

Das MUI-Xtra für Macromedia Director

Was ist das MUI-Xtra?

MUI steht für "Macromedia User Interface". Es ist ein Xtra, mit dem man als Entwickler systemeigene Dialogboxen entwerfen kann, im besten Falle sogar cross-platform. Das XTRA unterstützt allerdings nur Windows95/MacOs, nicht aber Windows 3.x. Über NT kann ich keine Aussagen machen. Der Lingo-Befehl, der dem MUI am nächsten kommt, ist *alert*, mit dem man eine Dialogbox aufrufen kann, in der ein Text dargestellt wird. Das MUI-Xtra aber bietet mehr: es können alle erdenklichen Dialogboxen programmiert werden, von einer einfachen JA/NEIN-Auswahl bis hin zur Dateiauswahl mit Checkboxes und Radiobuttons. Dabei werden die Systemsprache und das farbliche Layout des zugrundeliegenden Betriebssystems mit eingebunden. Selbst die Dialogboxen in Director selber wurden mit dem MUI-Xtra erstellt.

Was muß man haben?

Das MUI Xtra wird seit Director 6 ausgeliefert und wird bei der Installation standardmäßig in den Unterordner "Xtras" mitinstalliert. Es befindet sich wenig Dokumentation in der Hilfe bzw. auf der Installations-CD zu diesem Xtra, und auch bei Macromedia muß man kräftig wühlen, um dazu Informationen zu bekommen. Die beste Einsteigerquelle, welche ich auch hier im weiteren Verlauf benutzen werde, ist ein englischsprachige Einführung und eine Beispielfilm, welches gezippt [hier](#) auf der Website bzw. im Original bei [updateStage](#) zu finden sind. Dort gibt es auch einen Link vom Entwickler des Xtras, John Ware, unter dem eine [Programmierenreferenz im Word-Format](#) zu dem MUI zu finden ist. Ein absolutes Muß, wenn man später selber loslegen möchte. Bei weiterem Interesse an dieser Einführung werde ich auch versuchen, diese Referenz ins Deutsche zu übersetzen. *Dank an Gretchen MCDowell von www.updateStage.com, die uns die Benutzung des Primers und des Beispielfilms gestattete.*

Wie lege ich los?

Als erstes sollte man sich das oben erwähnte Beispielfilm [hier](#) oder von <ftp://ftp.shore.net/members/update/muihowto.zip> herunterladen, um einen Eindruck zu bekommen, was das MUI kann. Nach dem Starten des Movies kann man auf der Bühne einen Knopf drücken, und es erscheint eine kleine Dialogbox mit einem PopUp-Menü, aus dem man aus drei Einträgen einen auswählen kann. Sehr spannend... :-)

Wenn man das Movie sich genauer anschaut, so sieht man, das die ganze dazugehörige Magie in einem CastMember Script der Schaltfläche abgelegt ist:

```
on mouseUp
  global newCustomDialog,listOfItemsToDisplay
  if the machineType = 256 then
    set modal = TRUE
  else
    set modal = FALSE
  end if
  set newCustomDialog = new(xtra "MUI")
```

Die einführenden Befehle sind soweit klar: bei Mausclick auf die Fläche werden zwei globale Variablen deklariert, dann wird geprüft, auf welcher Maschine das

Programm läuft (PC/Mac) und anschließend eine Boolesche Variable namens *modal* auf TRUE gesetzt oder nicht. Anschließend wird eine neue Instanz des Xtras initialisiert.

Was nun folgt, ist der eigentliche Teil für das Layout des Dialoges:

```
set myWinPropList = GetWindowPropList(newCustomDialog)
```

Zunächst wird mit Hilfe der neuen MUI-Instanz eine Property-Liste (Eigenschaftsliste) mit Standardwerten erzeugt. Das geschieht mit Hilfe des GetWindowPropList-Befehls, der auf die neue Instanz des Extras angewendet wird. Danach steht folgendes in myWinPropList:

```
[#type: #normal, #name: "window", #callback: "nothing", #Mode: #Data, #xPosition: 100, #yPosition: 120, #width: 200, #height: 210, #modal: 1, #toolTips: 0, #closeBox: 1, #canZoom: 0]
```

Wie man sehen kann, ist dies eine Eigenschaftsliste mit vielen Knöpfen und Reglern. Diese Eigenschaftsliste kann wiederum Eigenschaftslisten beinhalten. Aber dazu später mehr... Im Moment soll nur interessant sein, daß man mit dieser Liste die Erscheinung des Dialogfensters beeinflusst.

```
-- set up properties for the window
setProp(myWinPropList, #name, "George")
setProp(myWinPropList, #modal, modal)
setProp(myWinPropList, #callback, "tellWhatHappened")
```

In diesem Abschnitt werden manche vordefinierte Werte der soeben abgeleiteten Eigenschaftsliste verändert. Dies wäre die Eigenschaft **#name** (nun "George"), **#modal** (bekommt den Wert der oben definierten Variable modal in Abhängigkeit vom Rechnersystem zugewiesen), sowie der Wert **#callback**, dem ein sogenannter Callbackhandler zugewiesen wird, der sich um die Verarbeitung des Ergebnisses der Dialogbox kümmert. Wenn also die Dialogbox geschlossen worden ist, wird anschließend dieser Handler aufgerufen. Man beachte die Übergabe des Handler-Namens in Anführungszeichen. Dieser Handler übernimmt die Auswertung der Benutzerauswahl in der Dialogbox.

Damit wäre soeben ein Fenster definiert worden, welches in seiner Caption-Bar "George" als Titel trägt, auf dem Mac keine Möglichkeit zum Schließen außer den Buttons hätte und nach erfolgter Darstellung den Handler "tellWhatHappened" aufruft.

In diesem Abschnitt wurde somit die Erscheinung des Fensters definiert.

Die Inhalte des Fensters

Es folgt im Listing die Definition des PopUp-Menüs sowie seiner Inhalte:

```
-- popUp list item
set popUp = GetItemPropList(newCustomDialog)
```

Wiederum wird zunächst eine Standardliste aus dem Xtra erzeugt, aber diesmal mit dem GetItemPropList-Befehl im Gegensatz zum ersten GetWindowPropList. Der Befehl GetItemPropList weist darauf hin, daß man hier Inhalte der DialogBox definiert. Es steht nun folgende Eigenschaftsliste in popUp:

```
[#value: 0, #type: #checkBox, #Attributes: [], #title: "title", #tip: "tip", #locH: 20, #locV: 24, #width: 200, #height: 210, #enabled: 1]
```

Auch hier sind jede Menge Einstellmöglichkeiten für die Inhalte des Fensters gegeben. Der Einfachheit halber werden aber nur bestimmte Werte der Liste

verändert:

```
setProp(popUp, #type, #popUpList)
setProp(popUp, #title, "Pick One")
```

```
setProp(popUp, #attributes, [#valueList: ["first", "sec", "third"]])
```

Die Eigenschaft **#type** wird auf `#popUpList` gesetzt, was darauf hinweist, daß man eine PopUp-Liste haben möchte. Es folgen **#title**, mit dem Titel des Items gesetzt wird (erscheint sonst nicht weiter auf dem Bildschirm), sowie **#attributes**, dem eine Eigenschaftsliste übergeben wird, die wiederum eine lineare Liste enthält [`#valueList: ["first", "sec", "third"]`], in der die Auswahlinhalte stehen. Wenn man hier einen Eintrag verändert, so erscheint dies im PopUp-Menü.

Damit wäre die PopUp-Liste definiert.

Es fehlt noch der "Abbrechen"-Knopf:

```
-- cancel button item
set cancelButton = GetItemPropList(newCustomDialog)
setProp(cancelButton, #type, #pushButton)
setProp(cancelButton, #title, "Cancel")
```

Wieder das gleiche Prozedere: ableiten einer Standardliste, verändern einiger wichtige Inhalte. Der **#type** wird auf `#pushButton` gesetzt (Knopf), auf dem Knopf soll "Cancel" stehen. Man kann den **#title** für den Knopf hier zwar auch verändern, jedoch funktioniert danach der Knopf nicht mehr richtig, da der weiter oben angegebene Callbackhandler den Titel des Knopfes prüft. Wenn dieser nicht "Cancel" ist, so wird die Dialogbox nicht mehr geschlossen...

Damit wäre auch der Abbruch-Knopf definiert, und der Inhalt des Fensters somit komplett.

Trotzdem ist hier noch nicht Schluß. Damit der neue Dialog dem Xtra übergeben werden kann, *muß es für jedes Fenster zwei zwingende Inhalte geben*: einen **#windowbegin** und einen **#windowend**. Die beiden Eigenschaften umgeben die Inhalte des Fensters wie eine Klammer. Selbst wenn es keinerlei Knöpfe oder PopUp-Menüs geben sollte, müssen diese beiden Items definiert werden. Sie werden wie andere Items per setzen der **#type**-Eigenschaft definiert:

```
-- window start and window end required items
set winStart = GetItemPropList(newCustomDialog)
setProp(winStart, #type, #windowBegin)

set winEnd = GetItemPropList(newCustomDialog)
setProp(winEnd, #type, #windowEnd)
```

Damit hätten wir als Inhalte des Fensters:

1. einen WindowBegin-Begrenzer,
2. eine PopUp-Liste,
3. einen Abbruch-Button sowie
4. einen WindowEnd-Begrenzer.

Jedes dieser Elemente besteht aus einer Eigenschaftsliste, in der bestimmte Werte verändert wurden. Bei jedem dieser Elemente wurde auf jeden Fall der **#type** verändert, je nach seiner Funktion (z.B. `#windowBegin` bzw. `#populist`). Zusätzlich konnten in Abhängigkeit des **#type** weitere Werte mittels des **#attribute** als Liste übergeben werden (z.B. bei der PopUp-Liste, bei der damit die Werte der Liste an sich übergeben wurden).

Zu diesen Elementenlisten kommt noch die `MyWinPropList`-Eigenschaftsliste für das äußere Aussehen des Fensters an sich hinzu. Mit ihr wird nur der äußere Rahmen des Fensters definiert, also sein Titel und die grafische Erscheinung.

Initialisieren und Darstellen des Dialogs

Der neu erstellte Dialog kann nun erstellt und angezeigt werden:

```
-- construct the itemList from the items you just built
set listOfItemsToDisplay = list(winStart,popUp,cancelButton,winEnd)
```

Dazu werden alle im Fenster darzustellenden Inhalte (also der Anfangsbegrenzer, die popUpListe, der Pushbutton sowie der Endbegrenzer) in eine große Liste zusammengefaßt. Die Dialogbox wird initialisiert, indem zwei Listen dem "initialize"-Befehl übergeben werden:

1. Die Fensterliste für das äußere Aussehen und
2. die große Liste mit allen im Fenster darzustellenden Inhalten:

```
initialize(newCustomDialog,[
#windowPropList:myWinPropList,#windowItemList:listofItemsToDisplay])
```

Nach der Initialisierung kann der Dialog angezeigt werden:

```
run(newCustomDialog)

if modal = FALSE then
  windowOperation(newCustomDialog,#show)
end if

end
```

Der letzte Abschnitt prüft die Variable modal, welche am Anfang in Abhängigkeit von der Plattform gesetzt wurde: auf dem PC ist sie TRUE, auf einem Mac FALSE. Der Mac benötigt diesen weiteren Befehl, um den neuen Dialog überhaupt anzuzeigen (windowOperation (newCustomDialog, #show)).

Zusammenfassung

Eine Dialogbox besteht für das MUI-Xtra aus...

- dem Fenster und
- den im Fenster darzustellenden Inhalten, den "Items".

Diesen beiden Objekten wird Rechnung getragen, indem man beim Aufruf einer eigenen Dialogbox zwei Listen übergibt: eine für das Aussehen des Fensters, und eine mit allen Inhalten des Fensters.

Dies ist das allgemeine Prozedere für die Definition einer Systemdialogbox mit dem MUI.

Da es beliebig viele Inhalte in einem Fenster geben darf, wird die zweite Liste in den meisten Fällen eine verschachtelte Liste sein, in der sich weitere Unterlisten verstecken können; zumindest so viele, wie es Elemente in dem Fenster geben soll plus zwei weiteren als Anfangs- und Endbegrenzer für die Fensterinhalte.

Eine gesamte Dialogbox wird also definiert als...

- Fensterliste
- Itemliste
 - ItemListeAnfangsbegrenzer
 1. Erstes selbstdefiniertes Item
 2. Zweites selbstdefiniertes Item
 3. Drittes selbstdefiniertes Item
 4. etc...
 - ItemListeEndbegrenzer

Wenn man diesen Aufbau verstanden hat, dann ist man gewappnet für Schlimmeres...

Besteht Interesse an der Fortführung dieses Themas? Weitere Themen, die nun folgen würden, wären die Beschreibung der einzelnen Listen, der benutzten Xtra-Befehle, Listtypen und programmiertechnischen Feinheiten für verschiedenen Items (PopUp-Liste, Schieberegler, Radiobuttons, Checkboxes, etc.)

Feedback bitte auf die Direct@r_Liste oder an [Christophe Leske](#).

Danke.

Das Listing

```

on mouseUp

    global newCustomDialog, listOfItemsToDisplay

    if the machineType = 256 then
        set modal = TRUE
    else
        set modal = FALSE
    end if

    set newCustomDialog = new(xtra "MUI")
    set myWinPropList = GetWindowPropList(newCustomDialog)

    -- set up properties for the window
    setProp(myWinPropList, #name, "George")
    setProp(myWinPropList, #modal, modal)
    setProp(myWinPropList, #callBack, "tellWhatHappened")

    -- popUp list item
    set popUp = GetItemPropList(newCustomDialog)
    setProp(popUp, #type, #popUpList)
    setProp(popUp, #title, "Pick One")

setProp(popUp, #attributes, [#valueList: ["first", "sec", "third"]])

    -- cancel button item
    set cancelButton = GetItemPropList(newCustomDialog)
    setProp(cancelButton, #type, #pushButton)
    setProp(cancelButton, #title, "Cancel")

    -- window start and window end required items
    set winStart = GetItemPropList(newCustomDialog)
    setProp(winStart, #type, #windowBegin)
    set winEnd = GetItemPropList(newCustomDialog)
    setProp(winEnd, #type, #windowEnd)

    -- construct the itemList from the items you just built

```

```
set listOfItemsToDisplay = list(winStart,popUp,cancelButton,winEnd)

initialize(newCustomDialog,[
#windowPropList:myWinPropList,#windowItemList:listofItemsToDisplay])

    run(newCustomDialog)

    if modal = FALSE then
        windowOperation(newCustomDialog,#show)
    end if
end
```