HTWM

Hochschule für Technik und Wirtschaft Mittweida

Technikumsplatz 17
09648 Mittweida

X-Programmierung
Qt und Motif

Erstellt von:
Isabel Drost (if99wP1)

SendMail@isabel-drost.de
http://www.isabel-drost.de

Einführung
Drei Möglickeiten unter X zu programmieren:

- Low – Level - Programmierung
- Programmierung mit Motif
- Programmierung mit dem Toolkit Qt

Fachabschluß:
- Beleg
- Yellow Card mit 70% für Erfolg
- Beleg kombinierbar mit Semesterprojekt oder Corba

Literatur
- www.htwm.de/geiler/intranet
- O'Reilly im Netz kostenfrei verfügbar

1 Einführung in X

1.1 X und das Client – Server – Modell

- Netzwerkfähiges, graphisches System zur Entwicklung von graphischen Oberflächen
- X – Server realisiert Ein und Ausgaben für dieses System

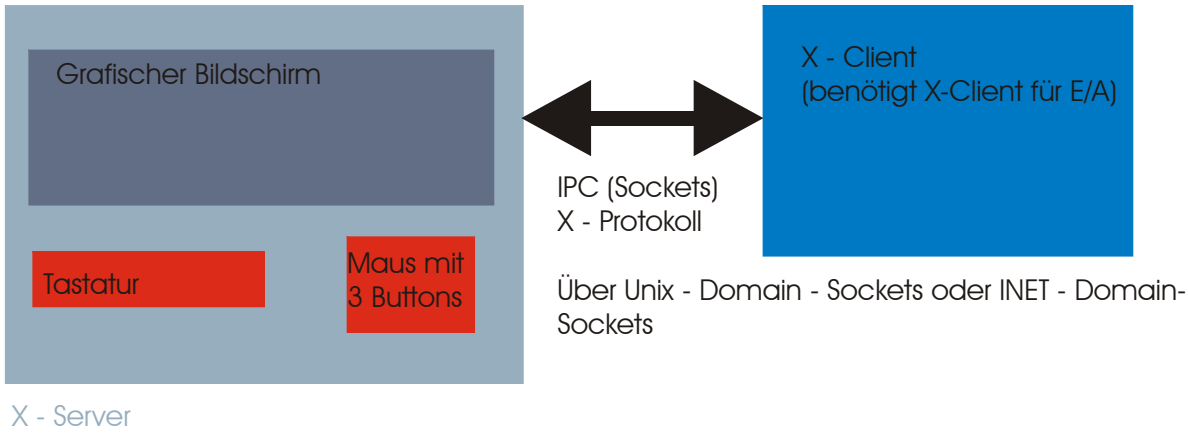


Abbildung 1-1; Client - Server - Modell unter X

1.1.1 Probleme:

- Unterschiedliche Monitorauflösung (Breite X Höhe)
- Unterschiedliche Farbauflösungen auf den Zielmonitoren
- Unterschiedliche Tastaturlayouts

1.1.2 Vorteile des X-Windowsystems

- plattformunabhängig
- netzwerkfähig

1.2 Arbeitsweise von Client und Server

- asynchrone Arbeitsweise: Puffer auf Client und Serverseite

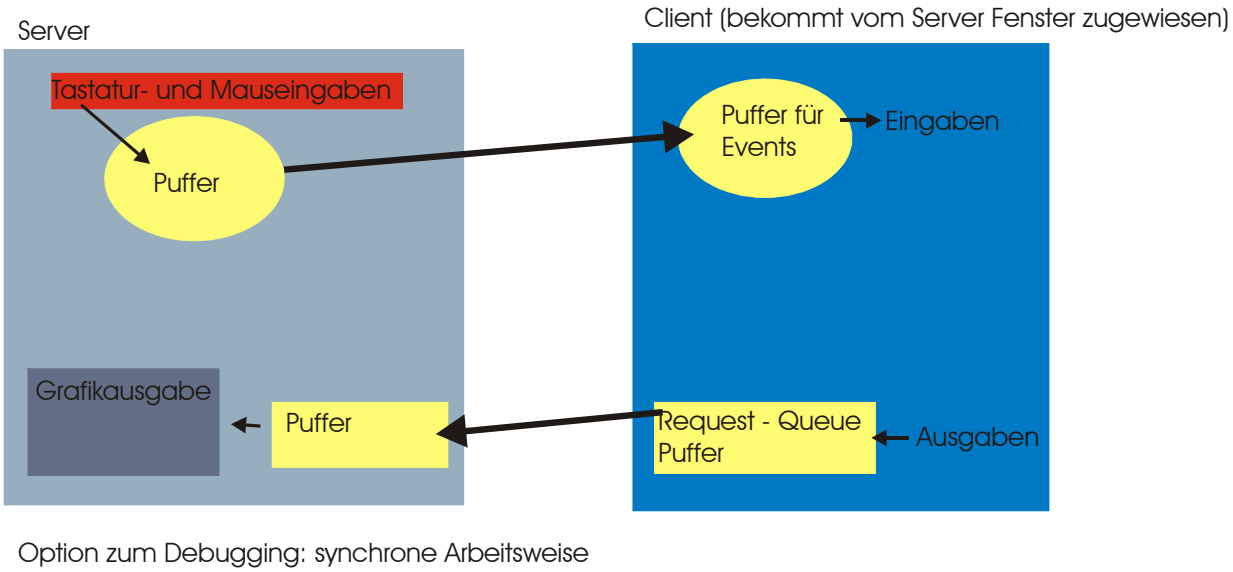


Abbildung 1-2; Gepufferte Arbeitsweise

1.3 Fensterhierarchie

- wird auf dem X-Server geführt, Wurzel ist das Root-Fenster (stellt gesamten Bildschirm dar)
- je X-Client mindestens ein Top-Level-Window

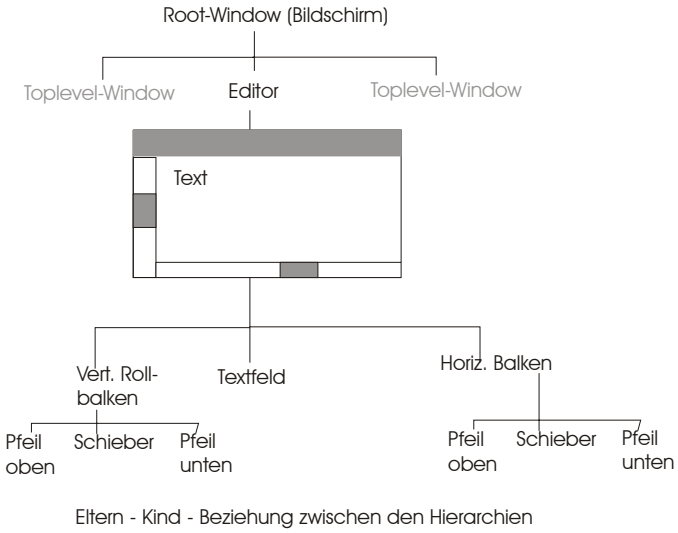


Abbildung 1-3; Hierarchien der X-Clients

1.4 Client – der Windowmananager

1.4.1 Aufgaben

- Dekoration der Fenster
- Größenänderung, Positionsänderung
- Fenster schließen, Iconisierung
- Organisierung der Überlagerung von Fenstern

1.4.2 Beispiele

- twm – Tab – Window – Manager
- fvwm, kde, gnome, enlightenment, iceWM...

1.5 Programmierung unter X

- grundlegende (c-) – Bibliothek Xlib (stellt mehr als 400 Funktionen des X-Protokolls zur Verfügung und führt sie aus) - „Assemblersprache von X“ (z.B. ups, xwge,)
- Xt – Intrinsics gestattat OOAufbau von Bausteinbibliotheken, stellt aber selbst keinen Baustein zur Verfügung (ca. 200 Fkt.)
- Baustein – (Widget –) Bibliotheken mit Xt: Xaw (Bsp: Xedit, Xman, Xfig, gv), Xaw3D, Motif
- widget-Bibliothek die auf XLib aufbaut (Qt, GTK, FLTK, XForms, WxWindows, ...)

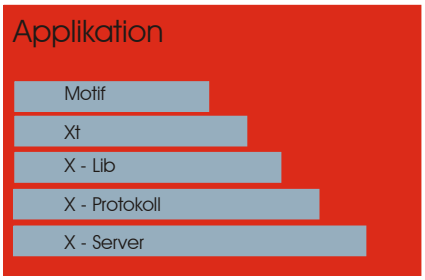


Abbildung 1-4; Applikations - Bibliotheksschicht

2 Motif – Programmierung

2.1 Prinzipien

2.1.1 Programmiermodell Interaktiver Systeme

Abbildung 2-1; Programmiermodell interaktiver Systeme

Xt: die Schleife wurde gekapselt

void XtAppMainLoop(XtAppContext app)

- Somit besteht das Hauptprogramm hauptsächlich in der Initialisierung.
-

2.1.2 Klassenhierarchie Motif – Widgets

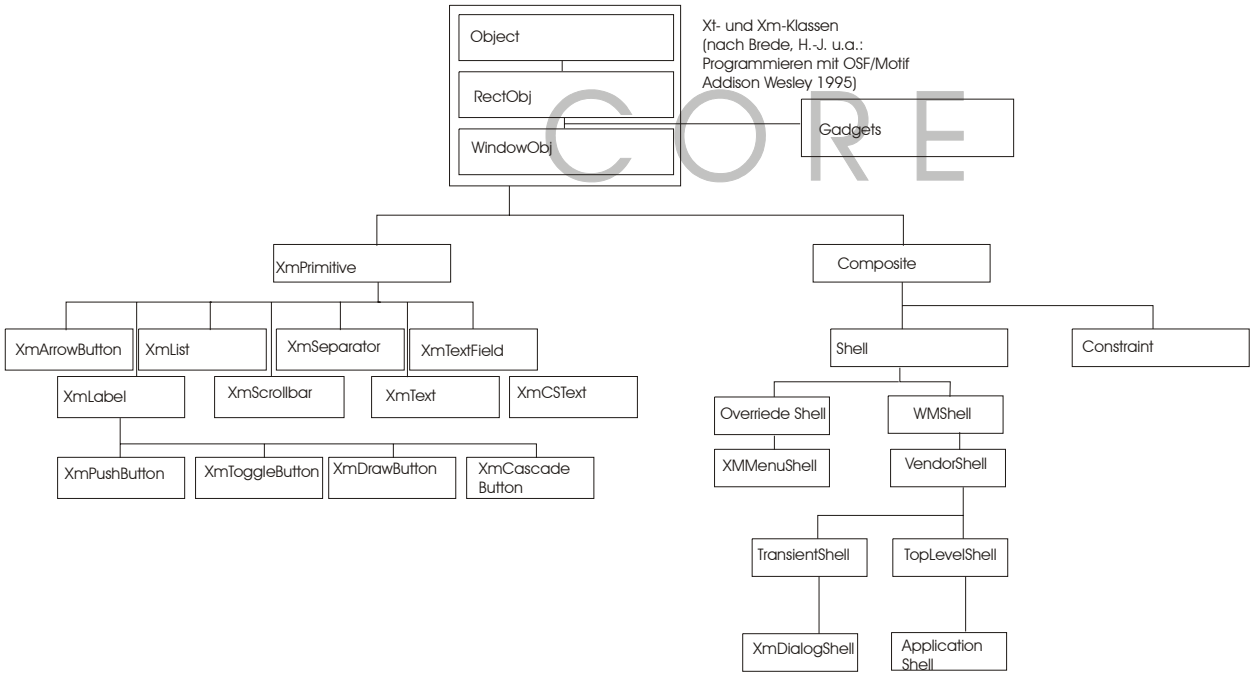


Abbildung 2-2; Klassenhierarchie Motif - Widgets

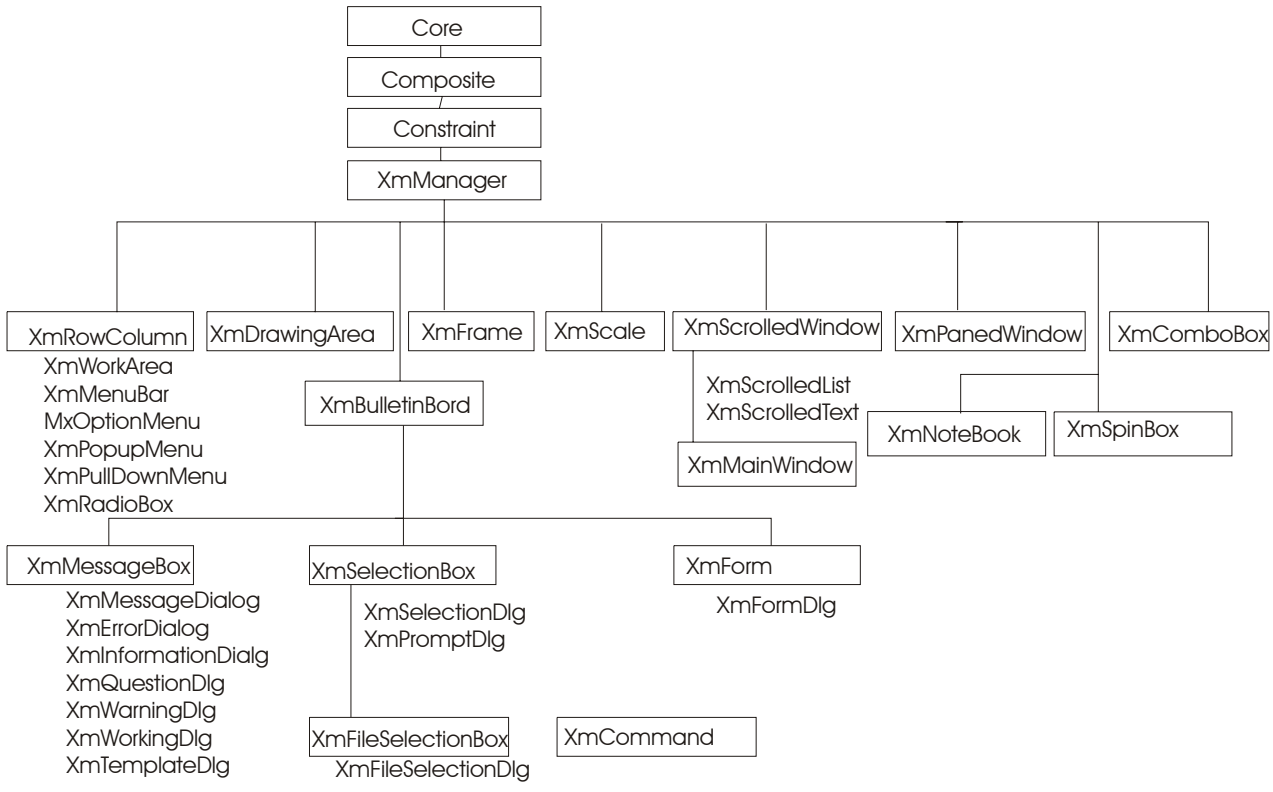


Abbildung 2-3; Motif - Managerklassen

2.1.3 Ressourcen

- enthalten Widgetparameter Motif ist stark konfigurierbar
- werden zur Laufzeit geladen und müssen dann auch interpretiert werden
- macht die Fenstersysteme relativ langsam

- Festlegung im Programm
- Festlegung in der Ressourcen-Datei
- Festlegung in einer Fallbackliste (wird genutzt, wenn keine Ressourcendatei nicht gefunden wird)

2.1.3.1 Aufbau einer Ressourcendatei:

```
                name.widgetname. ... .Ressourcen:Wert
                  .'.
               .*'   '*.
          Appklasse  Programmname
```

`Beispiel: Hello.button.foreground:red`

- Bei den Ressourcennamen die GROß-/ kleinschreibung beachten!
- Jokerzeichen sind möglich

2.1.3.2 Ressourcen-Typen

- Boolsche Werte (true, on, 1, yes / false, of, 0, no)
- Aufzählungstypen (XmALIGNMENT_CENTER)
- Größenangaben (XmNunitType:XmPIXELS oder XmCENTIMETERS)
  ab Motif 2.0 können die Maßeinheiten direkt hinter die Größen geschrieben werden, z.B. 0.5cm
  Einheiten: pix, pixel, pixels/ cm, centimeters, centimeter/ in/ pt/ fu (fontunits)
- weiter mögliche Werte: Strings, fonts, Farben (/usr/lib/X11/rgb.txt enthält die Farbnamen)

2.1.3.3 Wo werden Ressourcen festgelegt?

Der Widgetprogrammierer legt neue Ressourcen fest.

Der Applikationsprogrammier kann Ressourcen abfragen und setzen. In seinem Prog. direkt gesetzte Ressourcen können nachträglich nicht mehr gesetzt werden.

Der Anwender und der Sysadmin können best. Ressourcen nach seinen Wünschen einstellen.

2.2 Initialisierungsteil einer Motif-Anwendung

2.2.1 Überblick

Xm- und Xt-Funktionen in zwei Ausführungen:

a) Mit variabler Argumentliste

```
...XtVa...(Funktionsarg, Ressource,
Wert, Ressource, Wert, ..., NULL);
```

Vorteile:
- übersichtlich
- relativ wenige Fehlermöglichkeiten

Nachteile:
- Ressourcenänderung ist während der Laufzeit nicht möglich, sondern sie müssen während der Übersetzungszeit in Typ und Anzahl feststehen

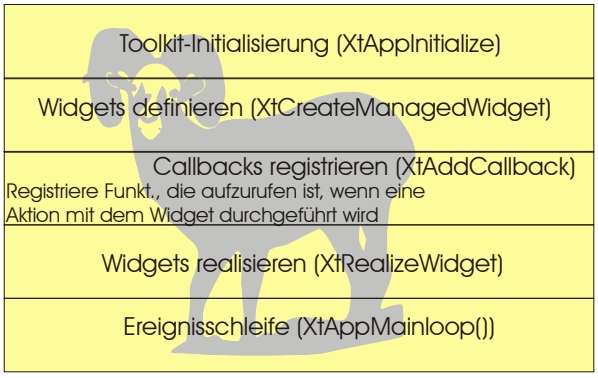


Abbildung 2-4; Initialisierungsvorgang einer Motifanwendung

b) Mit Argumentfeld

```
Xt ... (Funktionsargumente, Arglist args, Cardinal nargs)
args ... Feld mit Ressourcen/Wert – Paaren
nargs ... Anzahl der gültigen Argumente
```

Vorteile:
- Ressourcen können zur Laufzeit bereitgestellt werden

Nachteile:
- Fehleranfällig

Struktur einer Struktur arg:

```
typedef struct
{
      String name;
      XtArgVal value
}Arg, Arglist;
```

Makro zum Setzen der Werte im Argumentfeld

```
void XtSetArg(Arg arg, String name, XtArgVal value)
```

Beispiel zur Initialisierung mittels Xm/Xt-Funktion mit Argumentfeld

```
Arg args(4);
int i=0;
XtSetArg(args[i], XmNheight, 100); i++;
```

2.2.2 Toolkit-Initialisierung

siehe Arbeitsmaterial (C:_data\studium\SemesterVII\03_X_Programmierung\MotifFunkt.pdf)

Funktionen:
- Initialisierung des Toolkit (XtToolkitInitialize( ))
- Erzeugen von Applikationskontext
- Display eröffnen (Kommandozeilenoption –Display oder Umgebungsvariable DISPLAY) (XtOpenDisplay( ))
- Erzeugen des Shellwidgets (XtAppCreateShell( ))

Auswahl von Ressourcen der Shell (insg. 70, davon 68 geerbte):
- XmNheight, XmNwidth (Fensterhöhe , Fensterbreite)
- XmNmaxHeight, XmNmaxWidth (maximale Höhe/ Breite)
- XmNminHeight, XmNminWidth
- XmNmaxAspectX, XmNmaxAspectY (maximales Verhältnis von X/Y)
- XmNminAspectX, XmNminAspectY

2.2.3 Erzeugen und Initialisieren von Widgets

(siehe Arbeitsmaterial)

2.2.4 Registrieren von Callbacks und Eventhandlern

(siehe Material)

2.2.5 Realisieren von Widgets

XtRealizeWidget(Widget w)

2.2.6 Ereignisschleife

XtAppMainLoop(XtAppContext apc)

2.2.7 Der Ressourcen-Editor editres

- interaktive Anwendung von Ressourcen
- Dokumentation des Widgetbaumes
- Vorbereitung:
  - `#include<X11/Xmu/Editres.h>`
  - `XtAddEventHandler(shell, NoEventMask, True, _XEditResCheckMessages, (XtPointer) NULL);`
  - `Linken mit –lXmu`

2.3 Compound Strings

- mehrzeilige Zeichenketten, unterschiedliche Zeichensätze und Fonts, Farben und unterschiedliche Durchstreichungen

2.3.1 Zeichensätze(Fonts) im X-W.System

Utilities: xfontset
Fontdarstellung:
`-foundry-family-weight-slant-swdth-adstyl-pxlsz-plsz-resx-vesy-`

2.3.2 Aufbau eines CS

Folge von Komponenten:
- Text
- Text-Type (einf. (8 – Bit ASCII-Code) /Multibyte (Ein Teil der Zeichen wird über 16 Bit dargestellt, ein anderer Teil aus 8 bit)/Widechar (Unicode – 16 pro Zeichen))
- Separator
- Schreibrichtung
- Tab
- Gestaltungskennzeichen

Beispiel:

TAG GrosserFont	DIRECTION R_to_L	TEXT "grosser Text"	SEPARTOR	TAG KleinerFont	DIRECTION L_to_R	TEXT "Kleiner Text"

Abbildung 2-5; Beispiel zum Aufbau eines Compound Strings

```
xtime.button.fontlist: bx10=smallFont, 9x15=bigFont
s1=XmStringCreate("Zeit", "bigFont");
s2=XmStringCreate("übernehmen", "smallFont");
ButtonString=XmStringConcat(s1, s2);
```

3 Bausteine des Motifwidgetsets

3.1 Primitive

3.1.1 Label

siehe Material

3.2 Manager Widgets

3.2.1 Überblick

- Klasse Manager ist eine Superklasse für alle anderen Managerklassen
- bietet selbst keine direkt von ihr angelegten Instanzen an
- Scrolled Window = Default Manager für ein Fenster
- MainWindow = Fenster mit Kommandobereich
- BulletinBoard = Fläche, auf der Bausteine positioniert werden können
- Form = Allgemeines, sehr komplexes Formular
- RowColumn = Anordnung von Widgets in der Horizontalen oder Vertikalen (wesentlich zur Realisierung von Menüs, Menüleisten)
- Frame = Rahmen mit 3D-Aussehen zur Hervorhebung einzelner Sachen
- PaintWindow = zwei vertikal/horizontal angeordnete Einzelfenster (z.B. Netscape Mailer)
- DrawingArea

3.2.2 Superklassen der Manager

- Composite – Constraint

3.2.3 Bulletin Board

- Superklasse für Form, Messagebox, SelectionBox
- XmNx, XmNy, XmNwidth, XmNheight

3.2.4 Form

- stellt einen rechteckigen, leeren Raum zur Verfügung, in dem z.B. Buttons und Labels absolut oder relativ zueinander untergebracht werden können
- XmNfrachtionbase n teilt diesen Raum in n Zeilen und n Spalten ein
- ConstraintRessourcen:
  - XmN[bottom, top, left, right]Attachment
  - XmATTACH_NONE
  - XmATTACH_FORM
  - XmATTACH_OPPOSITE_FORM
  - XmATTACH_WIDGET
  - XmATTACH_OPPOSITE_WIDGET
  - XmATTACH_POSTION
  - XmATTACH_SELF
  - XmN[...]Widget

3.2.5 RowColumn

- XmNnmColumns
- XmNoientation (XmVERTICAL, XmHORIZONTAL)
- XmNpacking (XmPACK_TIGHT, _COLUMN, _NONE)

3.2.6 PanedWindow

Window mit Sash zum Verändern der Größe der einzelnen Panes (siehe auch ddd oder Netscape-Mailer)
Name der Ressource: XmNseparatorOn, XmNspacing
Constraint-Ressourcen: XmNpaneMaximum, XmNpaneMinimum, XmNpositionIndex

3.2.7 Drawing Area

- erlaubt graph. Grundoperatoren auf Basis von XLib
- Motif kann die Behandlung von Events, die sonst automat. erfolgt (resize/ expose) nicht realisieren
- Motif bietet Callbacks an, die für die Auswertung des Events übernehmen können (Für Mouseevents sind Eventhandler zu installieren)

Folge:
Beim Programmieren muß vorgesorgt werden, dass mein Verändern der Fenstergröße der Fensterinhalt wieder hergestellt wird.

Möglichkeit 1: Führe ein Backup des Fensterinhaltes im Speicher.
Möglichkeit 2: Wiederholung aller Zeichenkommandos.

Grafikprimitive aus der <X11/Xlib.h>
XDrawArc, XDrawPoint, XDrawLine, XDrawImageString, XDrawRectangle, XDrawText, XFillArc, XFillPolygon, XFillRectangle

Wichtige gemeinsame Parameter:
- Display xdisp (XtDisplay(widget) um das Display herauszukriegen, die das Widget aktuell nutzt)
- Drawable draw (Ausgabeziel, das entweder ein Window (siehe Fkt. XtWindow(drawingArea)), eine Pixmap oder ein Graphics Context ist)

Erzeugen eines GraphicsContext:

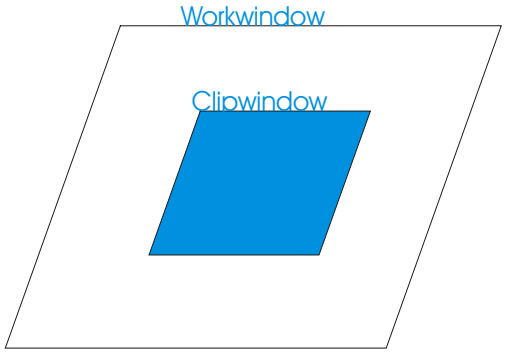
a) Mit Mitteln der X – Lib:
GC XCreateGC(Display *d, Drawable dr, unsigned long mask, XGCValues *val) (Standardeinstellungen des GC finden sich im Manual)

b) Mittels Xt-Funktionen
Vor allem für die gedacht, die selber Widgets schreiben.
GC XtGetGC(Widget w, XtGCMask mask, XGCValues xval)
Mit dieser Funktion darf einmal ein GC mit diesen Parametern erstellt werden. Anschließend kann der GC nicht mehr verändert werden. Es handelt sich um einen shared GC.

GC XtAllocateGC(Widget w, Cardinal depth, XtGCMask mask, XGCValues xval, XtGCMask dmask, XtGCMask dontcare);
mask – sofort gültig, dmask – später veränderbare Einträge, dontcare – Einträge spielen in meiner Anwendung keine Rolle

XtReleaseGC(Widget w, GC gc) – GC eines Widgets wird wieder frei gegeben

3.2.8 Scrolled Window



Arten der Steuerung:
- XmNscrollingPolicy (Standard) XmAPPLICATION_DEFINED
  - Applikation muss Rollleisten erzeugen und bedienen
  - Appl. muss auf Bedienung der Rollleisten reagieren
- XmAUTOMATIC – automat. Scrollen
  - Applikationsprogrammierer muss sich nicht um Rollleisten etc. kümmern
  - Anzeige der Rollleisten durch XmNscrollBarDisplayPolicy bestimmt (XmSTATIC (immer), oder XmAS_NEEDED)

3.2.9 Weitere Managerressourcen

ReadOnly: XmNChildren (WidgetList), XmNumChildren (Anzahl)
Boolean XtlsSubClass(Widget obj, WidgetClass ObjClass)

3.3 Ausgewählte Primitiv-Widgets

3.3.1 ListWidget

- Anzeige einer Auswahlliste (Strings), Auswahlmöglichkeiten
- Zeilennummerierung beginnt mit 1
- typisch: als Kind eines ScrolledWindow zu erzeugen → Conviniencefunktion ScrolledList

Auswahlmodi: (XmNSelectionPolicy)
- einfache Auswahl (XmSINGLE_SELECT)
- Browse Auswahl (XmBROWSE_SELECT)
- mehrfache Auswahl (XmMULTIPLE_SELECT) = zusammenhängende Bereiche selektierbar
- erweitert (XmEXTENDED_SELECT) = mehere Bereiche selectierbar

Funktionen für die Arbeit mit ListWindow:
- XmListAddItem()
- XmListAddItemUnselected()
- XmListDeleteItem()
- XmListDeleteAllItems()
- XmListSelectItem()
- XmListDeSelectItem()
- XmListGetMatchPos() ... Bool
- XmListItemExists() ... Bool

3.3.2 ToggleButton

- Ein-/ Ausschalter
- optischer Indikator und Text oder Pixmap als Teil davon

Zusammenfassung mehrer ToggleButtons: Manager muss Row-Column-Widget sein.

```
XmVaCreateSimpleRadioBox(
        Widget parent,
        String name,
        int intialy_pressed_button,
        XtCallbackProc cb /*Buttonnummer wird als Clientdata übergeben*/,
        ...
        NULL);
```

... – enthält XmRADIOBUTTON, label, NULL, NULL, NULL

```
XmVaCreateSimpleCheckBox(
        Widget parent,
        String name,
        int intialy_pressed_button,
        XtCallbackProc cb /*Buttonnummer wird als Clientdata übergeben*/,
        ...
        NULL);
```

... – enthält XmCHECKBUTTON, label, NULL, NULL, NULL

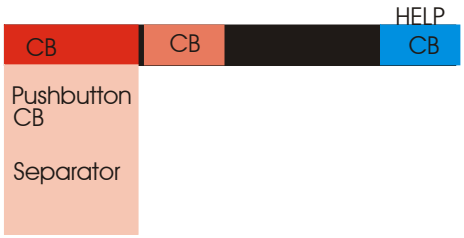
3.3.3 Text

- Editor, Clipboard

3.4 Menüs

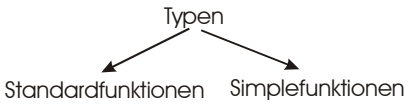
3.4.1 Überblick

- Pulldownmenüs (Help-Button ist hier immer an der rechten Seite der Menüleiste)
- Popupmenüs, die durch den dritten Mausbutton aktiviert werden (Context Menü, auslösbar über einen Eventhandler, nicht über einen Callback)
- Option Menüs, mit der Möglichkeit, eine aus n Möglichkeiten auszuwählen



Menüs können mit dem Tier off abreißbar gestaltet werden. Menüs sind keine eigenen Widgets, sondern Gruppen von Widgets, genauer aus Containern und dem Menü zugehörigen Widgets wie z.B. Pushbuttons.

3.4.2 Funktionen zur Erzeugung von Menüs



Standardfunktionen stellen nur Container zur Verfügnung und können z.B. genutzt werden können, um Menüs dynamisch zur Laufzeit zu erstellen und dabei Einträge wegzulassen, oder neu hinzuzufügen.
Simplefunktionen sind hingegen zwar einfacher, aber auch statischer. Es exisitieren Funktionen mit Variabler Argumentliste oder mit Argumentfeld.

3.4.2.1 Simplefunktionen mit Variabler Argumentliste

Parameter:

- Ressourcen Name-Wert-Paare
- Parametergruppen, Kennungabhängiger Aufbau

3.4.2.2 Pulldownmenüs

widget XmVSimpleMenuBar(Widget parent, String name, ... NULL)

Menüleiste ist ein RowColumnWidget. Seine Einträge bestehen aus CascadeButtons (Gadgets, Namen: button_0, button_1, ...).

XmVaCASCADE_BUTTON, label (compound String), mnemonic, die Ressource XmNMenuHelpWidget entspricht dem Hilfebutton und wird grundsätzlich am rechten Rand positioniert.

Erzeugen der Menüeinträge:
Widget XmVaCreateSimplePulldownMenu(Widget parent, String name, int button_no, XtCallbackProc cb, ...NULL)

Der Callback cb wird immer aufgerufen, wenn ich einen der Einträge im Menü auswähle, egal, welchen. button_no entspricht der Nummer des buttons, die ich oben vergeben habe.

In den Menüs sind die Ressourcen XmVaPUSHBUTTON, XmVaRADIOBUTTONS, XmVaCHECKBUTTONS, XmVaCASCADEBUTTON, XmVaSEPARATOR, XmVaDOUBLE_SEPARATOR oder XmVaTITLE erlaubt.

Den Ressourcentypen sind folgende Werte hinzuzufügen:

Gebe ich einen Title in meinem Menü an, ist zusätzlich ein lable anzugeben. Beim CascadeButton wie üblich lable und mnemonic. Bei den Push-/ Radio-/ Checkbuttons ist je label, menmonic, accelerator und acc-text anzugeben. Ein Accelerator ist dabei der übliche HotKey, der Text beschreibt den Accelerator. Die Acc. sind nicht immer zur Bedienung möglich, da sie häufig schon vom Windowmanager wie KDE abgefangen und verwendet werden.

3.4.2.3 Popupmenüs

... werden ausgelöst über einen Eventhandhandler. Es gibt zwei Möglichkeiten, beim Auslösen vorzugehen: entweder ich baue das Menü bei jedem Aufruf neu auf und zerstöre es, soll es nicht mehr angezeigt werden, oder ich baue es einmal auf und manage es erst, wenn es angezeigt werden soll – zum zerstören wird ein unmanage automatisch durchgeführt, wenn das PopupFenster den Fokus verliert.

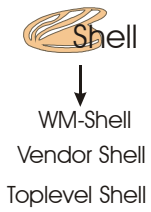
Funktion: XmVaCreateSimplePopupMenu(...)

3.4.2.4 OptionMenü

Wird wie alle anderen Menüs auch, aus CB und Pushbuttons erstellt – das was NS zum Absturz bringt.

3.5 Dialoge

... sind Widgets, die eine eigene Shell benötigen.



WM – Shell ... WindowManager Shell

Dialogboxen: erzeugt das Innere eines Dialoges, die Shell rundrum muss ich selber bauen
Dialoge: erzeugt sowohl eine Dialogbox als auch eine Standartshell rundrum (Fenster)

Funktionen: XmCreate ... Dialog(...) oder XmCreate ... Box(...)

Standarddialoge: Error, Information, Message, Question, Warning, Working. Standarddialoge bestehen meist aus einem Label, einer Standardpixmap sowie ein bis drei Auswahlmöglichkeiten.

Es gibt Dialoge mit einer SelectionBox (Selection). Diese beinhalten dann eine ScrolledList enthält, aus der der Anwender eine Auswahl treffen kann. Weiterhin gibt es einen Prompt-Dialog, bei dem ich ein Kommando eingeben kann, sowie einen FileSelection – Dialog, der ein standardisiertes Dateiauswahlfenster öffnet.

3.5.1 Wie werden Dialoge angezeigt und geschlossen?

Möglichkeit 1:
XtManageChild(); zum Anzeigen
XtUnmanageChild(); zum Schließen

Möglichkeit 2:
XtPopup(); zum Anzeigen
XtPopdown(); zum Schließen

Beide Möglichkeiten sollten nicht vermischt verwendet werden. Die vom Xt-Toolkit – Variante ist die zweite Methode.

Es gibt zwei Prinzipien nach denen ich vorgehen kann (wie beim normalen Popup):

- einmaliges Erstellen des Dialoges und Anzeigen bei Bedarf
- Erstellen des Dialoges bei Bedarf und Zerstören, wenn er geschlossen wird. Freigeben des Dialogs vermeidet Speicherlöcher ☺

Beim Erstellen des Dialoges kann ich festlegen, ob die anderen Fenster meiner Anwendung bedienbar bleiben:

- XmNdialogStyle, XmDIALOG_MODELESS (Alle anderen Anwendung sowie meine eigene lassen sich weiter bedienen)
- XmNdialogStyle, XmDIALOG_PRIMARY_APPLICATION_MODEL (Nur das Elternfenster des Dialogs nimmt keine Eingaben entgegen.)
- XmNdialogStyle, XmDIALOG_FULL_APPLICATION_MODEL (Die gesamte Elternanwendung des Dialogs nimmt keine Eingaben entgegen)
- XmNdialogStyle, XmDIALOG_SYSTEM_MODEL (Alle anderen Awendungen des Systems werden gesperrt, dies kann zu Problemen führen, insbesondere dann, wenn der Dialog nicht ganz oben auf dem Desktop zu sehen ist.)

4 Erweitertes Eventhandling

4.1 Verarbeitung von Tastatureingaben

4.1.1 Tastaturmapping von X11

Tastatureingaben erzeugen aus HWsignalen Tstatur- und serverabhängigen Code → Keycode. KeyCode und ein Status Byte (Control, Shift) werden in KeySym (16bit – Code vergl. <X11/keysymdef.h> umgewandelt. In der ersten Spalte darin wird jedem Keycode ein eindeutiges Symbol zugeordnet. Nahezu alle Tasten sind mehrfach belegt. Modifier Tabelle legt fest, welche Tasten als Kennung für Mehrfachbelegungen dienen. Für die Mehrfachbelegung werden Shift, ModeSwitch genutzt.

Beispielaufbau der Modeswitch Table:

Keycode	ohneShift	mit Shift	mit Altgr	mit Altgr+Shift
48	ä	Ä		
58	m	M	µ	

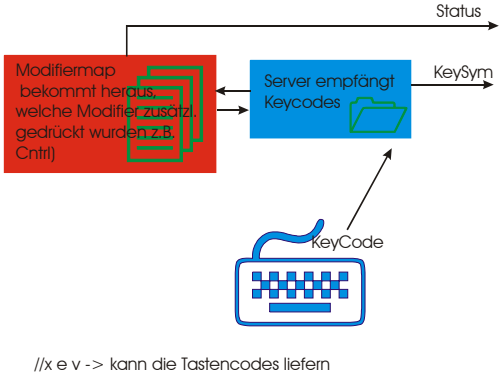


Abbildung 4-1; Tastaturmapping unter X11

Mittels xmodmap kann die Tastatur für X11 angepasst werden. (man xmodmap for further information). Mittels xmodmap kann auch das Pointermapping der Maus angepasst werden.

4.1.2 Die Translation Table

4.1.2.1 Allgemeines

... bestimmt, wie eine CallbackFkt. arbeitet.

Beispiel:
Translation Table
<Btn1Down> Arm()
<Btn1Up> Activate()
Disarm()

4.1.2.2 Syntax

Translation = Ereignisfolge : Aktionen
Ereignisfolge=Ereignis (',' Ereignis)
Ereignis=[modifier] '<'Ereignis'>'[anzahl] [detail]
Aktionen=[Aktion]
Aktion=aktionsname '('[Argumente]')'
Argumente=string(',' string)
detail=Tastensymbol aus <X11/keysymdef.h>, ohne Präfix xx_

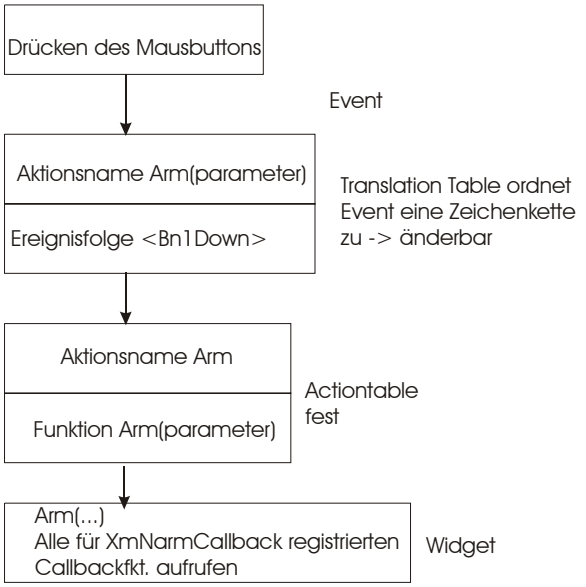


Abbildung 4-2; Arbeitsweise von Translation- und Actiontable

Details werden insbesondere angegeben, um spezielle Tasten zu benennen.

Wiederholungsfaktor, wenn ein Erignis mehrfach auftreten muss (z.B. Doppelklick) vgl. auch Ressource multiClickTime.

Beispiel für eine Tranlationtable für PushButtonWidget:
<Bn1Down>: Arm() \n\
<Bn1Up>: Activate() Disarm() \n\
<Key>space: ArmAndActivate

Translations müssen jeweils durch NewLine abgeschlossen werden.

4.1.2.3 Setzen und Ändern

- Im Programm selbst
  1) Schritt: Übersetzen in internes Format (XtTranslations XtParseTranslationTable(String s)
  2) Schritt: Installation der übersetzten Tabelle
     XtSetValues() – ersetzt komplette Tabelle
     void XtAugmentTrnlations(Widget w, XtTranslations trantable) – ergänzt Table um transtable, alte Einträge bleiben, auch wenn sie neu gesetzt werden
     void XtOverride Translatons(Widget w, XtTranslations transTable) – wie oben, überschreibt alte Einträge
- In Ressourcendatei
  #replace oder #override oder #augment
  Beispiel:
  XmPushButton.tranlations: \
  #override \
  <Btn2Down>:Arm()\n\
  <Btn2Up>: Activate() Disarm()

  (legt Pushbutton um auf 2. Taste, Auch hier Backslash und Enter nicht vergessen.

4.1.2.4 Installation eigener Aktionen

Actions:

- widgetspezifisch von Widgetprogrammierer festgelegt
- zusätzliche applikationsweite Aktionsroutinen installierbar
- Installation möglichst unmittelbar nach XtAppInitialize() erforderlich
- Aktion kann via Translationstabelle genutzt werden, dazu ist diese Tabelle zu ergänzen

Beispiel einer Aktionsroutine:
typedef void (*XtActionProc)(

Widget w,	//auslösendes Widget
Xevent* eventptr,	//Pointer auf EventStruktur
String parameter[ ],	//Feld auf Stringparameter
Cardinal* parameter)	//Pointer auf Parameteranzahl

Aktionsroutine:
```
void beepAct(Widget w, XEvent* ev, Sting *params, Cardinal *cnt)
{
   int anz=1;
   if(params==1){anz=atoi(params[0]);}
   for(;anz>0;anz--)
       XBell(XtDisplay(w), 0);
}
```

Registrieren der Routine:
```
void XtAppAddActions(XtAppcontext con, XtActionsRec actions[], Cardinal account)
```

Aufbau der Struktur XtActionsRec:
```
typedef struct{
      string action_name;
      XtActionProc proc;
   }XtActionsRec, *XtActionList;
```

Konvention: Verwendung des Namens der Aktionroutine als Aktionsname

4.2 Erweiterung der Ereignisbehandlung

4.2.1 Instllation zusätzlicher Eingbemedien

- gestattet Einbindung zusätzlicher Datenkanäle (Netzwerk, Pipes) in Eventsystem
- Erfüllung best. Bedinungen führt zum Aufruf des Eventhandlers

Registrierung des Datenkanals und des Eventhandlers:
```
XtInputId XtAppAddInput(
      XtAppContext app,       // AppKontext
      int Filedes,            //Datenkanal
      XtPointer Condition     //XtInputReadMask oder XtInputWriteMask oder
      XtINputExceptMask
      XtInputCallbackProc cb  //Inputcallback
      XtPointer client_data);
```

Aufruf des Inputhandlers:
```
void inputHanlder(XtPointer client_data, int*fildes, XtInputId*inputIaddr)
```

Deinstallation des Datenkanals:
```
void XtRemoveInput(XtInputId InputID)
```

4.2.2 Timer

ermöglichen einmalige Abarbeitung einer Rotuine nach fester Zeit (muss sich selbst neu installieren)
```
XtIntervalId XtAppAddTimeOut(
      XtAppContext pp,
      unsigned long milliseconds,
      XtTimerCallbackProc timerCB,
      XtPointer client_data);
```

Arbeite ich unter X11 mit Signalen, dann ist zu beachten, dass aus Signalhandlern heraus keinesfalls X-bezüglichen Aufrufe von Funktionen geschehen sollten, da dann Ausgaben/Anforderungen ins X-Protokoll an Stellen geschrieben werden, die unspezifiziert sind. Wenn, dann muss das Signal in obiges Eventsystem eingebunden werden. Aus der Signalroutine heraus sollten keine Veränderungen an meiner X-Applikation vorgenommen werden.

Entfernen des Timers:
```
void XtRemoveTimeOut(timer)
```

4.2.3 Work-Prozeduren

Wenn die Ereignisschleife kein Ereignis zu verarbeiten hat, wird i.d.R. auf das nächste Ereignis gewartet. Hier kann Hintergrund(Work-)prozedur abgearbeitet werden.

```
XtWorkProcId XtAppddWorkProc(XtAppContext app, XtWorkProc work, XtPointer
client_data);
```

Aufbau der WorkProzedur:
```
Boolean workProc(XtPointer client_data)
```
liefert True, wenn Workprozedur erneut aufgerufen werden soll, fals, wenn Workprozedur fertig ist.

Entfernen der Prozedur:
```
void XtRemoveWorkProc(XtWordProcID work)
```

5 Sicherheitsaspekte des X11-Systems

Die Kommunikation zw. X-Server und X-Client basiert auf der Annahme eines gutmütigen Nutzers. Jeder mit Zugang zum X-Server hat über ihn die volle Kontrolle: kann jede Mausbewegung, jeden Tastendruck abhören, der zum Server geschickt wird. Es ist also notwendig, den Zugang zum Server so zu beschränken, dass nur der ihn nutzen kann, der ihn aktuell nutzen muss.

5.1 Zugangskontrollmechanismen

- **hostbasiert** (unsicher, weil er den Zugang auf alle Nutzer eines best. Host beschränkt – jeder, der sich von Fern auf dem jeweiligen Rechner, kann meinen Server beeinflussen) via xhost läßt sich die Beschränkung einstellen.
  Auch heute noch werden viele Programme ausgeliefert, die genau diese Methode in ihrer Readme empfehlen, wenn ich als isabel in X bin und via su root geworden bin: man gebe xhost+ ein – schon kann die Installation gestartet werden (su auf X von isabel zugreifen), aber gleichzeitig hat auch der Rest der Welt theoretisch Zugriff auf meinen X-Server.
- **nutzerbasiert** ("magic cookie"-Methode: Wenn der Nutzer, bevor er seinen X11-Server startet, sich eine Zufallszahl ausdenkt (oder das System dies für ihn tut), dann kennt nur der Nutzer und sein X-Server diese Nummer. Will der Nutzer den X-Server den X-Server nutzen, muss der Nutzer die entsprechende Nummer vorweisen können. Ist zu finden unter ~/.Xauthority) Haken an der Sache: Das X-Protokoll läuft immernoch unverschlüsselt über das Netz, sobald ein Lauscher mein Magiccookie hat, kann er sich wieder angucken, was ich mache.
- **ssh-Zugriff** auf den Server, wenn der Client sich remote verbindet, um die Kommunikation zw. Client und Server zu verschlüsseln.

5.2 Anwendungen mit Paßworteingabe

- Paßworteingabe nach dem Motto "weißer Adler auf weißem Grund"
- Tastaturfokus exklusiv auf ein spezielles Fenster konzentrieren (XGrabKeyboard( )), Tastatur ist von einer anderen Anwendung auf dem gleichen Server nicht abhörbar, Nutzer ist aber gezwungen, das Paßwort einzugeben, da der Rest von X nicht mehr bedienbar ist.