This document contains some information about a simplified interface to the OSF/Motif programming interface.

It was written by Nils Schmeisser and is restricted by the GNU General Public License Agreement. You may not redistribute nor resale this document or parts of it. If you use the document or parts of it you should place the folloing in your index section:

"XAPP - simplified interface to OSF/Motif programming interface";
Nils Schmeisser

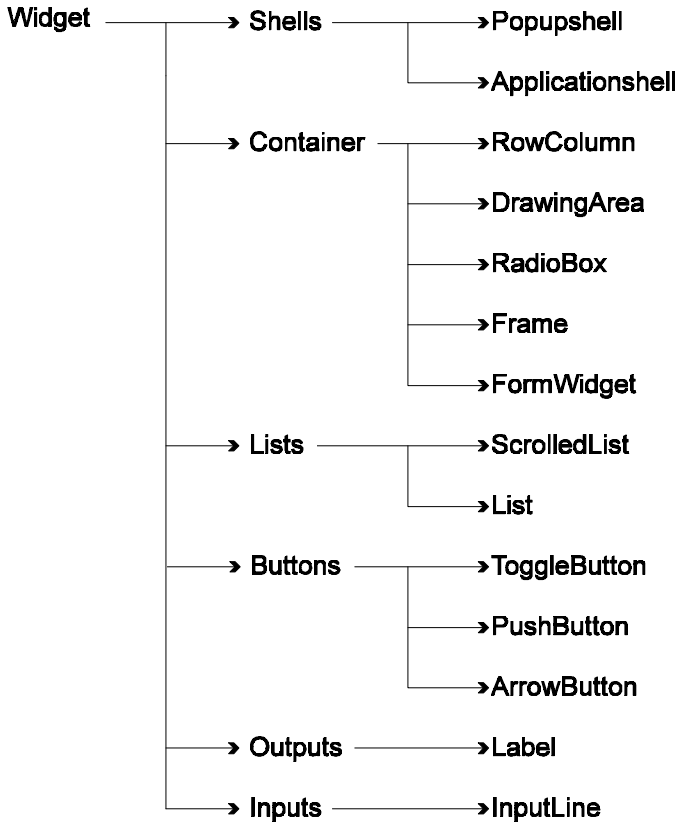
XAPP

Contents:

1. Introduction 1
2. Internals 2
3. Types 4
4. Functions 4
   4.1. Initialisation 4
   4.2. Creation functions 5
   4.3. Control functions 5
5. Programming Hints 7
   Style-Guide Appendix A
   Constants Appendix B

1. Introduction:

This module provides several functions for X programming.
The main "object" handled under X is the ***Widget*** type, which is a kind of geometric object that provides several functions, depending on the implementation for each widget type.
The tree on this page might give a short over overview on the available widgets.



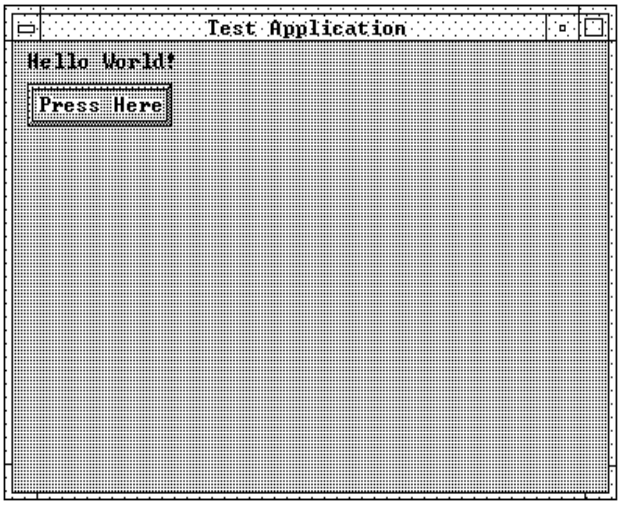
Reffering to this tree the folling short exapmles discusses how to program using this module.

Example 1:

```
#include "xapp.h"

Display *app_display;                   // the application display
XtAppContext app_context;               // the application context
Widget topshell;                        // the container which holds all widgets

void main() {
    Widget container;

        // initialising the X enviroment for the application, creating a display, an application context
        // and a topshell widget

    topshell=xInitApplication("Test Application",&app_display,app_context);

        // to make a mainwindow visible on the screen at least one widget should be inserted to the topshell
        // this is again a container

    container=xCreateRowColumn(XmVERTICAL,topshell);

        // inserting a string

    xCreateLabel("Hello World!",container);

        // inserting a pushbutton
```

```
        xCreatePushButton("Press here",kbNO,NULL,container);

            // now calling the main event handler

        XtAppMainLoop(app_context);
    }
```

Now a window looking like the following should appear on the screen.



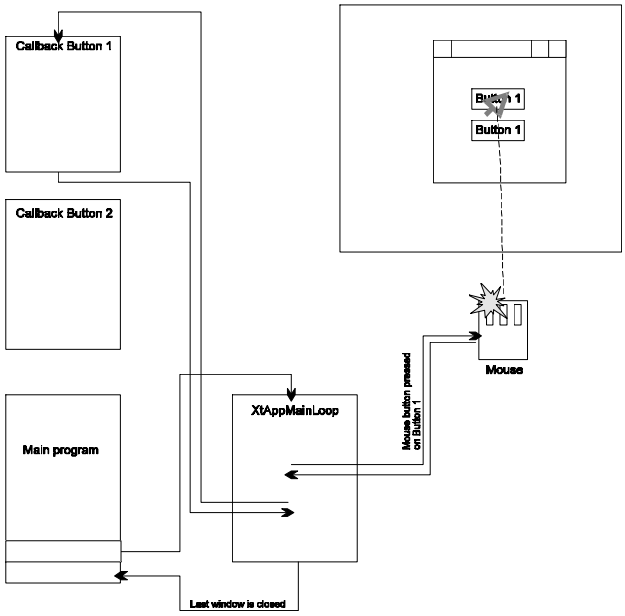
It should now be possible to klick the "Press Here" Button and to do some basics like moving and resizing the window. Closing the window is only possible by double clicking or selecting the style icon.
A full style guide of the supported widgets can be found in Appendix A.

2. Internals:

A X program normaly works in the following steps:

1. Initializing the X Toolkit
2. Creating the Widgets
3. calling to ***XtAppMainLoop***.

To initialize the X Toolkit a simple call to ***xInitApplication*** is sufficient. This simple call does several things. It first initializes the toolkit itself, then creates a display, a context and a topshell widget. The last is neccessary to hold other widgets. The ***xInitApplication*** also initializes some standard cursors which might be accessed through the ***xSetCursor*** function, keyboard constants, which assign a key to a ***pushbutton*** and finally mouse button constants which are used for determing what mouse button(s) are pressed.
To be able to react on inputs the user give by mouse or keyboard, the ***X-Toolkit*** takes care, that a ***Callback-Function*** assigned to a widget is called any time the widget is selected, clicked, how ever, if an event defined by the user occurs on the widget. The following digramm describes the principle, X is working with:



X takes care on **all** events. That means, assigning a callback function is the only task of the programmer. To assign a callback function to a widget, the xapp comes with a function called ***xAddCallback*** which requires a widget and a function of type ***XtCallbackProc*** which is defined as:

(XtCallbackProc*)(Widget widget,caddr_t in_data,caddr_t out_data)

Where ***widget*** is the widget which is assigned the callback function, ***in_data*** are the incoming data and ***out_data*** are the outgoing data. How should example 1 look like, if an output on pressing the button is excpected?

Example 2:

```
#include "xapp.h"

Display *app_display;                   // the application display
XtAppContext app_context;               // the application context
Widget topshell;                        // the container which holds all widgets

void button_callback(Widget widget,caddr_t in_data,caddt_t out_data) {
    printf("Button pressed.\n");
}

void main() {
    Widget container;
    Widget button;

    topshell=xInitApplication("Test Application",&app_display,app_context);
    container=xCreateRowColumn(XmVERTICAL,topshell);
    xCreateLabel("Hello World!",container);
    button=xCreatePushButton("Press here",kbNO,NULL,container);

    // this is the assignement statement

    xAddCallback(button,button_callback);

    XtAppMainLoop(app_context);
}
```

3. Types:

These types are used by the X Toolkit. They should neigther be changed, modified or redefined:

Display	the display which is used to display all objects
XtAppContext	the applications context
Cursor	the cursor type
Window	the window type
Widget	the widget type
Gadget	the gadget type (not used in xapp)
XtCallbackProc	the callback function type
caddr_t	adress type

Display
used as it is, no predefined values.

XtAppContext
used as it is, no predefined values.

Cursor
predefined values are: ***xArrowCursor, xHourglassCursor*** and ***xPencilCursor***. To define or use more cursors, reffer to [1].

Window
is the same as ***unsigned long***, if ***0*** then window identifies the screen.

Widget
used as it is, no predefined values.

Gadget
used as it is, no predefined values (not used in xapp).

XtCallbackProc
defined as (XtCallbackProc*)(Widget widget,caddr_t in_data,caddr_t out_data)

caddr_t
used as it is, mostly pointer to data structure

4. Functions:

4.1. Initializing:

Widget xInitApplication(char *app_name,
Display **app_display,XtAppContext &app_context)
Creates and initializes the ***display***, the ***application context*** and a ***topshell widget***. Should be the first statement in any X program. The display is returned in ***app_display***, the application context in ***app_context*** and the topshell widget is returned by the function itself.

void xInitStandardCursor(Display *display)
Initializes the standard cursors. These cursors are available through the ***xArrowCursor***, the ***xHourglassCursor*** and the ***xPencilCursor*** constants. This function is called automaticly by the ***xInitApplication*** function.

4.2. Creation functions:

Widget xCreatePopupShell(char *name,int x,int y,int w,int h,Widget Parent)
Creates at position (x,y) container with size (w,h) which should hold not more than one child. This container may be made visible by calling to:

Widget xCreateRowColumn(int flag,Widget Parent)
Creates a ***RowColumn*** conatiner widget. The container might be horizontal or vertical oriented. The orientation is indicated by the ***flag*** which may be set to:

XmHORIZONTAL	for horizontal orientation
XmVERTICAL	for vertical orientation.

Widget xCreateDrawingArea(int width,int height,Widget parent)
Creates a ***DrawinArea*** container of size (width,height).

Widget xCreateRadioBox(Widget Parent)
Creates a ***RadioBox*** container which will only hold ***ToggleButton***(s).

Widget xCreateFrame(Widget Shell)
Creates a ***Frame***.

Widget xCreateList(Widget Parent)
Creates a ***List***.

Widget xCreateScrolledList(Widget Parent)
Creates a ***ScrolledList***.

Widget xCreateToggleButton(char *Text,Boolean value,XtCallbackProc Proc, Widget Parent)
Creates a ***ToggleButton*** and assignes a callback function to it if the ***Proc*** argument is not equal to ***NULL***.

Widget xCreatePushButton(char *label,char mnemonic,XtCallbackProc Proc, Widget Parent)
Creates a ***PushButton*** and assignes a callback function to it if the ***Proc*** argument is not equal to ***NULL***. The pushbutton may be activated by a keystroke, which is defined by the ***mnemonic*** argument. If the mnemonic is set to ***kbNO*** then no ket is assigned to the pushbutton. See ***Mnemonics*** in the style guide section.

Widget xCreateArrowButton(int direction,XtCallbackProc Proc,Widget Parent)
Creates an ***ArrowButton*** and assignes a callback function to it if the ***Proc*** argument is not equal to ***NULL***.

Widget xCreateLabel(char *name,Widget Parent)
Creates an ***Label*** (static text).

Widget xCreateInputLine(char *name,char *value,int length,Widget Parent)
Creates a container carriyng a ***Label*** and a ***Text*** widget. The text widget is set to the value given by the ***value*** parameter.

4.3. Control functions:

void xPopup(Widget popupshell,int reg=true)
Makes a ***PopupShell*** widget visible. If the default argument ***reg*** is set to ***false*** then the window, assigned to the popupshell is not registered to the xapp internal applications window list.

void xPopdown(Widget popupshell,int reg=true)
Makes a ***PopupShell*** widget unvisible. If the ***reg*** parameter is set to ***false*** then the popupshell is not registered to the list mentioned above.

void xSetLabel(Widget w, char *new_label)
Replaces the contents of a ***Label*** by ***new_label***.

void xSetInputLineValue(Widget widget,double value)
void xSetInputLineValue(Widget widget,float value)
void xSetInputLineValue(Widget widget,long value)
void xSetInputLineValue(Widget widget,int value)
void xSetInputLineValue(Widget widget,char *value)
Sets the value of an ***InputLine***.

float xGetInputLineValue(Widget widget)
Gets the input line value as floating point number. If another format should be returned, use the ***char *XmTextGetString(Widget widget)***.

void xAddCallback(Widget widget,XtCallbackProc Proc)
Assignes a callback function to a widget.

void xSetCursor(Display *display,Window window,Cursor cursor)
Sets the cursor for the window ***window*** and its children. The cursor constants are mentioned above.

void xSetCursor(Display *display,Cursor cursor)
Sets the applications cursor, this means, the cursor is set for all windows, belonging to the current application. The windows have to be registered by the ***xRegisterWindow*** function.

Boolean xGetMousePosition(Display *display,Window window,int *x,int *y)
Returns ***true*** if the mouse pointer is in the window, specified by the ***window*** parameter. If the pointer is inside, the returned values for the x and y position are valid and local for the given window. If a window is known only as widget, use the ***XtWindow(Widget widget)*** function to determin the assigned window.

unsigned int xGetMouseButton(Display *display)
Returns the state of the mouse buttons. They may be checked by the mouse button constants: ***xLEFTBUTTON***,
xMIDDLEBUTTON and
xRIGHTBUTTON.

Window xWhereisMouse(Display *display,int *x,int *y)
Returns the window ID and the local mouse pointer position if the mouse pointer is in the given display.

void xRegisterWindow(Window w)
Registers the window w to the xapp internal window list.

void xRegisterWindow(Widget w)
Registers the window assigned to the widget w to the xapp internal window list.

void xUnregisterWindow(Window w)
Removes the window w from the internal list.

void xUnregisterWindow(Widget w)
Removes the window assigned to the widget w from the internal list.

5. Programming hints:

Every program should start with the statement *#include "xapp.h"*. This statement takes care on all necessary includes for X programming. If one is programming on an IBM RISC machine and wants to use the GL interface too, it is absolutly necessary, that the GL include statement is written **after** the xapp include statement.
The rest of the job is quit easy, just plan your applications outfit, it is a good idea to draw a little sketch including the container widgets, and then simply create the widgets step by step, using the ***xCreate*** functions.

A. Style-Guide:

Cursor shapes

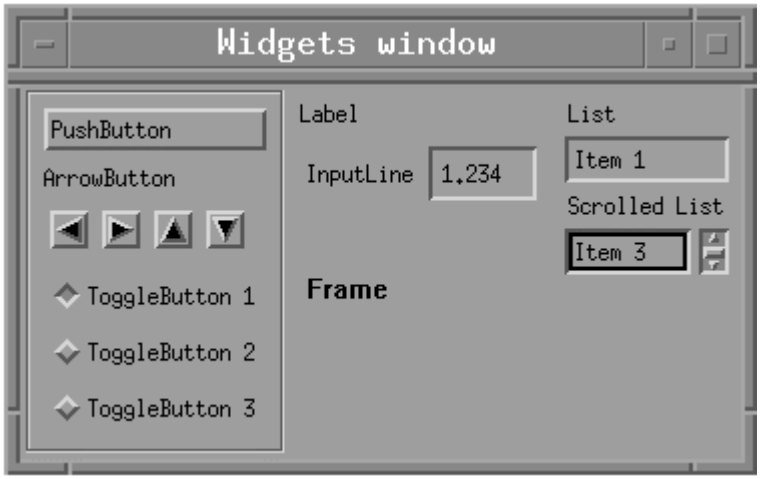
Cursor shapes of predefined cursors

xArrowCursor

xHourglassCursor

xPencilCursor

Widget style guide



B. Constants:

Keyboard mnemonics

const char kbNO	= '\0'	no key

The ***kbAlt*** and ***kbAltShift*** key mnemonics are built as follows:

const char kbAlt<key>	= '<char>'	Alt-<key>
const char kbAltShift<key>	= '<shift>-<char>'	Alt-Shift-<key>

Mouse Button Constants

const int xLEFTBUTTON	= 256
const int xMIDDLEBUTTON	= 512
const int xRIGHTBUTTON	=1024

Cursor Constants

```
extern Cursor xArrowCursor
extern Cursor xHourglassCursor
extern Cursor xPencilCursor
```