu werden, wenn der Benutzer den Status eines Auswahlkästchen umschaltet, müssen Sie das Attribut `Listview.ValueChange` überwachen.

Beachten Sie auch, dass `Listviewcolumn.Checkbox`, `Listviewcolumn.Editable` und `Listviewcolumn.Icon` sich gegenseitig ausschließen. Sie können keine Auswahlkästchen-Spalten erstellen, die editierbar sind oder Symbole anzeigen können.

TYP

Boolesch

ANWENDBARKEIT

I

25.4 Listviewcolumn.Editable

BEZEICHNUNG

Listviewcolumn.Editable – erlaubt das Bearbeiten von Spalteneinträgen

BESCHREIBUNG

Setzen Sie diesen Wert auf `True`, um die Bearbeitung der Elemente in dieser Spalte durch den Benutzer zu ermöglichen. Der Benutzer kann dann alle Elemente in dieser Spalte bearbeiten, indem er einen langsamen Doppelklick ausführt, d.h. die linke Maustaste zweimal hintereinander langsam drückt.

Wenn Sie nur die Bearbeitung einiger Elemente in der Spalte erlauben möchten, müssen Sie dieses Attribut auf `True` setzen und das Attribut `Listview.StartEditing` überwachen. `Listview.StartEditing` wird immer dann ausgelöst, wenn der Benutzer versucht, ein Element zu bearbeiten und Ihr Callback-Funktion kann `False` zurückgeben, um die Bearbeitung bestimmter Elemente zu verbieten. Siehe Abschnitt 24.33 [Listview.StartEditing], Seite 129, für Details.

Um benachrichtigt zu werden, wenn sich der Wert eines Listeneintrags ändert, weil der Benutzer ihn bearbeitet hat, müssen Sie das Attribut `Listview.ValueChange` überwachen.

Um die Bearbeitung eines Listeneintrags manuell zu starten, rufen Sie die Methode `Listview.Edit` auf.

Auch `Listviewcolumn.Checkbox` und `Listviewcolumn.Editable` schließen sich gegenseitig aus. Sie können keine editierbaren Auswahlkästchen-Spalten (Checkbox) anlegen.

Beachten Sie, dass unter AmigaOS und kompatiblen Betriebssystemen dieses Attribut mindestens MUI 4.0 erfordert.

TYP

Boolesch

ANWENDBARKEIT
I

25.5 Listviewcolumn.Hide

BEZEICHNUNG
Listviewcolumn.Hide – blendet Spalten ein oder aus

BESCHREIBUNG
Mit diesem Attribut können Sie einzelne Spalten der Listenansicht ein- oder ausblenden. Beachten Sie, dass die Spalten immer noch da sind, sie werden nur unsichtbar sein. Daher dürfen Sie versteckte Spalten nicht vergessen, wenn Sie Listenansicht-Einträge (ListView) mit `Listview.Insert` oder ähnlichen Methoden ändern.

Auf Nicht-AmigaOS-Systemen wird dieses Attribut nur für das DataView-Widget unterstützt. Wenn Sie eine Listenansicht erstellen und `Listviewcolumn.Hide` auf `True` gesetzt ist, wechselt RapaGUI automatisch zum DataView-Widget. Wenn Sie dieses Attribut nicht zum Zeitpunkt der Erstellung angeben, sondern es später mit `moai.Set()` festlegen möchten, müssen Sie explizit ein DataView-Widget anfordern, indem Sie das Attribut `Listview.ForceMode` setzen.

TYP
Boolesch

ANWENDBARKEIT
ISG

25.6 Listviewcolumn.Icon

BEZEICHNUNG
Listviewcolumn.Icon – aktiviert Symbole für diese Spalte

BESCHREIBUNG
Setzen Sie dies auf `True`, wenn die Einträge in der Listenansicht in dieser Spalte Symbole verwenden sollen. In diesem Fall müssen Sie der Methode `Listview.Insert` die Hollywood-Pinsel als Symbole übergeben.

Bitte lesen Sie auch das Kapitel über den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Unter AmigaOS und kompatiblen ist die Unterstützung von Symbolen nur mit MUI 4.0 oder höher verfügbar.

TYP
Boolesch

ANWENDBARKEIT
I

25.7 Listviewcolumn.Sortable

BEZEICHNUNG
Listviewcolumn.Sortable – Spalte wird sortierbar

BESCHREIBUNG
Setzen Sie dies auf `True`, um die Spalte als sortierbar zu kennzeichnen. Sie können dann mit `Listview.Insert` Einträge in sortierter Reihenfolge einfügen oder die Methode `Listview.Sort` aufrufen, um alle Einträge sortieren zu lassen. Auf einigen Plattformen setzt RapaGUI auch Sortierpfeile in den Spaltentitel, die der Benutzer anklicken kann, um die Spalteneinträge zu sortieren.

Beachten Sie, dass es pro Listenansicht nur eine sortierbare Spalte geben kann.

TYP
Boolesch

ANWENDBARKEIT
ISG

25.8 Listviewcolumn.Title

BEZEICHNUNG
Listviewcolumn.Title – setzt/ermittelt die Spaltenüberschrift

BESCHREIBUNG
Setzt oder ermittelt den Titel der Spalte. Der Titel wird immer oben in der Listenansicht angezeigt und verschwindet nicht, wenn die Listenansicht gescrollt wird.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

25.9 Listviewcolumn.Width

BEZEICHNUNG
Listviewcolumn.Width – setzt/ermittelt die Spaltenbreite

BESCHREIBUNG
Stellen Sie die Spaltenbreite in Pixel ein oder ermitteln Sie sie. Der Standardwert ist -1, d.h. die Spalte sollte so groß wie der größte Eintrag sein.

TYP
Zahl

ANWENDBARKEIT
ISG

26 Listviewitem-Klasse (Listenansichts-Element)

26.1 Übersicht

Die Klasse Listviewitem (Listenansichts-Element) kann bei der Erstellung von Listenansichten (Listview) verwendet werden, um bereits bei der Objekterstellung Listenansichts-Elemente hinzuzufügen. Siehe Abschnitt 24.1 [Listview-Klasse], Seite 113, für Details.

Listviewitem-Klassen-Einträge müssen immer in eine `<column>`-Definition eingebettet werden. Der XML-Tag ist `<item>`. Siehe Abschnitt 25.1 [Listviewcolumn-Klasse], Seite 133, für Details.

Beachten Sie, dass Sie Instanzen dieser Klasse nicht mit `moai.CreateObject()` erstellen können. Stattdessen müssen Sie `Listview.Insert` verwenden, um zur Laufzeit neue Elemente zu erstellen. Außerdem können Sie keine Attribute von Listenansichts-Elemente mit dieser Klasse ändern. Wenn Sie den Text oder das Symbol eines Listenansichts-Elementes ändern möchten, müssen Sie stattdessen `Listview.Rename` verwenden.

26.2 Listviewitem.Icon

BEZEICHNUNG

Listviewitem.Icon – setzt das Symbol für das Listenansichts-Elementes

BESCHREIBUNG

Setzen Sie dieses Attribut auf den Identifikator eines Hollywood-Pinsels, um ein Symbol neben dem Text des Eintrags hinzuzufügen. Sie müssen auch `Listviewcolumn.Icon` auf `True` setzen, wenn Sie Symbole in einer Listenansichts-Spalte (Listview) verwenden wollen.

Um das Symbol eines Listeneintrags später zu ändern, können Sie die Methode `Listview.Rename` verwenden.

Unter AmigaOS und kompatiblen ist die Unterstützung von Symbolen nur mit MUI 4.0 oder höher verfügbar.

TYP

Zahl

ANWENDBARKEIT

I

27 Menu-Klasse (Menü)

27.1 Übersicht

Die Menu-Klasse (Menü) kann verwendet werden, um ein einzelnes Abroll-Menü zu erstellen, das dann mit Elementen gefüllt werden sollte, die von der Menuitem-Klasse abgeleitet sind. Menüs werden normalerweise in ein `<menubar>`-Objekt (Menüleiste) eingebettet. Siehe Abschnitt 28.1 [Menubar-Klasse], Seite 143, für Details. Alternativ können sie auch als Untermenüs anderer Menüobjekte eingebettet werden.

Es ist auch möglich, ein Menüobjekt als Kontextmenü zu einem der Widgets Ihres Fensters mit dem Attribut `Area.ContextMenu` hinzuzufügen. Das Kontextmenü erscheint immer dann, wenn der Benutzer mit der rechten Maustaste auf das Widget klickt, welches das Kontextmenü enthält. Im Folgenden finden Sie ein Beispiel für das Hinzufügen eines Menüobjekts als Kontextmenü zu einem Texteditor-Widget:

```
<menu title="Context menu" id="ctxtmenu">
   <item>Cut</item>
   <item>Copy</item>
   <item>Paste</item>
</menu>

<window>
...
   <texteditor contextmenu="ctxtmenu"/>
...
</window>
```

Beachten Sie, dass das Attribut `Menu.Title` nur unter AmigaOS und kompatiblen Betriebssystemen verwendet wird, wenn Sie die Menu-Klasse zum Erstellen von Kontextmenüs verwenden. Kontextmenüs unter Windows, Linux und Mac OS zeigen keinen Titel an.

27.2 Menu.Append

BEZEICHNUNG

Menu.Append – fügt ein losgelöstes Objekt als letztes Menü-Element hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Append", obj)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` angegebene losgelöste Objekt dem durch `id` angegebenen Menüobjekt hinzuzufügen. Das losgelöste Objekt wird als letztes Element des Menüs hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufrufen des Befehls `moai.CreateObject()` oder durch explizites Trennen ihrer MOAI-Objekte mithilfe der Methode `Menu.Remove` erstellt werden.

EINGABEN

`id`	ID des Menu-Objekts
`obj`	ID des anzuhängenden Objekts

27.3 Menu.Disabled

BEZEICHNUNG
Menu.Disabled – setzt/ermittelt den Deaktivierungsstatus des Menüs

BESCHREIBUNG
Aktiviert (`False`) oder deaktiviert (`True`) das gesamte Menü. Ausserdem können Sie den Deaktivierungsstatus ermitteln.

TYP
Boolesch

ANWENDBARKEIT
ISG

27.4 Menu.Insert

BEZEICHNUNG
Menu.Insert – fügt das losgelöstes Objekt nach dem angegebenen Element ein

ÜBERSICHT
```
moai.DoMethod(id, "Insert", obj, pred)
```

BESCHREIBUNG
Mit dieser Methode kann das durch `obj` angegebene gelöste Objekt in das durch `id` angegebene Menüobjekt eingefügt werden. Das gelöste Objekt wird nach dem in `pred` angegebenen Element hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufrufen des Befehls `moai.CreateObject()` oder durch explizites Trennen ihrer MOAI-Objekte mithilfe der Methode `Menu.Remove` erstellt werden.

EINGABEN

`id`	ID des Menu-Objekts
`obj`	ID des einzufügenden Objekts
`pred`	das Objekt wird nach diesem Element eingefügt

27.5 Menu.NoAutoKey

BEZEICHNUNG

Menu.NoAutoKey – deaktiviert die automatische Tastaturkürzel-Generierung

BESCHREIBUNG

Setzen Sie dieses Attribut, um die automatische Tastaturkürzel-Generierung zu deaktivieren. Standardmäßig wird das Zeichen nach einem Unterstrich in einem Menütitel wie eine Tastaturkürzel behandelt, mit der Sie über die Tastatur auf den Menüeintrag zugreifen können. Wenn Sie dieses Attribut setzen, werden Unterstreichungszeichen niemals als Tastaturkürzel behandelt und in Menüeinträgen normal angezeigt.

Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details.

TYP

Boolesch

ANWENDBARKEIT

I

27.6 Menu.Prepend

BEZEICHNUNG

Menu.Prepend – fügt ein losgelöstes Objekt als erstes Menü-Element hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Prepend", obj)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` angegebene losgelöste Objekt in dem durch `id` angegebene Menüobjekt hinzuzufügen. Das losgelöste Objekt wird als erstes Element des Menüs hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöste in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufrufen des Befehls `moai.CreateObject()` oder durch explizites Trennen ihrer MOAI-Objekte mithilfe der Methode `Menu.Remove` erstellt werden.

EINGABEN

`id`	ID des Menu-Objekts
`obj`	ID des einzufügenden Objekts

27.7 Menu.Remove

BEZEICHNUNG

Menu.Remove – entfernt/löst ein Objekt von dem Menü

ÜBERSICHT

```
moai.DoMethod(id, "Remove", obj)
```

BESCHREIBUNG

Mit dieser Methode kann das angegebene Objekt aus dem angegebenen Menü entfernt werden. Nachdem diese Methode beendet wurde, wird das angegebene Objekt seinen Zustand von angehängt auf losgelöst ändern. Das bedeutet, dass Sie es nun mit der Methode wie `Menu.Insert` an ein anderes Menü anhängen oder mit `moai.FreeObject()` aus dem Speicher löschen können.

EINGABEN

`id`	ID des Menu-Objekts
`obj`	ID des Objekts, welches losgelöst wird

27.8 Menu.Title

BEZEICHNUNG

Menu.Title – setzt oder ermittelt den Menütitel

BESCHREIBUNG

Setzt oder ermittelt den Titel des Menüs.

Wenn der Titel einen Unterstrich enthält, richtet RapaGUI automatisch das auf diesen Unterstrich folgende Zeichen als Tastaturkürzel ein. Wenn Sie dieses Verhalten nicht wünschen, setzen Sie `Menu.NoAutoKey` auf `True`.

TYP

Zeichenkette

ANWENDBARKEIT

ISG

28 Menubar-Klasse (Menüleiste)

28.1 Übersicht

Die Menubar-Klasse dient zur Verwaltung von einer Menüleiste, die über das Attribut `Window.Menubar` an Fenster angehängt werden kann. Eine Menüleiste enthält eine Anzahl von Elementen, die Objekte der Menu-Klasse sind, wobei jedes von ihnen genau ein Menü beschreibt.

In einer XML-Datei wird ein Menübaum mit den Tags `<menubar>`, `<menu>` und `<item>` definiert. Hier ist eine Beispiel-Definition einer einfachen Menüleiste:

```
<menubar id="mymenubar">
    <menu title="_File">
        <item>_New...</item>
        <item>_Open...</item>
        <item/>
        <item>_Save</item>
        <item>S_ave as...</item>
        <item/>
        <item>_Quit</item>
    </menu>
    <menu title="Edit">
        <item shortcut="Ctrl+X">_Cut</item>
        <item shortcut="Ctrl+C">C_opy</item>
        <item shortcut="Ctrl+V">_Paste</item>
    </menu>
    <menu title="?">
        <item>Se_ttings...</item>
        <item/>
        <item>A_bout...</item>
        <item>About _RapaGUI...</item>
    </menu>
</menubar>
```

Beachten Sie die Verwendung des Unterstrichzeichens im obigen XML-Code: Sie können dieses Zeichen verwenden, um das nächste Zeichen automatisch als Tastaturkürzel für den Menüeintrag zu kennzeichnen. Dies ist sehr nützlich, da viele Leute gerne die Tastatur anstelle der Maus benutzen, besonders wenn es darum geht, die gleichen Aktionen mehrmals zu wiederholen. Daher ist es immer eine gute Idee, Tastaturkürzel einzurichten. Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details.

Wenn Sie komplexere Tastaturkürzel benötigen, z.B. `Ctrl+V` zum Einfügen, können Sie das Attribut `Menuitem.Shortcut` verwenden, um eine solche Verknüpfung einzurichten. Beachten Sie, dass auf einigen Plattformen (z.B. Windows) beide Arten von Verknüpfungen gleichzeitig angegeben werden können: Verknüpfungen, die mit dem Unterstrichzeichen angegeben werden, und Verknüpfungen, die mit dem `Menuitem.Shortcut` definiert werden. Wenn eine bestimmte Plattform nur eine Art von Verknüpfung unterstützt, hat die in `Menuitem.Shortcut` angegebene Verknüpfung Vorrang vor der mit dem Unterstrichzeichen.

Beachten Sie auch die leeren `<item/>` im obigen XML-Code: Diese fügen eine Trennlinie in den Menübaum ein. Die Verwendung von Trennlinien macht Ihr Menü für den Endbenutzer besser lesbar. Nachdem Sie die XML-Definition oben geschrieben haben, können Sie die Menüleiste zu einem Ihrer Fenster hinzufügen, indem Sie das Attribut `Window.Menubar` wie folgt verwenden:

```
<window menubar="mymenubar">
...
</window>
```

Es ist sehr wichtig zu beachten, dass Sie Ihre Menüleisten im Bereich `<application>` definieren müssen, da Menüleisten globale Objekte sind und erst später an Fenster oder Widgets angehängt werden. Deshalb ist es nicht erlaubt, Menüleisten innerhalb eines `<window>` XML-Bereichs zu definieren.

28.2 Menubar.Append

BEZEICHNUNG

Menubar.Append – fügt ein losgelöstes Objekt als letztes Menüleisten-Element hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Append", obj)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` angegebene gelöste Objekt dem durch `id` angegebene Menüleisten-Objekt hinzuzufügen. Das losgelöste Objekt wird als letztes Element der Menüleiste hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufruf des Befehls `moai.CreateObject()` oder durch explizite Trennung von ihrem übergeordneten Objekt mit der Methode `Menubar.Remove` erzeugt werden.

EINGABEN

`id`	ID des Menüleisten-Objekts
`obj`	ID des hinzufügenden Objekts

28.3 Menubar.Insert

BEZEICHNUNG

Menubar.Insert – fügt ein losgelöstes Objekt nach dem angegebenen Menüleisten-Element hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Insert", obj, pred)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` angegebene losgelöste Objekt in das durch `id` angegebene Menüobjekt einzufügen. Das losgelöste Objekt wird nach

dem in `pred` angegebenen Element hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufruf des Befehls `moai.CreateObject()` oder durch explizite Trennung von ihrem übergeordneten Objekt mit der Methode `Menubar.Remove` erzeugt werden.

EINGABEN

`id`	ID des Menüleisten-Objekts
`obj`	ID des hinzufügenden Objekts
`pred`	Das Objekt wird nach diesem Objekt eingefügt

28.4 Menubar.Prepend

BEZEICHNUNG

Menubar.Prepend – fügt ein losgelöstes Objekt als erstes Menüleisten-Element hinzu

ÜBERSICHT

```
moai.DoMethod(id, "Prepend", obj)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um das durch `obj` angegebene gelöste Objekt dem durch `id` angegebene Menüleisten-Objekt hinzuzufügen. Das losgelöste Objekt wird als erstes Element der Menüleiste hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufruf des Befehls `moai.CreateObject()` oder durch explizite Trennung von ihrem übergeordneten Objekt mit der Methode `Menubar.Remove` erzeugt werden.

EINGABEN

`id`	ID des Menüleisten-Objekts
`obj`	ID des hinzufügenden Objekts

28.5 Menubar.Remove

BEZEICHNUNG

Menubar.Remove – löst ein Objekt von der Menüleiste

ÜBERSICHT

```
moai.DoMethod(id, "Remove", obj)
```

BESCHREIBUNG

Mit dieser Methode kann das angegebene Objekt aus der angegebenen Menüleiste entfernt werden. Nachdem diese Methode beendet ist, wird das angegebene Objekt seinen Zustand von angehängt auf losgelöst ändern. Das bedeutet, dass Sie es nun mit Methoden

wie `Menubar.Insert` an eine andere Menüleiste anhängen oder mit `moai.FreeObject()` aus dem Speicher löschen können.

EINGABEN

`id`	ID des Menüleisten-Objekts
`obj`	ID des zu loslösenden Objekts

29 Menuitem-Klasse (Menüelement)

29.1 Übersicht

Mit Hilfe der Menuitem-Klasse (Menüelement) kann ein einzelner Menüpunkt erstellt werden. Da solche Menüpunkte immer in einem `<menu>`-Tag eingebettet sein müssen, das wiederum normalerweise in einem `<menubar>`-Tag eingebettet ist. Siehe Abschnitt 28.1 [Menubar-Klasse (Menüleiste)], Seite 143, für Details.

Bitte beachten Sie, dass das XML-Tag für die Menuitem-Klasse nur `<item>` und nicht `<menuitem>` ist, wenn Sie Menüpunkte erstellen, die in einem `<menu>`-Baum eingebettet sind. Wenn Sie isolierte Menüeinträge erstellen, die nicht in einem `<menu>`-Baum eingebettet sind, müssen Sie `<menuitem>` verwenden, da RapaGUI sonst nicht wissen könnte, auf welche Klasse Sie sich beziehen, da `<item>` von vielen verschiedenen Klassen verwendet wird.

Wenn Sie keinen Text für den Menüpunkt angeben, wird ein Trennelement erzeugt. Dies ist nützlich, um bestimmte Menüpunkte zusammenzufassen, was zu einer besseren Lesbarkeit des Menüs führt.

Siehe Abschnitt 28.1 [Menubar-Klasse], Seite 143, für ein Beispiel.

29.2 Menuitem.Disabled

BEZEICHNUNG
Menuitem.Disabled – setzt/ermittelt den Deaktivierungsstatus des Menüpunktes

BESCHREIBUNG
Aktiviert (`False`) oder deaktiviert (`True`) den Menüpunkt. Ausserdem können Sie den Deaktivierungsstatus ermitteln.

TYP
Boolesch

ANWENDBARKEIT
ISG

29.3 Menuitem.Help

BEZEICHNUNG
Menuitem.Help – setzt/ermittelt den Hilfetext des Menüeintrag

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzt oder ermittelt den Hilfetext des Menüpunktes. Dieser Text wird automatisch in der Statusleiste des Fensters angezeigt, in dem sich das Menü befindet, wenn Sie mit der Maus über den Menüpunkt fahren. Dies ist für den Anwender sehr komfortabel, da es die Funktion der einzelnen Menüpunkte etwas ausführlicher erklärt. Siehe Abschnitt 43.1 [Statusbar-Klasse (Statusleiste)], Seite 195, für Details.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

29.4 Menuitem.NoAutoKey

BEZEICHNUNG
Menuitem.NoAutoKey – deaktiviert die automatische Tastaturkürzel-Generierung

BESCHREIBUNG
Setzen Sie dieses Attribut, um die automatische Tastaturkürzel-Generierung zu deaktivieren. Standardmäßig wird das Zeichen nach einem Unterstrich in einem Menüpunkt wie ein Tastaturkürzel behandelt, mit der Sie über die Tastatur auf den Menüpunkt zugreifen können. Wenn Sie dieses Attribut setzen, werden Unterstreichungszeichen niemals als Tastaturkürzel behandelt und in den Menüpunkten normal angezeigt.

Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details.

TYP
Boolesch

ANWENDBARKEIT
I

29.5 Menuitem.Selected

BEZEICHNUNG
Menuitem.Selected – benachrichtigt, ob ein Element ausgewählt ist

BESCHREIBUNG
Dieses Attribut wird ausgelöst, wenn der Benutzer einen Menüeintrag auswählt. RapaGUI überwacht dieses Attribut automatisch für alle Menü-Elemente, sodass Sie keine explizite Benachrichtigung über das Attribut MOAI.Notify anfordern müssen.

Bei Menüpunkten, die umgeschaltet werden können oder Teil einer Radio-Gruppe sind, gibt die Abfrage dieses Attributs den aktuellen Umschalt- oder Radio-Status des Menüpunktes zurück.

TYP
Boolesch

ANWENDBARKEIT
ISGN

29.6 Menuitem.Shortcut

BEZEICHNUNG

Menuitem.Shortcut – setzt ein benutzerdefiniertes Tastaturkürzel für den Menüpunkt

BESCHREIBUNG

Legen Sie ein benutzerdefiniertes Tastaturkürzel für einen Menüpunkt fest. Normalerweise werden Menüpunkt-Tastaturkürzel definiert, indem man einfach den Unterstrich im Titel des Menüpunkts verwendet, um eine Tastenkombination zu definieren. Siehe Abschnitt 3.10 [Tastaturkürzel], Seite 17, für Details. Es gibt jedoch bestimmte Fälle, in denen dies nicht ausreicht. Zunächst einmal sind Sie durch die Verwendung des Unterstrichzeichens zur Definition einer Tastenkombination auf ein Zeichen beschränkt, die Teil des Textes des Menüpunkts sind. In vielen Fällen werden Sie jedoch ein Tastaturkürzel definieren wollen, die nicht Teil des Textes ist. Z.B. das typische Tastaturkürzel für "Einfügen" ist `Ctrl+V`, obwohl es kein "V" im Wort "Einfügen" gibt. Hier können Sie mit `Menuitem.Shortcut` beliebige Tastenkombination für Ihre Menüpunkte einrichten.

Zusätzlich ist dieses Attribut auch nützlich, um komplexere Tastenkombination zu definieren, z.B. `Alt+F5` oder `Ctrl+Shift+X` etc. Die Definition solcher komplexen Tastenkombination ist auch nicht über die Standardunterstreichung der Menütexte möglich, daher müssen Sie dafür `Menuitem.Shortcut` verwenden.

Die hier angegebene Zeichenkette muss eine Kombination von Umschalttasten und anderen Tasten sein, die durch die Zeichen `+` oder `-` getrennt sind. Die folgenden Umschalttasten und Tasten werden derzeit erkannt:

`Ctrl` Dies gibt die Control-Taste unter Windows und Linux und die Befehlstaste auf AmigaOS und Mac OS an. Der Grund, warum dies der Befehlstaste auf AmigaOS und Mac OS zugeordnet wurde, ist Ihnen das Schreiben von portablem Code zu erleichtern. Unter Windows und Linux ist die Control-Taste die Standard-Zusatztaste, während AmigaOS und Mac OS die Befehlstaste als Standard-Zusatztaste verwenden. Wenn Sie `Ctrl` angeben, erhalten Sie immer die Standard-Zusatztasten des Systems. Wenn Sie auf AmigaOS und Mac OS die echte Control-Taste verwenden müssen, benutzen Sie `RawCtrl` (siehe unten). Wenn RapaGUI `Ctrl` nicht als spezielles Token behandelt, müssen Sie immer separaten Code für Windows und Linux im Gegensatz zu AmigaOS und Mac OS schreiben, d.h. Sie müssen bei Windows und Linux `Ctrl+V` zum Einfügen angeben und `Cmd+V` zum Einfügen auf AmigaOS und Mac OS. Da dies zu unnötigem Overhead (Zusatzarbeit für den Computer) führt, wird `Ctrl` als spezielle Taste behandelt, die dem standardmäßigen Menükürzel-Taste des Systems zugeordnet ist.

`Alt` Verwenden Sie die Alt-Taste als Umschaltstaste.

`Shift` Verwenden Sie die Shift-Taste als Umschaltstaste.

`RawCtrl` Unter AmigaOS und Mac OS können Sie mit dieser Umschaltstaste die echte Control-Taste überwachen. Wenn Sie stattdessen `Ctrl` verwenden, überwacht RapaGUI die Befehlstaste auf AmigaOS und Mac OS (siehe oben). Unter Windows und Linux ist `RawCtrl` identisch mit `Ctrl`.

`Up` Pfeil nach oben

`Down`	Pfeil nach unten
`Right`	Pfeil nach rechts
`Left`	Pfeil nach links
`Help`	Hilfe-Taste
`Del`	Lösch-Taste
`Backspace`	Rückschritt-Taste
`Tab`	Tabulator-Taste
`Return`	Return-Taste
`Enter`	Enter-Taste
`Esc`	Escape-Taste
`Space`	Leertaste
`F1 - F16`	Funktionstasten
`Insert`	Einfüge-Taste
`Home`	Anfangs-Taste
`End`	Ende-Taste
`PageUp`	Seite rauf
`PageDown`	Seite runter
`Print`	Drucken-Taste
`Pause`	Pause-Taste

Darüber hinaus können Sie auch englische Buchstaben von A bis Z sowie die Zahlen 0 bis 9 als Tastaturkürzel benutzen.

Beachten Sie bitte, dass die Angabe von Tastaturkürzel mit dem Unterstrich und `Menuitem.Shortcut` sich nicht gegenseitig ausschließen. Sie können tatsächlich beide angeben und auf einigen Systemen (z. B. Windows) wird RapaGUI sogar beide unterstützen. Unter Windows können Sie beispielsweise über die Tastatur auf Menü-Elemente zugreifen, indem Sie zunächst auf `Alt` und dann auf den Unterstreichungskürzel drücken, oder Sie können auch auf Menü-Elemente zugreifen, indem Sie ein vordefiniertes Tastaturkürzel drücken, z.B. `Ctrl+V` zum Einfügen. Auf Systemen, die beide Arten von Tastaturkürzel nicht unterstützen, hat die in `Menuitem.Shortcut` angegebenes Tastaturkürzel Vorrang.

TYP

Zeichenkette

ANWENDBARKEIT

I

29.7 Menuitem.Title

BEZEICHNUNG
Menuitem.Title – setzt/ermittelt den Menüpunkttext

BESCHREIBUNG
Setzt oder ermittelt den Menüpunkttext.

Wenn der Text einen Unterstrich enthält, richtet RapaGUI dem Unterstrich folgenden Zeichen automatisch als Tastaturkürzel ein. Wenn Sie dieses Verhalten nicht möchten, setzen Sie `Menuitem.NoAutoKey` auf `True`.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

29.8 Menuitem.Type

BEZEICHNUNG
Menuitem.Type – setzt/ermittelt den Menüpunkttyp

BESCHREIBUNG
Mit diesem Attribut kann der Typ für diesen Menüpunkt eingestellt werden. Folgende Typen sind derzeit möglich:

`Normal`	Normaler Menüpunkt.
`Toggle`	Umnschalt-Menüpunkt mit einem Häkchen.
`Radio`	Radio-Menüpunkt, das mit seinen benachbarten Radio-Menüpunkten eine Gruppe bildet. Es kann immer nur ein Menüpunkt der Gruppe ausgewählt werden.

TYP
Zeichenkette (mögliche Werte siehe oben)

ANWENDBARKEIT
IG

30 MOAI-Klasse

30.1 Übersicht

MOAI-Klasse ist die Oberklasse aller anderen MOAI-Klassen. Es verwaltet einerseits den internen Benachrichtigungsmechanismus, andererseits stellt es auch einige allgemeine Attribute bereit, die mit allen Objekttypen verwendet werden können, z.B. können Sie Benutzerdaten innerhalb Ihrer Objekte mit `MOAI.UserData` speichern.

30.2 MOAI.Class

BEZEICHNUNG
MOAI.Class – ermittelt den Namen der Klasse des Objekts

BESCHREIBUNG
Ermittelt den RapaGUI-Klassennamen eines Objekts.

TYP
Zeichenkette

ANWENDBARKEIT
G

30.3 MOAI.ID

BEZEICHNUNG
MOAI.ID – setzt die Objekt-ID

BESCHREIBUNG
Mit diesem Attribut kann die ID für ein MOAI-Objekt gesetzt werden. Sie müssen Ihren Objekten eindeutige IDs geben, damit Sie mit den Befehlen `moai.Set()`, `moai.Get()` und `moai.DoMethod()` darauf zugreifen können.

TYP
Zeichenkette

ANWENDBARKEIT
I

30.4 MOAI.NoNotify

BEZEICHNUNG
MOAI.NoNotify – deaktiviert Benachrichtigungen

BESCHREIBUNG
Beim Einrichten von Benachrichtigungen für Klassenattribute werden Ereignisse auch dann ausgelöst, wenn der Wert eines Attributs manuell durch Aufruf von `moai.Set()` geändert wird. Durch das Setzen von `MOAI.NoNotify` im selben Aufruf wird verhindert, dass die Benachrichtigung ausgelöst wird.

Beachten Sie, dass `MOAI.NoNotify` ein "einmaliges" Attribut ist. Es ist nur während des aktuellen Aufrufs von `moai.Set()` wirksam!

TYP
Boolesch

ANWENDBARKEIT
S

BEISPIEL
```
moai.Set("lv", "active", 5, "nonotify", true)
```
Der obige Code aktiviert den Listeneintrag Nummer 6, löst aber keine Benachrichtigung aus.

30.5 MOAI.Notify

BEZEICHNUNG
MOAI.Notify – richtet Benachrichtigungen ein

BESCHREIBUNG
Mit diesem Attribut legen Sie fest, welche Attribute Sie überwachen möchten. Immer wenn sich der Wert des Attributs ändert, wird RapaGUI Ihre Callback-Funktion ausführen. Mehrere Attribute müssen durch Semikolons/Strichpunkte (;) getrennt werden. Um beispielsweise die Attribute `Listview.Active` und `Listview.DoubleClick` zu überwachen, müssten Sie hier die Zeichenkette "active; doubleclick" übergeben.

Um Benachrichtigungen außerhalb von XML-Definition einzurichten oder zu entfernen, verwenden Sie den Befehl `moai.Notify()`. Siehe Abschnitt 6.11 [moai.Notify], Seite 49, für Details.

TYP
Zeichenkette

ANWENDBARKEIT
I

30.6 MOAI.NotifyData

BEZEICHNUNG
MOAI.NotifyData – setzt/ermittelt ereignisspezifische Benutzerdaten

BESCHREIBUNG
Mit diesem Attribut können Sie benachrichtigungsspezifische Benutzerdaten in einem Objekt definieren. Sie müssen hier eine Zeichenfolge übergeben, die eine oder mehrere Benachrichtigungen und Benutzerdaten für jede Benachrichtigung in der Zeichenfolge enthält. Wenn eine Benachrichtigung ausgelöst wird, die in der Zeichenfolge angegeben ist, empfängt die Callback-Funktion die Benutzerdaten, die in `MOAI.NotifyData` im Feld `NotifyData` der Ereignismeldung angegeben sind.

Die Zeichenkette, die Sie an dieses Attribut übergeben müssen, muss wie folgt formatiert sein: Name des Benachrichtigungsattributs gefolgt von einem Doppelpunkt (:), gefolgt

von einem Benutzerdaten-Zeichenkette, gefolgt von einem Semikolon (;). Die Sequenz kann dann beliebig oft wiederholt werden.

Zum Beispiel: "Active: foo; DoubleClick: bar;". Wenn die "Active"-Benachrichtigung ausgelöst wird, wird "foo" an die Callback-Funktion gesendet. Wenn das Attribut "DoubleClick" ausgelöst wird, wird "bar" gesendet.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Alle

ANWENDBARKEIT
ISG

30.7 MOAI.UserData

BEZEICHNUNG
MOAI.UserData – setzt/ermittelt die Benutzerdaten des Objekts

BESCHREIBUNG
Speichern Sie mit diesem Attribut beliebige Werte in einem Objekt. Sie können diesen Wert später direkt aus dem Objekt holen. Dies ist ein guter Mechanismus, um die Verwendung von globalen Variablen zu vermeiden.

Die hier angegebenen Benutzerdaten werden auch an Ihre Callback-Funktion übergeben, die Sie mit `InstallEventHandler()` in Hollywood installiert haben. Die Callback-Funktion erhält die in `MOAI.UserData` im Feld `MOAIUserData` der Ereignismeldung angegebenen Benutzerdaten. Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Alle

ANWENDBARKEIT
ISG

31 Pageview-Klasse (Seitenansicht)

31.1 Übersicht

Die Pageview-Klasse (Seitenansicht) bietet eine komfortable Möglichkeit, mehrere Seiten mit Informationen auf einmal anzuzeigen. Wenn Sie diese Klasse verwenden, müssen Sie nur eine Anzahl von Gruppen angeben, die die Elemente der Seitenansicht bilden. Diese Elemente können dann auf verschiedene Arten visualisiert werden, abhängig vom Attribut `Pageview.Mode`, das das Widget zum Durchblättern der Seiten bestimmt. Die folgenden Widgets sind derzeit für diesen Zweck verfügbar:

- Register-Widget (voreingestellt)
- Listen-Widget
- Auswahl-Widget (Choice)
- kein, d.h. Seiten können nur programmmäßig geändert werden (nützlich für Setup-Assistenten etc.)

Wenn Sie eine Seitenansicht einrichten, müssen Sie das Attribut `Group.Title` verwenden, um Titeltexte für Ihre Seitenansicht-Elemente zu definieren. Sie können auch das Attribut `Group.Icon` verwenden, um Symbole für Ihre Seitenansicht-Elemente hinzuzufügen.

Hier ist ein Beispiel für eine dreiseitige Seitenansicht-Gruppe:

```
<pageview>
   <vgroup title="Page 1">
       <listview>
          <column>
             <item>Entry</item>
          </column>
       </listview>
   </vgroup>
   <vgroup title="Page 2">
       <texteditor/>
   </vgroup>
   <vgroup title="Page 3">
       <button>Click me</button>
   </vgroup>
</pageview>
```

31.2 Pageview.Active

BEZEICHNUNG

Pageview.Active – setzt/ermittelt die aktive Seite

BESCHREIBUNG

Setzt oder ermittelt die aktive Seite. Das Seitenansicht-Widget zeigt nur diese Seite an, alle anderen Seiten werden ausgeblendet. Die Seitenindizes reichen von 0 für die erste Seite bis zu `Pageview.Pages`-1 für die letzte.

Sie können hier auch einen der folgenden speziellen Werte übergeben:

`First` Erste Seite.

`Last` Letzte Seite.

`Prev` Vorherige Seite.

`Next` Nächste Seite.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, um zu erfahren, wenn der Benutzer die Seiten wechselt.

TYP
Zahl oder Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
ISGN

31.3 Pageview.Append

BEZEICHNUNG
Pageview.Append – fügt der Seitenansicht eine neue Seite hinzu

ÜBERSICHT
`moai.DoMethod(id, "Append", obj[, active])`

BESCHREIBUNG
Diese Methode kann verwendet werden, um eine neue Seite zu einer Seitenansicht hinzuzufügen. Die durch `obj` angegebene neue Seite muss ein losgelöstes Gruppenobjekt sein. Dieses losgelöste Gruppenobjekt wird dann als letzte Seite der Seitenansicht hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Gruppenobjekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufruf des Befehls `moai.CreateObject()` oder durch explizite Trennung von ihrem übergeordneten Objekt mit der Methode `Pageview.Remove` erzeugt werden.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id` ID des Pageview-Objekts

`obj` ID des Gruppenobjekts, das als neue Seite hinzugefügt werden soll.

`active` optional: setzen Sie dies auf `True`, um die Seite nach dem Hinzufügen zu aktivieren.

31.4 Pageview.GetPageID

BEZEICHNUNG

Pageview.GetPageID – gibt die ID einer Seite innerhalb der Seitenansicht zurück

ÜBERSICHT

```
id$ = moai.DoMethod(id, "GetPageID", idx)
```

BESCHREIBUNG

Diese Methode gibt die ID der Seite am angegebenen Index `idx` innerhalb der Seitenansicht in `id$` zurück. `idx` kann eine absolute Zahl sein, die von 0 bis zur Anzahl der Seiten in der Seitenansicht minus 1 reicht, oder es kann einer der folgenden Sonderwerte sein:

`First`	Erste Seite.
`Last`	Letzte Seite.
`Active`	Aktive Seite.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Pageview-Objekts
`idx`	absoluter Index der gewünschten Seite oder spezielle Zeichenkettenkonstante (siehe oben)

RÜCKGABEWERTE

`id$`	ID des Seitenobjektes beim angegebenen Seitenansicht-Index

31.5 Pageview.Insert

BEZEICHNUNG

Pageview.Insert – fügt eine neue Seite in die Seitenansicht ein

ÜBERSICHT

```
moai.DoMethod(id, "Insert", obj, pos[, active])
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um eine neue Seite an der in `pos` angegebenen Stelle in einer Seitenansicht einzufügen. Die Einfügepositionen werden ab 0 gezählt, was die Position der ersten Seite markiert. Die durch `obj` spezifizierte neue Seite muss ein losgelöstes Gruppenobjekt sein. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte für dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufruf des Befehls `moai.CreateObject()` oder durch explizite Trennung von ihrem übergeordneten Objekt mit der Methode `Pageview.Remove` erzeugt werden.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Pageview-Objekts
`obj`	ID des Gruppenobjekts, das als neue Seite eingefügt werden soll
`pos`	gewünschte Einfügeposition für die neue Seite (ab 0)
`active`	optional: Setzen Sie dies auf `True`, um die Seite nach dem Hinzufügen zu aktivieren

31.6 Pageview.Mode

BEZEICHNUNG

Pageview.Mode – setzt das Browser-Widget zum Durchblättern der Seiten

BESCHREIBUNG

Wählen Sie das Browser-Widget ein, das dieses Pageview-Objekt verwenden soll. Dies kann einer der folgenden Werte sein:

`Tabs`	Verwendet Registerkarten, um durch die Seiten zu blättern. Dies ist der Standardmodus.
`List`	Verwendet ein Listenansichts-Widget (Listview), um durch die Seiten zu blättern.
`Choice`	Verwendet ein Auswahl-Widget (Choice), um durch die Seiten zu blättern.
`None`	Verwendet kein Browser-Widget. Dies bedeutet, dass der Benutzer nicht durch die Seiten blättern kann. Seiten können nur durch Setzen des Attributs `Pageview.Active` geändert werden. Dies ist nützlich für Setup- oder Installationsassistenten, die oft eine benutzerdefinierte Seitenansicht verwenden, die nur dann die Seite wechselt, wenn der Benutzer auf die Schaltfläche `"Nächste Seite"` oder `"Vorherige Seite"` klickt.

TYP

Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

I

31.7 Pageview.Multiline

BEZEICHNUNG

Pageview.Multiline – aktiviert mehrzeilige Registerkarten

PLATTFORMEN

Nur Windows

BESCHREIBUNG

Setzen Sie dies auf `True`, damit das Browser-Widget seine Tabs auf mehreren Zeilen anzeigt, anstatt Scroll-Schaltflächen hinzuzufügen, wenn die Anzahl der Tabs den verfügbaren Platz überschreitet. Offensichtlich macht dieses Attribut nur Sinn, wenn der Modus `Tabs` mit `Pageview.Mode` verwendet wird.

TYP
Boolesch

ANWENDBARKEIT
I

31.8 Pageview.Pages

BEZEICHNUNG
Pageview.Pages – ermittelt die Anzahl der Seiten in der Seitenansicht

BESCHREIBUNG
Mit diesem Attribut können Sie die aktuelle Anzahl der Seiten im Pageview-Objekt ermitteln.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

TYP
Zahl

ANWENDBARKEIT
G

31.9 Pageview.PlainBG

BEZEICHNUNG
Pageview.PlainBG – erzwingt einen normalen Gruppenhintergrund

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Standardmäßig werden die einzelnen Seitenansichts-Elementgruppen mit einer speziellen Hintergrundfarbe gezeichnet, um anzuzeigen, dass es Seitenansichts-Elemente sind. Setzen Sie dieses Attribut auf `True`, wenn Sie diesen speziellen Hintergrund nicht wünschen.

TYP
Boolesch

ANWENDBARKEIT
I

31.10 Pageview.Position

BEZEICHNUNG
Pageview.Position – legt die Position des Browser-Widgets fest

BESCHREIBUNG

Mit diesem Attribut können Sie konfigurieren, wo das Browser-Widget der Seitenansicht (Register-, Liste- oder Auswahl-Widget) erscheinen soll.

Die folgenden Werte werden durch dieses Attribut erkannt:

`Left` Setzt das Browser-Widget links neben die Seiten.

`Right` Setzt das Browser-Widget rechts neben die Seiten.

`Bottom` Setzt das Browser-Widget an den unteren Rand der Seiten.

`Top` Setzt das Browser-Widget an den oberen Rand der Seiten.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

TYP

Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

I

31.11 Pageview.Prepend

BEZEICHNUNG

Pageview.Prepend – stellt einen neue Seite der Seitenansicht voran

ÜBERSICHT

```
moai.DoMethod(id, "Prepend", obj[, active])
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um eine neue Seite einer Seitenansicht voranzustellen. Die durch `obj` angegebene neue Seite muss ein losgelöstes Gruppenobjekt sein. Dieses losgelöste Gruppenobjekt wird dann als erste Seite der Seitenansicht hinzugefügt. Nachdem diese Methode beendet wurde, ändert das angegebene Objekt seinen Zustand von losgelöst in angehängt. Deshalb dürfen Sie keine Befehle mehr für losgelöste Objekte auf dieses Objekt verwenden.

Losgelöste MOAI-Objekte können entweder durch Aufruf des Befehls `moai.CreateObject()` oder durch explizite Trennung von ihrem übergeordneten Objekt mit der Methode `Pageview.Remove` erzeugt werden.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id` ID des Pageview-Objekts

`obj` ID des Gruppenobjekts, das als erste Seite hinzugefügt werden soll

`active` optional: setzen Sie dies auf `True`, um die Seite nach dem Hinzufügen zu aktivieren

31.12 Pageview.Remove

BEZEICHNUNG

Pageview.Remove – entfernt eine Seite aus der Seitenansicht

ÜBERSICHT

```
moai.DoMethod(id, "Remove", obj)
```

BESCHREIBUNG

Diese Methode entfernt die von `obj` angegebene Seite aus der Seitenansicht. Das Gruppenobjekt, das die zu entfernente Seite darstellt, ändert dann seinen Zustand von angehängt auf losgelöst. Das bedeutet, dass Sie es nun mit Methoden wie `Group.Insert` und `Pageview.Append` an eine andere Seitenansicht oder eine andere Gruppe anhängen oder mit `moai.FreeObject()` aus dem Speicher löschen können.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Pageview-Objekts
`obj`	ID der zu entfernenden Seite

32 Popcolor-Klasse (Farbdialog)

32.1 Übersicht

Die Popcolor-Klasse (Farbdialog) erstellt ein Widget, mit dem der Benutzer eine Farbe auswählen kann. Farbdialog-Widgets enthalten in der Regel eine Schaltfläche, die beim Anklicken einen Dialog öffnet, der den Benutzer zur Auswahl einer Farbe auffordert. Dies alles wird automatisch vom Widget verwaltet und Sie müssen nichts tun, außer das Widget irgendwo in Ihrem Fensterlayout zu platzieren.

Sie können den Titel des Farbdialog mit dem Attribut `Popcolor.Title` auf dem Popcolor-Objekt festlegen. Um die aktuelle Farbe eines Popcolor-Objekts zu erhalten oder zu setzen, können Sie das Attribut `Popcolor.RGB` verwenden.

32.2 Popcolor.RGB

BEZEICHNUNG
Popcolor.RGB – setzt die Farbe oder gibt sie zurück

BESCHREIBUNG
Setzen oder holen Sie die Farbdialog-Farbe. Wenn Sie eine Benachrichtigung zu diesem Attribut einrichten, werden Sie benachrichtigt, wenn sich die Farbe ändert. Aus dem Hollywood-Skript wird die Farbe als einfacher Zahlenwert angegeben, der für jede Komponente 8 Bit enthält. Wenn Sie die Farbe in der XML-Datei angeben, muss sie als 6-stellige Zeichenkette mit dem #-Zeichen als Präfix übergeben werden (genau wie in HTML).

TYP
Zahl

ANWENDBARKEIT
ISGN

32.3 Popcolor.Title

BEZEICHNUNG
Popcolor.Title – setzt den Farbdialog-Fenstertitel

BESCHREIBUNG
Setzt den Titel für das Farbdialog-Fenster.

TYP
Zeichenkette

ANWENDBARKEIT
I

33 Popfile-Klasse (Dateidialog)

33.1 Übersicht

Popfile-Klasse (Dateidialog) erstellt ein Widget, mit dem der Benutzer eine Datei auswählen kann. Dateidialog-Widgets enthalten normalerweise eine Schaltfläche, die beim Anklicken einen System-Dialog öffnet, in dem der Benutzer aufgefordert wird, eine Datei auszuwählen. Dies alles wird automatisch vom Widget verwaltet und Sie müssen nichts anderes tun, als das Widget irgendwo in Ihrem Fensterlayout zu platzieren.

Sie können den Titel des Popup-Dialogs mit dem Attribut `Popfile.Title` auf dem Popfile-Objekt setzen und die Auswahl des Benutzers mit dem Attribut `Popfile.File` ermitteln.

33.2 Popfile.File

BEZEICHNUNG
Popfile.File – setzt oder ermittelt den aktuellen Dateidialog-Pfad

BESCHREIBUNG
Ermitteln oder setzen Sie den aktuellen Pfad von diesem Popfile-Objekt.

Sie können für dieses Attribut auch eine Benachrichtigung einrichten, um benachrichtigt zu werden, wenn der Benutzer eine neue Datei für dieses Popfile-Objekt auswählt.

TYP
Zeichenkette

ANWENDBARKEIT
ISGN

33.3 Popfile.Pattern

BEZEICHNUNG
Popfile.Pattern – setzt das Filtermuster für den Datei-System-Dialog

BESCHREIBUNG
Hiermit setzen Sie das Filtermuster für den Datei-System-Dialog. Nur Dateien, die diesem Filtermuster entsprechen, können im System-Dialog ausgewählt werden. Das Filtermuster ist ein Zeichenkette, der eine Reihe von Dateierweiterungen enthält. Diese Endungen müssen durch das Zeichen '|' getrennt werden. Zum Beispiel: "voc|wav|8svx|16sv|iff|aiff" zeigt nur Dateien an, die eine dieser Erweiterungen haben.

Wenn Sie dieses Attribut nicht setzen, sind alle Dateien auswählbar.

TYP
Zeichenkette

ANWENDBARKEIT
I

33.4 Popfile.SaveMode

BEZEICHNUNG
Popfile.SaveMode – aktiviert den Speichermodus

BESCHREIBUNG
Setzen Sie diesen Tag auf `True`, um den Datei-System-Dialog in den Speichermodus zu versetzen.

TYP
Boolesch

ANWENDBARKEIT
I

33.5 Popfile.Title

BEZEICHNUNG
Popfile.Title – setzt den Titel des System-Dialog

BESCHREIBUNG
Legen Sie den Titel für den Datei-System-Dialog fest.

TYP
Zeichenkette

ANWENDBARKEIT
I

34 Popfont-Klasse (Schriftdialog)

34.1 Übersicht

Die Popfont-Klasse (Schriftdialog) erstellt ein Widget, mit dem der Benutzer eine Schriftart auswählen kann. Schriftdialog-Widgets enthalten normalerweise eine Schaltfläche, die beim Anklicken einen System-Dialog öffnet, in dem der Benutzer eine Schriftart auswählen kann. Dies alles wird automatisch vom Widget verwaltet und Sie müssen nichts anderes tun, als das Widget irgendwo in Ihrem Fensterlayout zu platzieren.

Sie können den Titel des Popup-Dialogs mit dem Attribut `Popfile.Title` auf dem Popfile-Objekt setzen und die Auswahl des Benutzers mit dem Attribut `Popfont.Font` ermitteln.

34.2 Popfont.Font

BEZEICHNUNG
Popfont.Font – setzt/ermittelt die aktuelle Schriftauswahl

BESCHREIBUNG
Ermittelt oder setzt die aktuelle Schriftartenauswahl des Benutzers. Die Popfont-Klasse verwendet einen Schrägstrich zur Trennung von Schriftname und -größe, z.B. `Arial/48`.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, wenn der Benutzer eine neue Schriftart mit diesem Popfont-Objekt auswählt.

TYP
Zeichenkette

ANWENDBARKEIT
ISGN

34.3 Popfont.MaxSize

BEZEICHNUNG
Popfont.MaxSize – stellt die maximale Schrifthöhe ein

BESCHREIBUNG
Zeigt nur Schriften an, die gleich oder kleiner als diese Größe sind.

TYP
Zahl

ANWENDBARKEIT
I

34.4 Popfont.MinSize

BEZEICHNUNG
Popfont.MinSize – stellt die minimale Schrifthöhe ein

BESCHREIBUNG
Zeigt nur Schriften an, die gleich oder größer als diese Größe sind.

TYP
Zahl

ANWENDBARKEIT
I

34.5 Popfont.Title

BEZEICHNUNG
Popfont.Title – setzt den Titel des Schriftdialog-Fensters

BESCHREIBUNG
Setzt den Titel für das Schriftdialog-Fenster.

TYP
Zeichenkette

ANWENDBARKEIT
I

35 Poppath-Klasse (Verzeichnisdialog)

35.1 Übersicht

Die Poppath-Klasse (Verzeichnisdialog) erstellt ein Widget, mit dem der Benutzer einen Pfad/ein Verzeichnis auswählen kann. Verzeichnisdialog-Widgets enthalten normalerweise eine Schaltfläche, die beim Anklicken einen Dialog öffnet, in dem der Benutzer einen Pfad auswählen kann. Dies alles wird automatisch vom Widget verwaltet und Sie müssen nichts anderes tun, als das Widget irgendwo in Ihrem Fensterlayout zu platzieren.

Sie können den Titel des Verzeichnisdialog mit dem Attribut `Poppath.Title` setzen und die Auswahl des Benutzers mit dem Attribut `Poppath.Path` ermitteln.

35.2 Poppath.Path

BEZEICHNUNG
Poppath.Path – setzt den aktuellen Verzeichnisdialog-Pfad oder gibt ihn zurück

BESCHREIBUNG
Liefert oder setzt den aktuellen Pfad dieses Poppath-Objekts.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, die ausgelöst wird, sobald der Benutzer einen neuen Pfad für dieses Poppath-Objekt auswählt.

TYP
Zeichenkette

ANWENDBARKEIT
ISGN

35.3 Poppath.Title

BEZEICHNUNG
Poppath.Title – setzt den Titel des Dialogfensters

BESCHREIBUNG
Setzen Sie die Titelzeichenfolge für das Dialogfenster.

TYP
Zeichenkette

ANWENDBARKEIT
I

36 Progressbar-Klasse (Fortschrittsbalken)

36.1 Übersicht

Ein Fortschrittsbalken (Progressbar-Klasse) ist ein Widget, das Aufgaben oder Prozesse visualisiert, die einige Zeit benötigen. Normalerweise wird der Rest der Anwendung blockiert, während ein Fortschrittsbalken angezeigt wird.

Hier ist ein Beispiel, wie in XML ein Fortschrittsbalken definiert wird:

```
<progressbar/>
```

36.2 Progressbar.Horiz

BEZEICHNUNG
Progressbar.Horiz – stellt die Ausrichtung des Fortschrittsbalken ein

BESCHREIBUNG
Für einen vertikalen Fortschrittsbalken setzen Sie dies auf `False`. Standardmäßig wird ein horizontaler Fortschrittsbalken erstellt.

TYP
Boolesch

ANWENDBARKEIT
I

36.3 Progressbar.Level

BEZEICHNUNG
Progressbar.Level – stellt den altuellen Pegel ein oder gibt ihn zurück

BESCHREIBUNG
Stellen Sie den aktuellen Pegel des Fortschrittsbalkens ein. Der neue Pegel des Fortschritts muss zwischen 0 und `Progressbar.Max` liegen. Sie können auch eine Benachrichtigung für dieses Attribut einrichten. Dies kann nützlich sein, wenn Sie ein anderes Widget aktualisieren möchten, wenn sich der aktuelle Pegel ändert.

TYP
Zahl

ANWENDBARKEIT
ISGN

36.4 Progressbar.Max

BEZEICHNUNG
Progressbar.Max – legt den Maximalen Bereich vom Fortschrittsbalken fest

BESCHREIBUNG
Stellen Sie den maximalen Bereich für den Fortschrittsbalken ein. Der Standardwert ist 100.

TYP
Zahl

ANWENDBARKEIT
ISG

37 Radio-Klasse

37.1 Übersicht

Die Radio-Klasse erstellt eine Reihe von sich gegenseitig ausschließenden Schaltflächen (Buttons), die dem Benutzer eine einzige Auswahl ermöglicht. Die Schaltflächen sind innerhalb einer Gruppe mit einem Rahmen und einem optionalen Rahmentitel eingebettet.

Wenn Sie ein Radio-Widget in XML definieren, müssen Sie den Tag `<item>` verwenden, um das Radio-Widget mit Elementen zu füllen. Jedes Radio-Widget muss mindestens ein Element enthalten.

Hier ein Beispiel für einen XML-Auszug zum Erstellen eines Radio-Widgets:

```
<radio id="printer">
    <item>HP Deskjet</item>
    <item>NEC P6</item>
    <item>Okimate 20</item>
</radio>
```

37.2 Radio.Active

BEZEICHNUNG
Radio.Active – setzt oder ermittelt die aktive Radio-Schaltfläche

BESCHREIBUNG
Legen Sie die aktive Schaltfläche im Radio-Widget fest oder rufen Sie sie ab, beginnend mit dem Index 0 für die erste Schaltfläche bis zur Anzahl aller Schaltflächen-1 für die letzte Schaltfläche.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, um zu erfahren, dass der Benutzer eine neue Schaltfläche auswählt.

TYP
Zahl

ANWENDBARKEIT
ISGN

37.3 Radio.Title

BEZEICHNUNG
Radio.Title – legt den Titel für den Radio-Rahmen fest

BESCHREIBUNG
Der Text, den Sie hier angeben, wird als Titeltext in dem Rahmen angezeigt, der um die Radio-Schaltflächen gezeichnet wird.

TYP
Zeichenkette

ANWENDBARKEIT
I

38 Rectangle-Klasse (Rechteck)

38.1 Übersicht

Die Rectangle-Klasse (Rechteck) erstellt einfach leere Rechteck-Objekte, die frei skalierbar sind. Dies mag auf den ersten Blick nicht sehr nützlich erscheinen, aber in der Tat werden häufig Rechteck-Objekte benötigt, um das GUI-Layout zu verfeinern. Sie können für alle Arten von Layouts-Aufgaben verwendet werden, z.B. zur Steuerung der Ausrichtung von Widgets mit fester Größe und vieles mehr.

Zusätzlich werden oft Rechteck-Objekte als Füllfläche neben Objekten benötigt, die selbst nicht größenveränderbar sind. Das Objekt `<radio>` ist beispielsweise nicht größenveränderbar. Um zu verhindern, dass diese statischen Radio-Objekte die Größenänderung Ihrer gesamte GUI blockiert, können Sie sie einfach mit `<rectangle>`-Objekten auffüllen. Hier ist ein Beispiel für ein Radio-Objekt, das mit einem unsichtbaren Rechteck aufgefüllt ist:

```
<hgroup>
  <rectangle/>
  <radio id="printer">
    <item>HP Deskjet</item>
    <item>NEC P6</item>
    <item>Okimate 20</item>
  </radio>
   <rectangle/>
</hgroup>
```

Durch die Verwendung von Rechtecken auf beiden Seiten des Radio-Box-Objekts wird die Radio-Box automatisch innerhalb des verfügbaren GUI-Raums zentriert. Wenn Sie nur ein Rechteck-Objekt vor dem Radio-Objekt verwenden, wird das Radio-Objekt automatisch rechtsbündig ausgerichtet. Ein Rechteck-Objekt nach dem Radio-Objekt würde zu einer linken Ausrichtung des Radio-Objekts führen.

Wenn Sie Objekte mit fester Größe benötigen, verwenden Sie die HSpace- oder VSpace-Klasse (HorizontalAbstand oder VertikalAbstand). Siehe Abschnitt 20.1 [HSpace-Klasse], Seite 103, für Details. Siehe Abschnitt 57.1 [VSpace-Klasse], Seite 261, für Details.

Die Rechteckklasse definiert keine Attribute.

39 Scrollbar-Klasse (Bildlaufleiste)

39.1 Übersicht

Die Scrollbar-Klasse erstellt ein Widget, das eine horizontale oder vertikale Bildlaufleiste darstellt. Ein isoliertes Scrollbar-Objekt macht nicht viel Sinn. Das ist der Grund, warum Scrollbar-Objekte normalerweise mit einem anderen Widget verbunden sind, indem das Attribut `Scrollbar.Target` verwendet wird. Beispielsweise können Sie Bildlaufleisten mit einem Objekt der Hollywood-Klasse verbinden, um ein benutzerdefiniertes Widget zu erstellen, welches Grafiken abhängig von der aktuellen Position der Bildlaufleiste anzeigt.

39.2 Scrollbar.Horiz

BEZEICHNUNG
Scrollbar.Horiz – stellt die Richtung der Bildlaufleiste ein

BESCHREIBUNG
Legen Sie fest, ob Sie eine horizontale (`True`) oder vertikale (`False`) Bildlaufleiste wünschen. Standardmäßig wird eine vertikale Bildlaufleiste erstellt.

TYP
Boolesch

ANWENDBARKEIT
IG

39.3 Scrollbar.Level

BEZEICHNUNG
Scrollbar.Level – setzt/ermittelt die aktuelle Position des Schiebers

BESCHREIBUNG
Setzt oder ermittelt die aktuelle Position des Schiebers. In der Regel möchten Sie eine Benachrichtigung für dieses Attribut einrichten, wenn sich die Position des Schiebers ändert. So können Sie dynamisch auf diese Änderungen reagieren und ein weiteres Widget aktualisieren, das Ihre Bildlaufleiste steuern soll.

TYP
Zahl

ANWENDBARKEIT
ISGN

39.4 Scrollbar.Range

BEZEICHNUNG
Scrollbar.Range – setzt/ermittelt den Bereich der Bildlaufleiste

BESCHREIBUNG
Setzt oder ermittelt den Bereich der Bildlaufleiste.

TYP
Zahl

ANWENDBARKEIT
ISG

39.5 Scrollbar.StepSize

BEZEICHNUNG
Scrollbar.StepSize – setzt/ermittelt die Schrittweite der Bildlaufleiste

BESCHREIBUNG
Stellen Sie die Anzahl der zu scrollenden Pixel ein, wenn der Benutzer auf eine der Schaltflächen der Bildlaufleiste klickt.

TYP
Zahl

ANWENDBARKEIT
ISG

39.6 Scrollbar.Target

BEZEICHNUNG
Scrollbar.Target – setzt das Ziel-Widget für die Bildlaufleiste

BESCHREIBUNG
Verwenden Sie dieses Attribut, um das Widget zu definieren, das von dieser Bildlaufleiste gesteuert werden soll. Dies sollte immer angegeben werden, da sich die Größe des Schiebers in der Bildlaufleiste verändert, wenn sich die Größe des Ziel-Widgets ändert. Wenn Sie hier ein Ziel-Widget angeben, aktualisiert RapaGUI automatisch die Größe des Schiebers in der Bildlaufleiste anhand der Größenänderung des Ziel-Widget. Andernfalls müssen Sie dies manuell vornehmen, indem Sie `Scrollbar.Visible` setzen.

TYP
MOAI-Objekt

ANWENDBARKEIT
I

39.7 Scrollbar.UseWinBorder

BEZEICHNUNG
Scrollbar.UseWinBorder – fügt eine Bildlaufleiste in den Fensterrahmen ein

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Setzen Sie dieses Attribut, damit RapaGUI die Bildlaufleiste in den Fensterrahmen setzt, anstatt ein Widget zu erstellen, welches in ein Gruppenobjekt eingefügt werden kann.

Vor der Verwendung von `Scrollbar.UseWinBorder` müssen Sie zunächst die Randbildlaufleisten für das übergeordnete Fenster aktivieren, indem Sie die entsprechenden Attribute der Window-Klasse (Fenster) setzen. Wenn Sie beispielsweise die Bildlaufleiste in den unteren Fensterrand setzen möchten, müssen Sie zuerst das Attribut `Window.UseBottomBorderScroller` setzen. Siehe Abschnitt 58.1 [Window-Klasse], Seite 263, für Details.

Beachten Sie auch, dass es natürlich nur zwei Bildlaufleisten im Fensterrand geben kann: Einen am linken oder rechten sowie einen am unteren Fensterrand.

Die folgenden Werte werden durch dieses Attribut erkannt:

`Left` Benutzt den linken Fensterrand.

`Right` Benutzt den rechten Fensterrand.

`Bottom` Benutzt den unteren Fensterrand.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
I

39.8 Scrollbar.Visible

BEZEICHNUNG
Scrollbar.Visible – setzt/ermittelt die Anzahl der sichtbaren Einträge

BESCHREIBUNG
Setzt oder ermittelt die Anzahl der sichtbaren Einträge und aktualisiert die Größe des Schiebers in der Bildlaufleiste entsprechend. Wenn Sie kein Ziel-Widget mit `Scrollbar.Target` definiert haben, müssen Sie die Größe des Schiebers mit diesem Attribut aktualisieren, sobald sich die Größe des Ziel-Widgets ändert.

TYP
Zahl

ANWENDBARKEIT
ISG

40 Scrollcanvas-Klasse (Bildlaufleinwand)

40.1 Übersicht

Die Scrollcanvas-Klasse (Bildlaufleinwand) erstellt eine Zeichnungsfläche mit angehängten Bildlaufleisten. Sie können benutzerdefinierte Grafiken über eine Zeichnungs-Callback-Funktion auf dieser Zeichnungsfläche darstellen, die automatisch aufgerufen wird, wenn Inhalte gezeichnet werden sollen. Sie müssen nur eine Benachrichtigung für das Attribut `Scrollcanvas.Paint` einrichten, und Ihre Zeichnungs-Callback-Funktion wird immer dann aufgerufen, wenn Inhalte gezeichnet werden müssen.

Dasselbe kann auch erreicht werden, indem ein Widget, das von Hollywood-Klasse abgeleitet ist, in eine Bildlaufleisten-Gruppe integriert wird oder ein solches Hollywood-Widget mit Bildlaufleisten verbunden wird. Scrollcanvas-Klasse ist jedoch in einigen Fällen vorzuziehen, da sie speziell für Verschiebeinhalte optimiert ist. Sie versucht, das Zeichnen zu minimieren, indem OS-Widgets verwendet werden, die speziell zum Anzeigen von verschiebbaren Inhalt entwickelt wurden. Daher ist diese Klasse normalerweise schneller als die beiden oben beschriebenen Lösungen.

Sie können die Dimensionen der Zeichnungsfläche mithilfe von `Scrollcanvas.VirtWidth` und `Scrollcanvas.VirtHeight` festlegen. Da die Scrollcanvas-Klasse Bildlaufleisten verwendet, können die Abmessungen der Zeichnungsfläche natürlich viel größer sein als die physischen Abmessungen des Bildlaufleinwand-Widgets. Wie bei allen anderen Klassen können Sie diese physischen Dimensionen mit den generischen Attributen `Area.Width` und `Area.Height` festlegen.

Um ein vollständiges Neuzeichnen Ihres Widgets zu erzwingen, führen Sie einfach die Methode `Area.Redraw` auf Ihrem Objekt aus. Wenn Sie `Area.Redraw` für Ihr Objekt ausführen, wird Ihre Zeichnungs-Callback-Funktion aufgerufen, so dass Sie die Zeichnungsfläche entsprechend aktualisieren können.

40.2 Scrollcanvas.AutoBars

BEZEICHNUNG

Scrollcanvas.AutoBars – stellt die Sichtbarkeit der Bildlaufleiste ein

BESCHREIBUNG

Standardmäßig werden die Bildlaufleisten automatisch ausgeblendet, wenn sie nicht benötigt werden. Wenn Sie dieses Verhalten nicht wünschen, setzen Sie dieses Attribut auf `False`. In diesem Fall sind die Bildlaufleisten immer sichtbar, auch wenn sie nicht benötigt werden.

TYP

Boolesch

ANWENDBARKEIT

I

40.3 Scrollcanvas.Paint

BEZEICHNUNG

Scrollcanvas.Paint – benachrichtigt, wenn in die Zeichnungsfläche gemalt wird

BESCHREIBUNG

Richten Sie für dieses Attribut eine Überwachung ein, damit die Callback-Funktion immer dann aufgerufen wird, wenn Inhalte auf die Zeichnungsfläche gemalt werden müssen. Ihre Callback-Funktion kann dann den Zeichnungsflächen-Inhalt abhängig von der aktuellen Position der Bildlaufleiste zeichnen.

RapaGUI übergibt den Identifikator eines Hollywood-Pinsels, dessen Größe genau so groß ist wie der sichtbare Bereich Ihres Bildlaufleinwand-Widgets, an Ihre Callback-Funktion. Sie müssen dann den gewünschten Inhalt auf diesen Pinsel zeichnen. Sie können also nur in dem Rechteck zeichnen, das durch die vier Koordinaten `X`, `Y`, `Width` und `Height` definiert ist, die auch an Ihre Callback-Funktion übergeben werden. Diese vier Koordinaten beschreiben einen rechteckigen Bereich innerhalb der Dimensionen des Pinsels. Wenn ein vollständiges Neuzeichnen benötigt wird, sind `X` sowie `Y` gleich 0 und `Width` sowie `Height` entsprechen den Abmessungen des Pinsels. Meistens wird jedoch nur ein partielles Neuzeichnen benötigt und dann müssen Sie nur auf den Teil des Pinsels zeichnen, der durch diese Koordinaten definiert ist.

Die folgenden zusätzlichen Argumente werden an Ihre Callback-Funktion übergeben:

`Brush:` Enthält den Identifikator eines Pinsels, auf den Sie zeichnen müssen. Verwenden Sie den Befehl `SelectBrush()` von Hollywood, um diesen Pinsel als Ausgabegerät in Ihrer Callback-Funktion auszuwählen. Vergessen Sie nicht, `EndSelect()` aufzurufen, wenn Sie fertig sind!

`ViewWidth:`
Enthält die sichtbare Breite des Widgets. Die ist auch identisch mit der Breite des Pinsels, der an Ihre Callback-Funktion übergeben wird.

`ViewHeight:`
Enthält die sichtbare Höhe des Widgets. Auch die ist identisch mit der Höhe des Pinsels, der an Ihre Callback-Funktion übergeben wird.

`ScrollX:` Enthält die Position des horizontalen Schiebers.

`ScrollY:` Enthält die Position des vertikalen Schiebers.

`VirtWidth:`
Enthält die virtuelle Breite Ihres Widgets. Dies ist der mit `Scrollcanvas.VirtWidth` gesetzte Wert.

`VirtHeight:`
Enthält die virtuelle Höhe Ihres Widgets. Dies ist der mit `Scrollcanvas.VirtHeight` gesetzte Wert.

`X:` Enthält die x-Position innerhalb des Pinsels, an der Sie mit dem Zeichnen beginnen sollen. Siehe oben für Details.

`Y:` Enthält die y-Position innerhalb des Pinsels, an der Sie mit dem Zeichnen beginnen sollen. Siehe oben für Details.

`Width:` Enthält die Anzahl der Spalten, die Sie auf den Pinsel malen sollten (ab `X`). Siehe oben für Details.

`Height:` Enthält die Anzahl der Zeilen, die Sie auf den Pinsel malen sollten (ab `Y`). Siehe oben für Details.

Um die absolute Position des Inhalts zu berechnen, der auf die Zeichnungsfläche gezeichnet werden soll, fügen Sie einfach die Koordinaten `ScrollX+X` sowie `ScrollY+Y` hinzu und Sie sind fertig.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Boolesch

ANWENDBARKEIT
N

40.4 Scrollcanvas.Scroll

BEZEICHNUNG
Scrollcanvas.Scroll – scrollt die Zeichnungsfläche

ÜBERSICHT
```
moai.DoMethod(id, "Scroll", x, y)
```

BESCHREIBUNG
Scrollt die Zeichnungsfläche an die in `x` und `y` angegebene Position.

EINGABEN

`id` ID des Scrollcanvas-Objekts

`x` gewünschte neue x-Position

`y` gewünschte neue y-Position

40.5 Scrollcanvas.StepSize

BEZEICHNUNG
Scrollcanvas.StepSize – setzt/ermittelt die Schrittweite der Bildlaufleisten

BESCHREIBUNG
Legen Sie die Anzahl der zu scrollenden Pixel fest, wenn der Benutzer auf eine der Schaltflächen der Bildlaufleiste klickt. Ausserdem können Sie die zu scrollenden Pixel auch ermitteln.

TYP
Zahl

ANWENDBARKEIT
ISG

40.6 Scrollcanvas.UseLeftBorder

BEZEICHNUNG
Scrollcanvas.UseLeftBorder – verwendet linke statt rechte Bildlaufleiste im Fensterrahmen

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Bei der Verwendung von `Scrollcanvas.UseWinBorder` werden die Bildlaufleisten standardmäßig in den rechten und unteren Fensterrahmen eingefügt. Wenn Sie die vertikale Bildlaufleiste stattdessen im linken Fensterrahmen haben möchten, legen Sie das Attribut `Scrollcanvas.UseLeftBorder` fest.

Vergessen Sie nicht, das Attribut `Window.UseLeftBorderScroller` ebenfalls festzulegen.

TYP
Boolesch

ANWENDBARKEIT
I

40.7 Scrollcanvas.UseWinBorder

BEZEICHNUNG
Scrollcanvas.UseWinBorder – verwendet die Bildlaufleisten im Fensterrahmen

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Setzen Sie dieses Attribut, damit RapaGUI die Bildlaufleisten der Zeichnungsfläche in den Fensterrahmen einfügt, anstatt sie als normale Widgets zu erstellen.

Standardmäßig werden die Bildlaufleisten in den rechten und unteren Fensterrand gesetzt. Wenn Sie stattdessen die vertikale Bildlaufleiste im linken Fensterrand haben möchten, setzen Sie das Attribut `Scrollcanvas.UseLeftBorder`.

Bevor Sie `Scrollcanvas.UseWinBorder` verwenden, müssen Sie zunächst die Rahmen-Bildlaufleisten für das übergeordnete Fenster aktivieren, indem Sie die entsprechenden Fensterklassenattribute festlegen. Für die Standardkonfiguration (rechte und untere Rahmen-Bildlaufleisten) müssten Sie die Attribute `Window.UseBottomBorderScroller` und `Window.UseRightBorderScroller` festlegen. Siehe Abschnitt 58.1 [Window-Klasse], Seite 263, für Details.

TYP
Boolesch

ANWENDBARKEIT
I

40.8 Scrollcanvas.VirtHeight

BEZEICHNUNG
Scrollcanvas.VirtHeight – setzt/ermittelt die Höhe der Zeichnungsfläche

BESCHREIBUNG
Setzt oder ermittelt die Höhe der Zeichnungsfläche in Pixel. Dies wird als "virtuelle Höhe" bezeichnet, da meist nur ein Teil davon tatsächlich sichtbar ist. Der Benutzer kann mit den Bildlaufleisten den gesamten Inhalt der Zeichnungsfläche verschieben/durchblättern.

Dieser Wert muss angegeben werden.

TYP
Zahl

ANWENDBARKEIT
ISG

40.9 Scrollcanvas.VirtWidth

BEZEICHNUNG
Scrollcanvas.VirtWidth – setzt/ermittelt die Breite der Zeichnungsfläche

BESCHREIBUNG
Setzt oder ermittelt die Breite der Zeichnungsfläche in Pixel. Dies wird als "virtuelle Breite" bezeichnet, da meist nur ein Teil davon tatsächlich sichtbar ist. Der Benutzer kann mit den Bildlaufleisten den gesamten Inhalt der Zeichnungsfläche verschieben/durchblättern.

Dieser Wert muss angegeben werden.

TYP
Zahl

ANWENDBARKEIT
ISG

41 Scrollgroup-Klasse (Bildlaufgruppe)

41.1 Übersicht

Die Scrollgroup-Klasse (Bildlaufgruppe) ist eine spezielle Variante der Group-Klasse, die Bildlaufleisten zu Gruppen hinzufügt, um es zu ermöglichen, Gruppen zu verwenden, die größer als der verfügbare GUI-Bereich sind. Der Benutzer kann einfach durch die Gruppe mit den an die Gruppe angehängten Bildlaufleiste blättern. Wenn genügend Platz für die gesamte Gruppe vorhanden ist, werden die Bildlaufleisten automatisch ausgeblendet, es sei denn, sie sind explizit als sichtbar konfiguriert.

Scrollgruppen können ähnlich wie normale Gruppen erstellt werden. Hier ist ein XML-Beispiel:

```
<scrollgroup>
    <radio>
       <item>Amiga 500</item>
       <item>Amiga 1200</item>
       <item>Amiga 4000</item>
    </radio>
    <listview>
       <column/>
    </listview>
</scrollgroup>
```

Der obige XML-Code bettet ein Radio- und ein Listview-Objekt (Listenansicht) in eine Scrollgruppe ein.

41.2 Scrollgroup.AutoBars

BEZEICHNUNG

Scrollgroup.AutoBars – stellt die Sichtbarkeit der Bildlaufleisten ein

BESCHREIBUNG

Standardmäßig werden die Bildlaufleisten automatisch ausgeblendet, wenn sie nicht benötigt werden. Wenn Sie dieses Verhalten nicht wünschen, setzen Sie dieses Attribut auf `False`. In diesem Fall sind die Bildlaufleisten immer sichtbar, auch wenn sie nicht benötigt werden.

TYP

Boolesch

ANWENDBARKEIT

I

41.3 Scrollgroup.Horiz

BEZEICHNUNG

Scrollgroup.Horiz – setzt die Ausrichtung der Scrollgruppen

BESCHREIBUNG
Setzen Sie dies auf `True`, um eine horizontale Scrollgruppe zu erstellen. Standardmäßig verwenden Scrollgruppen die vertikale Ausrichtung.

TYP
Boolesch

ANWENDBARKEIT
I

41.4 Scrollgroup.UseWinBorder

BEZEICHNUNG
Scrollgroup.UseWinBorder – verwendet Fenster-Bildlaufleisten für diese Scrollgruppe

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Setzen Sie dieses Attribut, damit RapaGUI die Bildlaufleisten der Scrollgruppe in den Fensterrand setzt, anstatt sie als normale Widgets zu erstellen.

Vor der Verwendung von `Scrollgroup.UseWinBorder` müssen Sie zunächst die Rand-Bildlaufleisten für das übergeordnete Fenster aktivieren, indem Sie die entsprechenden Fensterklassenattribute setzen. Wenn Sie beispielsweise eine Bildlaufleiste in den unteren Fensterrand einfügen möchten, müssen Sie zuerst das Attribut `Window.UseBottomBorderScroller` setzen. Siehe Abschnitt 58.1 [Window-Klasse], Seite 263, für Details.

TYP
Boolesch

ANWENDBARKEIT
I

42 Slider-Klasse (Schieberegler)

42.1 Übersicht

Die Slider-Klasse (Schieberegler) erstellt ein Widget, mit dem der Benutzer einen numerischen Wert mithilfe eines Schiebers anpassen kann, der vor und zurück gezogen werden kann, um den Wert zu ändern. Schieberegler können abhängig vom Attribut `Slider.Horiz` horizontale oder vertikale Ausrichtung verwenden. Standardmäßig wird ein horizontaler Schieberegler erstellt.

Hier ist ein XML-Beispiel für einen Schieberegler, mit dem der Benutzer eine Zahl zwischen 0 und 100 konfigurieren kann:

```
<slider min="0" max="100"/>
```

42.2 Slider.Horiz

BEZEICHNUNG
Slider.Horiz – setzt/ermittelt die Ausrichtung des Schiebereglers

BESCHREIBUNG
Geben Sie an, ob Sie einen horizontalen (`True`) oder vertikalen (`False`) Schieberegler wünschen. Standardwert ist `True`, was bedeutet, dass ein horizontaler Schieberegler erstellt wird.

TYP
Boolesch

ANWENDBARKEIT
I

42.3 Slider.Level

BEZEICHNUNG
Slider.Level – setzt/ermittelt die aktuelle Position des Schiebers

BESCHREIBUNG
Setzt/ermittelt die aktuelle Position des Schiebers im Schieberegler. Dieser Wert liegt immer zwischen `Slider.Min` und `Slider.Max`.

Sie können auch eine Benachrichtigung über dieses Attribut einrichten, um zu erfahren, wenn sich die Position des Schiebers ändert.

TYP
Zahl

ANWENDBARKEIT
ISGN

42.4 Slider.Max

BEZEICHNUNG
Slider.Max – setzt/ermittelt den Maximalwert des Schiebereglers

BESCHREIBUNG
Setzt oder ermittelt den Maximalwert des Schieberegler-Objekts.

TYP
Zahl

ANWENDBARKEIT
ISG

42.5 Slider.Min

BEZEICHNUNG
Slider.Min – setzt/ermittelt den Minimalwert des Schiebereglers

BESCHREIBUNG
Setzt oder ermittelt den Minimalwert des Schieberegler-Objekts. Dieser kann auch kleiner als 0 sein.

TYP
Zahl

ANWENDBARKEIT
ISG

42.6 Slider.Quiet

BEZEICHNUNG
Slider.Quiet – zeigt den aktuellen Schieberegler-Wert nicht an

BESCHREIBUNG
Stellen Sie dies auf `True`, damit der Schieberegler seinen aktuellen Wert ausblendet. Normalerweise wird der aktuelle Wert entweder im Schieber des Schiebereglers oder in einem Text-Widget neben dem Schieberegler angezeigt.

TYP
Boolesch

ANWENDBARKEIT
I

42.7 Slider.Release

BEZEICHNUNG
Slider.Release – benachrichtigt, wenn der Benutzer den Schieber loslässt

BESCHREIBUNG
Dieses Attribut wird immer dann gesetzt, wenn der Benutzer den Schieber loslässt. Sie können für dieses Attribut eine Benachrichtigung einrichten, um zu erfahren, wenn der Benutzer den Schieber loslässt.

TYP
Boolesch

ANWENDBARKEIT
N

42.8 Slider.Reverse

BEZEICHNUNG
Slider.Reverse – kehrt die Richtung des Schiebereglers um

BESCHREIBUNG
Setzen Sie dieses Attribut auf `True`, um die Richtung des Schiebereglers umzukehren.

TYP
Boolesch

ANWENDBARKEIT
I

43 Statusbar-Klasse (Statusleiste)

43.1 Übersicht

Die Statusbar-Klasse (Statusleiste) erstellt ein Widget am unteren Rand des Fensters, um einige Statusinformationen anzuzeigen. Statusleisten enthalten ein oder mehrere Elemente der Statusbaritem-Klasse (Statusleistenelement), die alle entweder feste oder variable Längen haben können.

Wenn Sie eine Statusleiste in XML erstellen, müssen Sie angeben, wie viele Felder sie enthalten soll, indem Sie den Tag `<item>` verwenden, um Elemente der Statusbaritem-Klasse zu Ihrer Statusleiste hinzuzufügen. Hier ist ein Beispiel für eine Statusleiste mit drei Feldern:

```
<statusbar>
   <item id="first">Welcome to my application</item>
   <item id="second"/>
   <item id="third" width="20"/>
</statusbar>
```

Die Breite der einzelnen Felder können Sie mit dem Attribut `Statusbaritem.Width` einstellen. Oben weisen wir nur dem dritten Feld eine feste Breite zu. Die anderen beiden verwenden variable Breiten.

Der Text von Statusleisten-Elementen kann später durch Setzen des Attributs `Statusbaritem.Text` geändert werden. Beachten Sie jedoch, dass dies oft nicht notwendig ist, da der in den Attributen `Menuitem.Help` und `Toolbarbutton.Help` enthaltene Text automatisch in der Statusleiste angezeigt wird und keinen zusätzlichen Code von Ihrer Seite benötigt.

Es kann nur eine Statusleiste pro Fenster geben und sie muss immer das letzte Element der Wurzelgruppe des Fensters sein. RapaGUI akzeptiert keine Statusleisten an beliebigen Positionen im GUI-Layout. Sie befinden sich also immer am unteren Rand des Fensterlayouts.

Beachten Sie auch, dass es nicht möglich ist, eigenständige Instanzen dieser Klasse mit `moai.CreateObject()` zu erzeugen. Statusleisten müssen immer im Kontext eines Fensters erstellt werden. Wenn Sie also mit `moai.CreateObject()` Statusleisten erstellen wollen, müssen Sie immer ein komplettes Fenster erstellen und die Statusleiste in diese Fensterdefinition einbetten.

Die Statusbar-Klasse definiert selbst keine Attribute oder Methoden. Siehe Abschnitt 44.1 [Statusbaritem-Klasse], Seite 197, für alle notwendigen Informationen.

44 Statusbaritem-Klasse (Statusleistenelement)

44.1 Übersicht

Die Statusbaritem-Klasse (Statusleistenelement) wird verwendet, um Textfelder zu erstellen, die als untergeordnete Elemente für Statusbar-Klasse (Statusleiste) verwendet werden. Sie können keine unabhängigen Instanzen der Statusbaritem-Klasse erstellen. Sie müssen immer in die Statusbar-Klasse eingebettet werden. Siehe Abschnitt 43.1 [Statusbar-Klasse], Seite 195, für Details.

44.2 Statusbaritem.Text

BEZEICHNUNG
Statusbaritem.Text – setzt oder ermittelt den Text des Statusleisteneintrags

BESCHREIBUNG
Setzt den Text des Statusleisteneintrags oder gibt ihn zurück.

TYP
Zeichenkette

ANWENDBARKEIT
SG

44.3 Statusbaritem.Width

BEZEICHNUNG
Statusbaritem.Width – stellt die Breite des Statusleisteneintrags ein

BESCHREIBUNG
Stellen Sie die gewünschte Breite der Statusleisteneintrags ein. Dies kann entweder ein absoluter Pixelwert für ein Feld mit fester Breite oder ein negativer Wert sein, der einen Anteil für ein Feld mit variabler Breite angibt. Der verbleibende Platz für alle Einträge mit variabler Breite wird entsprechend dem Absolutwert dieser Zahl zwischen ihnen aufgeteilt (Absolutwert ist die Zahl ohne Vorzeichen). Ein variabler Eintrag mit einer Breite von -2 erhält doppelt so viel Platz wie ein Eintrag mit einer Breite von -1 und so weiter (Absolutwert von -1 ist 1, von -2 ist 2).

Um z.B. ein Eintrag mit fester Breite der Breite 100 im rechten Teil der Statusleiste und zwei weitere Einträge zu erstellen, die entsprechend 66% bzw. 33% des verbleibenden Platzes erhalten, würden Sie die Breitenfelder der Elemente auf -2, -1 bzw. 100 setzen.

Voreingestellt ist -1.

TYP
Zahl

ANWENDBARKEIT
I

45 Text-Klasse

45.1 Übersicht

Die Text-Klasse erzeugt Widgets, die eine oder mehrere Zeilen Text anzeigen, ähnlich wie die Widgets der Label-Klasse (Beschriftung). Ein Unterschied besteht darin, dass die Text-Klasse mehrere Textzeilen sowie Rahmen um den Text unterstützt. Widgets, die von der Text-Klasse erzeugt werden, sind in der Größe veränderbar, während die Label-Klasse horizontal nicht skalierbar ist.

Bitte beachten Sie, dass die Text-Klasse weder automatischen Zeilenumbruch noch Textformatierung unterstützt. Wenn Sie diese Funktionen haben wollen, müssen Sie die Textview-Klasse (Textanzeige) verwenden. Siehe Abschnitt 48.1 [Textview-Klasse], Seite 223, für Details.

Hier ist ein Beispiel für die Verwendung des Tags `<text>` in XML:

```
<text>Hello World</text>
```

45.2 Text.Align

BEZEICHNUNG
Text.Align – setzt die Textausrichtung oder gibt sie zurück

BESCHREIBUNG
Stellen Sie die gewünschte Ausrichtung für den Text ein. Dies kann einer der folgenden Werte sein:

`Left`	Links ausgerichtet. Dies ist voreingestellt.
`Right`	Rechts ausgerichtet.
`Center`	Zentriert ausgerichtet.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
ISG

45.3 Text.Frame

BEZEICHNUNG
Text.Frame – zeichnet einen Rahmen um das Text-Widget

BESCHREIBUNG
Setzen Sie dies auf `True`, um einen Rahmen um das Text-Widget zu ziehen.

TYP
Boolesch

ANWENDBARKEIT
I

45.4 Text.Text

BEZEICHNUNG
Text.Text – setzt die Textzeichenfolge oder gibt sie zurück

BESCHREIBUNG
Setzen oder ermitteln Sie die Textzeichenfolge. Der Text kann Zeilenumbrüche enthalten.

TYP
Zeichenkette

ANWENDBARKEIT
SG

46 Texteditor-Klasse

46.1 Übersicht

Die Texteditor-Klasse erstellt mehrzeilige Texteingabe-Widgets mit den meisten Funktionen eines normalen Texteditors. Es unterstützt auch die Textformatierung über bestimmte Zeichencodes, wenn `Texteditor.Styled` festgelegt wurde. Dadurch können Sie bestimmte Stile (Fett, Kursiv, Unterstrichen) für Ihren Text aktivieren sowie die Farbe Ihres Textes ändern. Siehe Abschnitt 3.11 [Textformatierungscodes], Seite 18, für Details. Der Inhalt des XML-Tags wird als anfänglicher Inhalt des Texteditor-Widgets verwendet.

Hier ist ein XML-Beispiel, wie Sie ein Texteditor-Objekt in Ihre GUI einbinden können:

```
<texteditor>Enter your text here!</texteditor>
```

Beachten Sie, dass es einen kleinen Unterschied zwischen dem Texteditor-Widget unter Windows und allen anderen Plattformen gibt: Windows' natives Texteditor-Widget verwendet zwei Zeichen für einen Zeilenumbruch (CRLF, d.h. Wagenrücklauf und Zeilenvorschub), während auf allen anderen Plattformen nur ein Zeilenvorschubzeichen verwendet wird. Dies führt zu dem Problem, dass fest codierte Index- oder Bereichspositionen nicht vollständig portierbar sind, da Windows immer zwei Zeichen zum Starten einer neuen Zeile verwendet. Dieses Problem müssen Sie beachten.

Beachten Sie auch, dass diese Klasse auf AmigaOS und kompatiblen Systemen die Erweiterung `TextEditor.mcc` benötigt.

46.2 Texteditor.Align

BEZEICHNUNG
Texteditor.Align – setzt/ermittelt die Textausrichtung

BESCHREIBUNG
Setzt oder ermittelt die Textausrichtung. Folgende Werte sind möglich:

`Left` Links ausgerichtet.

`Right` Rechts ausgerichtet.

`Center` Zentriert ausgerichtet.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
ISG

46.3 Texteditor.AreaMarked

BEZEICHNUNG
Texteditor.AreaMarked – benachrichtigt, wenn Text makriert ist

BESCHREIBUNG
Dieser Tag wird auf `True` gesetzt, wenn Text markiert wurde, und zurück auf `False`, wenn nichts (mehr) markiert ist. Sie können eine Benachrichtigung mit diesem Tag erstellen und Ihre Schaltflächen wie Ausschneiden und Kopieren deaktiviert lassen, wenn nichts markiert ist.

TYP
Boolesch

ANWENDBARKEIT
GN

46.4 Texteditor.Bold

BEZEICHNUNG
Texteditor.Bold – setzt/ermittelt den Schriftstil Fett

BESCHREIBUNG
Dieser Tag zeigt an, ob der Cursor gerade Fett schreibt oder der Block fett formatierter Text ist oder nicht. Sie können eine Benachrichtigung für diesen Tag einrichten, um sich über Stiländerungen zu informieren. Sie können dieses Tag auf `True` für Fett oder `False` für die originale Schriftdicke setzen, wenn Sie den Stil ändern möchten.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

TYP
Boolesch

ANWENDBARKEIT
SGN

46.5 Texteditor.Clear

BEZEICHNUNG
Texteditor.Clear – löscht den gesamten Text

ÜBERSICHT
```
moai.DoMethod(id, "Clear")
```

BESCHREIBUNG
Dadurch wird der gesamte Text im Widget gelöscht. Beachten Sie, dass dies keine Benachrichtigung für `TextEditor.HasChanged` auslöst. Wenn Sie den Text löschen und eine Benachrichtigung für `TextEditor.HasChanged` erhalten möchten, löschen Sie den Text einfach manuell, indem Sie `TextEditor.Text` setzen.

EINGABEN

`id`	ID des Texteditor-Objekts

46.6 Texteditor.Color

BEZEICHNUNG
Texteditor.Color – setzt/ermittelt die aktuelle Textfarbe

BESCHREIBUNG
Mit diesem Attribut kann die aktuelle Textfarbe eingestellt werden. Jeder Text, der nach dem Setzen dieses Attributs eingegeben wird, erscheint in der angegebenen Farbe. Um die Farbe des vorhandenen Textes zu ändern, verwenden Sie stattdessen die Methode `Texteditor.SetColor`. Die Farbe muss als 24-Bit-RGB-Wert übergeben werden.

Sie können auch eine Benachrichtigung über dieses Attribut einrichten, um zu erfahren, wenn der Cursor über Text in einer anderen Farbe bewegt wurde.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

TYP
Zahl

ANWENDBARKEIT
SGN

46.7 Texteditor.Copy

BEZEICHNUNG
Texteditor.Copy – kopiert den markierten Text

ÜBERSICHT
```
moai.DoMethod(id, "Copy")
```

BESCHREIBUNG
Kopiert den aktuell ausgewählten Text in die Zwischenablage.

EINGABEN

`id`	ID des Texteditor-Objekts

46.8 Texteditor.CursorPos

BEZEICHNUNG
Texteditor.CursorPos – setzt/ermittelt die Cursorposition

BESCHREIBUNG
Mit diesem Tag können Sie die Position des Cursors setzen oder ermitteln. Das Anfangszeichen beginnt an Position 0.

Sie können auch eine Benachrichtigung für diesem Tag einrichten, um zu erfahren, wann der Cursor bewegt wird.

TYP
Zahl

ANWENDBARKEIT
SGN

46.9 Texteditor.Cut

BEZEICHNUNG
Texteditor.Cut – schneidet den markierten Text aus

ÜBERSICHT
```
moai.DoMethod(id, "Cut")
```

BESCHREIBUNG
Schneidet den aktuell markierten Text aus und legt ihn in die Zwischenablage.

EINGABEN

`id`	ID des Texteditor-Objekts

46.10 Texteditor.GetSelection

BEZEICHNUNG
Texteditor.GetSelection – ermittelt die Textauswahl

ÜBERSICHT
```
start, end = moai.DoMethod(id, "GetSelection")
```

BESCHREIBUNG
Diese Methode gibt die Start- und Endposition des aktuell ausgewählten Textes zurück. Das erste Zeichen des Gesamtextes steht an Position 0. Ist kein Bereich markiert, wird für beide Werte -1 zurückgegeben.

EINGABEN

`id`	ID des Texteditor-Objekts

RÜCKGABEWERTE

`start`	Startversatz des markierten Textes
`end`	Endversatz des markierten Textes

46.11 Texteditor.GetText

BEZEICHNUNG
Texteditor.GetText – exportiert einen Teil des aktuellen Textes

ÜBERSICHT
```
t$ = moai.DoMethod(id, "GetText", start, end)
```

BESCHREIBUNG
Diese Methode exportiert den Teil des aktuellen Textes zwischen `start` und `end` und gibt ihn zurück. Positionen werden im Gesamttext ab 0 gezählt.

EINGABEN

`id`	ID des Texteditor-Objekts
`start`	Startposition des gewünschten Blocks
`end`	Endposition des gewünschten Blocks

RÜCKGABEWERTE

`t$`	Text der angegebenen Blockkoordinaten

46.12 Texteditor.GetXY

BEZEICHNUNG

Texteditor.GetXY – wandelt die Indexposition in Spalte und Zeile um

ÜBERSICHT

```
x, y = moai.DoMethod(id, "GetXY", pos)
```

BESCHREIBUNG

Diese Methode konvertiert die angegebene Indexposition in ihre Spalten- und Zeilenkoordinaten. Sowohl Spalten- als auch Zeilenzähler beginnen bei Index 0.

EINGABEN

`id`	ID des Texteditor-Objekts
`pos`	zu konvertierende Indexposition

RÜCKGABEWERTE

`x`	Spaltenoffset des angegebenen Indexes
`y`	Zeilenoffset des angegebenen Indexes

46.13 Texteditor.HasChanged

BEZEICHNUNG

Texteditor.HasChanged – benachrichtigt, wenn sich der Textinhalt ändert

BESCHREIBUNG

Dieses Attribut wird gesetzt, wenn sich der Inhalt des Textes ändert. Sie können eine Benachrichtigung zu diesem Attribut einrichten, um sich über diese Änderungen zu informieren.

TYP

Boolesch

ANWENDBARKEIT

GN

46.14 Texteditor.Insert

BEZEICHNUNG
Texteditor.Insert – fügt Text ein

ÜBERSICHT
```
moai.DoMethod(id, "Insert", t$, pos$)
```

BESCHREIBUNG
Dadurch wird der in `t$` angegebene Text an der in `pos$` angegebenen Position eingefügt. Die Position des eingefügten Textes kann eine der folgenden Werte sein:

`Cursor`	Fügt den Text an der aktuellen Cursorposition ein.
`Top`	Fügt den Text ganz am Anfang ein.
`Bottom`	Hängt den Text am Schluss an.

EINGABEN

`id`	ID des Texteditor-Objekts
`t$`	der einzufügende Text
`pos$`	Einfügeposition (siehe oben für mögliche Werte)

46.15 Texteditor.Italic

BEZEICHNUNG
Texteditor.Italic – setzt/ermittelt den Schriftstil Italic

BESCHREIBUNG
Dieser Tag zeigt an, ob der Cursor gerade kursiv schreibt oder der Block kursiv formatierter Text ist oder nicht. Sie können eine Benachrichtigung für diesen Tag einrichten, um sich über Stiländerungen zu informieren. Wenn Sie den Stil ändern möchten, können Sie diesen Tag auf `True` für Kursiv setzen oder `False`, um Kursiv aufzuheben.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

TYP
Boolesch

ANWENDBARKEIT
SGN

46.16 Texteditor.Mark

BEZEICHNUNG
Texteditor.Mark – markiert den Text

ÜBERSICHT
```
moai.DoMethod(id, "Mark", start, end)
```

BESCHREIBUNG
Diese Methode markiert den Text in dem durch `start` und `end` begrenzten Bereich. Das erste Zeichen des Gesamttextes beginnt bei Position 0.

EINGABEN

`id`	ID des Texteditor-Objekts
`start`	Startposition
`end`	Endposition

46.17 Texteditor.MarkAll

BEZEICHNUNG
Texteditor.MarkAll – markiert den gesamten Text

ÜBERSICHT
```
moai.DoMethod(id, "MarkAll")
```

BESCHREIBUNG
Markiert den gesamten Text im Widget.

EINGABEN

`id`	ID des Texteditor-Objekts

46.18 Texteditor.MarkNone

BEZEICHNUNG
Texteditor.MarkNone – löscht den ausgewählten Text

ÜBERSICHT
```
moai.DoMethod(id, "MarkNone")
```

BESCHREIBUNG
Löscht den ausgewählten Text.

EINGABEN

`id`	ID des Texteditor-Objekts

46.19 Texteditor.NoWrap

BEZEICHNUNG
Texteditor.NoWrap – deaktiviert den Wortumbruch

BESCHREIBUNG
Standardmäßig verwenden Texteditor-Widgets automatischen Wortumbruch, wenn Wörter über den verfügbaren Widget-Platz hinaus laufen. Setzen Sie dieses Tag auf `True`, wenn Sie das nicht wollen. In diesem Fall verwendet das Texteditor-Widget eine horizontale Bildlaufleiste, anstatt Wörter in die nächste Zeile einzufügen.

TYP
Boolesch

ANWENDBARKEIT
I

46.20 Texteditor.Paste

BEZEICHNUNG
Texteditor.Paste – fügt den Text aus der Zwischenablage ein

ÜBERSICHT
```
moai.DoMethod(id, "Paste")
```

BESCHREIBUNG
Fügt den Text aus der Zwischenablage in das Texteditor-Widget ein.

EINGABEN

`id`	ID des Texteditor-Objekts

46.21 Texteditor.ReadOnly

BEZEICHNUNG
Texteditor.ReadOnly – setzt den Texteditor in den Nur-Lese-Modus

BESCHREIBUNG
Um das Widget in den Nur-Lese-Modus zu versetzen, setzen Sie diesen Tag auf `True`.

Dies ist wahrscheinlich nicht sehr nützlich, da es die Textview-Klasse (Textanzeige) für die Anzeige von nicht editierbarem Text gibt. Siehe Abschnitt 48.1 [Textview-Klasse], Seite 223, für Details.

TYP
Boolesch

ANWENDBARKEIT
I

46.22 Texteditor.Redo

BEZEICHNUNG
Texteditor.Redo – hebt ein vorangehendes Undo auf

ÜBERSICHT
```
moai.DoMethod(id, "Redo")
```

BESCHREIBUNG
Mit Redo können Sie ein vorangegangenes Undo des Texteditor-Widgets wieder aufheben.

EINGABEN

`id`	ID des Texteditor-Objekts

46.23 Texteditor.RedoAvailable

BEZEICHNUNG
Texteditor.RedoAvailable – benachrichtigt, wenn Redo verfügbar ist

BESCHREIBUNG
Dieser Tag wird auf `True` gesetzt, wenn der Benutzer ein vorangehendes Undo aufheben kann. Sie können eine Benachrichtigung für diesem Tag erstellen und Ihre Redo-Schaltfläche deaktivieren, wenn es kein Undo zum aufheben hat.

TYP
Boolesch

ANWENDBARKEIT
GN

46.24 Texteditor.SetBold

BEZEICHNUNG
Texteditor.SetBold – aktiviert/deaktiviert den Stil Fett des Textblocks

ÜBERSICHT
```
moai.DoMethod(id, "SetBold", start, end, flag)
```

BESCHREIBUNG
Diese Methode aktiviert oder deaktiviert den fettgedruckten Stil des Textblocks, der durch die Koordinaten `start` und `end` definiert ist. Wenn das Argument `flag` auf `True` gesetzt ist, wird der angegebenen Textblock fett ausgegeben, andernfalls wird der fette Stil aus dem Textblock entfernt.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

EINGABEN

`id`	ID des Texteditor-Objekts
`start`	Startposition (Gesamttext beginnt bei 0)
`end`	Endposition
`flag`	mit `True` wird der Textblock fett ausgegeben, mit `False` hingegen aufgehoben

46.25 Texteditor.SetColor

BEZEICHNUNG

Texteditor.SetColor – ändert die Farbe des Textblocks

ÜBERSICHT

```
moai.DoMethod(id, "SetColor", start, end, color)
```

BESCHREIBUNG

Diese Methode ändert die Farbe eines Textblocks, der durch die Koordinaten `start` und `end` definiert ist. Die Farbe muss als 24-Bit-RGB-Wert in `color` übergeben werden.

Um die Farbe zu ändern, die für neu eingefügten Text verwendet werden soll, verwenden Sie stattdessen das Attribut `Texteditor.Color`.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

EINGABEN

`id`	ID des Texteditor-Objekts
`start`	Startposition (Gesamttext beginnt bei 0)
`end`	Endposition
`color`	gewünschte Farbe für den Text

46.26 Texteditor.SetItalic

BEZEICHNUNG

Texteditor.SetItalic – aktiviert/deaktiviert den Stil Italic des Textblocks

ÜBERSICHT

```
moai.DoMethod(id, "SetItalic", start, end, flag)
```

BESCHREIBUNG

Diese Methode aktiviert oder deaktiviert den Kursivdruck des Textblocks, der durch die Koordinaten `start` und `end` definiert ist. Wenn das Argument `flag` auf `True` gesetzt ist, wird der angegebenen Textblock kursiv ausgegeben, andernfalls wird der kursive Stil aus dem Textblock entfernt.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

EINGABEN

`id`	ID des Texteditor-Objekts
`start`	Startposition (Gesamttext beginnt bei 0)
`end`	Endposition
`flag`	mit `True` wird der Textblock kursiv ausgegeben, mit `False` hingegen aufgehoben

46.27 Texteditor.SetUnderline

BEZEICHNUNG

Texteditor.SetUnderline – aktiviert/deaktiviert den Stil Unterstrichen des Textblocks

ÜBERSICHT

```
moai.DoMethod(id, "SetUnderline", start, end, flag)
```

BESCHREIBUNG

Diese Methode aktiviert oder deaktiviert den Unterstreichungsstil des Textblocks, der durch die Koordinaten `start` und `end` definiert ist. Wenn das Argument `flag` auf `True` gesetzt ist, wird der angegebenen Textblock unterstrichen ausgegeben, andernfalls wird der Unterstreichungsstil aus dem Textblock entfernt.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

EINGABEN

`id`	ID des Texteditor-Objekts
`start`	Startposition (Gesamttext beginnt bei 0)
`end`	Endposition
`flag`	mit `True` wird der Textblock unterstrichen ausgegeben, mit `False` hingegen aufgehoben

46.28 Texteditor.Styled

BEZEICHNUNG

Texteditor.Styled – aktiviert die Textformatierung

BESCHREIBUNG

Setzen Sie dieses Attribut auf `True`, um die Textformatierung für dieses Widget zu aktivieren. Wenn diese Option auf `True` gesetzt ist, können Sie in dem Text, den Sie an dieses Widget übergeben, spezielle Textformatierungscodes verwenden. Siehe Abschnitt 3.11 [Textformatierungscodes], Seite 18, für Details.

TYP

Boolesch

ANWENDBARKEIT

I

46.29 Texteditor.Text

BEZEICHNUNG

Texteditor.Text – setzt/ermittelt den Inhalt des Texteditors

BESCHREIBUNG

Verwenden Sie dieses Attribut, um den Inhalt des Texteditor-Objekts zu setzen oder zu ermitteln.

Die hier angegebene Zeichenkette kann Textformatierungscodes verwenden, wenn `Texteditor.Styled` auf `True` gesetzt wurde. Siehe Abschnitt 3.11 [Textformatierungscodes], Seite 18, für Details.

Beachten Sie, dass das Setzen dieses Attributs eine Benachrichtigung an `TextEditor.HasChanged` auslöst. Wenn Sie das nicht wollen, setzen Sie einfach das Attribut `MOAI.NoNotify` auf `True` in Ihrem Aufruf mit `moai.Set()`.

TYP
Zeichenkette

ANWENDBARKEIT
SG

46.30 Texteditor.Underline

BEZEICHNUNG
Texteditor.Underline – setzt/ermittelt den Schriftstil Unterstrichen

BESCHREIBUNG
Dieser Tag zeigt an, ob der Cursor oder Block über unterstrichenem Text steht oder nicht. Sie können eine Benachrichtigung für diesen Tag einrichten, um sich über Stiländerungen zu informieren. Sie können diesen Tag auf `True` (unterstrichen) oder `False` (unterstrichen aufgehoben) setzen, wenn Sie den Stil ändern möchten.

Beachten Sie, dass `Texteditor.Styled` auf `True` gesetzt werden muss, um diese Fähigkeit zu benutzen.

TYP
Boolesch

ANWENDBARKEIT
SGN

46.31 Texteditor.Undo

BEZEICHNUNG
Texteditor.Undo – Undo macht die letzte Operation rückgängig

ÜBERSICHT
```
moai.DoMethod(id, "Undo")
```

BESCHREIBUNG
Mit Undo wird die letzte Operation des Texteditor-Widgets rückgängig gemacht.

EINGABEN

`id` ID des Texteditor-Objekts

46.32 Texteditor.UndoAvailable

BEZEICHNUNG

Texteditor.UndoAvailable – benachrichtigt, wenn Undo verfügbar ist

BESCHREIBUNG

Dieser Tag wird auf `True` gesetzt, wenn der Benutzer seine Aktion(en) rückgängig machen kann. Sie können eine Benachrichtigung für diesen Tag erstellen und Ihre Undo-Schaltfläche deaktivieren, wenn es nichts rückgängig zu machen gibt.

TYP

Boolesch

ANWENDBARKEIT

GN

47 Textentry-Klasse (Texteingabe)

47.1 Übersicht

Die Textentry-Klasse generiert die bekannten einzeiligen Texteingabe-Widgets, die zur Anzeige und Eingabe einer Text-Zeichenkette verwendet werden können.

Standardmäßig haben Texteingabe-Widgets keine Beschriftung (Label) neben sich. Wenn Sie eine Beschriftung neben Ihrem Texteingabe-Widget haben möchten, müssen Sie es in eine `<hgroup>` einfügen und dann die Label-Klasse verwenden, um eine Beschriftung daneben zu setzen.

Der Inhalt des XML-Tags wird als Anfangsinhalt des Texteingabe-Widgets verwendet.

Hier ist ein XML-Beispiel für ein Texteingabe-Widget:

```
<textentry id="mytextentry"/>
```

Beachten Sie, dass die Textentry-Klasse nur einzeilige Widgets unterstützt. Wenn Sie ein mehrzeiliges Texteingabe-Widget benötigen, müssen Sie stattdessen die Texteditor-Klasse verwenden. Siehe Abschnitt 46.1 [Texteditor-Klasse], Seite 201, für Details.

47.2 Textentry.Accept

BEZEICHNUNG
Textentry.Accept – setzt die Zeichen, die vom Texteingabe-Widget akzeptiert werden

BESCHREIBUNG
Setzen Sie dies auf eine Zeichenkette, die Zeichen enthält, welche vom Widget akzeptiert werden. Nützlich zum Beispiel, wenn Sie nur die Eingabe von Zahlen erlauben wollen. In diesem Fall würden Sie `Textentry.Accept` auf `"0123456789"` setzen.

Sie können auch `Textentry.Reject` verwenden, um Zeichen selektiv abzulehnen.

TYP
Zeichenkette

ANWENDBARKEIT
I

47.3 Textentry.Acknowledge

BEZEICHNUNG
Textentry.Acknowledge – benachrichtigt, wenn der Benutzer RETURN drückt

BESCHREIBUNG
Immer wenn der Benutzer die Taste Return drückt, wird dieses Attribut auf `True` gesetzt. Sie können diese Benachrichtigung überwachen und die entsprechenden Maßnahmen ergreifen.

Das Drücken der TAB-Taste oder das Klicken mit der Maus zum Deaktivieren des Widgets löst `Textentry.Acknowledge` nicht aus.

TYP
Boolesch

ANWENDBARKEIT
N

47.4 Textentry.AdvanceOnCR

BEZEICHNUNG
Textentry.AdvanceOnCR – aktiviert das nächste Objekt, wenn RETURN gedrückt wird

BESCHREIBUNG
Wenn Sie dies auf `True` setzen, verhält sich das Drücken von RETURN wie das Drücken der TAB-Taste, d.h. es gibt den Fokus auf das nächste Widget im Fenster.

TYP
Boolesch

ANWENDBARKEIT
I

47.5 Textentry.Copy

BEZEICHNUNG
Textentry.Copy – kopiert den markierten Text

ÜBERSICHT
```
moai.DoMethod(id, "Copy")
```

BESCHREIBUNG
Kopiert den aktuell ausgewählten Text in die Zwischenablage.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Textentry-Objekts

47.6 Textentry.CursorPos

BEZEICHNUNG
Textentry.CursorPos – setzt/ermittelt die Cursorposition

BESCHREIBUNG
Setzt oder ermittelt die aktuelle Position des Cursor.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

TYP
Zahl

ANWENDBARKEIT
SG

47.7 Textentry.Cut

BEZEICHNUNG
Textentry.Cut – schneidet den markierten Text aus

ÜBERSICHT
```
moai.DoMethod(id, "Cut")
```

BESCHREIBUNG
Diese Methode schneidet den aktuell markierten Text aus und legt ihn in der Zwischenablage ab.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Textentry-Objekts

47.8 Textentry.GetSelection

BEZEICHNUNG
Textentry.GetSelection – ermittelt den Bereich der Textauswahl

ÜBERSICHT
```
start, end = moai.DoMethod(id, "GetSelection")
```

BESCHREIBUNG
Gibt den Bereich des aktuell ausgewählten Textes zurück, der bei `start` anfängt und bei `end` aufhört. Der Versatz vom gesamten Text beginnt bei 0, wobei mit 0 das erste Zeichen angegeben wird. Wenn kein Text markiert ist, wird für beide Werte -1 zurückgegeben.

EINGABEN

`id`	ID des Textentry-Objekts

RÜCKGABEWERTE

`start`	der Beginn des Bereichs
`end`	das Ende des Bereichs

47.9 Textentry.Insert

BEZEICHNUNG
Textentry.Insert – fügt den Text ein

ÜBERSICHT
```
moai.DoMethod(id, "Insert", t$)
```

BESCHREIBUNG

Dadurch wird der in `t$` angegebener Text an der aktuellen Cursorposition eingefügt.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Textentry-Objekts
`t$`	Text, der eingefügt wird

47.10 Textentry.Mark

BEZEICHNUNG

Textentry.Mark – markiert den Text

ÜBERSICHT

```
moai.DoMethod(id, "Mark", start, end)
```

BESCHREIBUNG

Markiert den Text von der ersten Position `start` bis zum Zeichen an der letzten Position `end`. Das erste Zeichen vom gesamten Text steht an der Position 0.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Textentry-Objekts
`start`	Position des ersten zu markierenden Zeichens
`end`	Position des letzten zu markierenden Zeichens

47.11 Textentry.MarkAll

BEZEICHNUNG

Textentry.MarkAll – markiert den gesamten Text

ÜBERSICHT

```
moai.DoMethod(id, "MarkAll")
```

BESCHREIBUNG

Mit dieser Methode markieren Sie den gesamten Text im Widget.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Textentry-Objekts

47.12 Textentry.MarkNone

BEZEICHNUNG
Textentry.MarkNone – entfernt die Markierung

ÜBERSICHT
```
moai.DoMethod(id, "MarkNone")
```

BESCHREIBUNG
Diese Methode entfernt jede Textauswahl. Der Text wird komplett unmarkiert, wenn diese Methode beendet ist.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id`	ID des Textentry-Objekts

47.13 Textentry.MaxLen

BEZEICHNUNG
Textentry.MaxLen – stellt die maximale Eintragslänge ein

BESCHREIBUNG
Legt die maximale Anzahl von Zeichen fest, die eingegeben werden kann.

TYP
Zahl

ANWENDBARKEIT
IG

47.14 Textentry.Password

BEZEICHNUNG
Textentry.Password – setzt das Widget in den Passwortmodus

BESCHREIBUNG
Wenn Sie dieses Attribut setzen, wird die Benutzereingabe ausgeblendet. Nützlich vor allem bei der Eingabe von Passwörtern.

TYP
Boolesch

ANWENDBARKEIT
I

47.15 Textentry.Paste

BEZEICHNUNG
Textentry.Paste – fügt den Text aus der Zwischenablage ein

ÜBERSICHT
```
moai.DoMethod(id, "Paste")
```

BESCHREIBUNG
Fügt den Text aus der Zwischenablage in das Texteingabe-Widget ein.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id` ID des Textentry-Objekts

47.16 Textentry.Redo

BEZEICHNUNG
Textentry.Redo – hebt ein vorangehendes Undo auf

ÜBERSICHT
```
moai.DoMethod(id, "Redo")
```

BESCHREIBUNG
Diese Methode hebt ein vorangehendes Undo in einem Texteingabe-Widget auf.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id` ID des Textentry-Objekts

47.17 Textentry.Reject

BEZEICHNUNG
Textentry.Reject – setzt die Zeichen, welche vom Texteingabe-Widget abgelehnt werden

BESCHREIBUNG
Setzen Sie dieses Attribut auf eine Zeichenkette, die Zeichen enthält, die vom Widget abgelehnt werden sollen. Wenn Sie beispielsweise die Eingabe von Zahlen nicht zulassen möchten, setzen Sie `Textentry.Reject` auf `"0123456789"`.

Sie können auch `Textentry.Accept` verwenden, um Zeichen selektiv zu akzeptieren.

TYP
Zeichenkette

ANWENDBARKEIT
I

47.18 Textentry.Text

BEZEICHNUNG

Textentry.Text – setzt/ermittelt den Inhalt vom Texteingabe-Widget

BESCHREIBUNG

Setzen oder ermitteln Sie den Inhalt eines Texteintrags-Widgets.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, um benachrichtigt zu werden, wenn sich der Inhalt des Widgets ändert. Beachten Sie, dass dies dazu führt, dass bei jedem Tastendruck ein Ereignis ausgelöst wird.

TYP

Zeichenkette

ANWENDBARKEIT

SGN

47.19 Textentry.Undo

BEZEICHNUNG

Textentry.Undo – Undo macht die letzte Operation rückgängig

ÜBERSICHT

```
moai.DoMethod(id, "Undo")
```

BESCHREIBUNG

Mit Undo nehmen Sie die letzte Operation des Texteingabe-Widgets zurück.

Unter AmigaOS und kompatiblen Betriebssystemen erfordert diese Fähigkeit mindestens MUI 4.0.

EINGABEN

`id` ID des Textentry-Objekts

48 Textview-Klasse (Textanzeige)

48.1 Übersicht

Die Textview-Klasse (Textanzeige) ist eine Klasse, die größere Textmengen mit Unterstützung für Zeilenumbruch und Textformatierung in einem Widget anzeigen kann. Der Inhalt des XML-Tags wird als Anfangsinhalt des Textanzeige-Widgets verwendet.

Hier ist ein Beispiel für die Verwendung des XML-Tags `<textview>`:

```
<textview>Hello World</textview>
```

Hier ist das gleiche Beispiel in fett ausgegeben:

```
<textview styled="true">\33bHello World</textview>
```

Die hier angegebene Zeichenkette kann Textformatierungscodes verwenden, wenn `Textview.Styled` auf `True` gesetzt wurde. Siehe Abschnitt 3.11 [Textformatierungscodes], Seite 18, für Details.

48.2 Textview.Align

BEZEICHNUNG
Textview.Align – setzt die Textausrichtung oder gibt sie zurück

BESCHREIBUNG
Legen Sie die gewünschte Textausrichtung für mehrzeiligen Text fest. Dies kann einer der folgenden Werte sein:

`Left`	Links ausgerichtet.
`Right`	Rechts ausgerichtet.
`Center`	Zentriert ausgerichtet.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
ISG

48.3 Textview.Styled

BEZEICHNUNG
Textview.Styled – aktiviert die Textformatierung

BESCHREIBUNG
Setzen Sie dieses Attribut auf `True`, um die Textformatierung für dieses Widget zu aktivieren. Wenn auf `True` gesetzt, können Sie spezielle Textformatierungscodes in dem Text verwenden, die Sie an dieses Widget übergeben. Siehe Abschnitt 3.11 [Textformatierungscodes], Seite 18, für Details.

TYP
Boolesch

ANWENDBARKEIT
I

48.4 Textview.Text

BEZEICHNUNG
Textview.Text – setzt den Inhalt des Textview-Objekts oder gibt ihn zurück

BESCHREIBUNG
Hiermit wird der Text gesetzt, der vom Textanzeige-Widget angezeigt werden soll. Aber Sie können auch den Text aus dem Textanzeige-Widget auslesen. Der Text kann Zeilenumbrüche enthalten und wird automatisch umgebrochen, sobald er die Grenzen des Widgets erreicht. Mehrzeiliger Text wird entsprechend der mit `Textview.Align` eingestellten Ausrichtung ausgegeben.

Die hier angegebene Zeichenkette kann Textformatierungscodes verwenden, wenn `Textview.Styled` auf `True` gesetzt wurde.

TYP
Zeichenkette

ANWENDBARKEIT
SG

49 Toolbar-Klasse (Symbolleiste)

49.1 Übersicht

Die Toolbar-Klasse (Symbolleiste) erstellt eine Leiste mit Bildschaltflächen, die normalerweise am oberen Rand eines Fensters platziert wird. Die Schaltflächen können in verschiedenen Ansichtsmodi angezeigt werden: Entweder als Bild, Bild und Text oder nur als Text. Normalerweise werden Schaltflächen in der Symbolleiste nur als Bilder angezeigt.

Wenn Sie ein Symbolleisten-Widget im XML-Code definieren, müssen Sie immer mindestens eine Symbolleistenschaltfläche hinzufügen. Dies geschieht mit der Toolbarbutton-Klasse (Symbolleistenschaltflächen). Hier ist eine Beispiel-XML-Definition einer Symbolleiste mit sechs Schaltflächen:

```
<toolbar>
   <button icon="1">Open</button>
   <button icon="2">Save</button>
   <button/>
   <button icon="3">Cut</button>
   <button icon="4">Copy</button>
   <button icon="5">Paste</button>
   <button/>
   <button icon="6">Help</button>
</toolbar>
```

In der obigen XML-Definition verwendet die Schaltfläche 1 in der Symbolleiste den Hollywood-Pinsel 1 als Bild, Schaltfläche 2 den Pinsel 2 und so weiter. Beachten Sie die leeren `<button/>`-Definitionen: Diese erzeugen Füll-Elemente, um eine zusammengehörende Gruppe von Schaltflächen, optisch vom Rest der Schaltflächen zu trennen. Symbolleisten-Schaltflächen können viele weitere Optionen wie spezielle Bilder für ausgewählte und deaktivierte Zustände, Tooltips und mehr verwenden. Siehe Abschnitt 50.1 [Toolbarbutton-Klasse], Seite 227, für Details.

Es kann nur eine Symbolleiste pro Fenster geben und sie muss immer das erste Element der Wurzelgruppe des Fensters sein. RapaGUI akzeptiert keine Symbolleisten an beliebigen Positionen im GUI-Layout. Sie befinden sich also entweder oben (horizontale Symbolleisten) oder links vom Fenster (vertikale Symbolleisten).

Beachten Sie, dass es nicht möglich ist, eigenständige Instanzen dieser Klasse mit `moai.CreateObject()` zu erzeugen. Symbolleisten müssen immer im Kontext eines Fensters erstellt werden. Wenn Sie also Symbolleisten mit `moai.CreateObject()` erzeugen wollen, müssen Sie immer ein komplettes Fenster erstellen und die Symbolleiste in diese Fensterdefinition einbetten.

AmigaOS-Benutzer beachten bitte auch, dass diese Klasse die Erweiterung `TheBar.mcc` benötigt.

49.2 Toolbar.Horiz

BEZEICHNUNG

Toolbar.Horiz – setzt die Ausrichtung der Symbolleiste

BESCHREIBUNG
Boolesch-Wert, um anzugeben, ob die Schaltflächen der Symbolleiste horizontal (`True`) oder vertikal (`False`) angeordnet werden sollen. Standardwert ist `True`.

TYP
Boolesch

ANWENDBARKEIT
I

49.3 Toolbar.ViewMode

BEZEICHNUNG
Toolbar.ViewMode – stellt den Ansichtsmodus der Symbolleiste ein

BESCHREIBUNG
Legt den Ansichtsmodus der Symbolleiste fest. Die folgenden Modi werden unterstützt:

`TextGfx`	Schaltflächen in der Symbolleiste werden als Text und Bilder angezeigt. Dies ist auch die Standardeinstellung.
`Gfx`	Schaltflächen in der Symbolleiste werden nur als Bilder angezeigt.
`Text`	Schaltflächen in der Symbolleiste werden nur als Text angezeigt.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
I

50 Toolbarbutton-Klasse (Symbolleistenschaltfläche)

50.1 Übersicht

Die Toolbarbutton-Klasse (Symbolleistenschaltflächen) ist eine Unterklasse der Toolbar-Klasse (Symbolleiste). Sie kann nicht allein verwendet werden, sondern muss immer innerhalb einer Toolbar-Klassendefinition gekapselt werden. Toolbarbutton-Klasse erstellt eine einzelne Schaltfläche für seine Toolbar-Klasse.

Bitte beachten Sie, dass das XML-Tag für diese Klasse nur `<button>` ist, nicht `<toolbarbutton>`. Siehe Abschnitt 49.1 [Toolbar-Klasse], Seite 225, für ein Beispiel.

50.2 Toolbarbutton.Disabled

BEZEICHNUNG
Toolbarbutton.Disabled – aktiviert/deaktiviert die Schaltfläche

BESCHREIBUNG
Aktiviert oder deaktiviert eine Symbolleisten-Schaltfläche.

TYP
Boolesch

ANWENDBARKEIT
ISG

50.3 Toolbarbutton.Help

BEZEICHNUNG
Toolbarbutton.Help – setzt den Hilfetext für die Symbolleisten-Schaltfläche

BESCHREIBUNG
Legen Sie den Hilfetext der Symbolleisten-Schaltfläche fest. Dieser Text wird automatisch in der Statusleiste des Fensters angezeigt, in dem sich die Symbolleiste befindet, wenn die Maus über die Schaltfläche in der Symbolleiste bewegt wird. Dies ist für den Anwender sehr komfortabel, da es die Funktion der einzelnen Schaltflächen der Symbolleiste auf elegante Weise erklärt. Siehe Abschnitt 43.1 [Statusbar-Klasse (Statusleiste)], Seite 195, für Details.

Beachten Sie, dass dies nicht dasselbe wie `Toolbarbutton.Tooltip` ist. Kurzinfos (Tooltips) werden als kleine Fenster direkt neben der Symbolleisten-Schaltfläche angezeigt, sobald Sie mit der Maus darüber fahren. Der Hilfetext wird dagegen sofort in der Statusleiste des Fensters angezeigt, sobald sich der Mauszeiger über einer Schaltfläche in der Symbolleiste befindet.

TYP
Zeichenkette

ANWENDBARKEIT
I

50.4 Toolbarbutton.Icon

BEZEICHNUNG
Toolbarbutton.Icon – setzt das Symbol für die Symbolleisten-Schaltfläche

BESCHREIBUNG
Legt das Symbol für diese Schaltfläche der Symbolleiste fest. Dies muss auf den Identifikator eines Hollywood-Pinsels gesetzt werden.

Bitte lesen Sie auch das Kapitel Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

TYP
Zahl

ANWENDBARKEIT
I

50.5 Toolbarbutton.Pressed

BEZEICHNUNG
Toolbarbutton.Pressed – benachrichtigt, wenn eine Schaltfläche gedrückt wird

BESCHREIBUNG
Dieses Attribut wird ausgelöst, wenn der Benutzer eine Schaltfläche drückt. RapaGUI überwacht automatisch dieses Attribut für alle Schaltflächen der Symbolleiste, so dass Sie nicht explizit eine Benachrichtigung mit dem Attribut MOAI.Notify anfordern müssen.

TYP
Boolesch

ANWENDBARKEIT
N

50.6 Toolbarbutton.Selected

BEZEICHNUNG
Toolbarbutton.Selected – Auswahlzustand umschalten/benachrichtigen

BESCHREIBUNG
Hiermit können Sie den Auswahlzustand einer Umschalt- sowie Radio-Schaltfläche umschalten oder den aktuellen Zustand überwachen.

RapaGUI überwacht automatisch dieses Attribut für alle Schaltflächen der Symbolleiste, so dass Sie nicht explizit eine Benachrichtigung mit dem Attribut MOAI.Notify anfordern müssen.

TYP
Boolesch

ANWENDBARKEIT
ISGN

50.7 Toolbarbutton.Tooltip

BEZEICHNUNG
Toolbarbutton.Tooltip – setzt die Kurzinfo für Symbolleisten-Schaltfläche

BESCHREIBUNG
Legt die Kurzinfo der Symbolleisten-Schaltfläche fest.

Beachten Sie, dass dies nicht dasselbe wie `Toolbarbutton.Help` ist. Kurzinfos werden als kleine Fenster direkt neben der Symbolleisten-Schaltfläche angezeigt, wenn Sie mit der Maus darüber fahren. Der Hilfetext wird dagegen sofort in der Statusleiste des Fensters angezeigt, sobald sich der Mauszeiger über einer Schaltfläche in der Symbolleiste befindet.

TYP
Zeichenkette

ANWENDBARKEIT
I

50.8 Toolbarbutton.Type

BEZEICHNUNG
Toolbarbutton.Type – legt den Typ der zu erstellenden Schaltfläche fest

BESCHREIBUNG
Dieses Attribut definiert die Art der Schaltfläche in der Symbolleiste, die Sie haben möchten. Folgende Typen sind derzeit möglich:

`Normal` Normale Schaltfläche in der Symbolleiste.

`Toggle` Schaltfläche in der Symbolleiste, die zwischen zwei Zuständen umgeschaltet werden kann.

`Radio` Schaltfläche in der Symbolleiste, die mit den benachbarten Schaltflächen eine Gruppe bildet. Es kann jeweils nur eine Schaltfläche der Gruppe ausgewählt werden.

TYP
Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT
I

51 Treeview-Klasse (Baumansicht)

51.1 Übersicht

Die Treeview-Klasse (Baumansicht) kann verwendet werden, um komplexe Datenstrukturen mithilfe einer Baumstruktur zu visualisieren. Die Daten werden als Hierarchie dargestellt, in der jedes Element eine unendliche Anzahl von Unterelementen enthalten kann. Elemente, die Unterelemente enthalten, werden Knoten genannt, während Elemente ohne Unterelemente Blätter heissen. RapaGUIs Treeview-Klasse ist sehr mächtig und unterstützt mehrspaltige Bäume, Auswahlkästchen (Checkboxen), bearbeitbare Knoten und Blätter sowie Symbole für die einzelnen Baumansicht-Elemente.

Beim Erstellen einer Baumansicht im XML-Code müssen Sie immer mindestens eine Spalte hinzufügen. Dies geschieht mit Treeviewcolumn-Klasse (Baumansichtspalten). Hier ist ein Beispiel für eine minimale Baumansicht-Definition mit nur einer einzigen Spalte:

```
<treeview>
   <column/>
</treeview>
```

Es ist auch möglich, einige Einträge zum Zeitpunkt der XML-Definition in die Baumansicht einzufügen. Dies kann mit den Tags `<node>` und `<leaf>` geschehen, die sich auf Treeviewnode-Klasse (Baumansichtknoten) bzw. Treeviewleaf-Klasse (Baumansichtblätter) beziehen. Ein Knoten der Baumansicht enthält immer nur einen einzigen Eintrag, auch bei mehrspaltigen Baumansichten. Blätter hingegen müssen jedoch so viele Einträge definieren, wie es Spalten in der Baumansicht gibt. Sie müssen also ein `<item>`-Tag pro Spalte innerhalb des `<leaf>` verwenden, um diese einzelnen Elemente zu erstellen. Hier ist eine Beispieldefinition:

```
<treeview>
   <column/>
   <node name="CPU">
      <leaf><item>Model: Motorola MPC 7447 Apollo V1.1</item></leaf>
      <leaf><item>CPU speed: 999 Mhz</item></leaf>
      <leaf><item>FSB speed: 133 Mhz</item></leaf>
      <leaf><item>Extensions: performancemonitor altivec</item></leaf>
   </node>
   <node name="Machine">
      <leaf><item>Machine name: Pegasos II</item></leaf>
      <leaf><item>Memory: 524288 KB</item></leaf>
      <leaf><item>Extensions: bus.pci bus.agp</item></leaf>
   </node>
   <node name="Expansion buses">
      <node name="PCI/AGP">
         <leaf><item>Vendor 0x11AB Device 0x6460</item></leaf>
      </node>
   </node>
   <node name="Libraries">
      <leaf><item>0x6c7d4a58: exec.library V53.34</item></leaf>
```

```
      </node>
      <node name="Devices">
         <leaf><item>0x6ff8fba4: ramdrive.device V52.6</item></leaf>
      </node>
      <node name="Tasks">
         <node name="input.device">
            <leaf><item>Stack: 0x6ff4b000 - 0x6ff5b000</item></leaf>
            <leaf><item>Signals: SigWait 0x00000000</item></leaf>
            <leaf><item>State: Task (Waiting)</item></leaf>
         </node>
      </node>
   </treeview>
```

Wie Sie sehen können, haben wir eine einspaltige Baumansicht mit dem obigen XML-Code erstellt. So müssen wir `<item>` nur einmal pro `<leaf>`-Definition verwenden. Für mehrspaltige Bäume müssten Sie so viele `<item>`-Tags verwenden, wie es Spalten in Ihrer Baumansicht gibt. Siehe Abschnitt 53.1 [Treeviewleaf-Klasse], Seite 249, für Details. Siehe Abschnitt 54.1 [Treeviewleafitem-Klasse], Seite 255, für Details.

Im Beispiel haben wir auch das Attribut `Treeviewnode.Name` verwendet, um jedem unserer Knoten einen Namen hinzuzufügen. Es gibt noch einige weitere Attribute, mit denen Sie das Aussehen Ihrer Knoten anpassen können. Siehe Abschnitt 55.1 [Treeviewnode-Klasse], Seite 257, für Details. Um beispielsweise auf ein Element (Knoten oder Blatt) in einem Treeview-Objekt zu verweisen, müssen Sie ihm eine eindeutige ID-Zeichenfolge zuweisen. Sie können dann auf dieses Element verweisen, indem Sie einfach diesen ID-Zeichenfolge verwenden. IDs werden entweder bei der Objekterstellung im XML-Code oder beim Aufruf von `Treeview.InsertNode` oder `Treeview.InsertLeaf` vergeben.

Im obigen Beispiel definieren wir keine Attribute für die Spalte der Baumansicht. Es ist jedoch möglich, das Aussehen von Baumspalten anzupassen. Sie können beispielsweise das Attribut `Treeviewcolumn.Title` verwenden, um jeder Ihrer Spalten eine Titelleiste hinzuzufügen. Darüber hinaus können Sie Ihren Spalten auch Auswahlkästchen (Checkboxen) hinzufügen und die Bearbeitung von Spaltenelementen erlauben. Siehe Abschnitt 52.1 [Treeviewcolumn-Klasse], Seite 245, für Details.

Beachten Sie, dass RapaGUI möglicherweise zwei verschiedene Arten von Widgets für diese Klasse verwendet: Wenn Sie eine Strukturansicht erstellen, die keine erweiterten Funktionen (z.B. mehrere Spalten, Kontrollkästchen, bearbeitbare Elemente, Dekorationen) verwendet, erstellt RapaGUI eventuell ein einfacheres Baumansichts-Widget für Sie, weil einige Betriebssysteme zwei verschiedene Baumansicht-basierte Widgets anbieten. RapaGUI wird das Light-Widget verwenden, falls Ihre Baumansicht keine der erweiterten Funktionen verwendet, da es schneller ist. Wenn Sie das nicht möchten, können Sie RapaGUI zwingen, immer eine vollständige Baumansicht zu verwenden, indem Sie das Attribut `Treeview.ForceMode` auf das entsprechende Tag setzen. Siehe Abschnitt 51.10 [Treeview.ForceMode], Seite 236, für Details.

51.2 Treeview.AbortEditing

BEZEICHNUNG

Treeview.AbortEditing – benachrichtigt, wenn der Benutzer die Bearbeitung abbricht (V1.1)

BESCHREIBUNG

Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, wenn der Benutzer einen Bearbeitungsvorgang an einem Element abbricht, z.B. durch Drücken der Escape-Taste oder durch Klicken außerhalb des Elementbearbeitungs-Widgets.

Beachten Sie, dass Sie `Treeviewcolumn.Editable` auf `True` setzen müssen, wenn Blätter in der jeweiligen Spalte editierbar sein sollen. Um Knoten editierbar zu machen, müssen Sie `Treeview.EditableNodes` auf `True` setzen.

Ihre Callback-Funktion wird mit den folgenden zusätzlichen Argumenten aufgerufen:

`Item:` ID des Knotens oder Blattes, den der Benutzer bearbeitet hat, als er den Vorgang abgebrochen hat.

`Column:` Spaltenindex des Elements, welches der Benutzer bearbeitet hat, als er die Bearbeitung abgebrochen hat. Für Knoten ist dies immer 0.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP

Boolesch

ANWENDBARKEIT

N

51.3 Treeview.Active

BEZEICHNUNG

Treeview.Active – setzt/ermittelt das aktive Baumelement

BESCHREIBUNG

Setzen Sie dieses Attribut, um das angegebene Baumelement zu aktivieren. An dieses Attribut muss die ID des zu aktivierenden Eintrags übergeben werden. IDs werden entweder bei der Objekterstellung im XML-Code oder beim Aufruf von `Treeview.InsertNode` oder `Treeview.InsertLeaf` vergeben.

Um die Markierung des aktiven Eintrags aufzuheben, übergeben Sie einfach den speziellen Wert `Off`. Wenn dieses Attribut abgerufen wird, gibt es die ID des aktiven Baumelements und `Off` zurück, falls es keinen aktiven Eintrag gibt.

Sie können auch eine Benachrichtigung für `Treeview.Active` einrichten, um über Änderungen der Auswahl informiert zu werden. Die ID des aktiven Baumelements wird dann in `TriggerValue` zurückgegeben.

TYP

Zeichenkette

ANWENDBARKEIT
SGN

51.4 Treeview.Alternate

BEZEICHNUNG
Treeview.Alternate – verwendet wechselnde Zeilenfarben

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dieses Attribut auf `True`, damit die Baumansicht mit wechselnden Zeilenfarben erscheint.

TYP
Boolesch

ANWENDBARKEIT
I

51.5 Treeview.Close

BEZEICHNUNG
Treeview.Close – schließt den Knoten in der Baumansicht

ÜBERSICHT
```
moai.DoMethod(id, "Close", node$)
```

BESCHREIBUNG
Diese Methode schliesst den angegebenen Knoten und alle seine Unterknoten in der Baumansicht.

Sie können die folgenden speziellen Werte für `node$` übergeben:

`Root`	Der Wurzelknoten. Die Übergabe dieses speziellen Wertes schließt alle Knoten.
`Active`	Der aktive Knoten.

EINGABEN

`id`	ID des Treeview-Objekts
`node$`	ID des zu verwendenden Baumknotens oder spezieller Wert (siehe oben)

51.6 Treeview.DoubleClick

BEZEICHNUNG
Treeview.DoubleClick – benachrichtigt, wenn Doppelklicks auf Baum-Elemente ausgeführt werden

BESCHREIBUNG
Sie können eine Benachrichtigung für dieses Attribut einrichten, um bei Doppelklicks auf Baum-Elemente zu reagieren. Das Baum-Element, auf das der Benutzer doppelt geklickt hat, wird in `TriggerValue` übergeben.

TYP
Boolesch

ANWENDBARKEIT
N

51.7 Treeview.DropFile

BEZEICHNUNG
Treeview.DropFile – benachrichtigt, wenn Datei(en) über dem Widget abgelegt werden (V1.1)

BESCHREIBUNG
Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI Ihre Callback-Funktion aus, sobald eine oder mehrere Dateien auf das Baumansicht-Widget gezogen wurden. Das Feld `TriggerValue` der Nachrichtentabelle wird auf eine Tabelle gesetzt, die eine Liste aller Dateien enthält, die auf das Widget gezogen wurden.

Außerdem enthält die Meldungstabelle die folgenden zwei zusätzlichen Felder:

`X:` Die x-Position, an der der Benutzer die Datei(en) abgelegt hat. Dies ist relativ zur linken Ecke des Widgets.

`Y:` Die y-Position, an der der Benutzer die Datei(en) abgelegt hat. Dies ist relativ zur oberen Ecke des Widgets.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

Beachten Sie, dass `Treeview.DropFile` nur dann ausgelöst wird, wenn `Treeview.DropTarget` zuerst auf `True` gesetzt wurde.

TYP
Boolesch

ANWENDBARKEIT
N

51.8 Treeview.DropTarget

BEZEICHNUNG
Treeview.DropTarget – konfiguriert, ob Dateien abgelegt werden können (V1.1)

BESCHREIBUNG
Setzen Sie dies auf `True`, wenn Dateien auf diesem Baumansicht-Widget abgelegt werden können. Sie können das Attribut `Treeview.DropFile` überwachen, um zu erfahren, wenn der Benutzer eine oder mehrere Dateien darauf ablegt.

TYP
Boolesch

ANWENDBARKEIT
ISG

51.9 Treeview.EditableNodes

BEZEICHNUNG
Treeview.EditableNodes – erlaubt die Bearbeitung von Baum-Knoten

BESCHREIBUNG
Setzen Sie dies auf `True`, um die Bearbeitung von Baum-Knoten durch den Benutzer zu ermöglichen. Der Benutzer kann dann alle Knoten im Baum bearbeiten, indem er einen langsamen Doppelklick ausführt, d.h. die linke Maustaste zweimal hintereinander langsam drückt.

Wenn Sie nur die Bearbeitung einiger Knoten im Baum erlauben möchten, müssen Sie dieses Attribut auf `True` setzen und das Attribut `Treeview.StartEditing` überwachen. `Treeview.StartEditing` wird immer dann ausgelöst, wenn der Benutzer versucht, einen Knoten zu bearbeiten und Ihr Callback-Funktion kann `False` zurückgeben, um die Bearbeitung bestimmter Elemente zu verbieten. Siehe Abschnitt 51.17 [Treeview.StartEditing], Seite 243, für Details.

Um benachrichtigt zu werden, wenn sich der Wert eines Baum-Knotens ändert, weil der Benutzer ihn bearbeitet hat, müssen Sie das Attribut `Treeview.ValueChange` überwachen.

Um die Bearbeitung eines Baum-Knotens manuell zu starten, rufen Sie die Methode `Treeviewnode.Edit` auf.

Beachten Sie, dass dieses Attribut nur für Baum-Knoten gilt. Wenn Sie möchten, dass auch Beschriftungen der Blätter bearbeitet werden können, müssen Sie das Attribut `Treeviewcolumn.Editable` auf `True` setzen.

Beachten Sie auch, dass unter AmigaOS und kompatiblen Betriebssystemen diese Fähigkeit mindestens MUI 4.0 erfordert.

TYP
Boolesch

ANWENDBARKEIT
I

51.10 Treeview.ForceMode

BEZEICHNUNG
Treeview.ForceMode – überscheibt den Standardmodus der Baumansicht

BESCHREIBUNG
RapaGUI kann abhängig von Ihren Einstellungen zwei verschiedene Widgets für die Treeview-Klasse verwenden. Wenn beispielsweise eine einspaltige Baumansicht

ohne Überschriften verwendet wird, benutzt RapaGUI möglicherweise aus Gründen der Effizienz ein anderes Widget, falls das Host-Betriebssystem ein solches Widget bereitstellt. Unter Windows beispielsweise wird RapaGUI in diesen Fällen das Treeview-Control anstelle einer voll funktionsfähigen Baumansicht verwenden. Wenn Sie das nicht möchten, setzen Sie dieses Attribut auf das gewünschte Widget und RapaGUI wird versuchen, es zu verwenden.

Die folgenden Modi werden derzeit erkannt:

`Normal` Wählt automatisch das Widget aus, das am besten passt. Dies ist die Voreinstellung.

`Treeview` Verwendet ein Treeview-Widget. Treeview-Widgets unterstützen nur einspaltige Bäume ohne editierbare Elemente und ohne Auswahlkästchen (Checkboxen), ohne Titel und ohne spezielle Dekorationen.

`Dataview` Verwenden Sie ein Dataview-Widget. Unterstützt alles, verwendet aber derzeit eine generische Implementierung unter Windows.

Beachten Sie, dass dieses Attribut unter AmigaOS und Kompatiblen keine Auswirkung hat, da RapaGUI auf diesen Plattformen immer das gleiche Widget verwendet.

TYP

Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

I

51.11 Treeview.GetEntry

BEZEICHNUNG

Treeview.GetEntry – ermittelt Informationen über den Baumeintrag

ÜBERSICHT

```
found, table = moai.DoMethod(id, "GetEntry", item, position)
```

BESCHREIBUNG

Ermittelt Informationen über einen Baumeintrag. Der Baumeintrag kann entweder als absoluter Index oder auch relativ zu einem bestimmten Knoten oder Blatt angegeben werden. So können Sie den gesamten Baum durchqueren. Siehe unten für ein Beispiel.

`item` gibt das Baumelement an, das als Referenzpunkt verwendet werden soll. Dies kann ein Knoten oder ein Blatt sein. Beachten Sie, dass `item` im Gegensatz zu allen anderen Methoden oder Attributen der Treeview-Klasse kein Zeichenketten-Identifikator, sondern ein spezieller Wert sein muss, der von dieser Methode im Feld `UID` (siehe unten) zurückgegeben oder durch Abfragen von `Treeviewleaf.UID` oder `Treeviewnode.UID` für ein Objekt abgerufen wird. Alternativ kann es einer der folgenden speziellen Werte sein:

`Root` Verwendet den Wurzelknoten.

`Active` Verwendet das aktive Element.

`position` kann entweder eine Zahl oder ein spezieller Wert sein. Die Übergabe einer Nummer wird nur unterstützt, wenn das angegebene `item` ein Knoten ist. In diesem

Fall gibt die Zahl den Index des Elementes an, über das Sie Informationen erhalten möchten, beginnend bei 0, d.h. die Übergabe von 5 würde hier Informationen über das sechste Element des in `item` übergebenen Knotens zurückgeben. Alternativ können Sie die folgenden Sonderwerte in `position` übergeben:

`Head` Gibt Informationen über das erste Element des Knotens zurück.

`Tail` Gibt Informationen über das letzte Element des Knotens zurück.

`Active` Gibt Informationen über das aktive Element zurück.

`Next` Gibt den nächsten Eintrag im Baum nach `item` zurück.

`Previous` Gibt den vorherigen Eintrag im Baum vor `item` zurück.

`Parent` Gibt Informationen über den übergeordneten Objektteil von `item` zurück.

Diese Methode gibt zwei Werte zurück: Der erste Rückgabewert `found` ist ein boolesches Flag, das angibt, ob ein Element gefunden wurde oder nicht. Wenn der erste Rückgabewert `True` ist, ist der zweite Rückgabewert `table` eine Tabelle, bei dem folgende Felder initialisiert sind:

`Items` Dies ist eine Tabelle, die die Einträge für alle Spalten dieses Baum-Eintrags enthält. Beachten Sie, dass wenn der Eintrag ein Knoten ist, diese Tabelle nur einen Eintrag enthält, da sich Knoten nicht über mehrere Spalten erstrecken können.

`Node` `True`, wenn der gefundene Eintrag ein Knoten ist, `False`, wenn es ein Blatt ist.

`ID` Zeichenkette-Objekt-Identifikator dieses Baumeintrags.

`UID` Interne ID dieses Baumeintrags. Dies ist die einzige ID, die Sie im Argument `node` dieser Methode übergeben dürfen. Die Übergabe von Standard-Zeichenketten-Objekt-Identifikatoren ist bei dieser Methode nicht erlaubt. Sie können diesen Wert für nachfolgende Aufrufe von `Treeview.GetEntry` im Argument `node` verwenden. Siehe oben für weitere Informationen und unten für ein Beispiel. Sie können UIDs auch erhalten, indem Sie das Attribut `Treeviewleaf.UID` oder `Treeviewnode.UID` ermitteln.

EINGABEN

`id` ID des Treeview-Objekts

`item` spezieller Identifikator, der von dieser Methode zurückgegeben wird oder ein spezieller Wert (siehe oben)

`position` Index des Eintrags, der ermittelt wird oder ein spezieller Wert (siehe oben)

RÜCKGABEWERTE

`found` boolesches Flag, das angibt, ob ein Eintrag gefunden wurde oder nicht

`table` Tabelle mit Informationen zum gefundenen Eintrag

BEISPIEL

```
Function p_DumpListTree(id$, node, indent)
   Local found, t = moai.DoMethod(id$, "GetEntry", node, "Head")
```

```
    While found = True
       If indent > 0
          DebugPrint(RepeatStr(" ", indent) ..
              IIf(t.Node = True, "+", ""), Unpack(t.items))
       Else
          DebugPrint(IIf(t.Node = True, "+", ""), Unpack(t.items))
       EndIf
       If t.Node = True Then p_DumpListTree(id$, t.uid, indent + 4)
       found, t = moai.DoMethod(id$, "GetEntry", t.uid, "Next")
     Wend
EndFunction

p_DumpListTree("mytreeview", "root", 0)
```

Der obige Code zeigt, wie man den kompletten Inhalt einer Baumansicht unter Beibehaltung seiner Struktur ausgibt.

51.12 Treeview.HRules

BEZEICHNUNG
Treeview.HRules – zeichnet horizontale Linien zwischen den Zeilen

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dies auf `True`, um für die Baumansicht horizontale Linien zwischen den Zeilen zu aktivieren.

TYP
Boolesch

ANWENDBARKEIT
I

51.13 Treeview.InsertLeaf

BEZEICHNUNG
Treeview.InsertLeaf – fügt ein neues Blatt in den Baum ein

ÜBERSICHT
`moai.DoMethod(id, "InsertLeaf", id$, node$, pred$, [icon1,] entry1$, ...)`

BESCHREIBUNG
Fügt ein neues Blatt an der durch `node$` und `pred$` definierten Stelle in die Baumansicht ein. Ein Blatt ist ein Baumelement, das keine Elemente hat. `id$` muss ein eindeutiger Zeichenketten-Identifikator sein, den Sie verwenden werden, um auf das neu eingefügte Baum-Blatt zu verweisen.

Die Eingabedaten für das neue Blatt müssen in den letzten Parametern übergeben werden. Wenn die Baumansicht mehrere Spalten hat, müssen Sie für alle Spalten der Baumansicht individuelle Eingabedaten übergeben. Die Eingabedaten bestehen aus einer Zeichenkette und, wenn für die Spalte das Attribut `Treeviewcolumn.Icon` gesetzt ist, einem Symbol für jede Spalte. Das Symbol muss vor der Zeichenkette übergeben werden und es muss ein Identifikator eines Hollywood-Pinsels sein, der dann als Symbol für den Eintrag verwendet werden soll. Wenn `Treeviewcolumn.Icon` nicht gesetzt ist, müssen Sie das Attribut `icon` weglassen und nur Textdaten für den Baum-Eintrag übergeben. Wenn Sie `Treeviewcolumn.Icon` auf `True` gesetzt haben und kein Symbol in dieser bestimmten Zeile und Spalte anzeigen möchten, können Sie auch den speziellen Wert -1 übergeben. In diesem Fall zeigt RapaGUI kein Symbol an, obwohl `Treeviewcolumn.Icon` auf `True` gesetzt wurde. Bitte lesen Sie auch das Kapitel über den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Wenn eine Spalte ein Auswahlkästchen (Checkbox) enthält, müssen Sie `"On"`, `"True"` oder `"1"` übergeben, um das Auswahlkästchen zu aktivieren und irgend einen anderen Text, um das Auswahlkästchen zu deaktivieren.

In `node$` müssen Sie den Knoten übergeben, in dessen Liste das neue Blatt eingefügt wird. Dies kann der Zeichenketten-Identifikator eines Knotens oder einer der folgenden Sonderwerte sein:

`Root` Der Wurzelknoten.

In `pred$` müssen Sie den Knoten oder das Blatt angeben, der zum Vorgänger des einzufügenden Blattes wird, d.h. das neue Blatt wird nach dem in `pred$` angegebenen Element eingefügt. Dies kann der Zeichenketten-Identifikator eines Knotens oder eines Blattes oder einer der folgenden speziellen Werte sein:

`Head` Als erstes Element des Knotens einfügen.

`Tail` Als letztes Element des Knotens einfügen.

`Active` Fügt das neue Blatt nach dem aktiven Element ein. Wenn es keinen aktiven Eintrag gibt, wird das Objekt als letztes untergeordnetes Element eingefügt.

Beachten Sie, dass unter AmigaOS und kompatiblen Betriebssystemen diese Fähigkeit mindestens MUI 4.0 erfordert.

EINGABEN

`id` ID des Treeview-Objekts

`id$` eindeutiger Zeichenketten-Identifikator für das neue Baum-Blatt

`node$` Zeichenketten-ID des einzufügenden Knotens oder spezieller Wert (siehe oben)

`pred$` Zeichenketten-ID des Vorgängerknotens/-Blattes oder speziellen Wertes (siehe oben)

`icon1` optional: Symbole für den Eintrag in der ersten Spalte; diese muss nur übergeben werden, wenn die Spalte das Icon-Attribut gesetzt hat.

`entry1$` Eintrag zum Einfügen in die erste Spalte

`...` mehr Einträge, wenn der Baum mehrere Spalten hat

51.14 Treeview.InsertNode

BEZEICHNUNG

Treeview.InsertNode – fügt einen neuen Knoten in den Baum ein

ÜBERSICHT

```
moai.DoMethod(id, "InsertNode", id$, node$, pred$, entry$[, icon])
```

BESCHREIBUNG

Fügt einen neuen Knoten in die Baumansicht an der Position ein, die durch `node$` und `pred$` definiert ist. Ein Knoten ist ein Baumansichtselement, das untergeordnete Elemente haben und vom Benutzer geöffnet werden kann. `id$` muss ein eindeutiger Zeichenketten-Identifikator sein, den Sie verwenden werden, um auf den neu eingefügten Baum-Knoten zu verweisen.

`entry$` gibt die gewünschte Beschriftung für den neuen Knoten an. Sie können auch die ID eines Hollywood-Pinsels im optionalen Argument `icon` übergeben, wenn neben der Knotenbeschriftung ein Symbol angezeigt werden soll. Beachten Sie, dass Knotensymbole immer unterstützt werden. Sie müssen `Treeviewcolumn.Icon` nicht auf `True` setzen, um Symbole neben den Knotenbeschriftungen anzuzeigen. `Treeviewcolumn.Icon` gilt nur für Blätter; Knoten unterstützen immer Symbole. Bitte lesen Sie auch den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

In `node$` müssen Sie den Knoten übergeben, dessen Liste zum Einfügen des neuen Knotens verwendet wird. Dies kann der Zeichenketten-Identifikator eines Knotens oder einer der folgenden speziellen Werte sein:

`Root` Der Wurzelknoten.

In `pred$` müssen Sie den Knoten oder das Blatt angeben, der zum Vorgänger des einzufügenden Knotens wird, d.h. der neue Knoten wird nach dem in `pred$` angegebenen Element eingefügt. Dies kann der Zeichenketten-Identifikator eines Knotens oder eines Blattes oder einer der folgenden speziellen Werte sein:

`Head` Als erstes Element des Knotens einfügen.

`Tail` Als letztes Element des Knotens einfügen.

`Active` Fügt das neue Blatt nach dem aktiven Element ein. Wenn es keinen aktiven Eintrag gibt, wird das Objekt als letztes untergeordnetes Element eingefügt.

Beachten Sie, dass unter AmigaOS und kompatiblen Betriebssystemen diese Fähigkeit mindestens MUI 4.0 erfordert.

EINGABEN

`id` ID des Treeview-Objekts

`id$` eindeutiger Zeichenketten-Identifikator für den neuen Baum-Knoten

`node$` Zeichenketten-ID des einzufügenden Knotens oder spezieller Wert (siehe oben)

`pred$`	Zeichenketten-ID des Vorgängerknotens/-Blattes oder speziellen Wertes (siehe oben)
`entry$`	gewünschte Knotenbeschriftung
`icon`	optional: gewünschtes Knotensymbol (standardmäßig -1, d.h. kein Symbol)

51.15 Treeview.Open

BEZEICHNUNG

Treeview.Open – öffnet einen Baumknoten

ÜBERSICHT

```
moai.DoMethod(id, "Open", node$[, all])
```

BESCHREIBUNG

Öffnet den angegebenen Baum-Knoten. Wenn das optionale Argument `all` auf `True` gesetzt ist, werden auch alle Elemente des angegebenen Knotens geöffnet.

Sie können folgende spezielle Werte für `node$` übergeben:

`Root`	Der Wurzelknoten. Da der Wurzelknoten nie sichtbar ist, werden automatisch alle Elemente des unsichtbaren Wurzelknotens geöffnet.
`Active`	Der aktive Knoten.

EINGABEN

`id`	ID des Treeview-Objekts
`node$`	Zeichenketten-ID des zu öffnenden Knotens oder spezieller Wert (siehe oben)
`all`	optional: `True`, um auch alle Knotenelemente zu öffnen (Voreingestellt: `False`)

51.16 Treeview.Remove

BEZEICHNUNG

Treeview.Remove – entfernt ein Baumelement

ÜBERSICHT

```
moai.DoMethod(id, "Remove", item$)
```

BESCHREIBUNG

Entfernt ein Element aus einer Baumansicht (entweder ein Blatt oder einen kompletten Knoten). Sie müssen die Zeichenketten-ID des zu entfernenden Eintrags in `item$` übergeben. Alternativ können Sie auch einen der folgenden Sonderwerte in `item$` übergeben:

`Active`	Entfernt das aktive Element.

EINGABEN

`id`	ID des Treeview-Objekts
`item$`	ID des zu entfernenden Baumeintrags oder spezieller Wert (siehe oben)

51.17 Treeview.StartEditing

BEZEICHNUNG

Treeview.StartEditing – benachrichtigt, wenn ein Element bearbeitet wird

BESCHREIBUNG

Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI die Callback-Funktion immer dann aus, wenn der Benutzer ein Baumansichtselement bearbeiten möchte, indem er langsam darauf doppelklickt oder wenn das Skript die Methoden `Treeviewleaf.Edit` oder `Treeviewnode.Edit` ausführt. Ihre Callback-Funktion kann dann die Bearbeitungsanfrage des Benutzers erlauben oder verbieten. Um die Anfrage zu verbieten, muss Ihre Callback-Funktion `False` zurückgeben. Um die Anfrage zuzulassen, muss sie hingegen `True` zurückgeben.

Beachten Sie, dass Sie `Treeviewcolumn.Editable` auf `True` setzen müssen, wenn Blätter in der entsprechenden Spalte bearbeitbar sein sollen. Um Knoten editierbar zu machen, müssen Sie `Treeview.EditableNodes` auf `True` setzen.

Ihre Callback-Funktion wird mit folgenden zusätzlichen Argumenten aufgerufen:

`Item:` ID des Knotens oder Blattes, den der Benutzer bearbeiten möchte.

`Column:` Spaltenindex des Eintrags, den der Benutzer bearbeiten möchte. Für Knoten ist dies immer 0.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP

Boolesch

ANWENDBARKEIT

N

51.18 Treeview.ValueChange

BEZEICHNUNG

Treeview.ValueChange – benachrichtigt, wenn sich der Wert des Baumansicht-Elements ändert

BESCHREIBUNG

Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, führt RapaGUI die Callback-Funktion aus, sobald sich der Wert eines Baumansichts-Elementes geändert hat, weil der Benutzer das Auswahlkästchen (Checkbox) aktiviert oder das Element bearbeitet hat. Beachten Sie, dass `Treeview.ValueChange` nicht ausgelöst wird, wenn der Wert des Elements mit den Methoden `Treeviewleaf.SetItem` oder `Treeviewleaf.SetState` geändert wurde. Es wird auch nicht ausgelöst, wenn `Treeviewnode.Name` zum Ändern der Beschriftung des Elements verwendet wurde.

Für Elemente in Auswahlkästchen-Spalten wird `TriggerValue` entweder auf `True` oder `False` gesetzt, was den neuen Auswahlkästchen-Status widerspiegelt. Für Elemente in Textspalten enthält `TriggerValue` den neuen Elementtext.

Außerdem wird Ihre Callback-Funktion mit folgenden zusätzlichen Argumenten aufgerufen:

`Item:` ID des Eintrags (Knoten oder Blatt), dessen Wert sich geändert hat.

`Column:` Spaltenindex des Elements, dessen Wert sich geändert hat. Für Knoten ist dies immer 0.

Siehe Abschnitt 3.6 [Benachrichtigungen der Attribute], Seite 11, für Details.

TYP
Boolesch oder Zeichenkette (abhängig vom Spaltentyp)

ANWENDBARKEIT
N

51.19 Treeview.VRules

BEZEICHNUNG
Treeview.VRules – zeichnet vertikale Linien zwischen den Spalten

BESCHREIBUNG
Setzen Sie dies auf `True`, um vertikale Linien zwischen den Spalten für die Baumansicht zu aktivieren.

TYP
Boolesch

ANWENDBARKEIT
I

52 Treeviewcolumn-Klasse (Baumansichtspalten)

52.1 Übersicht

Die Treeviewcolumn-Klasse (Baumansichtspalten) wird beim Erstellen von Baumansichten benötigt. Sie erlaubt Ihnen, verschiedene Attribute für die Spalten Ihrer Baumansichten anzugeben.

Die Treeviewcolumn-Klasse muss immer in eine `<treeview>`-Definition eingebettet sein. Ihr XML-Tag ist `<column>`. Siehe Abschnitt 51.1 [Treeview-Klasse], Seite 231, für Details.

Beachten Sie, dass Sie keine Instanzen dieser Klasse mit `moai.CreateObject()` erzeugen können. Die Spaltennummern der Baumansicht sind derzeit statisch, d.h. Sie können zur Laufzeit keine Spalten hinzufügen oder entfernen.

52.2 Treeviewcolumn.Align

BEZEICHNUNG

Treeviewcolumn.Align – setzt/ermittelt die Spaltenausrichtung

BESCHREIBUNG

Stellen Sie die Spaltenausrichtung ein oder rufen Sie sie ab. Dies kann einer der folgenden Werte sein:

`Left` Links ausgerichtet. Dies ist auch voreingestellt.

`Right` Rechts ausgerichtet.

`Center` Zentriert ausgerichtet.

Auf Nicht-AmigaOS-Systemen wird dieses Attribut nur für das Dataview-Widget unterstützt. Wenn Sie eine Baumansicht erstellen und `Treeviewcolumn.Align` auf einen anderen Wert als `Left` gesetzt ist, wird RapaGUI automatisch zum Dataview-Widget wechseln. Wenn Sie dieses Attribut nicht zum Zeitpunkt der Erstellung angeben, sondern es später mithilfe von `moai.Set()` festlegen möchten, müssen Sie explizit ein Dataview-Widget anfordern, indem Sie das Attribut `Treeview.ForceMode` setzen.

TYP

Zeichenkette (siehe oben für mögliche Werte)

ANWENDBARKEIT

ISG

52.3 Treeviewcolumn.Checkbox

BEZEICHNUNG

Treeviewcolumn.Checkbox – setzt die Spalte in den Auswahlkästchen-Modus

BESCHREIBUNG

Setzen Sie dies auf `True`, um diese Spalte als Auswahlkästchen-Spalte (Checkbox) zu markieren. Auswahlkästchen-Spalten zeigen Auswahlkästchen anstelle von Text an. Immer wenn der Text eines Eintrags in einer Auswahlkästchen-Spalte auf `"On"`, `"True"`

oder "1" gesetzt ist, wird das Auswahlkästchen ausgewählt. Alle anderen Texte führen zu einem nicht markierten Auswahlkästchen.

Sie können die Zustände der Auswahlkästchen mit der Methode `Treeviewleaf.SetState` ändern. Ebenso ist es möglich, den Status eines Auswahlkästchen über die Methode `Treeviewleaf.GetState` abzurufen.

Um benachrichtigt zu werden, wenn der Benutzer den Status eines Auswahlkästchen umschaltet, müssen Sie das Attribut `Treeview.ValueChange` überwachen.

Beachten Sie auch, dass `Treeviewcolumn.Checkbox`, `Treeviewcolumn.Editable` und `Treeviewcolumn.Icon` sich gegenseitig ausschließen. Sie können keine Auswahlkästchen-Spalten erstellen, die editierbar sind oder Symbole anzeigen.

Auswahlkästchen in der ersten Spalte der Baumansicht werden derzeit unter Windows, Mac OS und Linux nicht unterstützt.

TYP

Boolesch

ANWENDBARKEIT

I

52.4 Treeviewcolumn.Editable

BEZEICHNUNG

Treeviewcolumn.Editable – erlaubt das Bearbeiten von Spaltenblättern

BESCHREIBUNG

Setzen Sie dies auf `True`, um den Benutzern die Bearbeitung der Blätter in dieser Spalte zu erlauben. Der Benutzer wird dann in der Lage sein, alle Blätter in dieser Spalte zu bearbeiten, indem er einen langsamen Doppelklick ausführt, d.h. langsam die linke Maustaste zweimal hintereinander drückt.

Wenn Sie nur einige Blätter in der Spalte bearbeiten möchten, müssen Sie dieses Attribut auf `True` setzen und das Attribut `Treeview.StartEditing` überwachen. `Treeview.StartEditing` wird dann ausgelöst, wenn der Benutzer versucht, ein Blatt zu bearbeiten, und Ihre Callback-Funktion kann `False` zurückgeben, um die Bearbeitung bestimmter Elemente zu verbieten. Siehe Abschnitt 51.17 [Treeview.StartEditing], Seite 243, für Details.

Um benachrichtigt zu werden, wenn sich der Wert eines Baumansicht-Blattes ändert, weil der Benutzer es bearbeitet hat, müssen Sie das Attribut `Treeview.ValueChange` überwachen.

Um die Bearbeitung eines Baumansicht-Blattes manuell zu starten, rufen Sie die Methode `Treeviewleaf.Edit` auf.

Bitte beachten Sie, dass dieses Attribut nur für Baumansicht-Blätter gilt. Wenn Sie möchten, dass auch Knotenbeschriftungen editierbar sind, müssen Sie das Attribut `Treeview.EditableNodes` auf `True` setzen.

Außerdem schließen sich `Treeviewcolumn.Checkbox` und `Treeviewcolumn.Editable` gegenseitig aus. Sie können keine editierbaren Auswahlkästchen-Spalten erstellen.

Beachten Sie, dass unter AmigaOS und kompatiblen Betriebssystemen diese Fähigkeit mindestens MUI 4.0 erfordert.

TYP
Boolesch

ANWENDBARKEIT
I

52.5 Treeviewcolumn.Hide

BEZEICHNUNG
Treeviewcolumn.Hide – blendet die Baumansicht ein oder aus

BESCHREIBUNG
Mit diesem Attribut können Sie einzelne Baumspalten ein- oder ausblenden. Beachten Sie, dass die Spalten immer noch da sind, sie sind nur unsichtbar. So dürfen Sie beim Hinzufügen von Baumansicht-Blättern mit `Treeview.InsertLeaf` versteckte Spalten nicht vergessen.

Auf Nicht-AmigaOS-Systemen wird dieses Attribut nur für das Dataview-Widget unterstützt. Wenn Sie eine Baumansicht erstellen und `Treeviewcolumn.Hide` auf `True` gesetzt ist, benutzt RapaGUI automatisch die Dataview-Widget. Wenn Sie dieses Attribut bei der Erstellung nicht angeben, es aber später mit `moai.Set()` setzen wollen, müssen Sie explizit ein Dataview-Widget anfordern, indem Sie das Attribut `Treeview.ForceMode` setzen.

TYP
Boolesch

ANWENDBARKEIT
ISG

52.6 Treeviewcolumn.Icon

BEZEICHNUNG
Treeviewcolumn.Icon – aktiviert Blattsymbole für diese Spalte

BESCHREIBUNG
Setzen Sie dies auf `True`, wenn Baumansicht-Blätter in dieser Spalte Symbole verwenden. In diesem Fall müssen Sie der Methode `Treeview.InsertLeaf` ein Hollywood-Pinsel als Symbol übergeben.

Beachten Sie, dass dieses Attribut nur für Baumansicht-Blätter gilt. Baumansicht-Knoten können Symbole verwenden, ohne ein bestimmtes Attribut zu setzen. Symbole für Baumansicht-Knoten sind immer verfügbar, aber für Blätter müssen Sie dieses Attribut zuerst auf `True` setzen.

Bitte lesen Sie auch den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

Beachten Sie auch, dass unter AmigaOS und kompatiblen Betriebssystemen diese Fähigkeit mindestens MUI 4.0 erfordert.

TYP
Boolesch

ANWENDBARKEIT
I

52.7 Treeviewcolumn.Title

BEZEICHNUNG
Treeviewcolumn.Title – setzt/ermittelt den Titel der Spalte

BESCHREIBUNG
Setzt oder ermittelt den Titel der Spalte. Der Titel wird immer oben in der Baumansicht angezeigt und verschwindet nicht, wenn die Baumansicht gescrollt wird.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

52.8 Treeviewcolumn.Width

BEZEICHNUNG
Treeviewcolumn.Width – setzt/ermittelt die Spaltenbreite

BESCHREIBUNG
Setzt oder ermittelt die Spaltenbreite in Pixel. Die Voreinstellung ist -1, was bedeutet, dass die Spalte so groß wie ihr größter Eintrag sein sollte.

TYP
Zahl

ANWENDBARKEIT
ISG

53 Treeviewleaf-Klasse (Baumansichtblätter)

53.1 Übersicht

Die Treeviewleaf-Klasse (Baumansichtblätter) wird beim Erstellen von Baumansichten benötigt. Sie erlaubt Ihnen, die einzelnen Blätter der Baumansicht zu definieren und verschiedene Attribute für sie zu setzen. Jedes Blatt Ihrer Baumansicht muss einen Eintrag für jede Spalte Ihrer Baumansicht enthalten. So müssen Ihre `<leaf>`-Definitionen so viele `<item>`-Definitionen enthalten, wie es Spalten in Ihrer Baumansicht gibt. Siehe Abschnitt 54.1 [Treeviewleafitem-Klasse], Seite 255, für Details.

Die Treeviewleaf-Klasse muss immer in eine `<treeview>`-Definition eingebettet werden. Siehe Abschnitt 51.1 [Treeview-Klasse (Baumansicht)], Seite 231, für Details.

Beachten Sie, dass Sie keine Instanzen dieser Klasse mit `moai.CreateObject()` erzeugen können. Stattdessen müssen Sie `Treeview.InsertLeaf` verwenden, um Blätter zur Laufzeit zu erzeugen.

53.2 Treeviewleaf.Edit

BEZEICHNUNG

Treeviewleaf.Edit – startet die Bearbeitung von Blattelementen durch das Programm

ÜBERSICHT

```
moai.DoMethod(id, "Edit", column)
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um die Bearbeitung von Blattelementen durch das Programm zu starten. Normalerweise wird die Bearbeitung eines Eintrags vom Benutzer durch einen langsamen Doppelklick auf einen Eintrag gestartet. Diese Methode stellt eine Alternative zu diesem Benutzermechanismus dar.

Dies funktioniert nur, wenn `Treeviewcolumn.Editable` für die jeweilige Baumansicht-Spalte auf `True` gesetzt wurde.

Wenn der Benutzer die Bearbeitung beendet hat, wird das Attribut `Treeview.ValueChange` ausgelöst.

Beachten Sie, dass wenn Sie eine Überwachung für das Attribut `Treeview.StartEditing` installiert haben, diese Callback-Funktion zuerst um Erlaubnis fragt, bevor die Bearbeitung tatsächlich gestartet wird.

Um zu erfahren, wenn Bearbeitungsvorgänge abgebrochen werden, können Sie das Attribut `Treeview.AbortEditing` überwachen.

Beachten Sie, dass unter AmigaOS und kompatiblen Betriebssystemen diese Fähigkeit mindestens MUI 4.0 erfordert.

EINGABEN

`id`	ID des Blattes
`column`	Spaltenindex des zu bearbeitenden Blattelements

53.3 Treeviewleaf.GetDisabled

BEZEICHNUNG

Treeviewleaf.GetDisabled – ermittelt den deaktivierten Status des Auswahlkästchen

ÜBERSICHT

```
state = moai.DoMethod(id, "GetDisabled", column)
```

BESCHREIBUNG

Gibt den deaktivierten Zustand des Auswahlkästchen (Checkbox) in der angegebenen Spalte zurück. Dies ist entweder `True` für deaktiviert oder `False` für aktiviert.

EINGABEN

`id`	ID des Blattes
`column`	Spaltenindex des Auswahlkästchen

RÜCKGABEWERTE

`state`	`True` wenn das Auswahlkästchen deaktiviert ist, sonst `False`

53.4 Treeviewleaf.GetIcon

BEZEICHNUNG

Treeviewleaf.GetIcon – ermittelt die ID vom Symbol des Blattelements

ÜBERSICHT

```
br = moai.DoMethod(id, "GetIcon", column)
```

BESCHREIBUNG

Gibt die Hollywood-Pinsel-ID des in der angegebenen Spalte verwendeten Blattsymbols zurück. Wenn das angegebene Element kein Symbol hat, wird -1 zurückgegeben.

EINGABEN

`id`	ID des Blattes
`column`	Spaltenindex des Elements

RÜCKGABEWERTE

`br`	ID eines Hollywood-Pinsels oder -1, wenn es kein Symbol gibt.

53.5 Treeviewleaf.GetItem

BEZEICHNUNG

Treeviewleaf.GetItem – ermittelt die Beschriftung des Blattelements

ÜBERSICHT

```
t$ = moai.DoMethod(id, "GetItem", column)
```

BESCHREIBUNG

Gibt den Beschriftungstext des Elements in der angegebenen Blattspalte zurück.

EINGABEN

`id` ID des Blattes

`column` Spaltenindex des Elements

RÜCKGABEWERTE

`t$` Beschriftung des angegebenen Elements

53.6 Treeviewleaf.GetState

BEZEICHNUNG

Treeviewleaf.GetState – ermittelt den Umschaltstatus des Auswahlkästchen

ÜBERSICHT

```
state = moai.DoMethod(id, "GetState", column)
```

BESCHREIBUNG

Gibt den Umschaltstatus des Auswahlkästchen (Checkbox) in der angegebenen Spalte zurück. Dies ist entweder `True`, wenn das Auswahlkästchen aktiviert oder `False`, wenn sie deaktiviert ist.

EINGABEN

`id` ID des Blattes

`column` Spaltenindex des Auswahlkästchen

RÜCKGABEWERTE

`state` `True`, wenn des Auswahlkästchen markiert ist, ansonsten `False`

53.7 Treeviewleaf.SetDisabled

BEZEICHNUNG

Treeviewleaf.SetDisabled – setzt den Deaktivierungsstatus des Auswahlkästchen

ÜBERSICHT

```
moai.DoMethod(id, "SetDisabled", column, state)
```

BESCHREIBUNG

Legt den deaktivierten Status des Auswahlkästchen (Checkbox) in der angegebenen Spalte fest. Übergeben Sie `True`, um das Auswahlkästchen zu deaktivieren oder `False`, um sie zu aktivieren.

EINGABEN

`id` ID des Blattes

`column` Spaltenindex des Auswahlkästchen

`state` `True`, wenn des Auswahlkästchen deaktiviert werden soll, andernfalls `False`.

53.8 Treeviewleaf.SetIcon

BEZEICHNUNG

Treeviewleaf.SetIcon – setzt das Symbol für das Blattelement

ÜBERSICHT

```
moai.DoMethod(id, "SetIcon", column, br)
```

BESCHREIBUNG

Fügt dem angegebenen Blatteintrag ein Symbol hinzu. Sie müssen die ID eines Hollywood-Pinsels übergeben, der als Symbol in `br` verwendet werden soll. Um ein Symbol aus einem Blatteintrag zu entfernen, übergeben Sie -1 in `br`.

Bitte lesen Sie auch den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

EINGABEN

`id`	ID des Blattes
`column`	Spaltenindex des Blattelements
`br`	ID eines Hollywood-Pinsels, der als Symbol verwendet werden soll oder -1, um das Element-Symbol zu entfernen.

53.9 Treeviewleaf.SetItem

BEZEICHNUNG

Treeviewleaf.SetItem – setzt die Beschriftung des Blattelements

ÜBERSICHT

```
moai.DoMethod(id, "SetItem", column, l$)
```

BESCHREIBUNG

Diese Methode übeschreibt die Beschriftung mit dem in `l$` angegebenen Text im `id` angegebenen Blattelement.

EINGABEN

`id`	ID des Blattes
`column`	Spaltenindex des Blattelements
`l$`	gewünschte neue Beschriftung für das Blattelement

53.10 Treeviewleaf.SetState

BEZEICHNUNG

Treeviewleaf.SetState – setzt den Umschaltstatus des Auswahlkästchen

ÜBERSICHT

```
moai.DoMethod(id, "SetState", column, state)
```

BESCHREIBUNG
Setzt den Umschaltstatus des Auswahlkästchen (Checkbox) in der angegebenen Spalte. Übergeben Sie `True`, um das Auswahlkästchen anzukreuzen/Häckchen zu setzen oder `False`, um das Kreuz/Häckchen zu entfernen.

EINGABEN

`id`	ID des Blattes
`column`	Spaltenindex des Blattelements
`state`	`True`, wenn das Auswahlkästchen angekreuzt werden soll, ansonsten `False`

53.11 Treeviewleaf.UID

BEZEICHNUNG
Treeviewleaf.UID – ermittelt die UID des Blattes

BESCHREIBUNG
Ermittelt die UID des angegebenen Baumblattes ab. Diese UID wird benötigt, wenn Sie die Methode `Treeview.GetEntry` ausführen.

TYP
MOAI-Objekt

ANWENDBARKEIT
G

54 Treeviewleafitem-Klasse (Baumansichtblätterelement)

54.1 Übersicht

Die Treeviewleafitem-Klasse (Baumansichtblätterelement) wird beim Erstellen von Baumansichten (Treeview) benötigt. Es erlaubt Ihnen, die einzelnen Elemente für jede Spalte eines Baumansicht-Blattes zu definieren. Sie müssen so viele `<item>`-Definitionen verwenden, wie es Spalten in Ihrer Baumansicht gibt. Da diese Klasse eine Unterklasse von der Treeviewleaf-Klasse (Baumansichtblätter) ist, müssen Ihre `<item>`-Definitionen in den Tag `<leaf>` eingebettet werden. Blatt-Elemente können mit einer Beschriftung und optional mit einem Symbol versehen werden. Siehe Abschnitt 51.1 [Treeview-Klasse], Seite 231, für ein Beispiel.

Beachten Sie, dass Sie keine Instanzen dieser Klasse mit `moai.CreateObject()` erzeugen können. Stattdessen müssen Sie `Treeview.InsertLeaf` verwenden, um zur Laufzeit Blatt-Elemente zu erstellen.

54.2 Treeviewleafitem.Icon

BEZEICHNUNG

Treeviewleafitem.Icon – setzt das Symbol

BESCHREIBUNG

Setzen Sie dieses Attribut auf eine Hollywood-ID eines Pinsels, um ein Symbol für das Baumansicht-Blatt zu erstellen. Bitte lesen Sie auch das Kapitel Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

TYP

Zahl

ANWENDBARKEIT

I

55 Treeviewnode-Klasse (Baumansichtknoten)

55.1 Übersicht

Die Treeviewnode-Klasse (Baumansichtknoten) wird beim Erstellen von Baumansichten benötigt. Es erlaubt Ihnen, die einzelnen Knoten der Baumansicht zu definieren und verschiedene Attribute für sie zu setzen. Beachten Sie, dass Knoten im Gegensatz zu Baumansicht-Blättern auch bei mehrspaltigen Baumansichten nur einen einzigen Eintrag enthalten.

Die Treeviewnode-Klasse (Baumansichtknoten) muss immer in einer `<treeview>`-Definition eingebettet werden. Siehe Abschnitt 51.1 [Treeview-Klasse], Seite 231, für Details.

Beachten Sie, dass Sie keine Instanzen dieser Klasse mit `moai.CreateObject()` erzeugen können. Stattdessen müssen Sie `Treeview.InsertNode` verwenden, um Knoten während der Laufzeit zu erstellen.

55.2 Treeviewnode.Edit

BEZEICHNUNG

Treeviewnode.Edit – fordert den Benutzer auf, eine Knotenbeschriftung zu bearbeiten

ÜBERSICHT

```
moai.DoMethod(id, "Edit")
```

BESCHREIBUNG

Diese Methode kann verwendet werden, um die Bearbeitung von Knotenbeschriftungen vom Programm aus zu starten. Normalerweise wird die Bearbeitung der Knotenbeschriftung vom Benutzer durch einen langsamen Doppelklick auf ein Element gestartet. Diese Methode stellt eine Alternative zu diesem Benutzermechanismus dar.

Dies funktioniert nur, wenn `Treeview.EditableNodes` in der Treeview-Klasse auf `True` gesetzt wurde.

Wenn der Benutzer die Bearbeitung beendet hat, wird das Attribut `Treeview.ValueChange` ausgelöst.

Bitte beachten Sie, dass, wenn Sie eine Überwachung auf dem Attribut `Treeview.StartEditing` installiert haben, diese Callback-Funktion zuerst um Erlaubnis gefragt wird, bevor die Bearbeitung tatsächlich gestartet wird.

Wenn Sie erfahren möchten, welche Bearbeitungsvorgänge abgebrochen werden, können Sie das Attribut `Treeview.AbortEditing` überwachen.

Beachten Sie auch, dass diese Methode unter AmigaOS und kompatiblen Betriebssystemen nur unter MUI 4.0 oder höher verfügbar ist.

EINGABEN

`id` ID des Knotens

55.3 Treeviewnode.Icon

BEZEICHNUNG
Treeviewnode.Icon – setzt das Knotensymbol

BESCHREIBUNG
Setzen Sie dieses Attribut auf die ID des Hollywood-Pinsels, um ein Symbol für den Baumknoten zu erstellen. Um das Symbol aus einem Baumknoten zu entfernen, setzen Sie dieses Attribut auf -1.

Bitte lesen Sie auch das Kapitel über den Bilder-Cache von RapaGUI, um mehr über die Unterstützung von Symbolen in RapaGUI zu erfahren. Siehe Abschnitt 3.16 [Bilder-Cache], Seite 26, für Details.

TYP
Zahl

ANWENDBARKEIT
ISG

55.4 Treeviewnode.Name

BEZEICHNUNG
Treeviewnode.Name – setzt/ermittelt den Knotennamen

BESCHREIBUNG
Setzt oder ruft die Zeichenkette ab, die als Knotenname verwendet wird.

Beim Erstellen eines neuen Knotens ist dieses Attribut obligatorisch und muss immer festgelegt werden.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

55.5 Treeviewnode.UID

BEZEICHNUNG
Treeviewnode.UID – ermittelt die Knoten-UID

BESCHREIBUNG
Liefert die UID des angegebenen Baumknotens. Diese UID wird beim Ausführen der Methode `Treeview.GetEntry` benötigt.

TYP
MOAI-Objekt

ANWENDBARKEIT
G

56 VLine-Klasse (VertikalLinie)

56.1 Übersicht

Die VLine-Klasse (VertikalLinie) erzeugt einfach eine vertikale Trennlinie, die als separate Gruppe von Widgets verwendet werden kann. Es gibt auch eine HLine-Klasse (HorizontalLinie), welche horizontale Trennlinien erzeugt. Siehe Abschnitt 18.1 [HLine-Klasse], Seite 99, für Details.

Die VLine-Klasse hat keine Attribute.

57 VSpace-Klasse (VertikalAbstand)

57.1 Übersicht

Die VSpace-Klasse (VertikalAbstand) erzeugt einfach Objekte mit einer festen Pixelgröße. Dies wird typischerweise zur Feinabstimmung des GUI-Layouts verwendet. Um VSpace-Objekte zu erstellen, die frei skalierbar sind, können Sie stattdessen Rectangle-Klasse (Rechteck) verwenden. Siehe Abschnitt 38.1 [Rectangle-Klasse], Seite 177, für Details.

57.2 VSpace.Height

BEZEICHNUNG
VSpace.Height – setzt den vertikalen Abstand

BESCHREIBUNG
Legt den gewünschten vertikalen Abstand für dieses Objekt in Pixeln fest.

TYP
Zahl

ANWENDBARKEIT
I

58 Window-Klasse (Fenster)

58.1 Übersicht

Die Window-Klasse erstellt Top-Level-Fenster, die mit Gruppen von Widgets gefüllt werden können. Diese Gruppen sind Elemente der Top-Level-Fenster. Als solche werden Gruppen automatisch weitergeleitet, wenn sich die Größe ihres übergeordneten Objekts ändert. So können Sie flexible GUI-Layouts erstellen, die sich automatisch an den verfügbaren Bildschirmbereich anpassen. Zusätzlich können Sie auch Menüleisten und Symbolleisten an ein Fenster zuweisen.

Fenster sind Elemente der Application-Klasse (Anwendung). Wenn Sie Fensterobjekte dynamisch erstellen, müssen Sie diese zunächst dem Applications-Objekt hinzufügen, indem Sie `Application.AddWindow` aufrufen.

Wenn Sie Dialoge öffnen müssen, können Sie dies tun, indem Sie Objekte der Dialog-Klasse erstellen. Dialoge sind spezielle Top-Level-Fenster, die den Rest vom Programm blockieren, bis sie geschlossen werden. Dialoge werden typischerweise verwendet, wenn Benutzeraktionen erforderlich sind, um eine Aufgabe fortzusetzen oder um anzuzeigen, dass das Programm gerade beschäftigt ist. Ein Dialog könnte dann z.B. einen Fortschrittsbalken anzeigen. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details.

Hier ist ein minimales Beispiel für die Erstellung eines Fensters in XML:

```
<window title="Hello World!">
   <vgroup>
       <button>Hello World!</button>
   </vgroup>
</window>
```

Beachten Sie, dass das Wurzelelement eines Fensters immer ein einzelnes Gruppenobjekt sein muss, d.h. eine Instanz von Group-Klasse (Gruppen). Siehe Abschnitt 17.1 [Group-Klasse], Seite 87, für Details. In unserem Beispiel verwenden wir eine `<vgroup>` als Wurzelelement. Es ist nicht erlaubt, mehrere Elemente auf der Wurzelebene des Fensters zu haben, darum dürfen Sie nur ein einziges Gruppenobjekt als Wurzelelement verwenden.

Fenster, die im XML-Code definiert oder von `moai.CreateApp()` erstellt wurden, werden automatisch geöffnet, es sei denn, Sie setzen explizit `Window.Open` auf `False`, damit sie geschlossen bleiben. Fenster, die mit `moai.CreateObject()` erstellt wurden, werden dagegen nicht automatisch geöffnet. Sie müssen sie zunächst der Fensterliste des Programms hinzufügen, indem Sie die Methode `Application.AddWindow` aufrufen und dann `Window.Open` auf `True` setzen.

58.2 Window.Accelerator

BEZEICHNUNG

Window.Accelerator – setzt die Tastaturkürzel-Tabelle des Fensters

BESCHREIBUNG

Setzt die Tastaturkürzel-Tabelle (Accelerator-Tabelle) für dieses Fenster. Siehe Abschnitt 7.1 [Accelerator-Klasse], Seite 53, für Details.

TYP
MOAI-Objekt

ANWENDBARKEIT
I

58.3 Window.Activate

BEZEICHNUNG
Window.Activate – ändert den Aktivierungszustand des Fensters

BESCHREIBUNG
Setzen Sie dieses Attribut auf `True`, um das Fenster zu aktivieren.

Sie können auch eine Benachrichtigung für dieses Attribut einrichten, um zu erfahren, wann immer das Fenster aktiviert oder deaktiviert wird.

TYP
Boolesch

ANWENDBARKEIT
ISGN

58.4 Window.ActiveObject

BEZEICHNUNG
Window.ActiveObject – setzt/ermittelt das aktive Widget

BESCHREIBUNG
Setzt oder ermittelt das aktive Widget. Das aktive Widget ist das mit dem Tastaturfokus. Widgets können auch manuell mit der TAB-Taste aktiviert werden.

TYP
MOAI-Objekt

ANWENDBARKEIT
SG

58.5 Window.CloseGadget

BEZEICHNUNG
Window.CloseGadget – konfiguriert das Fensterschließsymbol

BESCHREIBUNG
Setzen Sie diese Option auf `False`, um ein Fenster ohne Fensterschließsymbol zu erstellen.

TYP
Boolesch

ANWENDBARKEIT
I

58.6 Window.CloseRequest

BEZEICHNUNG
Window.CloseRequest – benachrichtigt, wenn das Fensterschließsymbol gedrückt wurde

BESCHREIBUNG
Wenn Sie eine Benachrichtigung für dieses Attribut einrichten, wird das Fenster nicht automatisch geschlossen, wenn der Benutzer das Fensterschließsymbol drückt. Stattdessen wird Ihre Callback-Funktion aufgerufen und Sie müssen das Fenster manuell schließen, indem Sie `Window.Open` auf `False` setzen. Oder Sie können das Fenster offen lassen, wenn es Dinge gibt, die zuerst erledigt werden müssen.

Ein typischer Anwendungsfall für dieses Attribut ist die Frage, ob der Benutzer das aktuelle Projekt speichern möchte, bevor er das Programm beendet.

Wenn es keine Benachrichtigung für dieses Attribut gibt, werden beim Drücken des Fensterschließsymbols das Fenster automatisch geschlossen.

TYP
Boolesch

ANWENDBARKEIT
N

58.7 Window.DefaultObject

BEZEICHNUNG
Window.DefaultObject – setzt/ermittelt das Standard-Widget des Fensters

BESCHREIBUNG
Das Standard-Widget ist dasjenige, das beim Öffnen eines Fensters aktiv ist. Typischerweise kann dies ein "OK" oder eine andere Bestätigungsschaltfläche sein, so dass der Benutzer einfach RETURN drücken kann, um das Fenster zu schließen.

TYP
MOAI-Objekt

ANWENDBARKEIT
ISG

58.8 Window.DragBar

BEZEICHNUNG
Window.DragBar – konfiguriert die Zieh-Leiste des Fensters

BESCHREIBUNG
Setzen Sie dies auf `False`, wenn Sie nicht möchten, dass Ihr Fenster gezogen werden kann.

TYP
Boolesch

ANWENDBARKEIT
I

58.9 Window.Height

BEZEICHNUNG
Window.Height – stellt die Fensterhöhe ein oder ermittelt sie

BESCHREIBUNG
Hiermit können Sie die Fensterhöhe in Pixel ermitteln oder einstellen. Dies kann entweder ein absoluter Pixelwert oder einer der folgenden Sonderwerte sein:

`Default` Berechnet die Höhe aus den Standardgrößen aller Widgets.

`Screen:<1..100>`
Stellt die Höhe als Prozentsatz der Gesamthöhe des Hostbildschirms ein.

Standard für dieses Tag ist `Default`.

TYP
Zahl oder vordefinierter Sonderwert

ANWENDBARKEIT
IG

58.10 Window.HideFromTaskbar

BEZEICHNUNG
Window.HideFromTaskbar – konfiguriert die Sichtbarkeit der Taskleiste

PLATTFORMEN
Windows, Linux

BESCHREIBUNG
Setzen Sie dies auf `True`, wenn Ihr Fenster nicht in der Taskleiste erscheinen soll.

TYP
Boolesch

ANWENDBARKEIT
I

58.11 Window.Left

BEZEICHNUNG
Window.Left – setzt/ermittelt den linken Fensterrand

BESCHREIBUNG
Ermitteln oder stellen Sie die horizontale Position des Fensters ein. Sie können hier einen absoluten Pixelwert übergeben oder eines der folgenden Sonderwerte verwenden:

`Centered` Fenster im sichtbaren Bereich des Bildschirms zentrieren.

`Moused` Fenster unter dem Mauszeiger öffnen.

Standard für dieses Tag ist `Centered`.

TYP
Zahl oder Sonderwert

ANWENDBARKEIT
IG

58.12 Window.Margin

BEZEICHNUNG
Window.Margin – setzt den Fensterrand

BESCHREIBUNG
Legt den Fensterrand in Pixeln fest. Der Fensterrand ist als der Abstand zwischen dem Rahmen des Fensters und der Wurzelgruppe definiert.

TYP
Zahl

ANWENDBARKEIT
I

58.13 Window.MaximizeGadget

BEZEICHNUNG
Window.MaximizeGadget – konfiguriert das Fenstermaximiersymbol

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dies auf `False`, wenn Sie kein Maximiersymbol für Ihr Fenster haben möchten. Standardmäßig `False`, wenn `Window.Toolwindow` gesetzt ist, ansonsten `True`.

TYP
Boolesch

ANWENDBARKEIT
I

58.14 Window.Menubar

BEZEICHNUNG
Window.Menubar – stellt die Menüleiste des Fensters ein

BESCHREIBUNG
Stellen Sie die Menüleiste für dieses Fenster ein. Siehe Abschnitt 28.1 [Menubar-Klasse], Seite 143, für Details.

TYP
MOAI-Objekt

ANWENDBARKEIT
I

58.15 Window.MinimizeGadget

BEZEICHNUNG
Window.MinimizeGadget – konfiguriert das Fensterminimiersymbol

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dies auf `False`, wenn Sie kein Minimiersymbol für Ihr Fenster haben wollen. Standardmäßig `False`, wenn `Window.Toolwindow` gesetzt ist, ansonsten `True`.

TYP
Boolesch

ANWENDBARKEIT
I

58.16 Window.Open

BEZEICHNUNG
Window.Open – öffnet/schließt ein Fenster

BESCHREIBUNG
Setzen Sie dieses Attribut, um Ihre Fenster zu öffnen oder zu schließen.

Beachten Sie, dass Fenster, die im XML-Code definiert oder von `moai.CreateApp()` erstellt wurden, werden automatisch geöffnet, es sei denn, Sie setzen explizit `Window.Open` auf `False`, damit sie geschlossen bleiben. Fenster, die mit `moai.CreateObject()` erstellt wurden, werden dagegen nicht automatisch geöffnet. Sie müssen sie zunächst der Fensterliste des Programms hinzufügen, indem Sie die Methode `Application.AddWindow` aufrufen und dann `Window.Open` auf `True` setzen.

TYP
Boolesch

ANWENDBARKEIT
ISG

58.17 Window.Parent

BEZEICHNUNG
Window.Parent – definiert die übergeordneten Objekte des Fensters

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dies auf den Identifikator eines MOAI-Objekts, das das übergeordnete Element dieses Fensters werden soll. Wenn ein Fenster ein übergeordnetes Objektteil hat, wird das Fenster automatisch auch ausgeblendet, wenn der übergeordnete Objektteil ausgeblendet ist, und es wird auch über dem übergeordneten Objektteil zentriert sein, falls keine explizite Position angegeben wurde.

Beachten Sie, dass Sie die Dialog-Klasse verwenden sollten, wenn Sie möchten, dass das übergeordnete Fenster blockiert wird, während das Element geöffnet ist. Siehe Abschnitt 16.1 [Dialog-Klasse], Seite 83, für Details.

TYP
MOAI-Objekt

ANWENDBARKEIT
I

58.18 Window.PubScreen

BEZEICHNUNG
Window.PubScreen – setzt/ermittelt/definiert den Bildschirm des Fensters

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Mit diesem Attribut können Sie den Namen eines öffentlichen Bildschirms angeben, auf dem das Fenster geöffnet werden soll. Bitte verwenden Sie dieses Attribut nur, wenn es wirklich notwendig ist, da normalerweise der Benutzer Ihrer Anwendung derjenige sein sollte, der entscheidet, auf welchem Bildschirm er Ihre Anwendung mit den MUI-Einstellungen ausführen möchte.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

58.19 Window.ScreenTitle

BEZEICHNUNG
Window.ScreenTitle – setzt den Bildschirmtitel des Fensters

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Legen Sie den Text fest, der in der Titelleiste des Bildschirms angezeigt wird, wenn das Fenster aktiv ist.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

58.20 Window.SizeGadget

BEZEICHNUNG
Window.SizeGadget – definiert das Fenstergrößensymbol

BESCHREIBUNG
Setzen Sie dies auf `False`, wenn Sie kein Größensymbol für Ihr Fenster haben möchten.

TYP
Boolesch

ANWENDBARKEIT
I

58.21 Window.StayOnTop

BEZEICHNUNG
Window.StayOnTop – öffnet ein Fenster, das zu vorderst bleibt

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dies auf `True`, damit Ihr Fenster oben bei der Z-Reihenfolge der Fenster bleibt.

TYP
Boolesch

ANWENDBARKEIT
I

58.22 Window.Title

BEZEICHNUNG
Window.Title – setzt/ermittelt den Titel des Fensters

BESCHREIBUNG
Setzt oder ermittelt den Titel des Fensters.

Wenn Sie dieses Attribut nicht festlegen, verwendet RapaGUI den Titel, der in Hollywoods Präprozessor-Anweisung `@APPTITLE` angegeben wurde. Wenn `@APPTITLE` ebenfalls nicht angegeben wurde, wird der Standardtitel `"RapaGUI"` verwendet.

TYP
Zeichenkette

ANWENDBARKEIT
ISG

58.23 Window.Toolwindow

BEZEICHNUNG
Window.Toolwindow – kennzeichnet das Fenster als Werkzeugfenster

PLATTFORMEN
Windows, Linux, Mac OS

BESCHREIBUNG
Setzen Sie dies auf `True`, um Ihr Fenster im Werkzeugfensterdesign zu öffnen. Werkzeugfenster haben in der Regel einen kleineren Rahmen als normale Top-Level-Fenster.

TYP
Boolesch

ANWENDBARKEIT
I

58.24 Window.Top

BEZEICHNUNG
Window.Top – setzt/ermittelt die Fensteroberkante

BESCHREIBUNG
Stellen Sie die vertikale Position des Fensters ein. Sie können hier einen absoluten Pixelwert übergeben oder eines der folgenden Sonderwerte verwenden:

`Centered` Fenster im sichtbaren Bereich des Bildschirms zentrieren.

`Moused` Fenster unter dem Mauszeiger öffnen.

`Delta:<p>`
Öffnen Sie das Fenster `<p>` Pixel unterhalb der Titelleiste des Bildschirms.

Der Standardwert für dieses Tag ist `Centered`.

TYP
Zahl oder vordefinierte Sonderwerte

ANWENDBARKEIT
IG

58.25 Window.UseBottomBorderScroller

BEZEICHNUNG
Window.UseBottomBorderScroller – aktiviert die Bildlaufleiste am unteren Rand

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Setzen Sie dies auf `True`, um RapaGUI mitzuteilen, dass sich in Ihrem Fensterlayout ein Element befindet, das eine Bildlaufleiste im unteren Fensterrand platziert, z.B. mit `Scrollbar.UseWinBorder`.

TYP
Boolesch

ANWENDBARKEIT
I

58.26 Window.UseLeftBorderScroller

BEZEICHNUNG
Window.UseLeftBorderScroller – aktiviert die Bildlaufleiste am linken Rand

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Setzen Sie dies auf `True`, um RapaGUI mitzuteilen, dass sich in Ihrem Fensterlayout ein Element befindet, das eine Bildlaufleiste im linken Fensterrand platziert, z.B. mit `Scrollbar.UseWinBorder`.

TYP
Boolesch

ANWENDBARKEIT
I

58.27 Window.UseRightBorderScroller

BEZEICHNUNG
Window.UseRightBorderScroller – aktiviert die Bildlaufleiste am rechten Rand

PLATTFORMEN
Nur AmigaOS und kompatible Betriebssysteme

BESCHREIBUNG
Setzen Sie dies auf `True`, um RapaGUI mitzuteilen, dass es in Ihrem Fensterlayout ein Element gibt, das eine Bildlaufleiste in den rechten Fensterrand setzt, z.B. mit `Scrollbar.UseWinBorder`.

TYP
Boolesch

ANWENDBARKEIT
I

58.28 Window.Width

BEZEICHNUNG
Window.Width – setzt/ermittelt die Fensterbreite

BESCHREIBUNG
Legen Sie die Fensterbreite in Pixel fest. Dies kann entweder ein absoluter Pixelwert oder einer der folgenden Sonderwerte sein:

`Default` Berechnet die Breite aus den Standardgrößen aller Widgets.

`Screen:<1..100>`
Legt die Breite als Prozentsatz der Gesamtbreite des Hostbildschirms fest.

Standard für dieses Tag ist `Default`.

TYP
Nummer oder vordefinierte Sonderwerte

ANWENDBARKEIT
IG

Anhang A Lizenzen

A.1 wxWidgets license

wxWindows Library Licence, Version 3.1

Copyright (c) 1998-2005 Julian Smart, Robert Roebling et al

Everyone is permitted to copy and distribute verbatim copies of this licence document, but changing it is not allowed.

WXWINDOWS LIBRARY LICENCE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public Licence as published by the Free Software Foundation; either version 2 of the Licence, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public Licence for more details.

You should have received a copy of the GNU Library General Public Licence along with this software, usually in a file named COPYING.LIB. If not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

EXCEPTION NOTICE

1. As a special exception, the copyright holders of this library give permission for additional uses of the text contained in this release of the library as licenced under the wxWindows Library Licence, applying either version 3.1 of the Licence, or (at your option) any later version of the Licence as published by the copyright holders of version 3.1 of the Licence document.

2. The exception is that you may use, copy, link, modify and distribute under your own terms, binary object code versions of works based on the Library.

3. If you copy code from files distributed under the terms of the GNU General Public Licence or the GNU Library General Public Licence into a copy of this library, as this licence permits, the exception does not apply to the code that you add in this way. To avoid misleading anyone as to the status of such modified files, you must delete this exception notice from such code and/or adjust the licensing conditions notice accordingly.

4. If you write modifications of your own for this library, it is your choice whether to permit this exception to apply to your modifications. If you do not wish that, you must delete the exception notice from such code and/or adjust the licensing conditions notice accordingly.

A.2 MUI license

This application uses MUI - MagicUserInterface (c) Copyright 1992-97 by Stefan Stuntz. MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to pageview directly, feel free to send DM 30.- or US$ 20.- to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

Support and online registration is available at http://www.sasg.com/

A.3 Expat license

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

A.4 LGPL license

GNU Lesser General Public License Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library

does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which

must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Index

L

M

P

R

S

T

V

W