

Nutzung des Motif Windowmanagers

Das Motif-Konzept

Motif ist eine graphische Nutzerschnittstelle, die in sich Entwicklungswerkzeuge, eine Beschreibungssprache für graphische Darstellungen, Windowmanagement und Richtlinien für einen einheitlichen Stil vereinigt.

Toolkit

Das Motif-Toolkit stellt eine Bibliothek für die Generierung von Anwendungen zur Verfügung. Diese vermitteln dem Nutzer einen 3D-Eindruck, der einen leichteren Zugang zum Geschehen auf dem Bildschirm ermöglichen soll. Das Toolkit ist konform zu dem des Presentation-Managers (OS/2), was einen erleichterten Austausch zwischen PC- und Workstation-Umgebungen sichert.

Presentation

Motif besitzt eine Nutzer-Schnittstellen-Sprache (UIL - User Interface Language), die von Anwendungsentwicklern für die Beschreibung von visuellen Eigenschaften und Anfangszuständen der darzustellenden Komponenten genutzt werden kann.

Style

Um eine einheitliche Oberfläche zu garantieren, wurden für die Gestaltung von Anwendungen und einzelnen Komponenten in einem Style-Guide Design-Standards festgelegt.

Window

Der Motif Windowmanager (MWM) bietet Funktionen zur Bewegung, Größenänderung und zum Arrangieren von Windows, zum Reduzieren und Wiederherstellen von Icons, zum Darstellen von Menüs und Ausführen von Shell-Anweisungen und vielem mehr. Im weiteren wird nur noch auf die Nutzung und Anpassung des MWM eingegangen; darum seien hier noch einige markante Merkmale genannt:

- MWM unterstützt direkte Manipulationen graphischer Objekte durch das Objekt-Aktion-Modell (z.B. durch einen Desktop Manager). Ein Nutzer kontrolliert die Operation einer Anwendung durch Auswahl eines Windows, Menüs, Icons oder eines graphischen Objektes und wählt dann eine Aktion, die auf diese Objekte angewendet werden soll.
- MWM erlaubt ausschließliche Tastaturarbeit
- Der Nutzer kann über Konfigurationsdateien die MWM-Umgebung seinen Bedürfnissen anpassen.

Arbeiten mit dem MWM

Eigentlich ist es nicht nötig, über die Handhabung des MWM viele Worte zu verlieren, da er im Aussehen, in der Anordnung der Bedienelemente, in seiner Funktionalität und im Inhalt der Window-Menüs weitestgehend mit Windows 3.x identisch ist. Dennoch seien an dieser Stelle seine Besonderheiten hervorgehoben und einige (für den folgenden Abschnitt) notwendige Begriffsbildungen eingefügt.

Der Kontext-Begriff

Nachfolgend wird auf Maus- und Tastaturaktionen eingegangen, die sich immer auf die Position des Mauszeigers auf dem Bildschirm beziehen. Wird die linke Maustaste gedrückt, wenn der Mauszeiger auf einem Icon steht, dann erscheint ein anderes Menü, als würde sich der Mauszeiger im Root-Window befinden. Die Aktion ist wesentlich davon abhängig, auf welches Objekt der Mauszeiger auf dem Bildschirm weist. Eine Zusammenstellung der Kontexte ist in der Abbildung des Motif-Rahmens ersichtlich.

Die Maus

Im X-Window-System (und auch beim MWM) wird reger Gebrauch von 3-Tasten-Mäusen gemacht. Nutzer von 2-Tasten-Mäusen können durch gleichzeitiges Drücken der rechten und linken Taste die mittlere Maustaste emulieren. Jede Maustaste kann vier Zustände einnehmen:

Down	- Taste gedrückt gehalten
Up	- Taste loslassen
Click	- drücken und loslassen
Click2	- Doppel-Click

Mit Hilfe der Tastatur können noch weitere Mausaktionen definiert werden - folgende Modifikatoren sind möglich:

Ctrl	- rechte/ linke Ctrl (Strg)-Taste
Shift	- Umschalt-Taste
Meta	- Alt-Taste (deutsch: Alt_L, d.h. linke Alt-Taste)
Lock	- Caps_Lock
Mod1-5	- oft unbenutzt oder umdefiniert z.B. Mod2 = Num_Lock oder Mod3 = Alt_R (deutsch: AltGr)

Den Inhalt der Modifikatorenliste erhält man mit 'xmodmap -pm', die Tastaturdefinitionen mit 'xmodmap -pk' und die Mauszuweisungen mit 'xmodmap -pp'.

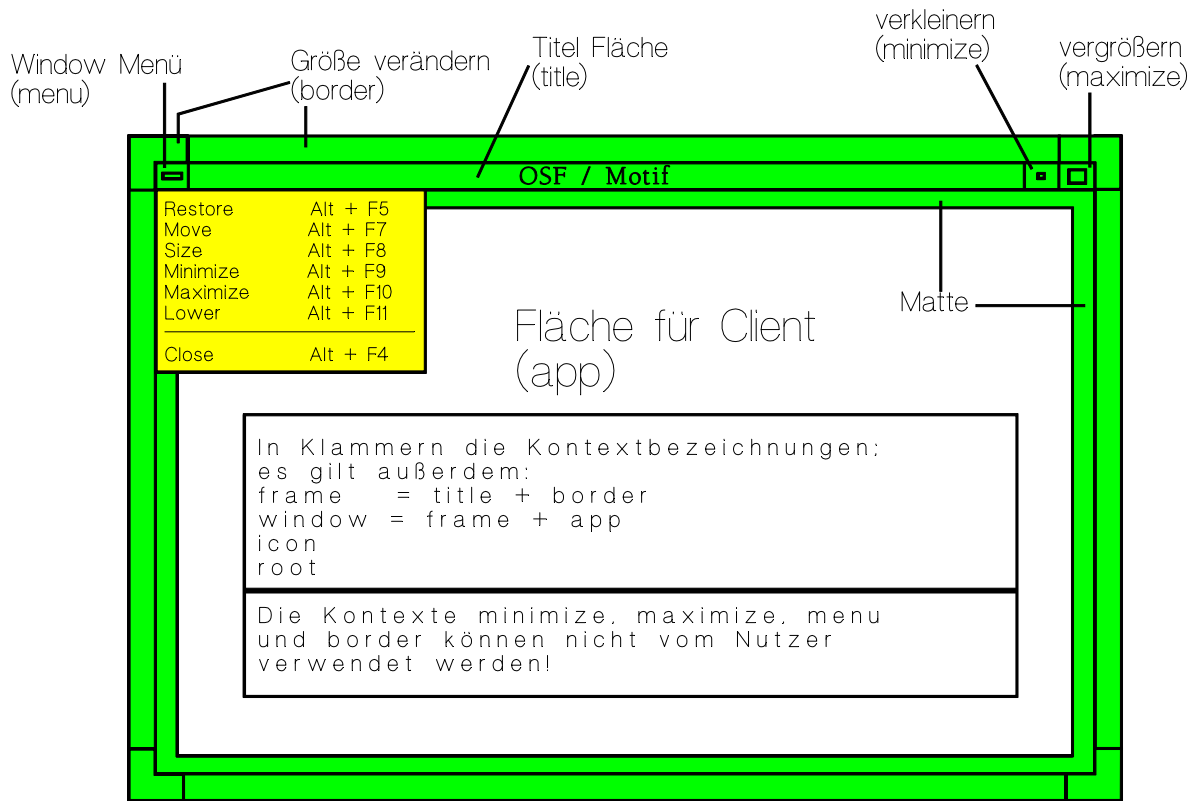


Abbildung: Der typische Motif-Rahmen einer X-Anwendung; Aussehen und Funktionalität können durch die Ressourcenparameter 'clientDecoration' und 'clientFunctions' eingeschränkt werden.

Die folgenden Maus-Definitionen liegen fest und können nicht mehr verändert werden:

Maus Aktion	Kontext	Funktion
<Btn1Down>	menu	Window-Menü
<Btn1Click2>	menu	Window killen
<Btn1Click>	minimize	ikonifizieren
<Btn1Click>	maximize	Window maximal
<Btn1Down>	title	Window bewegen
<Btn1Down>	window/icon	Window/Icon aktiv
<Btn1Down>	border	Größe ändern
<Btn1Click2>	icon	wieder herstellen
<Btn1Down>	icon	Icon bewegen

Die Tastatur

Alle Maus- und Menü-Aktionen lassen sich auch über die Tastatur auslösen. Alle Motif-Tasten-Festlegungen lassen sich durch den Nutzer in der Datei \$HOME/.mwmrc (worin sie auch definiert werden) verändern mit einer einzigen Ausnahme:

Meta Ctrl Shift <Key>exclam

(auf deutscher Tastatur: Alt Strg Umschalt <!>)

dient dem Schalten zwischen Motif-Default-Einstellungen und den Nutzer-Definitionen. Zweimaliges Anwenden dieser Tastenkombination kann zum Restart des MWM genutzt werden.

Anpassen der MWM-Umgebung

Pfade: Environment- und Sprachabhängigkeit

Es gibt für die Initialisierung einer Motif-Sitzung eine ganze Reihe von Konfigurationsdateien, deren Lage im Dateiensystem abhängig von gesetzten Environmentvariablen und von der Sprachunterstützung des Betriebssystems ist. Berücksichtigt das System nationalsprachliche Schriftsätze und Tastaturen, so wird die Environmentvariable \$LANG auf einen die Sprache kennzeichnenden String gesetzt (z.B. RS6000: LANG=Gr_GR). Entsprechend verlängern sich einige Pfade. Außerdem können noch zusätzliche Environmentvariable veränderte Orte für die Konfigurationsdateien festlegen.

Ressourcendateien

Die systemweiten MWM-Ressourcen-Informationen sind entweder enthalten in der Datei:

- /usr/lib/X11/\$LANG/app-defaults/Mwm oder
- /usr/lib/X11/app-defaults/Mwm oder man findet den Dateinamen in der Variablen:
- \$XFILESEARCHPATH

Nutzerspezifischen MWM-Ressourcen in der Datei:

- \$HOME/\$LANG/Mwm oder - \$HOME/Mwm oder Inhalt von - \$XUSERFILESEARCHPATH oder Inhalt von - \$XAPPLRESDIR/\$LANG/Mwm

Nutzerspezifische Ressourcen-Informationen über MWM und andere Clients in der Datei:

- \$HOME/.Xdefaults

Nutzer- und rechnerspezifische Ressourcen-Werte von MWM und anderen Clients in der Datei:

- \$HOME/.Xdefaults-hostname oder Inhalt von - \$XENVIRONMENT

Menü-, Maus- und Tastaturdefinitionen

Systemweite Informationen über MWM-Menüs und Maus- und Tastaturfestlegungen in:

- /usr/lib/X11/\$LANG/system.mwmrc oder - /usr/lib/X11/system.mwmrc

Nutzerspezifische Angaben zu Menüs und Maus- und Tastaturdefinitionen in:

- \$HOME/\$LANG/.mwmrc oder - \$HOME/.mwmrc

Menüs, Maus- und Tastendefinitionen

Falls im Home-Directory keine '.mwmrc' vorhanden sein sollte, dann hole man sich die 'system.mwmrc':

```
cp /usr/lib/X11/system.mwmrc $HOME/.mwmrc
(evtl. $LANG zwischenschieben - s.o.).
```

Veränderungen in der .mwmrc (wie auch in den Ressourcendateien .Xdefaults und Mwm) werden erst mit einem erneuten Start des MWM wirksam. Also nach dem Editieren muß im Root-Menü (Maus-Zeiger ins Root-Window fahren und linke Maustaste drücken) der Punkt "Restart..." ausgewählt werden.

MWM-Funktionen

Über Menüs bzw. Maus- oder Tastenaktionen können nur genau definierte MWM-Funktionen aktiviert werden. Nicht jede Funktion läßt sich in jedem Kontext nutzen. Zum Beispiel ist die Funktion 'f.resize' für die Größenveränderung von Windows zuständig - aber sie funktioniert nicht in den Kontexten root und icon, da diese sich nicht in ihrer Größe ändern lassen. Außerdem steuern Ressourcenparameter ganz wesentlich die Wirksamkeit der Funktionen. Existiert z.B. folgende Zeile in \$HOME/.Xdefaults:

```
Mwm*clientFunctions: -resize
```

dann ist es trotz vereinbarter 'f.resize'-Aktion in der Datei \$HOME/.mwmrc nicht mehr möglich, irgendein Window in seiner Größe zu verändern.

Für eine vollständige Funktionsdiskussion sei auf den MWM-Programmer's Guide oder den MWM-Manual-Eintrag verwiesen. An dieser Stelle werden

nur drei Funktionen kurz vorgestellt, mit denen man aber auch schon ganz schön weit kommt.

f.menu - Aufruf eines in .mwmrc vereinbarten Menüs; im Menü können weitere f.menu-Aufrufe enthalten sein (Untermenüs)

f.title - fügt eine Titelzeile im Menü ein

f.exec - nutzt /bin/sh zur Ausführung eines Kommandos

Struktur der Datei .mwmrc

In der Datei .mwmrc können mehrere Menüs vereinbart werden, die mit dem Schlüsselwort 'Menu' und einem kennzeichnenden Namen eingeleitet werden. In ihnen werden MWM-Funktionen oder Shell-Kommandos (f.exec) bestimmten Menüpunkten zugeordnet. Diese Menüs lassen sich dann durch Tasten- oder Mausaktionen aktivieren. Welche das sind, wird ebenfalls in Tabellen festgelegt. Es ist zwar möglich, mehrere Tasten- (bzw. Maus-) Vereinbarungs-Tabellen zu führen, aber nur eine von ihnen kann jeweils aktiv sein. Voreingestellt sind für die Tastaturfestlegung die Tabelle mit dem Namen 'DefaultKeyBindings' und für die Maus-aktionen 'DefaultButtonBindings' (ab Motif 1.1). Diesen Einstellungen können über die Ressourcen-parameter 'keyBindings' und 'buttonBindings' andere Tabellennamen zugewiesen werden - danach sollte der (erneute) Start des MWM erfolgen.

Menüs in .mwmrc

Syntax: Menu menu_name

```
{
  item1 [mnem] [acc] function [arg]
  item2 [mnem] [acc] function [arg]
  .
  item# [mnem] [acc] function [arg]
}
```

menu_name der Name, unter dem die Funktion f.menu das Menü erreicht

item der Name oder das Bitmap eines Menüpunktes; für Name gilt, daß Leerzeichen in doppelte Hochkommas eingeschlossen werden; Bitmaps werden mit einem vor den Pfadnamen vorangestelltem '@' angegeben

mnem hervorgehobener Buchstabe im Menünamen für Tastenabkürzung

acc Tastenkombination, mit der die Funktion auch ohne Menüaufruf erreicht werden kann

function [arg] MWM-Funktion evtl. mit Argumenten

Nutzereigene Mausaktionen

Syntax: Buttons *ButtonBindingSetName*

```
{
  button cont [/cont] function [arg]
  button cont [/cont] function [arg]
  .
  button cont [/cont] function [arg]
}
```

ButtonBindingSetName

Name, unter dem der Windowmanager die Tabelle mit den Mausaktionen sucht; voreingestellt ist 'DefaultButtonBindings'; ein anderer Name kann über den Ressourcenparameter 'buttonBindings' festgelegt werden

button legt die Maustaste einschließlich Zustand und Modifikator fest

cont Kontext

function MWM-Funktion

Nutzereigene Tastenaktionen

Syntax: Keys *KeyBindingSetName*

```
{
  key cont [/cont] function [arg]
  key cont [/cont] function [arg]
  .
  key cont [/cont] function [arg]
}
```

KeyBindingSetName

Name, unter dem MWM die Tabelle mit den Tastenaktionen sucht; voreingestellt ist 'DefaultKeyBindings'; ein anderer Name kann über den Ressourcenwert 'keyBindings' festgelegt werden

key Taste einschließlich Modifikator

cont Kontext

function MWM-Funktion

Beispiel .mwmrc

```
Menu mymenu
{
  "My Menu"          f.title
  "Clock"            _C  Alt<Key>c  f.exec "xclock &"
  "Window"           _W  Alt<Key>w  f.exec "xterm &"
  "Puzzle"           _P  Alt<Key>p  f.exec "puzzle &"
}
```

Keys DefaultKeyBindings

```
{
  <Key>F11          root      f.menu mymenu
  Ctrl<Key>F11     root      f.exec "xman &"
}
```

Buttons DefaultButtonBindings

```
{
  <Btn2Down>       root      f.menu mymenu
  <Btn3Click>      root      f.menu mymenu
  Ctrl<Btn3Click> root      f.exec "xman &"
}
```

Im Beispiel ist ein Menü 'mymenu' definiert, das im Kopf den Titel 'My Menu' führt. Es folgen drei Menüpunkte. In der ersten Zeile wird ein Menüpunkt 'Clock' definiert. Das darauffolgende '_C' zeigt an, daß in der Zeichenkette 'Clock' das 'C' unterstrichen werden und als Akkürzungstaste im Menü fungieren soll. Mit der Tastenkombination 'Alt <c>' kann die Uhr auch ohne aktives Menü und vor allem völlig kontextunabhängig (d.h. überall) aufgerufen werden. Nach der MWM-Funktion 'f.exec' folgt in doppelten Hochkommas der eigentliche xclock-Aufruf, wobei das abschließende '&' dafür sorgt, daß xclock als Hintergrundprozeß läuft.

Die Aktivierung dieses Menüs per Taste zeigt dann die Tabelle 'Keys DefaultKeyBindings'. Diese bereits bestehende Tabelle wurde in unserem Falle um zwei Einträge erweitert. Mit Drücken der Taste <F11> und dem Mauszeiger im Root-Window (Kontext='root') läßt sich das Menü auf den Bildschirm projizieren. Betätigt man dann noch die Taste <c>, dann erscheint die Uhr. Außerdem zeigt die letzte Zeile dieser Tabelle, wie mit Strg <F11> und der Maus im Root-Window das X-Manual direkt (ohne Menü-Eintrag) gestartet werden kann.

Die Einbindung des Menüs in die Maus-Definitionstabelle 'Buttons DefaultButtonBindings' zeigt zwei unterschiedliche Herangehensweisen. Mit <Btn2Down> (mittlere Maustaste gedrückt halten) im Root-Window ist das Menü nur solange sichtbar (und auch nutzbar) wie die Maustaste gedrückt bleibt. Man muß bei stetig gedrückter Maustaste mit dem Mauszeiger auf den gewünschten Menüpunkt fahren und dann die Maustaste loslassen. Anders bei <Btn3Click> (rechte Maustaste im Root-Window drücken und wieder loslassen) - nach der Mausaktion erscheint auf dem Bildschirm ein stehendes Menü, aus dem dann mit der rechten Maustaste ein Menüpunkt angeklickt oder mit der Abkürzungstaste ausgewählt werden kann. Das X-Manual bekommt man, wenn man bei gedrückter Control-Taste die rechte Maustaste im Root-Window abklickt.

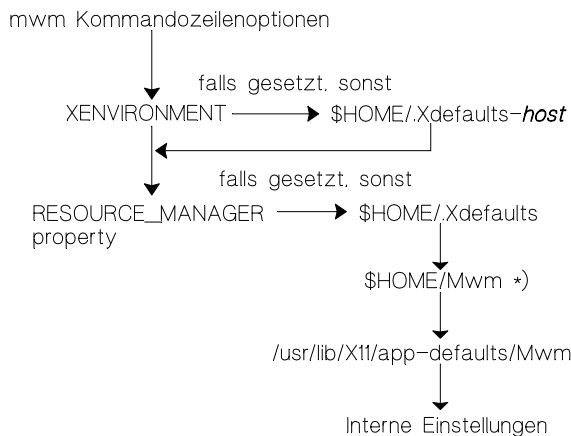
Wie bekommt man nun seine Uhr?

- irgendwo Alt<c> drücken
- im Root-Window <F11><c> drücken
- mittlere Maustaste im Root-Window drücken, zum Menüpunkt 'C'lock' ziehen und loslassen
- rechte Maustaste im Root-Window klicken, dann 'C'lock' mit Maus oder mit Taste <c> auswählen

Einfluß über Ressourcenparameter

Einleitung

Erscheinung und Verhalten von Windows, Window-Rahmen, Menüs und Icons werden durch Ressourcenparameter kontrolliert. Diese werden system- und/oder nutzerspezifisch in einer Reihe von Dateien definiert (siehe **Ressourcendateien**). Außerdem kann eine Ressourcen-Datenbasis über das RESOURCE_MANAGER-Property mit Hilfe des Programms 'xrdp' erstellt werden². Im folgenden Diagramm wird der Pfad illustriert, dem MWM zur Auswertung der Ressourcenparameter folgt (sind zwei gleichnamige Parameter an unterschiedlichen Orten mit verschiedenen Inhalten belegt, so wird nur der berücksichtigt, der zuerst vorgefunden wird):



Die Environmentvariable \$LANG kann noch eine Rolle spielen. Die Datei \$HOME/Mwm (*) wird erst ab Release 1.1 von Motif unterstützt. Im Folgenden kann nur eine kleine Auswahl von Ressourcenparametern diskutiert werden; die vollständige Übersicht entnehme man dem MWM-Programmer's-Guide.

Klassifikation

Der MWM unterscheidet zwischen drei Klassen von Ressourcenzuweisungen:

- MWM-Komponenten, d.h. alles was von MWM auf dem Bildschirm zu sehen ist (Window-Rahmen, Icons, Menüs und Feedback-Windows);
- Ressourcen, die Erscheinung und Verhalten von MWM beeinflussen und nicht einer einzelnen Komponente oder Klienten zugewiesen werden (z.B. Tastatureingabemodus);
- Ressourcen, die speziell einem Client zugewiesen werden können (z.B. Client-Dekoration und clientbezogene MWM-Funktionen);

Syntax

Einträge für den MWM in den nutzerspezifischen Ressourcendateien besitzen folgende Syntax:

Mwm*Subobjekt*Subobjekt*..*Ressource: Wert

Die *Subobjekte* dienen der eindeutigen Charakterisierung des MWM-Objektes, dem über das *Ressource*-Attribut ein *Wert* zugewiesen wird. Zum Beispiel wird im Folgenden der Font unseres Menüs auf die Größe 10x20 festgelegt:

Mwm*menu*mymenu*fontList: 10x20

Standardressourcen

Hinter- und Vordergrundfarben sowie verwendete Fonts aller MWM-Elemente (Window-Rahmen, Menüs, Icons und Feedback-Windows) besitzen einen einheitlichen Satz von Attributen. Da nur ein Window oder Icon für Tastaturein- und ausgaben aktiv sein kann, kann dieses noch extra farblich hervorgehoben werden.

inaktive Window/Icon-Ressource	aktive Window/Icon-Ressource	eingefärbte Fläche
foreground	activeForeground	Vordergrund (Text)
background	activeBackground	Hintergrund
topShadowColor	activeTopShadowColor	oberer und linker Rand
bottomShadowColor	activeBottomShadowColor	unterer und rechter Rand

Zum Setzen der Fontart und -größe wird das Attribut **fontList**

benutzt. Im Beispiel werden alle inaktiven MWM-Elemente hellgrau, das jeweils aktive Element grün und der Font auf die Größe 6x10 festgelegt:

```

Mwm*background:          LightGrey
Mwm*activeBackground:   Green
Mwm*fontList:            6x10
    
```

MWM-Komponenten

Um unterschiedliche MWM-Elemente mit eigenen Farben und Fonts zu versehen, wird folgende Syntax verwendet:

Mwm*{menu|icon|client|feedback}*Ressource: Wert

Zusätzlich können noch Ressourcenwerte für den Titelbalken aller Windows und für gekennzeichnete Menüs vereinbart werden:

Mwm*client*title*Ressource: Wert

Mwm*menu*Menüname*Ressource: Wert

Die Titelbalken aller inaktiven Windows türkis einzufärben, sähe dann so aus (Menü siehe oben):

Mwm*client*title*background: Turquoise

MWM-Gesamtverhalten

²Ressourcen-Management wird Thema eines ausführlicheren Artikels in der nächsten RZ-Mitteilung - darum wird hier nur erwähnt, was für die Arbeit mit dem MWM nötig erscheint.

Es kann immer nur ein Window für die Ein- und Ausgabe via Tastatur aktiv sein - wie dieses auserwählt wird, hängt von folgender Ressource ab:

Mwm*keyboardFocusPolicy: explicit (default)
Mit dem Default-Wert wird ein inaktives Window aktiviert, indem man es mit der Maus anklickt. Setzt man diese Ressource auf 'pointer' (Maus), dann wird ein Window schon dadurch aktiviert, daß der Mauszeiger auf das Window gefahren wird (ohne Klick!).

Der nächste Parameter sorgt dafür, daß ein verdecktes Window bei Aktivierung vollständig auf dem Bildschirm erscheint:

Mwm*focusAutoRaise: true
(default, falls Mwm*keyboardFocusPolicy: explicit)
Mwm*focusAutoRaise: false
(default, falls Mwm*keyboardFocusPolicy: pointer)
Im ersten Fall wird das aktivierte Window auf dem Windowstack nach oben befördert. Im zweiten Fall wird es zwar aktiviert, bleibt aber verdeckt - durch Anklicken des Rahmens kann es vollständig sichtbar werden, wenn dies in der Maus-Definitions-Tabelle der \$HOME/.mwmrc vereinbart wurde:
<Btn1Down> icon|frame f.raise

Beispiel

Einträge in \$HOME/.Xdefaults

```
Mwm*keyboardFocusPolicy: explicit
Mwm*focusAutoRaise: false
```

Maus-Tabelle der \$HOME/.mwmrc wie oben
Klickt man in die client-Fläche (d.h. innerhalb des Rahmens) eines inaktiven, verdeckten Windows, dann wird es zwar aktiv, bleibt aber verdeckt. Will man es vervollständigen, dann muß man den Rahmen anklicken. Die Möglichkeit, ein Window auch verdeckt zu aktivieren, ist bei der Durchführung eines Window-Dumps nützlich. Im aktiven, teils verdeckten Window kann der Dump-Befehl abgesetzt werden, während das zu "fotografierende" Window im Vordergrund unverdeckt bleiben kann.

Eine spezielle Icon-Ressource ermöglicht die Einrichtung eines Icon-Windows:

Mwm*useIconBox: false (default)
Mit dem Default-Wert 'false' werden die Icons im Root-Window plaziert - wird dagegen 'true' gesetzt, dann wird ein Extra-Window (die Icon-Box) generiert, in dem die Icons aufgehoben werden.

Client-Ressourcen

Jedes Window erhält standardmäßig vom MWM einen gewissen Funktionsumfang (clientFunctions) zugeordnet, die auch in der Dekoration der Windows (clientDecoration) wie minimize- und maximize-Knöpfe und resize-Rahmen erkennbar sind. Die Funktionalität wird wie folgt deklariert:

Mwm*clientFunctions: all (default)
Mwm*clientname*clientFunctions: Werte

Über den Default-Wert 'all' erhält jedes Window von Motif einen vollen Funktionsumfang. Dieser kann vom Nutzer für bestimmte Clients eingeschränkt werden. Die beeinflussbaren Funktionen sind:

Name	Beschreibung
all	alle fünf unteren Funktionen verfügbar
none	keine der fünf Funktionen möglich
resize	Größe des Window ändern
move	Window auf Bildschirm bewegen
minimize	Window in Icon verwandeln
maximize	Window auf Maximum vergrößern
close	Window schließen; Client beenden

Werden die Werte mit '-' eingeleitet, so geht MWM von einer vollen Funktionalität ('all') aus und zieht die folgenden Werte ab. Diese Möglichkeit soll das Beispiel illustrieren. In der Datei \$HOME/.xsession (in ihr wird nach Nutzervorgaben die X-Sitzung gestartet) sei folgende abschließende Zeile:

```
exec xterm -ls -C -name login
```

und in der Datei \$HOME/.Xdefaults:

```
MWM*login*clientFunctions: -close
```

Vom MWM aus kann dieses login-Window nicht beendet werden, da sowohl der Doppelklick auf das menu-Button als auch die Auswahl des Window-Menü-Punktes "Close" nicht mehr möglich sind

In ähnlicher Art und Weise kann die Dekoration eines Windows modifiziert werden.

Eine clientbezogene Icon-Ressource sei noch zum Schluß aufgeführt:

```
Mwm*clientname*iconImage: path/bitmap
```

Hier erhält man die Möglichkeit, den Windows selbstgefertigte Bildchen (mit dem Programm 'bitmap') als Icon zuzuordnen. Startet man eine xterm-Terminalemulation mit

```
xterm -name egon &
```

dann läßt sich mit

```
Mwm*egon*iconImage: $HOME/egon.bits
```

die Eigenproduktion 'egon.bits' als Icon verwenden.

Schlußbemerkung

Die Auswahl der Funktionen und Ressourcen war willkürlich und widerspiegelt nur einen kleinen Teil der Möglichkeiten des MWM. Ich hoffe dennoch, daß dieser Artikel das Interesse an der Gestaltung einer eigenen Motif-Umgebung wecken konnte. In den nächsten RZ-Mitteilungen wird ein Beitrag zu Hintergrund und Arbeitsweise einer Ressourcenverwaltung folgen, der eventuell verbliebene Unklarheiten aus diesem Artikel beseitigen kann.

Frank Sittel