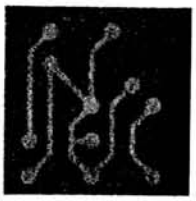


Parecer

Confirmo tanto o empenho revelado no estágio pelo aluno do 4º ano de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto, **José Manuel Pereira da Costa Moreira**, como a qualidade técnica do trabalho realizado "Desenvolvimento de uma Interface Gráfica para um Ambiente CAD 3D. Aplicação para Design de Calçado".

Considero assim o candidato merecedor da bolsa PRODEP.

Porto e INESC, 28 de Julho de 1993

Raul Fernando Almeida Moreira Vidal
Professor Associado da FEUP

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
Departamento de Engenharia Electrotécnica e de Computadores
Rua dos Bragas, 4099 Porto Codex, PORTUGAL
Telef. 351-2-317105/107/412/457 · Telex 27323 FEUP P · Telefax 351-2-319280

Parecer

Confirmo tanto o empenho revelado no estágio pelo aluno do 4º ano de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto, **José Manuel Pereira da Costa Moreira**, como a qualidade e o interesse científico do trabalho realizado e intitulado "Desenvolvimento de uma Interface Gráfica para um Ambiente CAD 3D. Aplicação para Design de Calçado".

Considero assim o candidato merecedor da bolsa PRODEP.

Porto e FEUP, 28 de Julho de 1993

Fernando Nunes Ferreira
Professor Catedrático da FEUP

UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA
Departamento de Engenharia Electrotécnica e de Computadores

Desenvolvimento de uma interface gráfica para um ambiente CAD 3D. Aplicação para design de calçado.

Relatório de estágio

José Manuel Pereira da Costa Moreira

1992/93

AGRADECIMENTOS

Agradeço aos meus orientadores de estágio, Prof. Fernando Nunes Ferreira, orientador pela FEUP, e Prof. Raul Vidal, orientador pelo INESC, por todo o apoio prestado ao longo da realização do trabalho.

Pretendo também endereçar um agradecimento ao PRODEP que subsidiou esta acção.

Porto e FEUP, 28 de Julho de 1993

José Manuel Pereira da Costa Moreira

MINISTÉRIO DA EDUCAÇÃO

DIRECÇÃO-GERAL DO ENSINO SUPERIOR

SUBPROGRAMA 4 - PRODEP

MEDIDA 4.3

REF. DO PROJECTO: 4.3/__/__/92/__

Metodologia da Avaliação

1. Metodologia Geral :

2. Cálculo da classificação :

Nota : Anexar enunciados de provas de avaliação, relatórios de estágio, dissertação

(mod.H)

Desenvolvimento de uma interface gráfica para um ambiente CAD 3D.

Aplicação para design de sapatos.

Relatório das actividades desenvolvidas no INESC-Porto durante o estágio no âmbito da bolsa PRODEP por :

José Manuel Pereira da Costa Moreira

sob orientação de:
Professor Fernando Nunes Ferreira

O objectivo deste trabalho era realizar um pequeno sistema de menus de fácil utilização e configuração.
A interface escolhida foi Motif (MWM) por ser um Window Manager muito divulgado.

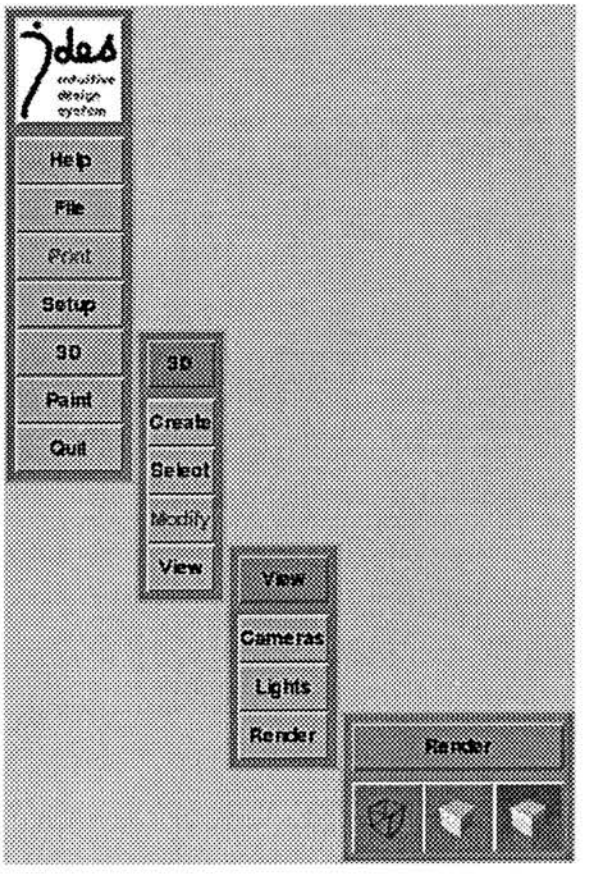
Umas das principais preocupações foi de tentar criar esse mini-sisema com um estílo própio. Por isso, toda a programação foi feita utilizando o "baixo nível" do X Window (XLib e XToolkit). Motif só foi utilizado para testes.

Por consequencia deste método de programação, este mini-sistema é portável e funçiona sobre OW (Open Windows), ou TWM (Tom Window Manager), etc...

Numa primeira fase, foi necessário aprender X Window e Motif.
Depois veio o desenvolvimento do sistema com:

- criação e testes.
- inserção no projecto IDES (Intuitive Design System) da responsabilidade do Engº Vasco Branco do INESC.

A inserção no IDES implicou a criação de icons (gráficos para os botões) de elevada definição e qualidade, com um aspecto a 3 dimensões.



I Progamação em X Window
II .Mini-sistema de menus.
III Listagens

PROGRAMAÇÃO EM X WINDOW

APRESENTAÇÃO

X Window é um sistema gráfico de janelas, desenvolvido no MIT em 1984, e foi obtido da conjugação de esforços entre dois laboratórios: Project Athena e Laboratory for Computer Science.

O objectivo era construir uma interface portável e comum, independente do "hardware", para evitar a necessidade de reprogramar grandes aplicações para cada máquina.

A versão actualé a X11 R5 (1992)

X Window foi adoptado pelos principais fabricantes de equipamento: DEC, HP, Sun, IBM, AT&T, ..., e as aplicações são facilmente transferidas de um sistema para outro.

O ambiente X Window é um ambiente amigável que permite utilizar vários terminais, possui ferramentas para edição de textos, "debuggers", etc...

Ele oferece a possibilidade de execução e de visualização simultâneas de várias aplicações.

Estrutura do sistema de janelas X Window

Sistema Base de Janelas (Base Window System)

Ele fornece mecanismos para a construção de interfaces, mas não os estilos dessas interfaces.

A independência é garantida pelo protocolo de comunicação (dispositivo/fabricante).

Protocolo de Comunicação (X Network Protocol)

Único meio de comunicação com SBJ (Sistema Dase de Janelas).

Transparência em relação às aplicações remotas.

Interface de Programação de Baixo Nível (Xlib)

Biblioteca de funções em C.

Interface de Programação de Alto Nível (X Toolkit)

Biblioteca de ferramentas/objectos em C.

1º Parte: Sistema de janelas X Window

1º Características do Sistema X Window

Saída:

As janelas ("Windows") são organizadas hierarquicamente com possibilidade de sobreposição. Cada aplicação é livre de criar as janelas de que necessita, movê-las, redimensioná-las, etc...
As operações gráficas são do tipo "raster" com endereços relativos à janela.
O texto tem uma alta qualidade devido à grande Quantidade de fontes existente.
O uso de cor pode ser feito através de paletas (desde 4 a 8 bits) ou cores directas (representação a 24 bits).
O Sistema X permite ainda a utilização de monitores monocromáticos.

Entrada:

Os dados introduzidos por teclado ou rato são distribuídos às aplicações sob a forma de eventos (controlado pelo SBJ).
X suporte uma grande variedade de teclados.

Comunicação entre aplicações:

Método "Cut and Paste" para dados não estruturados simples ("strings").
Comunicação de dados estruturados através de atributos ("X Properties").

Concorrência de aplicações:

Uma "workstation" pode ser compartilhada de forma independente (partilha do monitor, teclado, rato, etc...) por várias aplicações.

2º Utilização do Sistema X Window

Para um utilizador, uma "workstation" X é composta de um teclado, um rato e um monitor (onde coexistem janelas de várias aplicações).

O estilo de utilização é determinado pela interface seleccionada:
Motif, OpenWindows, Twm, etc...

As aplicações são variadas e com múltiplas funções:

- xterm: terminal de computador alfanumérico.
  Essencial para aproveitar software já existente.
  Possibilidade de ter vários terminais em simultâneo.
- bitmap: editor de bitmaps.
- xclock: relógio analógico ou digital.
- xcalc: calculadora.
- xload: monitor da performance da máquina.
- xedit: editor de texto.
- xman: visualizador de páginas de manual.

Xlib

As aplicações interagem com a "workstation" recebendo eventos e reagindo a eles.
Uma aplicação entra num "Main loop" infinito e todo o funcionamento é feito reagindo aos eventos que chegam e fazendo pedidos.

Toolkits.

O XToolkit é uma biblioteca de funções relacionadas com a interface homem-máquina.
Para o utilizador: objectivo de standardizar essa interface.
Para o programador: objectivo de simplificar o desenho e implementação das interfaces homem-máquina.
Toolkit básico: X Toolkit.

2º Parte: Conceitos básicos - XLib

Introdução

Cada aplicação comunica com uma workstation X através de uma conexão ao monitor (Display Connection).
- É criado um circuito de rede lógico entre a aplicação e a workstation.
- Toda a comunicação se processa através desta conexão.
- Aplicação necessita de autorização prévia para estabelecer conexão (xhost).

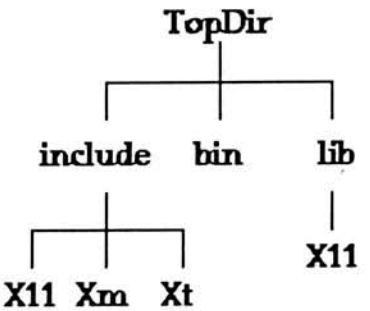
Filosofia cliente-servidor (client-server).
- Software servidor colocado na workstation.
- Aplicação cliente executada a partir de qualquer máquina.
- Servidor aceita pedidos de conexão de aplicações. Através dessas conexões recebe pedidos (requests) das aplicações e responde-lhes.
- Eventualmente, utiliza essas conexões para enviar eventos.

Funcionamento assíncrono entre aplicação-workstation
- Os pedidos das aplicações são guardados em buffers.

Protocolo de comunicação (tipos de mensagens)
- Pedidos unidireccionais (one-way protocol request messages) como operações gráficas.
  - O facto de serem bufferizados aumenta a performance da aplicação (não espera pela execução do pedido).
- Pedidos bidireccionais (round-trip request messages) como operações de pesquisa.
  - Ao pedido está associada uma resposta da workstation o que impede a sua bufferização.
- Eventos (event messages).
  - É a informação fornecida pela workstation à aplicação. É da responsabilidade desta informar a workstation dos tipos de eventos que está interessada em receber. Tipos mais comums: eventos de rato e teclado, gráficos e janelas e comunicação entre aplicações.
- Eventos de erros (error event messages).
  - A workstation notifica a aplicação de problemas na execução de pedidos desta.

Recursos
- Janelas (Windows) organizadas hierarquicamente a partir de uma "root window".
- Contextos gráficos (Graphic Contexts) utilizados para guardar informação sobre primitivas gráficas.
- Fontes de texto.
- Mapas de cores (Colormaps).
- Cursores.
- Áreas gráficas auxiliares (Pixmaps).

Organização do sistema.



Conexão ao Monitor

Nome da workstation (UNIX)
hostname:wks_number.screen_number
Caso não tenha um nome utilizável: variável DISPLAY
Ligações UDS (locais), TCP/IP (:) e DECNET (::).

Abrir conexão (uma aplicação pode ter várias abertas simultaneamente)
XOpenDisplay (display_name).

Fechar conexão
XCloseDisplay (Display *).

Janela raiz
Window XDefaultRootWindow(Display *);

Eventos

Pontos principais
Else serados pelo utilizador (rato ou teclado), por outras aplicações ou pela própria workstation.
De forma geral, cada aplicação só recebe os eventos que solicitou.
Os eventos são armazenados numa fila de espera até que a aplicação os analise.
Cada aplicação é responsável por analisar os seus próprios eventos.
A maior parte dos eventos indica:
o que aconteceu,
em que janela,
a janela raiz da anterior,
quando aconteceu, etc...

Categorias e tipos
Rato: ButtonPress, ButtonRelease, MotionNotify, EnterNotify, LeaveNotify.
Teclado: KeyPress, KeyRelease, FocusIn, FocusOut, ...
Gráficos em janelas: VisibilityNotify (Janela tornada visível ou invisível), Expose, ...
Alterações em janelas: CreateNotify, MapNotify, ...
Comunicação entre clientes: SelectionClear, ClientMessage, ...
Intercepção de pedidos de janelas: MapRequest, ResizeRequest, ...

Solicitar eventos
A solicitação é feita através da "event mask" de uma janela
XSelectInput(Display *, Window, long event_mask);
XCreateWindow,
XChangeWindowAttributes.

Máscaras: KeyPressMask, ButtonPressMask, EnterWindowMask, ...

Aceitação e análise de eventos
XNextEvent (Display *, XEvent *);
XPeekEvent(Display *, XEvent *);

Janelas

As janelas estão organizadas hierarquicamente debaixo da "root window".

Uma aplicação cria geralmente uma janela "Top Level".

Geometria:

Largura, altura, posição (x,y) em relação à janela mãe, largura da moldura e origem.

Vida:

Criação	(existe mas sem estar visualizada)
Unmapped	(podem-se alterar características da janela)
Mapped	(se alguma das antecessoras não está mapped)
Viewable	Unviewable
Visible	Obscured
Unobscured	Partially
Obscured	

Eventos relacionados com janelas:

CirculateNotify-	Alteração na pilha das janelas
ConfigureNotify-	Reconfiguração da janela
CreateNotify-	Criação de uma sub-janela
DestroyNotify-	Eliminação da janela
GravityNotify-	Alteração da posição da janela devido a alteração da superior.
MapNotify-	Mapeamento da janela
ReparentNotify-	Alteração da hierarquia das janelas
UnmapNotify-	Desmapeamento da janela
VisibilityNotify-	Alteração da visibilidade da janela
ClientMessage-	Evento enviado por outro cliente

Gráficos

Os gráficos são orientados a bitmaps.
Os dispositivos de saída são chamados Drawables (Window, Pixmap).
Os contextos gráficos (GC) comtem os atributos de visualização.
Sequência de operações (Graphic Pipeline)

Fase de selecção de pixeis (Pixel Selection Stage):
Fase de selcção de padrões (Patterning Stage):
Fase de "clipping" (GC Clip Stage),
Fase de visibilidade da janela (Window ClipStage)
Visualização (Raster Output Stage).

Primitivas Gráficas

Pontos:	ex XDrawPoint (...).
Linhas:	ex XDrawLine (...).
Arcos, Círculos e Elipses:	ex XSetArcMode (...), XDrawArc (...).
Rectângulos:	ex XFillRectangle(...).
Polígonos:	ex XFillPolygon(...).
Limpeza de áreas (só para janelas):	ex XClearWindow(...), XClearArea(...).
Cópia de áreas:	ex XCopyArea(...).

Eventos Associados às Primitivas Gráficas

Expose Event
    XMapWindow, XMapRaised
    XUnmapWindow
    XRaiseWindow, XLowerWindow
    Alteração de dimensões ou posição
    XClearArea
GraphicsExpose Event
NoExpose Event

Técnicas avançadas de Desenho de Primitivas Gráficas
Uso de Padrões: ex XFillStyle(...).
Uso de "Clipping": ex XSetClipMask(...).

Texto

Fontes (Fonts):
Tamanho dos caracteres.
Espaçamento dos caracteres: "fixed-pitch" ou "variable pitch".
Estilo dos caracteres.
Nomes identificam a fonte:
vrb-25
vri-31
9x15

Manipulação de Fontes:
Carregar , usar, libertar.
Desenho de Texto: ex XDrawString(...), XDrawText(...).

Cor

Classes Visuais
Estratégias de tradução dos valores de pixeis em cores.
PseudoColor:
Valor de pixel como índice de uma paleta (colormap). O conteúdo desta pode ser alterado.
DirectColor:
Valor de pixel é decomposto em três campos de 8 bits (R, G e B) que indexam três paletas independentes. O conteúdo delas pode ser alterado.
GrayScale:
Como PseudoColor, com a diferença que apenas um dos canais primários é utilizado na representação da cor.
StaticColor:
Igual a PseudoColor mas com uma paleta fixa.
TrueColor:
Igual a DirectColor mas com paletas fixas (rampa linear).
StaticGray:
Igual a GrayScale mas com paleta fixa.

Paletas (ColorMaps)
Conjunto de células de cor (XColor) associados às janelas.

Funções de pesquisa das capacidades da workstation
Estratégias de utilização de células de cor
Partilha de células de cor:
Simples, economia de recursos, aplicações portáveis para vários visuais.
Paletas (colormaps) standard:
ex: XA_RGB_DEFAULT_MAP, com XGetStandardColormap(...).
Células de cor privadas:
Manipulação de paletas
Criação, associação a uma janela, eliminação de paletas

Imagens, "Bitmaps" e "Pixmaps"

Pixmaps:

Área de desenho rectangular invísivel (existe em memória).
Pixmaps com profundidade de 1 bit: bitmaps.

Um Bitmap é um Pixmap com uma profundidade de 1 bit.
Um Bitmap file, é um ficheiro de texto com o formato:

```
#define name_width 16
#define name_height 16
#define name_x_hot 12
#define name_y_hot 1
static char name_bits[ ] = { 0x00, 0x00, 0x00, 0x10, 0x00, 0x18, ..... }
```

Funções de manipulação:
XReadBitmapFile (...), XWriteBitmapFile (...).

Imagens:
Problemas existentes na manipulação:
Grandes blocos de dados difíceis de transmitir.
Codificação de bits diferentes entre CPU's.
Relação entre a cor (valor) dos pixeis das imagens e a correspondente da paleta utilizada.
Funções de manipulação:
XCreateImage(...), XDestroyImage(...)...

Rato e Cursor

Eventos predefinidos permitem conhecer a posição e o estado do rato:
ButtonPress, MotionNotify.
O cursor (objecto visível) pode ser modificado tanto em cor como em forma.

Teclado

"Focus" do teclado:
XGetInputFocus(...).
XSetInputFocus(...).
O teclado é definido por:
Códigos: correspondentes a cada tecla (depende do teclado).
Mapas: indica o estado de cada tecla.
Símbolos: correspondentes aos vários caracteres.

3º Parte: X Toolkit

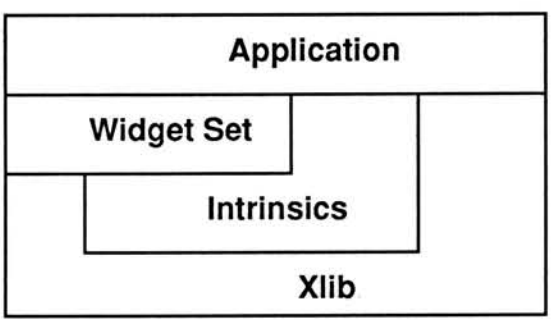
O X Toolkit é composto por um conjunto de funções base (Intrinsics) e por um núcleo de objectos (widgets). Ele não é responsável pelo estilo e consistência da interface, mas fornece um conjunto de ferramentas que simplificam o desenho de interfaces homem-máquina das aplicações.
A maioria das funções são destinadas ao desenhador da interface, e não ao programador da aplicação.

Terminologia:

Classe (Class): grupo a que pertence um objecto.
Instância (Instance): um objecto específico de uma classe.
Métodos (Method): funções implementadas numa classe.
SuperClasse (Superclass): grupo que engloba outras classes.
Objecto (Widget): algo que fornece uma abstracção da interface h/m.
Um objecto (widget) é a combinação de uma janela (Xlib) e a entrada/saída de dados associadas.

Modelo:

Cada objecto (widget) está contido numa janela X Window.
A informação respeitante a cada objecto é privada ao próprio e às suas subclasses. Toda a interacção com o objecto é realizada através dos métodos definidos para esse objecto.
A semântica de cada objecto está claramente separada da definição geométrica desse objecto.



Intrinsics:

Inicialização do Toolkit.
Gestão de objectos.
Geometria de objectos.
Gestão de memória.
Eventos de janelas e tempo.
Foco de janelas.
Contextos gráficos e pixmaps.
Erros e avisos.

Programação:

Objectos (Widgets):

Rectângulo com entrada e saída de dados associadas.
Cada objecto pertence a uma única classe (procedimentos e dados associados a esses objectos, eventualmente herdados pelas subclasses).
Na realidade, apontador para uma estrutura.

Instância de um objecto: estrutura própria da instância e estrutura de classe.

Objectos de base:

Core (campos comuns a todos os objectos)

Cada objecto possui um ficheiro .h com as declarações necessárias às aplicações e outro ficheiro .h, privado, acessível apenas pelas subclasses.

Instanciação de objectos (Widgets):
Organizados hierarquicamente em árvores.
Objectos compostos (composite) com gestão especial dos filhos ou primitivos.
Inicialização do X Toolkit:
Manipulação de contextos de aplicação (Application Context):
Manipular a conexão ao monitor (display):
Recursos (Resources):
ex: -background
-bordercolor
Quando existe, a Base de Dados de recursos é carregada a partir de:
- Ficheiro de definição de recursos de classe.
TopLevel/lib/X11/app-defaults/class
- Ficheiro de definição de recursos próprio do utilizador.
~/class
- Ficheiro de preferências do utilizador
~/.Xdefaults
- Ficheiro de recursos por workstation
~/.Xdefaults-host
- Linha de comandos (argv)
Criação de objectos:
Alocação dos objectos e inicialização dos recursos respectivos.
Objectos compostos são notificados dos objectos geridos.
Cada objecto cria uma janela X que é mapeada.
Widget XtCreateWidget (...), XtDestroyWidget (...);
Listas de argumentos (ArgList):
Criação de um objecto topo de aplicação (application shell):
Widget XtAppCreateShell (...);
Realização de um objecto (realize widget):
Informações sobre a janela de um objecto:
XtDisplay(...), XtWindow(...);

Objectos compostos (Composite Widgets):
Responsabilidade na gestão geral dos objectos contidos.
Responsabilidade na criação e destruição dos objectos filhos.
Responsabilidade nos arranjos geométricos dos objectos filhos.
Responsabilidade no mapeamento dos objectos filhos.
Funções de manipulação:
XtManageChildren (...), XtMapWidget (...)

Objectos principais (Shell Widgets):
Contêm os objectos de mais alto nível de uma aplicação e estabelecem a comunicação entre eles e o gestor de janelas.
Tipos públicos:
OverrideShell: utilizados por janelas que ignoram o gestor de janelas.
TransientShell: para janelas que podem ser manipuadas pelo GJ mas não podem ser iconificadas separadamente do resto da aplicação.
TopLevelShell: para janelas de alto nível.
ApplicationShell: usada pelas janelas principais de uma aplicação.
Outros tipos:
Shell: classe base para todas as outras "shells".
WMShell: contem campos necessários para o protocolo de comunicação com o gestor de janelas.
VendorShell: contem campos específicos utilizados por gestores de janelas de fabricantes.

Objectos "pop-up" (Pop-up Widgets):

Utilizados para criar janelas fora da hierarquia normal da aplicação.

Cada objecto "pop-up" tem uma janela que descende directamente da janela raiz.

Tipos:

Modeless: visível para o GJ, comportando-se como qualquer outra aplicação, para o utilizador.

Modal: pode ou não ser visível para o GJ, mas impede que qualquer evento que ocorra for da janela seja reconhecido.

Spring-loaded: invisível para o GJ e desactiva todos os eventos que não ocorram na janela.

Funções de manipulação:

XtCreatePopupShell (...);
XtPopup (...);
void XtPopDown(...);

Geometria dos objectos:

Controlada pelo objecto pai (embora possam indicar algumas preferências).

Alterações provocadas pelo pai:

XtMoveWidget, XtResizeWidget, XtConfigureWidget

Alterações pedidas pelo objecto filho:

XtMakeGeometryRequest, XtMakeResizeRequest

Alterações pedidas por aplicações:

XtSetValues (...).

Funções de alteração de geometria:

XtMakeGeometryRequest (...);
XtMakeResizeRequest (...);
XtMoveWidget (...);
void XtResizeWidget (...);
void XtConfigureWidget (...);

Gestão de eventos:

Os objectos não lêem directamente eventos nem manipulam o teclado e o rato.

Cada objecto tem um conjunto de funções que são executadas quando os eventos acontecem.

Uma aplicação típica consiste num código de inicialização seguido de um ciclo infinito de leitura de eventos (XtAppMainLoop). Normalmente esta é a única função de gestão de eventos utilizada.

Funções do gestor de eventos:

Adicionar ou remover novos eventos (interrupções temporizadas, ..).
Pesquisar o estado das fontes de eventos.
Adicionar ou remover funções que devem ser chamadas quando um evento ocorre.
Permitir ou desligar o envio de eventos para um determinado objecto.
Chamar a função correspondente quando um evento é lido.

Funções de registo de novas entradas de eventos:

Registo de ficheiros:

XtAppAddInput (...);
XtRemoveInput (XtInputId id).

Registo de interrupções temporizadas:

XtAppAddTimeOut (...);
XtRemoveTimeOut (...);

Função de manipulação do foco do teclado:

XtSetKeyboardFocus (...);

Função de pesquisa de eventos:

XtAppPending (...);
XtAppPeekEvent (...n);
XtAppPeekEvent (...);
XtAppProcessEvent (...);
XtDispatchEvent (...);
XtAppMainLoop (...);

Sensibilidade de objectos:

XtSetSensitive (...);
XtIsSensitive (...);

Filtragem de eventos:
- Movimento do cursor (campo compress_motion)
- Entrada / saída de janelas (campo compress_enterleave)
- Exposição de janelas (campo compress_expose)

Funções de tratamento de eventos
- XtAddEventHandler (...);
- XtRemoveEventHandler (...);

"Callbacks":

Funções executadas quando ocorre um determinado evento.

Manipulação de Callbacks.
- XtAddCallback (...);
- XtAddCallbacks (...);
- XtRemoveCallback (...);
- XtRemoveCallbacks (...);
- XtRemoveAllCallbacks (...);

Gestão de recursos:

Campo na estrutura de um objecto

Métodos de alteração dos valores:
- XtCreateWidget,
- XtSetValues,
- Ficheiro de recursos

Convenções de nomes
- Começam por minúsculas.
- Em relação aos campos da estrutura, perdem '_' e a letra seguinte é capitalizada.
- Os recursos 'Intrinsics' começam por XtN
- Ex:
  - background_pixel
  - backgroundPixel
  - XtNbackgroundPixel

Cada objecto herda os recursos das suas superclasses.

Funções de manipulação
- XtGetValues (...);
- XtSetValues (...);

Funções utilitárias

Gestão de memória
- char *XtMalloc (Cardinal size);
- char *XtCalloc (Cardinal num, Cardinal size);
- char *XtRealloc (char *ptr, Cardinal num);
- void XtFree (char *ptr);
- type *XtNew(type);
- String XtNewString (String string);

Partilha de Contextos Gráficos (Só leitura)
- GC XtGetGC (...);
- XtReleaseGC (...);

Translação de coordenadas
- XtTranslateCoords (...);

4º Parte: OSF/Motif

1º Motif Window Manager (MWM)

Composto por dois elementos fundamentais: o gestor de janelas (Mwm-"Motif Window Manager") e as aplicações Motif.
Aplicações criadas de duas formas: biblioteca de funções de criação e manipulação de objectos Motif ("Motif Toolkit") ou linguagem própria de criação de interfaces ("Motif User Interface Language").

Princípios de desenho das interfaces Motif

Camada visível de uma aplicação, manipulada pelo utilizador e através da qual ele indica as acções que devem ser executadas.
Evolução desde as interfaces em modo texto (técnicos) até às interfaces gráficas.
Objectivos principais:

permitir ao programador criar facilmente aplicações,
permitir que o utilizador use essa aplicação eficazmente e com satisfação

Deve adaptar-se ao nível de conhecimento de um utilizador:

reduzindo o tempo de aprendizagem de utilização,
facilitando a manipulação utilizadores experimentados.

Deve levar sempre em conta o utilizador final a que a aplicação se destina.
Regras:

Adoptar a perspectiva do utilizador e não a do programador;
Construir a interface com o auxílio dos utilizadores finais de forma a que o trabalho destes não sofra grandes alterações quando usarem a aplicação;
Fornecer ao utilizador o controle da aplicação (interface flexível);
Possibilitar a reconfiguração pelo utilizador e adaptação aos seus progressos;
Utilização de símbolos semelhantes aos do mundo real (botões, etc...);
Permitir manipulação directa de objectos (botões, elevadores, listas,etc...);
Permitir que o resultado de um comando seja utilizável como entrada para outra acção, por exemplo por criação de listas de valores;
Garantir que a aplicação responde com a maior rapidez possível a uma entrada de dados;
Manter a interface o mais natural (lógica) possível, de tal forma que o utilizador possa prever o resultado das suas acções;
Organizar os objectos por ordem de utilização, tentando reduzir ao máximo as necessidades de movimentação do utilizador;
Minimizar o contraste entre objectos para reduzir a distracção do utilizador pelas cores;
Manter a consistência da interface (componentes semelhantes operam da mesma maneira, não alterar a posição do cursor bruscamente, etc...);
Manter o utilizador informado acerca das acções da aplicação, fornecendo-lhe um "feedback" das suas próprias manipulações;
Antecipar erros prováveis e fornecer informação sobre eles quando ocorrem;

Gestor de janelas OSF/Motif

Organiza as janelas das aplicações de forma a que os programas se preocupem exclusivamente com as suas saídas de resultados.
Permite que o utilizador tenha um controle directo sobre objectos gráficos, através do rato ou do teclado, seleccionando janelas, menus ou ícones.
O gestor Mwm ("Motif Window Manager") associa a cada janela uma moldura com o seu nome e com vários botões auxiliares.

TitleBar,
TitleArea,
MinimizeButton,
MaximizeButton,
WindowMenuButton,

Restore, Move, Size, Minimize, Maximize, Lower, Close
ResizeBorderHandles
MainWindowMenu,

Configuração:
/usr/lib/X11/app-defaults/Mwm
~/.mwmrc (Resource Description File)
~/.Xdefaults

Exemplo de ".Xdefaults"

```
! app-defaults RESOURCE SPECIFICATIONS FOR
Mwm (mwm has precedence over Mwm)
!

Mwm*moveThreshold:          3
Mwm*buttonBindings:         DefaultButtonBindings
Mwm*keyBindings:            DefaultKeyBindings
Mwm*windowMenu:                   DefaultWindowMenu
Mwm*rootMenu:               RootMenu

!
! component appearance resources
!
*resizeBorderWidth:         7

! FONT stuff
*fontList:                  variable

! Use smaller fixed font for icons
*icon*fontList:             fixed

*highlight:  #980035005000
*borderColor:       #00000000ff00
*foreground:        #000000000000
*background:        #b900c200bc00
*doubleClickDelay: 250
*language_preference1: 6
*language_preference2: 6
Mwm*keyboardFocusPolicy:    explicit
!
! END OF RESOURCE SPECIFICATIONS FOR
MOTIF
!
XRnMotif.version:           6.17 (Motif)
#
```

Exemplo de ".mwmrc"

```
 #
# DEFAULT mwm RESOURCE DESCRIPTION FILE
(system.mwmrc)
#
```

```
# Root Menu Description
Menu RootMenu
{
    "Root Menu"         f.title
    no-label        f.separator
    "Clock"         f.exec "xclock &"
    "New Window"    f.exec "xterm -r &"
    "Calculator"    f.exec "xcalc  &"
    "Colormap"      f.exec "/usr2/X11R4M/usr/bin/X11/xcmap  &"
    no-label        f.separator
    "Shuffle Up"    f.circle_up
    "Shuffle Down"      f.circle_down
    "Refresh"       f.refresh
    no-label        f.separator
    "Restart..."    f.restart
    no-label        f.separator
    "Logout"        f.quit_mwm
}

Menu DefaultWindowMenu MwmWindowMenu
{
    "Restore"    _R    Alt<Key>F5                    f.normalize
    "Move"       _M    Alt<Key>F7                    f.move
    "Size"       _S    Alt<Key>F8                    f.resize
    "Minimize"   _n    Alt<Key>F9                    f.minimize
    "Maximize"   _x    Alt<Key>F10                   f.maximize
    "Lower"      _L    Alt<Key>F3                    f.lower
    no-label           f.separator
    "Close"      _C    Alt<Key>F4                    f.kill
}

Keys DefaultKeyBindings
{
    Shift<Key>Escape                    icon | window           f.post_wmenu
    Meta<Key>space  icon | window       f.post_wmenu
    Meta<Key>Tab    root | icon | window                        f.next_key
    Meta Shift<Key>Escape               root | icon | window    f.prev_key
    Meta Ctrl Shift<Key>exclam          root | icon | window    f.set_behavior
    Meta<Key>F6     window              f.next_key transient
}

Buttons DefaultButtonBindings
{
    <Btn1Down>      frame | icon            f.raise
    <Btn2Down>      frame | icon            f.post_wmenu
    <Btn1Down>      root    f.menu          RootMenu
    Meta<Btn1Down>          icon | window   f.lower
    Meta<Btn2Down>          window | icon   f.resize
    Meta<Btn3Down>          window          f.move
}
```

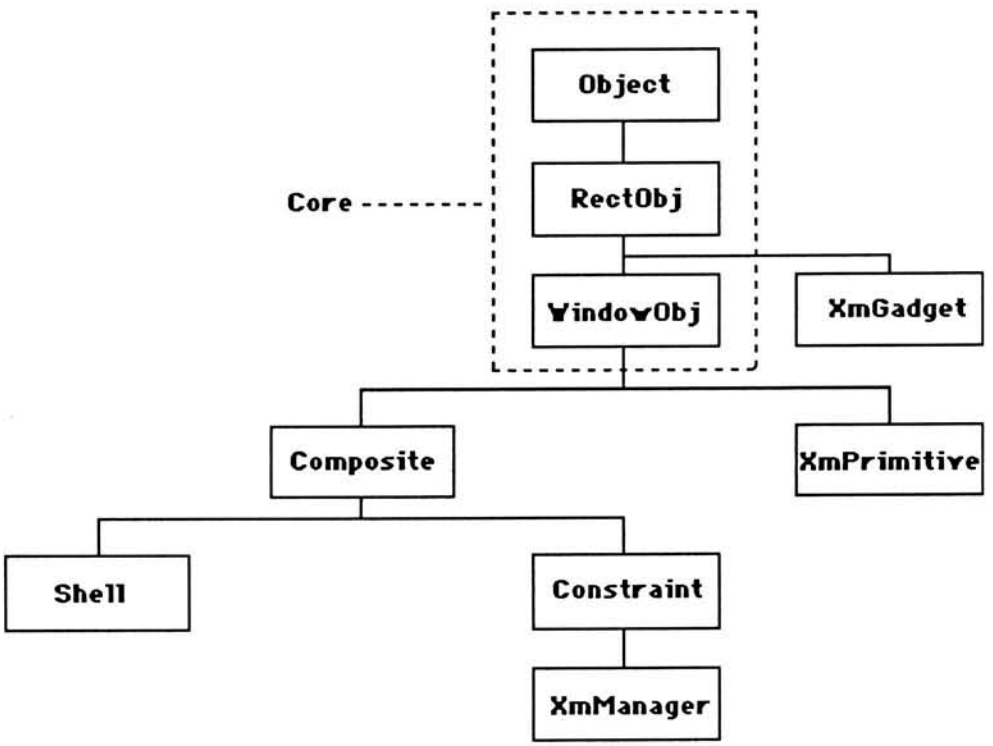
Objectos OSF/Motif

Definem a interface com o utilizador de uma aplicação. Baseado no Xt Toolkit.
O conjunto de objectos pode ser alargado pelo programador, combinando vários objectos num só ou introduzindo novos objectos.

As aplicações criadas com estes objectos são geradas mais rapidamente e têm mais garantias de respeitar o estilo Motif, tendo a desvantagem de gastar mais memória e serem ligeiramente mais lentas na execução.

Classes e hierarquias

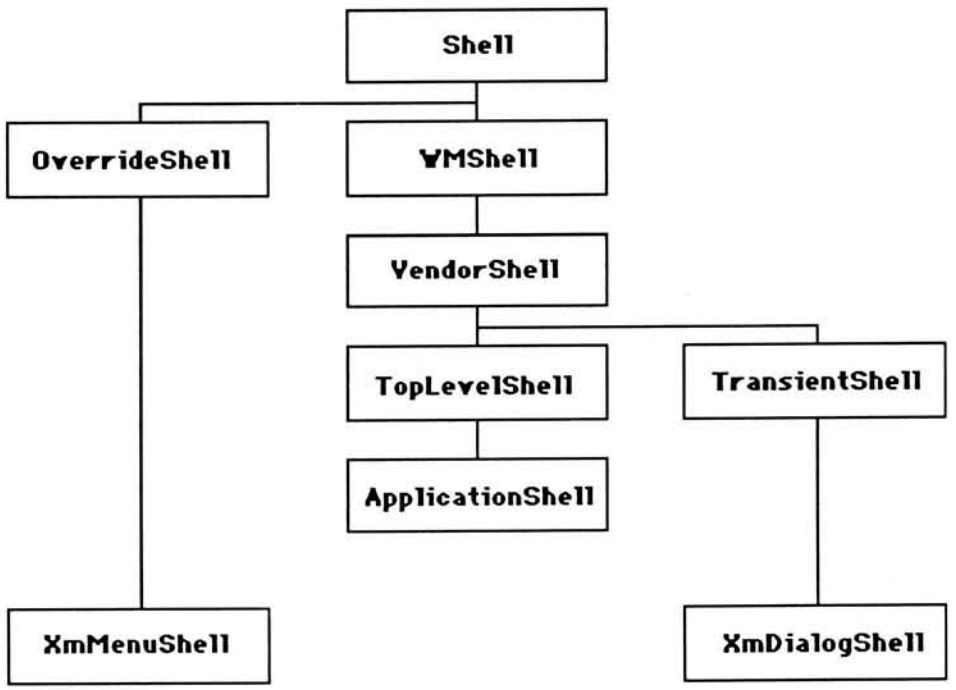
Cada objecto pertence a uma classe, organizada hierarquicamente, de tal forma que objectos de classes inferiores recebem caracteristicas das classes superiores.



Core é a classe base (XtIntrinsics).
Composite - contentores para os outros objectos (número variável de filhos).
Constraint - contentores com restrições adicionais.
Os objectos Motif (incluindo os objectos fornecidos pelo Intrinsics) distribuem-se por cinco categorias: "Shell", "Display", "Container", "Dialog" e "Gadgets".

Objectos Shell

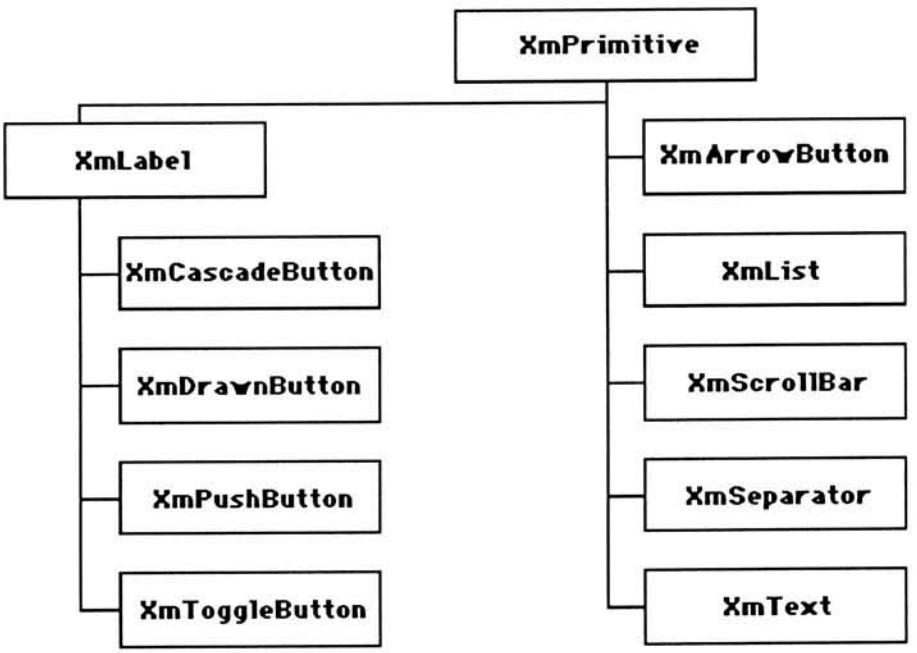
Objectos de alto nível que se encarregam da interface necessária entre as várias janelas de uma aplicação e o gestor de janelas.



Shell- classe base para os objectos Shell (tem um único filho).
OverrideShell- destinada às aplicações que ignoram os gestores de janelas.
WMShell- contem os recursos necessários ao protocolo de comunicação com o gestor de janelas.
VendorShell- contem recursos específicos para a interface com os gestores de janelas de um determinado fabricante.
TransientShell- classe utilizada para janelas manipuladas pelo gestor de janelas mas que não podem ser iconificadas individualmente.
TopLevelShell- utilizada para as janelas base.
ApplicationShell- usadas para as janelas base de aplicações.
XmDialogShell- instâncias deste objecto são utilizadas para janelas temporárias das aplicações (janelas de diálogo).
XmMenuShell- esta classe é utilizada como contentor de menus.

Objectos de apresentação ("Display Widgets")

Grupo de objectos que compoem a interface visual.



XmPrimitive- esta classe proporciona um conjunto extra de recursos aos outros objectos, como desenho da moldura, características de selecção, etc...

XmLabel- permite colocar uma etiqueta (texto ou gráficos) numa determinada zona da janela. O texto pode ser multidireccional, multilinha e multifonte.

XmCascadeButton- quando activado faz aparecer um menu.

XmDrawnButton- consiste num botão vazio com sombra onde o programador pode colocar um desenho. Quando seleccionado reage da mesma forma que o anterior.

XmPushButton- este objecto simula um botão rodeado por uma moldura sombreada. Quando é seleccionado a sombra é alterada para imitar o carregar do botão. A selecção deste objecto desencadeia uma acção que é determinada pelo programador.

XmToggleButton- semelhante ao XmPushButton na maneira de ser activado, corresponde no entanto à noção de interruptor lógico (0 ou 1) podendo existir nos dois estados.

XmArrowButton- consiste numa seta direccional com sombra. Quando seleccionado a sombra é alterada para dar a sensação de ter sido carregado. A direcção da seta é indicada pelo programador.

XmList- consiste num objecto que permite ao utilizador fazer uma selecção entre várias opções (sob a forma de texto). Eventualmente pode ser utilizada conjuntamente com um elevador ("ScrollBar") para visualizar mais opções.

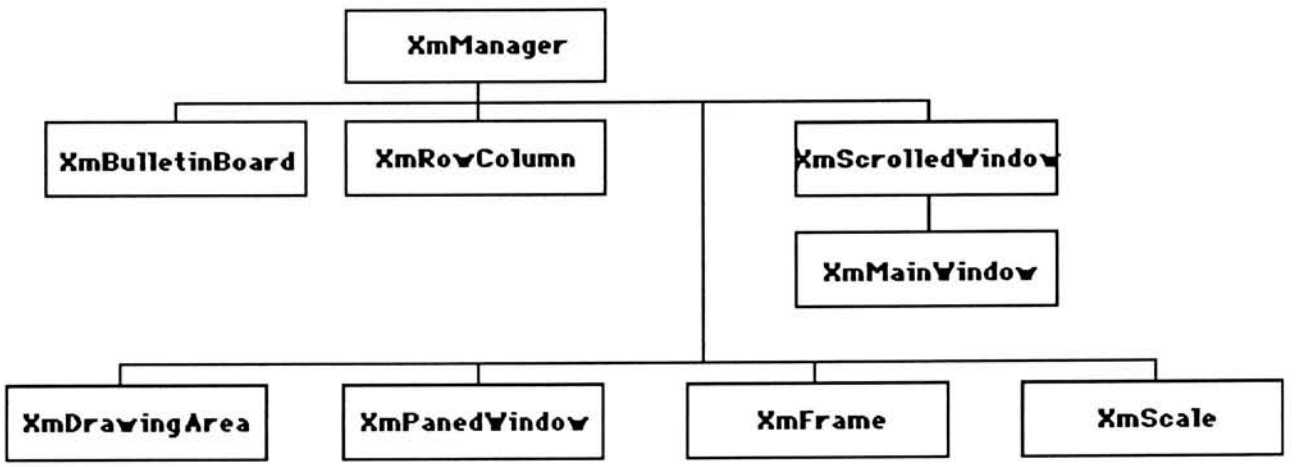
XmScrollBar- permite visualizar áreas que estão temporáriamente escondidas fazendo deslocar elevadores que podem estar colocados horizontalmente ou verticalmente.

XmSeparator- estabelece a separação entre vários ítems de um menu. Pode ser desenhado horizontalmente ou verticalmente e em vários estilos de linha.

XmText- proporciona um editor de texto com uma ou várias linhas. Pode ser utilizado para introdução de dados, visualização de documentos, etc...

Contentores ("Container Widgets")

Organizadores espaciais para os outros objectos:



XmManager- esta é uma Meta Classe no sentido de que existe apenas como classe de suporte para os outros objectos deste tipo. Proporciona mecanismos para os recursos visuais e contextos gráficos.

XmBulletinBoard- objecto composto, que proporciona regras geométricas simples de colocação de objectos filhos, evitando sobreposições.

XmRowColumn- objecto de uso geral, uma vez que pode ter como filho qualquer objecto de outro tipo. Permite organizar os objectos filhos em filas ou colunas, simetricamente ou não, controlando os espaçamentos e alinhamentos.

XmScrolledWindow- combina uma área visível (gráfica ou texto) com um ou dois elevadores, implementando uma janela de visualização de uma superfície maior. Essa janela de visualização pode ser deslocada com o auxílio dos elevadores.

XmMainWindow- proporciona um lay-out comum para as aplicações. Essa apresentação consiste numa barra de menus horizontal (XmMenuBar), numa janela de comandos (XmCommandWindow), uma zona de trabalho (work area) e elevadores (XmScrollBar).

XmDrawingArea- consiste num objecto vazio adaptável facilmente a varios objectivos. É um objecto importante utilizado para saídas gráficas.

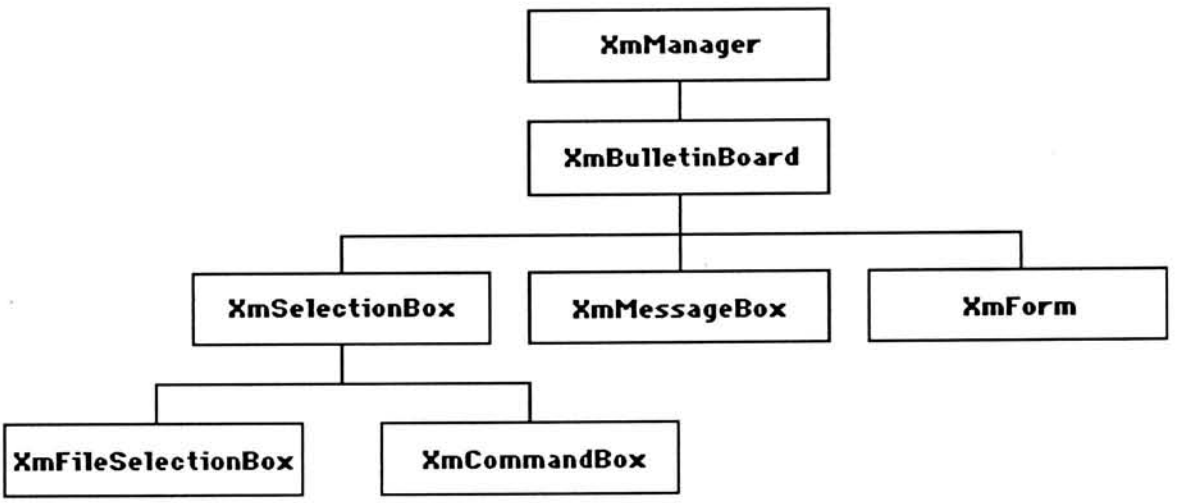
XmPanedWindow- objecto composto, que dispõe os objectos filhos verticalmente. A dimensão do objecto pai é ajustada automaticamente pelo número de objectos filhos.

XmFrame- é um objecto extremamente simples que funciona unicamente como uma moldura sombreada que engloba um outro objecto. É utilizado com os objectos que não tem esse recurso intrínseco.

XmScale- objecto com duas funções essenciais: usado pela aplicação para indicar um valor de uma determinada gama e permite que o utilizador introduza ou altere um valor dessa mesma gama. Esta selecção é feita ajustando um marcador ao longo de uma linha (simula um potenciómetro). Podem ser adicionados a este objecto valores da escala ajustáveis, texto, valores de máximo e minímo, etc...

Objectos de diálogo ("Dialog Widgets")

Objectos com um tempo de vida curto. Utilizados para tarefas de interacção, como a apresentação de mensagens, determinação de valores ou selecção de um conjunto de valores.



XmSelectionBox- permite fazer uma selecção de uma lista de alternativas (a lista pode deslizar com o auxílio de elevadores) ou de uma zona de edição.

XmMessageBox- permite fornecer informações ao utilizador. Inclui um símbolo gráfico que indica o tipo da informação (aviso, informação, pergunta, etc...).

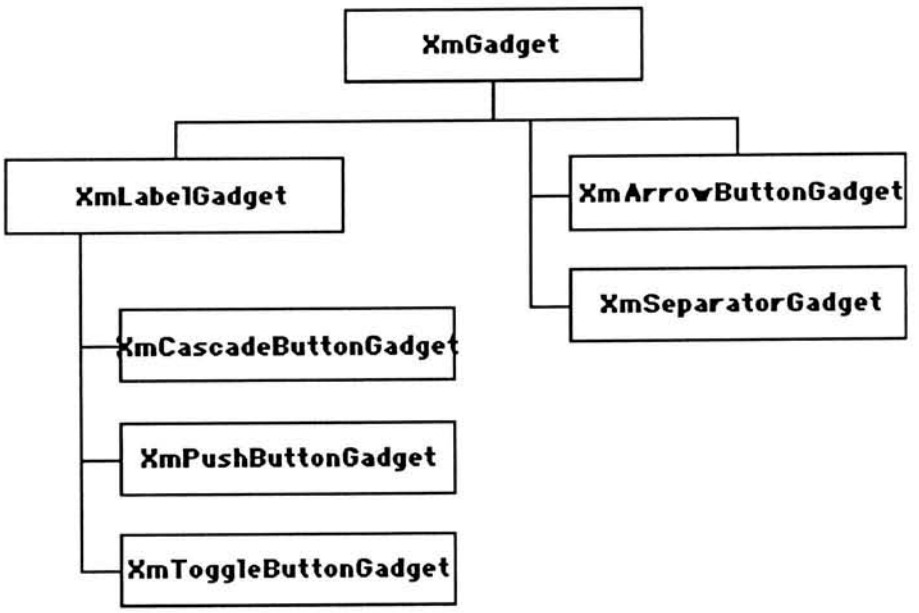
XmForm- é, tal como o XmBulletinBoard, um Container Object, uma vez que agrupa objectos filhos estabelecendo relações espaciais relativas entre eles.

XmFileSelectionBox- é um objecto relativamente complexo, permitindo seleccionar um ficheiro de uma lista. Apresenta ainda uma lista de directórios, uma janela de comando para edição do filtro de procura de ficheiros e outra para edição do nome do ficheiro.

XmCommand- é composto por duas áreas distintas: uma de historial de comandos (lista de comandos executados anteriormente), outra de introdução de novos comandos.

Gadgets

Objectos que, embora fornecendo a mesma funcionalidade dos objectos de visualização, melhoram a performance da aplicação em termos de rapidez de execução e de ocupação de memória, à custa da não criação de uma janela para cada objecto. Isto implica um menor número de possibilidades de manipulação destes objectos, por vezes compensada pelos ganhos em eficiência.



XmGadget- é uma meta classe, não instanciável, que funciona como suporte das outras, fornecendo-lhes recursos.
XmPushButtonGadget- tem, sensivelmente, as mesmas funcionalidades do XmPushButton.
XmArrowButtonGadget- funciona como o XmPushButtonGadget, embora apresentando o desenho de uma seta no botão em vez de texto.
XmLabelGadget- tem, sensivelmente, as mesmas funcionalidades do XmLabel.
XmSeparatorGadget- tem, sensivelmente, as mesmas funcionalidades do XmSeparator.
XmToggleButtonGadget- tem, sensivelmente, as mesmas funcionalidades do XmToggleButton.

Biblioteca de funções Motif

Dedicadas ao tratamento dos objectos Motif.
As principais são funções de conveniência para criar os objectos (começam por "XmCreate<objecto>"), alterar atributos desses objectos e para estabelecer comunicação com o gestor de janelas ou com outras aplicações.

Criação de objectos:
XmCreate<**WidgetName**> (...);
ex: XmCreateFileSelectionBox(...);
XmCreateMenuBar(...);

Manipulação de Strings:
XmString- estrutura de texto, contendo vários segmentos (frases), fonte e direcção utilizada.
ex: XmStringCopy (...)
XmStringDraw (...)

Funções gerais:
Manipulação do clipboard.
Manipulação de objectos.
Manipulação de pixmaps e imagens.
Manipulação de fontes.
Manipulação de listas.

2º Motif User Interface Language (UIL)

UIL ("User Interface Language") é uma linguagem de especificação para descrever o estado inicial de uma interface de uma aplicação Motif.

Descreve os objectos (por exemplo, menus, caixas de diálogo, "labels", etc...) usados na interface e especifica as funções chamadas por acção do utilizador.

O processo de criação de uma interface com o utilizador, usando este método é:

- Especificação da interface num módulo UIL.
- Compilação para gerar um ficheiro UID ("User Interface Definition").
- Dentro da aplicação, utilizam-se funções MRM ("Motif Resource Manager") para abrir o módulo UID e aceder às definições da interface.

Utilizando UIL, podem-se especificar os seguintes parâmetros:

- Objectos ("widgets" e "gadgets") que, no seu conjunto, compoem a interface.
- Atributos dos objectos especificados.
- Funções chamadas por actuação nos objectos ("Callback functions").
- A árvore de objectos da aplicação.

Vantagens na utilização de UIL e MRM:

- Facilidade de codificação da interface, uma vez que não é necessário conhecer as funções especifícas de criação de objectos, bastando mencionar quais os atributos que se querem alterar.
- Detecção de erros, uma vez que o compilador de UIL faz uma verificação prévia de problemas de construção de interfaces, como tipo errado de valores para argumentos, argumentos não suportados por um tipo particular de objecto, objectos filhos do tipo errado, etc...
- Separação entre funções da aplicação e funções de interface, uma vez que a interface é definida e compilada separadamente num módulo UIL. Isto torna possível, por exemplo, criar várias interfaces para o mesmo conjunto de funcionalidades (caso de criação de uma aplicação para vários países) ou alterar o aspecto da interface sem recompilar a aplicação.
- Prototipagem rápida, uma vez que se simplificam os processos da criação de interfaces. Isto permite, por exemplo, que os desenhadores da interface comuniquem mais com os utilizadores finais da aplicação. Por outro lado, como os programadores da aplicação trabalham mais ou menos independentemente dos desenhadores
da interface, o resultado final, ou seja, a aplicação
completa, fica concluída mais depressa.

A síntaxe da linguagem UIL, é semelhante à de uma linguagem de programação por objectos, uma vez que se definem blocos estruturados de argumentos, atributos e funções respeitantes a um objecto da interface.

MINI-SISTEMA DE MENUS

INTRODUÇÃO

Observação

Em toda a progrmação adoptou-se a notação seguinte:

todas as funções criadas levam um prefixo Xj.

Assim, o Menu tem o nome XjMenuWidget, e a Window tem XjWindowWidget.

O sistema é constituido, a nível do X Window, de dois Widgets:

- O Menu que corresponde ao conjunto de butões de um "Pull-Down" menu.
- O Window que é o Widget que contém cada um dos menus e pode egualemente conter outra coisa qualquer.

A listagem desses dois Widgets aparece em Anexo, assim como as rotinas elativas aos Pixmaps.

WINDOW

Um widget Window pode ser utilizado como widget "Parent" por qualquer widget do X Toolkit.
Ele tem toda a funçionalidade de um "Shell Widget".

Para preservar a independencia com o Window Manager, não existe decorações nestas janlas. porque as decorações são diferentes de um WM par outro

Este widget foi o primeiro a ser criado porque ele serve de "top shell" para cada um dos menus. Os menus são filhos de uma Window.

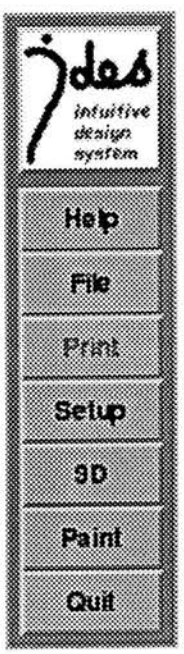
Uma Window pode ser do "tipo" fixa ou do tipo "movimentável".

As windows do tipo movimentável possuiem um botão no cabeçalho que o utilizador pode premir para movimenar a janela.

Esse botão comtem igualemente o nome da janela.

As windows do tipo fixa, como seu nome indica, não podem ser movimentados.

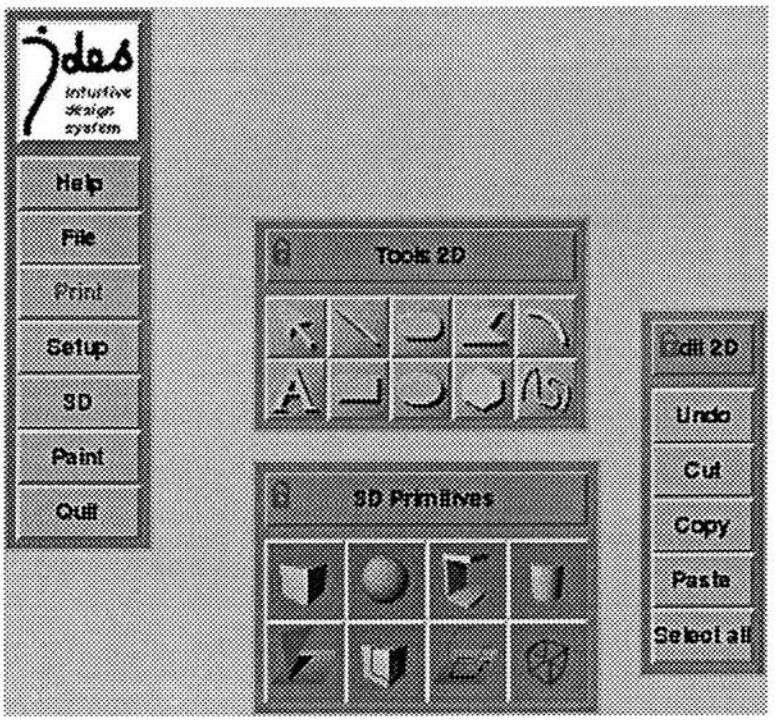
Estes menus fixos podem ter, em vez do botão de movimentação, um gráfico que é criado atraves das rotinas de manipulação de pixmaps. Pode-se assim, por exemplo, criar um logotipo para o program e colocar-lho nessa zona.



Uma Window pode estar bloqueada

Para bloquear uma janela, basta movimentar-lha. Aparece então um pequeno cadeado no lado esquerdo do botão de movimentação indicando que a janela não pode deaparecer.

Para desbloquear uma janela, é necessário premir com o rato sobre o cadeado. Nesse momento, a janela fecha-se (desaparece).



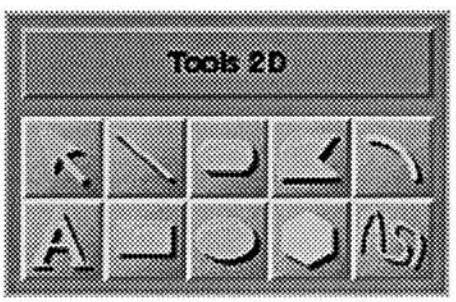
MENU

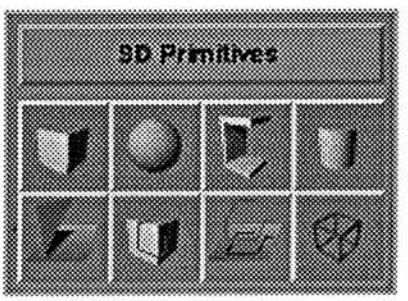
Os Menus são do tipo "PullDown Menus" do X Toolkit.

Um botão de menu pode ser texto ou gráfico (Pixmap)

Um menu deve, no entanto, ser constituido unicamente por butões de texto ou botões gráficos. Não é possível misturar os dois tipos.

Foi necessário implementar umas rotinas de manipulação de pixmaps Essas rotinas permitem uma inicialização rápida e fácil dos pixmaps para o programador.





Um botão pode estar activo ou inactivo.

No caso em que ele está activo, a string aparece a preto.

No caso em que ele está inactivo, a string aparece numa cor acinzentada para dar a idea de que ele não está acessível.

Organização horizontal ou vertical dos botões

Os botões estão organizados em linhas-colunas.

O programador pode colocar os botões como quizer:

ex: 2 linhas por 4 colunas ...

Importante: A única restrição é que o número de botões tem de ser igual ao produto Linhas x Colunas.

Os menus, herdam as características da janela onde eles se encaixam.

Os menus sem botão de movimentação são utilizados principalemente para criar menus principais.

Um menu pode ficar bloqueado. Ao premir um botão, o menu não desaparece.

PIXMAPS

Foram implementadas duas rotinas para a criação e instalação de pixmaps para os botões dos menus e para o "header" dos menus principais.

Essas rotinas podem ser chamadas atraves de dois "defines" XPM2PIXMAP_CMAP e XPM2PIXMAP para uma util o programador.

A duas lêem as imagens a partir de ficheiro (formato Xpm ou Bmp) e instalam os pixmaps.

A primeir instala a ColorMap que vem no ficheiro na palete ativa.

LISTAGENS

WINDOW

WindowP.h

```
#ifndef _WindowP_h
#define _WindowP_h

#include <X11/Xlib.h>
#include <X11/CompositeP.h>
#include "Window.h"
#include "Pixmap.h"

typedef struct {
    XtCallbackList lockProc;
    XtCallbackList closeProc;
    XtCallbackList restoreProc;
    Widgetshell, toplevel, header, top, lock;
    Widgetformwidget;
    Booleanlockable;
    Booleanlocked;
    Booleanno_top;
    Booleancloseparentonrealize;
    PixmapPlockpixmap, headerpixmap;
    WindowWidgettoclose;
} WindowPart;

typedef struct _WindowRec {
    CorePart core;
    CompositePart composite;
    WindowPart window;
} WindowRec;

typedef struct _WindowClass {
    int make_compiler_happy;
} WindowClassPart;

typedef struct _WindowClassRec {
    CoreClassPart core_class;
    CompositeClassPart composite_class;
    WindowClassPart window_class;
} WindowClassRec;

extern WindowClassRec windowClassRec;

#endif /* _WindowP_h */
```

Window.h

```
#ifndef _Window_h
#define _Window_h

#include <X11/Composite.h>

typedef struct _WindowRec *WindowWidget;
```

```
typedef struct _WindowClassRec *WindowWidgetClass;


#define XjNlockProc"lockproc"
#define XjNcloseProc"changeproc"
#define XjNarmCallback"armproc"
#define XjNlockable"lockable"
#define XjNlocked"locked"
#define XjNnoTop"notop"
#define XjNtoClose"toclose"
#define XjNcloseParentOnRealize"closeparentonrealize"
#define XjNlockPixmap"lockpixmap"
#define XjNheaderPixmap"headerpixmap"


extern WidgetClass windowWidgetClass;
extern Widget XjWindowAddChild(
#if NeedFunctionPrototypes
        Widget, char *, WidgetClass, Arg *, int
#endif
);

extern void XjCloseParentWindow(
#if NeedFunctionPrototypes
        Widge
#endif
);
#endif /* _Window_h */
```

Window.c

```
#include <stdio.h>
#include <math.h>
#include <X11/Xos.h>
#include <X11/cursorfont.h>
#include <X11/IntrinsicP.h>
#include <X11/Xresource.h>
#include <X11/CompositeP.h>
#include <X11/StringDefs.h>
#include <X11/Composite.h>
#include <Xm/Form.h>
#include <Xm/Frame.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/ToggleB.h>
#include <Xm/CascadeB.h>
#include <X11/ShellP.h>
#include <X11/Shell.h>

#include "menudefs.h"
#include "WindowP.h"
#include "cadena.xbm"


#define Offset(field)XtOffset(WindowWidget, field)
```

```
#define TOP_HEIGHT30
#define SHADOW_THICKNESS2
#define SHADOWS(2 * SHADOW_THICKNESS)
#define WINDOW_BORDER(3+SHADOWS)
#define WINDOW_BORDERS(2*WINDOW_BORDER)

Cursor cursor;
int n;
Arg wargs[256];

static Dimension        defHeaderWidth = 50;
static int          defParentButton = 0;


static XtResource resources[] = {
    {XtNx, XtCPosition, XtRPosition, sizeof(Position),
        Offset(core.x), XtRImmediate, (XtPointer)0},
    {XtNy, XtCPosition, XtRPosition, sizeof(Position),
        Offset(core.y), XtRImmediate, (XtPointer)0},
    {XtNwidth, XtCWidth, XtRDimension, sizeof(Dimension),
        Offset(core.width), XtRImmediate, (XtPointer)0},
    {XtNheight, XtCHeight, XtRDimension, sizeof(Dimension),
        Offset(core.height), XtRImmediate, (XtPointer)0},
    {XjNlockProc, XtCCallback, XtRCallback, sizeof(XtPointer),
        Offset(window.lockProc), XtRCallback, NULL},
    {XjNcloseProc, XtCCallback, XtRCallback, sizeof(XtPointer),
        Offset(window.closeProc), XtRCallback, NULL},
    {XjNtoClose, "ToClose", "ToClose", sizeof(XtPointer),
        Offset(window.toclose), "ToClose", (XtPointer) NULL},
    {XjNlocked, "Locked", "Locked", sizeof(Boolean),
        Offset(window.locked), "Locked", (XtPointer) False},
    {XjNlockable, "Lockable", "Lockable", sizeof(Boolean),
        Offset(window.lockable), "Lockable", (XtPointer) False},
    {XjNcloseParentOnRealize, "CloseParentOnRealize", "CloseParentOnRealize", sizeof(Boolean),
        Offset(window.closeparentonrealize), "CloseParentOnRealize", (XtPointer) False},
    {XjNnoTop, "NoTop", "NoTop", sizeof(Boolean),
        Offset(window.no_top), "NoTop", (XtPointer) False},
    {XjNlockPixmap, "LockPixmap", "LockPixmap", sizeof(XtPointer),
        Offset(window.lockpixmap), "LockPixmap", NULL},
    {XjNheaderPixmap, "HeaderPixmap", "HeaderPixmap", sizeof(XtPointer),
        Offset(window.headerpixmap), "HeaderPixmap", NULL},
};


void CloseChildWindow ();
void SetReadyToCloseWindow ();
static void Unlock ();
static void InitMoveWin ();
static void EndMoveWin ();
static void MoveWin ();
static Dimension getD();
static void ClassInitialize();
static void Initialize();
static void Realize();
static void InsertChild();
static void Redisplay();
static Boolean SetValues();
static XtGeometryResult GeometryManager();
```

```
Widget XjWindowAddChild();

static int must_close = 0;
static int can_move = 0;
static int off_x, off_y;
static int old_x, old_y, old_w, old_h;
static GC  gc;
static XSegmentoutline[4];
static Window Root_Window;
static Display *display;

static XtActionsRec actionTable[] = {
   {NULL, NULL},
};

WindowClassRec windowClassRec = {
   {
      (WidgetClass) &compositeClassRec,/* superclass */
      "Window",/* class_name */
      sizeof(WindowRec),/* size */
      NULL,/* class_initialize */
      NULL,/* class_part_initialize */
      FALSE,/* class_inited */
      Initialize,/* initialize */
      NULL,/* initialize_hook */
      Realize,/* realize */
      actionTable,/* actions */
      XtNumber(actionTable),/* num_actions */
      resources,/* resources */
      XtNumber(resources),/* resource_count */
      NULLQUARK,/* xrm_class */
      TRUE,/* compress_motion */
      TRUE,/* compress_exposure */
      TRUE,/* compress_enterleave   */
      FALSE,/* visible_interest */
      NULL,/* destroy */
      NULL,/* resize */
      XtInheritExpose,/* expose */
      SetValues,/* set_values */
      NULL,/* set_values_hook */
      XtInheritSetValuesAlmost,/* set_values_almost */
      NULL,/* get_values_hook */
      NULL,/* accept_focus */
      XtVersion,/* version */
      NULL,/* callback_private */
      NULL,/* tm_table */
      NULL,/* query_geometry */
      XtInheritDisplayAccelerator,/* display_accelerator */
      NULL/* extension */
   },{
/* composite_class fields */
      /* geometry_handler  */   GeometryManager,
      /* change_managed    */   XtInheritChangeManaged,
      /* insert_child */XtInheritInsertChild,
      /* delete_child */XtInheritDeleteChild,
      /* extension */   NULL
   },{
```

```
/* Window class fields */
        /* empty */0,
    }
};


WidgetClass windowWidgetClass = (WidgetClass)&windowClassRec;

static String argv0 = "Window";



static XtGeometryResult GeometryManager(w, request, reply)
        Widget w;
        XtWidgetGeometry *request;
        XtWidgetGeometry *reply;    /* RETURN */
{
    return XtGeometryYes;
}

static void Realize( cw, valueMask, attributes )
      WindowWidget cw;
      XtValueMask *valueMask;
      XSetWindowAttributes *attributes;
{
    if (cw->window.lockable == True) {
        cw->window.locked = False;
        XtUnmanageChild(cw->window.lock);
    }
    XtRealizeWidget(cw->window.shell);
    if (cw->window.top) {
        cursor = XCreateFontCursor(XtDisplay(cw->window.top), XC_fleur);
        XDefineCursor(XtDisplay(cw->window.top), XtWindow(cw->window.top), cursor);
    }
    XSelectInput(XtDisplay(cw->window.shell), XtWindow(cw->window.shell), ExposureMask | KeyPressMask);
    if (cw->window.closeparentonrealize && cw->window.toclose)
        SetReadyToCloseWindow(cw->window.toclose);
}

static Dimension getD(w, s)
          Widget w;
          String s;
{
Dimension result;
int v;
Arg arg[1];

    XtSetArg(arg[0],s,(XtArgVal) &result);
    XtGetValues(w, arg, 1);
    v = result;
    return(v);
}

static Boolean SetValues( current, request, desired )
      WindowWidget current,/* what I am */
                   request,/* what he wants me to be */
                   desired;/* what I will become */
{
```

```
int i;
    n=0;
    if (desired->core.x != current->core.x) {
        XtSetArg(wargs[n], XmNx, desired->core.x); n++;
    }
    if (desired->core.y != current->core.y) {
        XtSetArg(wargs[n], XmNy, desired->core.y); n++;
    }
    if (desired->core.width != current->core.width) {
        if (desired->window.header) {
          XtSetArg(wargs[n],XmNwidth,desired->core.width+WINDOW_BORDERS);n++;
            desired->core.width += WINDOW_BORDERS;
        } else {
            XtSetArg(wargs[n],XmNwidth,desired->core.width+WINDOW_BORDERS); n++;
            desired->core.width += WINDOW_BORDERS;
        }
    }
    if (desired->core.height != current->core.height) {
        XtSetArg(wargs[n], XmNheight, desired->core.height); n++;
    }
    XtSetValues(desired->window.shell, wargs, n);
    return(False);
}

static createWindowTop(cw)
WindowWidget cw;
{
Widget top, move;
    n=0;
    XtSetArg(wargs[n], XmNx, 0); n++;
    XtSetArg(wargs[n], XmNy, 0); n++;
    XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNheight, TOP_HEIGHT); n++;
    top = XtCreateManagedWidget("",xmFormWidgetClass,cw->window.toplevel,wargs,n);
    if (cw->window.lockable) {
        if (!cw->window.lockpixmap)
            cw->window.lockpixmap=XjCreateAllocatedPixmap(XtDisplay(cw),RootWindowOfScreen(XtScreen(cw)
        n=0;
        XtSetArg(wargs[n], XmNshadowThickness, 0); n++;
        XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
        XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
        XtSetArg(wargs[n], XmNx, (TOP_HEIGHT-SHADOWS-cadena_height)/2); n++;
        XtSetArg(wargs[n], XmNy, (TOP_HEIGHT-SHADOWS-cadena_height)/2); n++;
        XtSetArg(wargs[n], XmNlabelType, XmPIXMAP); n++;
        XtSetArg(wargs[n], XmNlabelPixmap, cw->window.lockpixmap->pixmap); n++;
        XtSetArg(wargs[n], XmNhighlightThickness, 0); n++;
        cw->window.lock = XtCreateWidget("",xmPushButtonWidgetClass,top,wargs,n);
        XtAddCallback(cw->window.lock, XmNactivateCallback, Unlock, cw);
    }
    n=0;
    XtSetArg(wargs[n], XmNtopShadowColor, 8); n++;
    XtSetArg(wargs[n], XmNbottomShadowColor, 9); n++;
    XtSetArg(wargs[n], XmNarmColor, 7); n++;
    XtSetArg(wargs[n], XmNbackground, 7); n++;
    XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
```

```
    XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNheight, TOP_HEIGHT); n++;
    XtSetArg(wargs[n], XmNhighlightThickness, 0); n++;
    XtSetArg(wargs[n], XmNlabelString, STRING(cw->core.name)); n++;
    move = XtCreateManagedWidget("",xmPushButtonWidgetClass,top,wargs,n);
    XtAddCallback(move, XmNarmCallback, InitMoveWin, cw);
    XtAddCallback(move, XmNdisarmCallback, EndMoveWin, cw);
    XtAddEventHandler(move,Button1MotionMask,False,MoveWin,cw);
    top->core.width = getD(move, XmNwidth);
    cw->window.top = top;
}

static createWindowHeader(cw)
WindowWidget cw;
{
Widget top, global, headerW;
Pixmap pixmap;
    if (cw->window.headerpixmap == (Pixmap) NULL) {
        cw->window.header = (Widget) NULL;
        return;
    }
    n=0;
    XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
    if (cw->window.top) {
        XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
        XtSetArg(wargs[n], XmNtopWidget, cw->window.top); n++;
    } else {
        XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_FORM); n++;
    }
    top = XtCreateManagedWidget("",xmFormWidgetClass,cw->window.toplevel,wargs,n);
    n=0;
    XtSetArg(wargs[n], XmNshadowType, XmSHADOW_OUT); n++;
    global = XtCreateManagedWidget("",xmFrameWidgetClass,top,wargs,n);
    n=0;
    XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
    XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
    XtSetArg(wargs[n], XmNlabelType, XmPIXMAP); n++;
    XtSetArg(wargs[n], XmNlabelPixmap, cw->window.headerpixmap->pixmap); n++;
    headerW = XtCreateManagedWidget("",xmLabelWidgetClass,global,wargs,n);top->core.height = getD(headerW
    top->core.width = getD(headerW, XmNwidth) + SHADOWS;
    cw->window.header = top;
}

static void Initialize( request, new )
      Widget request;/* what the client asked for */
      Widget new;/* what we're going to give him */
{
int i;
Dimension h=0;
XGCValues values;
    WindowWidget cw = (WindowWidget) new;
    n = 0;
    XtSetArg(wargs[n], XmNmwmFunctions, 0L); n++;
    XtSetArg(wargs[n], XmNmwmDecorations, 0L); n++;
    cw->window.shell=XtCreateApplicationShell(cw->core.name,topLevelShellWidgetClass,wargs,n);
    n = 0;
    XtSetArg(wargs[n], XmNshadowThickness, 2); n++;
    XtSetArg(wargs[n], XmNtopShadowColor, 8); n++;
```

```
    XtSetArg(wargs[n], XmNbottomShadowColor, 8); n++;
    XtSetArg(wargs[n], XmNbackground, 7); n++;
    XtSetArg(wargs[n], XmNhorizontalSpacing, WINDOW_BORDER); n++;
    XtSetArg(wargs[n], XmNverticalSpacing, WINDOW_BORDER); n++;
    cw->window.toplevel=XtCreateManagedWidget("",xmFormWidgetClass,cw->window.shell,wargs,n);
    cw->window.top = cw->window.header = (Widget) NULL;
    if (!cw->window.no_top)
        createWindowTop(cw);
    createWindowHeader(cw);
    h = WINDOW_BORDER;
    cw->core.width = 50 + SHADOWS+WINDOW_BORDERS;
    n = 0;
    XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
    if (cw->window.top) {
        h += WINDOW_BORDER;
        XtSetArg(wargs[1], XmNtopWidget, cw->window.top); n=2;
        h += getD(cw->window.top, XtNheight);
        cw->core.width = getD(cw->window.top, XtNwidth) + SHADOWS+WINDOW_BORDERS;
    }
    if (cw->window.header) {
        h += WINDOW_BORDER;
        XtSetArg(wargs[1], XmNtopWidget, cw->window.header); n=2;
        h += getD(cw->window.header, XtNheight);
        cw->core.width = getD(cw->window.header, XtNwidth) + WINDOW_BORDERS;
        cw->window.shell->core.width = cw->core.width;
    }
    XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
    cw->window.formwidget = XtCreateManagedWidget("", xmFormWidgetClass, cw->window.toplevel, wargs, n
    cw->core.height = h;

    values.function = GXxor;
    values.subwindow_mode = IncludeInferiors;
    values.foreground = 1;
    values.line_width = 2;
    gc =XCreateGC(XtDisplay(cw->window.shell),DefaultRootWindow(XtDisplay(cw->window.shell)),GCLineWic
}

static void Unlock (widget,cw,Data)
                Widget widget;
                WindowWidget cw;
                XmAnyCallbackStruct *Data;
{
    XtUnrealizeWidget(cw->window.shell);
    cw->window.locked = False;
    XtUnmanageChild(cw->window.lock);
    if (cw->window.toclose)
        CloseChildWindow(cw->window.toclose);
}

void CloseChildWindow (cw)
WindowWidget cw;
{
    if (!cw->window.locked)
        XtUnrealizeWidget(cw->window.shell);
    if (cw->window.toclose)
        CloseChildWindow(cw->window.toclose);
```

```
}

void MustCloseWindow (cw)
{
    if (!must_close)
        return;
    must_close = 0;

    CloseChildWindow(cw);

}

void SetReadyToCloseWindow (cw)
WindowWidget cw;
{
    must_close = 1;
}

#define AAA 2

static void InitMoveWin (widget,win,Data)
Widget widget;
            WindowWidget win;
XmAnyCallbackStruct *Data;
{
Window WRoot,WIn;
int PRx,PRy, PIx,PIy;
unsigned int unused;
    can_move = 1;
    (void)XQueryPointer(XtDisplay(win->window.shell),XtWindow(win->window.shell),&WRoot,&WIn,&PRx,&P
    off_x = PIx;
    off_y = PIy;
    old_x = win->window.shell->core.x;
    old_y = win->window.shell->core.y;
    old_w = getD(win->window.shell, XtNwidth)-AAA;
    old_h = getD(win->window.shell, XtNheight)-AAA;
    display = XtDisplay(widget);
    Root_Window = DefaultRootWindow(XtDisplay(win->window.shell));
    outline[0].x1 = outline[1].x1 = old_x+2;
    outline[0].x2 = outline[1].x2 = old_x-1+old_w;
    outline[0].y1 = outline[0].y2 = old_y+1;
    outline[1].y1 = outline[1].y2 = old_y+1+old_h;
    outline[3].y1 = outline[2].y1 = old_y;
    outline[3].y2 = outline[2].y2 = old_y+1+old_h;
    outline[3].x1 = outline[3].x2 = old_x+1;
    outline[2].x1 = outline[2].x2 = old_x+1+old_w;
    XDrawSegments(display,Root_Window,gc, outline, 4);
}

static void MoveWin (widget,win,event,continue_to_dispatch )
            Widget widget;
            WindowWidget win;
XMotionEvent *event;
            Boolean *continue_to_dispatch;
{
Window WRoot,WIn;
int PRx,PRy, PIx,PIy;
unsigned int unused;
```

```
int x, y;
    if (can_move) {
        x = event->x_root- off_x;
        y = event->y_root - off_y;
        old_x = x;
        old_y = y;
        XDrawSegments(display,Root_Window,gc, outline, 4);
        outline[0].x1 = outline[1].x1 = old_x+2;
        outline[0].x2 = outline[1].x2 = old_x-1+old_w;
        outline[0].y1 = outline[0].y2 = old_y+1;
        outline[1].y1 = outline[1].y2 = old_y+1+old_h;
        outline[3].y1 = outline[2].y1 = old_y;
        outline[3].y2 = outline[2].y2 = old_y+1+old_h;
        outline[3].x1 = outline[3].x2 = old_x+1;
        outline[2].x1 = outline[2].x2 = old_x+1+old_w;
        XDrawSegments(display,Root_Window,gc, outline, 4);
    }
    *continue_to_dispatch = 1;  /* tells whether to call other dispatchers */

}

static void EndMoveWin (widget,win,Data)
                Widget widget;
                WindowWidget win;
                XmAnyCallbackStruct *Data;
{
    XDrawSegments(display,Root_Window,gc, outline, 4);
    XMoveWindow(XtDisplay(win->window.shell),XtWindow(win->window.shell), old_x, old_y);
    win->core.x = win->window.shell->core.x = old_x;
    win->core.y = win->window.shell->core.y = old_y;
    if (win->window.lockable == True) {
        win->window.locked = True;
        XtManageChild(win->window.lock);
        cursor = XCreateFontCursor(XtDisplay(win->window.lock), XC_pirate);
        XDefineCursor(XtDisplay(win->window.lock),XtWindow(win->window.lock),cursor);
    }
    can_move = 0;
    if (win->window.toclose)
        CloseChildWindow(win->window.toclose);
}

Widget XjWindowAddChild(parent, name, widget_class, wargs, n)
Widget parent;
char *name;
WidgetClass widget_class;
Arg *wargs;
int n;
{
Widget W, result;
WindowWidget cw = (WindowWidget)parent;
    W = cw->window.formwidget;
    result = XtCreateManagedWidget(name, widget_class, W, wargs, n);
    return result;
}

void XjCloseParentWindow(form)
                Widget form;
{ XtUnrealizeWidget(XtParent(XtParent(form))); }
```

MENU

MenuP.h

```
#ifndef _MenuP_h
#define _MenuP_h
#include <X11/Xlib.h>
#include <X11/CompositeP.h>
#include "WindowP.h"
#include "Menu.h"
#include "Pixmap.h"

typedef struct MenuProcStructure {
                FunctionP     *function;
                XtPointer     *data;
                } MenuProcStructureDec;
typedef struct MenuProcStructure MenuProcS;

typedef struct {
    intarmbutton;
    Widgetwindow;
    Widgettoplevel;
    WindowWidgetarmsubmenu;
    Widgetbuttons;
    Widget*buttonslist;
    MenuWidgetparentmenu;
    intparentbutton;
    Booleantopmenu;
    Dimensionnbuttons;
    Dimensionncolumns;
    WindowWidget*submenulist;
    Dimensionbuttontype;
    Dimensionbuttonwidth;
    Dimensionbuttonheight;
    PixmapPListpixmaplist;
    PixmapPListarmpixmaplist;
    StringListlabellist;
    Position*ylist;
    MenuProcS*armcallbacks;
    MenuProcS*disarmcallbacks;
    MenuProcS*activatecallbacks;
    intnumcallbacks;
    ProcS*proclist;
} MenuPart;

typedef struct _MenuRec {
    CorePart core;
    MenuPart menu;
    WindowPart window;
} MenuRec;

typedef struct _MenuClass {
    int make_compiler_happy;
} MenuClassPart;
```

```
typedef struct _MenuClassRec {
    CoreClassPart core_class;
    MenuClassPart menu_class;
    WindowClassPart window_class;
} MenuClassRec;

#endif /* _MenuP_h */
```

Menu.h

```
#ifndef _Menu_h
#define _Menu_h

#include "Window.h"
#include "menudefs.h"

#ifndef PROC_DEF
#define PROC_DEF
typedef struct ProcStructure {
inttype;
intbutton;
FunctionP*function;
XtPointer*data;
} ProcStructureDec;
typedef struct ProcStructure ProcS;
#endif /* PROC_DEF */

typedef struct _MenuRec *MenuWidget;

typedef struct _MenuClassRec *MenuWidgetClass;

#define XjARM_CALLBACK1
#define XjDISARM_CALLBACK2
#define XjACTIVATE_CALLBACK3
#define XjNlockProc"lockproc"
#define XjNunlockProc"unlockproc"
#define XjNcloseProc"changeproc"
#define XjNactivateProc"activateproc"
#define XjNarmCallback"armproc"
#define XjNbuttonsList"buttonslist"
#define XjNarmButton"armbutton"
#define XjNarmSubmenu"armsubmenu"
#define XjNparent"parent"
#define XjNtopMenu"topmenu"
#define XjNlockable"lockable"
#define XjNparentMenu"parentmenu"
#define XjNparentButton"parentbutton"
#define XjNlockPixmap"lockpixmap"
#define XjNunlockPixmap"unlockpixmap"
#define XjNheaderPixmap"headerpixmap"
#define XjNnColumns"ncolumns"
#define XjNnButtons"nbuttons"
#define XjNbuttonType"buttontype"
```

```
#define XjNbuttonWidth"buttonwidth"
#define XjNbuttonHeight"buttonheight"
#define XjNpixmapList"pixmaplist"
#define XjNarmPixmapList "armpixmaplist"
#define XjNlabelList"labellist"
#define XjNnumCallbacks"numcallbacks"
#define XjNcallbacksList "callbackslist"
#define XjLABEL_TYPE1
#define XjPIXMAP_TYPE2

extern WidgetClass menuWidgetClass;

#endif /* _Menu_h */
```

Menu.c

```
#include <stdio.h>
#include <math.h>

#include <X11/Xos.h>
#include <X11/cursorfont.h>
#include <X11/IntrinsicP.h>
#include <X11/Xresource.h>
#include <X11/CompositeP.h>
#include <X11/StringDefs.h
#include <X11/Composite.h>
#include <Xm/Form.h>
#include <Xm/RowColumn.h>
#include <Xm/Frame.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/CascadeB.h>
#include <X11/ShellP.h>
#include <X11/Shell.h>

#include "Window.h"
#include "WindowP.h"
#include "menudefs.h"
#include "MenuP.h"
#include "cadena.xbm"

#define Offset(field)XtOffset(MenuWidget, field)
#define TOP_HEIGHT30
#define SHADOW_THICKNESS2
#define SHADOWS(2 * SHADOW_THICKNESS)
#define WINDOW_BORDER        (3+SHADOWS)
#define WINDOW_BORDERS       (2*WINDOW_BORDER)

static Dimension defHeaderWidth = 200;
static Dimension defButtonWidth = 200;
static Dimension defButtonHeight = 30;
static Dimension defLabelType = XjLABEL_TYPE;
static Dimension defNColumns = 1;
static Dimension defNButtons = 0;
static Dimension defNCallbacks = 0;
static int     defParentButton = 0;
static ProcS ProcList;
int n;
Arg wargs[256];

static XtResource resources[] = {
   {XjNlockProc, XtCCallback, XtRCallback, sizeof(XtPointer),
        Offset(window.lockProc), XtRCallback, NULL},
   {XjNcloseProc, XtCCallback, XtRCallback, sizeof(XtPointer),
        Offset(window.closeProc), XtRCallback, NULL},
   {XNlockable, "Lockable", "Lockable", sizeof(Boolean),
        Offset(window.lockable), "Lockable", (XtPointer) False},
   {XjNnoTop, "NoTop", "NoTop", sizeof(Boolean),
        Offset(window.no_top), "NoTop", (XtPointer) False},
   {XjNlockPixmap, "LockPixmap", "LockPixmap", sizeof(XtPointer),
```

```
        Offset(window.lockpixmap), "LockPixmap", NULL},
    {XjNheaderPixmap, "HeaderPixmap", "HeaderPixmap", sizeof(XtPointer),
        Offset(window.headerpixmap), "HeaderPixmap", NULL},
    {XjNbuttonsList, "ButtonsList", "ButtonsList", sizeof(XtPointer),
        Offset(menu.buttonslist), "ButtonsList", NULL},
    {XjNarmButton, "ArmButton", "ArmButton", sizeof(XtPointer),
        Offset(menu.armbutton), "ArmButton", (XtPointer) &defParentButton},
    {XjNarmSubmenu, "ArmSubmenu", "ArmSubmenu", sizeof(XtPointer),
        Offset(menu.armsubmenu), "ArmSubmenu", (XtPointer) NULL},
    {XjNparentMenu, "ParentMenu", "ParentMenu", sizeof(XtPointer),
        Offset(menu.parentmenu), "ParentMenu", NULL},
    {XjNparentButton, "ParentButton", "ParentButton", sizeof(XtPointer),
        Offset(menu.parentbutton), "ParentButton", (XtPointer) &defParentButton},
    {XjNtopMenu, "TopMenu", "TopMenu", sizeof(Boolean),
        Offset(menu.topmenu), "TopMenu", (XtPointer) False},
    {XjNnButtons, "NButtons", "NButtons", sizeof(Dimension),
        Offset(menu.nbuttons), "NButtons", (XtPointer) &defNButtons},
    {XjNnColumns, "NColumns", "NColumns", sizeof(Dimension),
        Offset(menu.ncolumns), "NColumns", (XtPointer) &defNColumns},
    {XjNbuttonType, "ButtonType", "ButtonType", sizeof(Dimension),
        Offset(menu.buttontype), "ButtonType", (XtPointer) &defLabelType},
    {XjNbuttonWidth, "ButtonWidth", "ButtonWidth", sizeof(Dimension),
        Offset(menu.buttonwidth), "ButtonWidth", (XtPointer) &defButtonWidth},
    {XjNbuttonHeight, "ButtonHeight", "ButtonHeight", sizeof(Dimension),
        Offset(menu.buttonheight), "ButtonHeight", (XtPointer) &defButtonHeight},
    {XjNpixmapList, "PixmapList", "PixmapList", sizeof(XtPointer)
        Offset(menu.pixmaplist), "PixmapList", NULL},
    {XjNarmPixmapList, "ArmPixmapList", "ArmPixmapList", sizeof(XtPointer),
        Offset(menu.armpixmaplist), "ArmPixmapList", NULL},
    {XjNlabelList, "LabelList", "LabelList", sizeof(XtPointer),
        Offset(menu.labellist), "LabelList", NULL},
    {XjNnumCallbacks, "NumCallbacks", "NumCallbacks", sizeof(int),
        Offset(menu.numcallbacks), "NumCallbacks", (XtPointer) &defNCallbacks},
    {XjNcallbacksList, "CallbacksList", "CallbacksList", sizeof(XtPointer),
        Offset(menu.proclist), "CallbacksList", (XtPointer) NULL},
};

static int activated = 0;
Cursor cursor_button = 0;
Cursor cursor_submenu = 0;

static XtActionsRec actionTable[] = {
    {NULL, NULL},
};

MenuClassRec menuClassRec = {
    {
        (WidgetClass) &widgetClassRec,/* superclass */
        "Menu",/* class_name */
        sizeof(MenuRec),/* size */
        NULL,/* class_initialize */
        NULL,/* class_part_initialize */
        FALSE,/* class_inited */
        Initialize,/* initialize */
        NULL,/* initialize_hook */
        Realize,/* realize */
        actionTable,/* actions */
        XtNumber(actionTable),/* num_actions */
```

```
        resources,/* resources */
        XtNumber(resources),/* resource_count */
        NULLQUARK,/* xrm_class */
        TRUE,/* compress_motion */
        TRUE,/* compress_exposure */
        TRUE,/* compress_enterleave   */
        FALSE,/* visible_interest */
        NULL,/* destroy */
        NULL,/* resize */
        XtInheritExpose,/* expose */
        SetValues,/* set_values */
        NULL,/* set_values_hook */
        XtInheritSetValuesAlmost,/* set_values_almost */
        NULL,/* get_values_hook */
        NULL,/* accept_focus */
        XtVersion,/* version */
        NULL,/* callback_private */
        NULL,/* tm_table */
        NULL,/* query_geometry */
        XtInheritDisplayAccelerator,/* display_accelerator */
        NULL/* extension */
    },{
/* Menu class fields */
        /* empty */0,
    },{
/* Window class fields */
        /* empty */0,
    }
};

WidgetClass menuWidgetClass = (WidgetClass)&menuClassRec;
static String argv0 = "Menu";

static XtGeometryResult GeometryManager(w, request, reply)
        Widget w;
        XtWidgetGeometry *request;
        XtWidgetGeometry *reply;    /* RETURN */
{
    return XtGeometryYes;
}

static void Realize( cw, valueMask, attributes )
      MenuWidget cw;
      XtValueMask *valueMask;
      XSetWindowAttributes *attributes;
{
int i;
Position x, y;
char str[256];
MenuWidget pmenu;
    pmenu = (MenuWidget)cw->menu.parentmenu;
    if (cw->menu.topmenu == True)
        x = y = 0;
    else {
        getXYbutton(pmenu, cw->menu.parentbutton, &x, &y);
        n = 0;
        XtSetArg(wargs[n], XmNx, x); n++;
        XtSetArg(wargs[n], XmNy, y); n++;
```

```
        XtSetValues(cw, wargs, n);
    }
    sprintf(str, "+%d+%d", x, y);
    n = 0;
    XtSetArg(wargs[n], XmNx, x); n++;
    XtSetArg(wargs[n], XmNy, y); n++;
    XtSetArg(wargs[n], XtNgeometry, str); n++;
    XtSetValues(((WindowWidget)cw->menu.window)->window.shell, wargs, n);
    n = 0;
    XtSetArg(wargs[n], XmNx, x); n++;
    XtSetArg(wargs[n], XmNy, y); n++;
    XtSetValues((WindowWidget)cw->menu.window, wargs, n);
    XtRealizeWidget (cw->menu.window);
    for (i=0; i<cw->menu.nbuttons; i++) {
        cursor_button = XCreateFontCursor(XtDisplay(cw->menu.buttonslist[i]), XC_hand2);
        cursor_submenu = XCreateFontCursor(XtDisplay(cw->menu.buttonslist[i]), XC_sb_right_arrow);
        if (cw->menu.submenulist[i])
            XDefineCursor(XtDisplay(cw->menu.buttonslist[i]), XtWindow(cw->menu.buttonslist[i]), cursor_submer
        else
            XDefineCursor(XtDisplay(cw->menu.buttonslist[i]), XtWindow(cw->menu.buttonslist[i]), cursor_button)
    }
}

static void getXYbutton(cw, button, x, y)
MenuWidget cw;
int button;
Position *x, *y;
{
    *x = getD(cw->menu.window,XmNx) + getD(cw->menu.window,XmNwidth);
    *y = getD(cw->menu.window,XmNy) +
            getD(cw->menu.window,XmNheight) +
            cw->menu.ylist[button];
}

static Dimension getD(w, s)
Widget w;
String s;
{
Dimension result;
int v;
Arg arg[1];
    XtSetArg(arg[0],s,(XtArgVal) &result);
    XtGetValues(w, arg, 1);
    v = result;
    return(v);
}

static Boolean SetValues( current, request, desired )
     MenuWidget current,/* what I am */
                request,/* what he wants me to be */
                desired;/* what I will become */
{
int i;
    if (desired->menu.armsubmenu != current->menu.armsubmenu) {
        desired->menu.submenulist[desired->menu.armbutton] = desired->menu.armsubmenu;
        n = 0;
        XtSetArg(wargs[n], XjNtoClose, desired->menu.window); n++;
        XtSetValues(desired->menu.armsubmenu, wargs, n);
```

```
    }
    return(False);
}

static createMenuButtons(cw)
MenuWidget cw;
{
 Widget top, tempW;
 Widget *columns;
 int i, j;
 Dimension vert, horiz;
 Dimension h=0, w=0;
 Position offset_y=0;
   horiz = cw->menu.ncolumns;
      if (!cw->menu.nbuttons || (cw->menu.nbuttons % horiz)) return;
   vert = cw->menu.nbuttons / horiz;
   columns = (Widget*)XtMalloc(sizeof(Widget) * horiz);
   n=0;
   top = XtCreateWidget("",xmFormWidgetClass,cw->menu.toplevel,wargs,n);
   for (i=0; i<horiz; i++) {
      n=0;
         if (!i && (i == (horiz-1))) XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
      if (i) {
         XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_WIDGET); n++;
         XtSetArg(wargs[n], XmNleftWidget, columns[i-1]); n++;
      }
         if (i == (horiz-1)) XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
      XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_FORM); n++;
      XtSetArg(wargs[n], XmNbottomAttachment, XmATTACH_FORM); n++;
      columns[i] = XtCreateManagedWidget("",xmFormWidgetClass,top,wargs,n);
   }
   for (i=0; i<cw->menu.nbuttons; i++) {
      j = i / vert;
      if (cw->menu.buttontype == XjPIXMAP_TYPE) {
         n=0;
         XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
         XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
         if (i-(j*vert)) {
            XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
            XtSetArg(wargs[n], XmNtopWidget, tempW); n++;
         }
         XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
         XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
         XtSetArg(wargs[n], XmNhighlightThickness, 0); n++;
         XtSetArg(wargs[n], XmNlabelType, XmPIXMAP); n++;
         XtSetArg(wargs[n], XmNlabelPixmap,cw->menu.pixmaplist[i]->pixmap); n++;
            if (cw->menu.armpixmaplist) XtSetArg(wargs[n],XmNarmPixmap, cw->menu.armpixmaplist[i]->pixı
         tempW = XtCreateManagedWidget("",xmPushButtonWidgetClass,columns[j],wargs,n);
         cw->menu.ylist[i] = offset_y + h;
         h += getD(tempW, XmNheight);
         if (getD(tempW, XmNwidth) > w)
            w = getD(tempW, XmNwidth);
      } else {
         n=0;
         XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
         XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
         XtSetArg(wargs[n], XmNheight, cw->menu.buttonheight); n++;
         if (i-(j*vert)) {
```

```
            XtSetArg(wargs[n], XmNtopAttachment, XmATTACH_WIDGET); n++;
            XtSetArg(wargs[n], XmNtopWidget, tempW); n++;
        }
        XtSetArg(wargs[n], XmNmarginHeight, 0); n++;
        XtSetArg(wargs[n], XmNmarginWidth, 0); n++;
        XtSetArg(wargs[n], XmNhighlightThickness, 0); n++;
        XtSetArg(wargs[n], XmNlabelType, XmSTRING); n++;
        XtSetArg(wargs[n], XmNlabelString, cw->menu.labellist[i]); n++;
        tempW = XtCreateManagedWidget("",xmPushButtonWidgetClass,columns[j],wargs,n);
        cw->menu.ylist[i] = offset_y + h;
        h += getD(tempW, XmNheight);
            if (getD(tempW, XmNwidth) > w) w = getD(tempW, XmNwidth);
    }
    cw->menu.buttonslist[i] = tempW;
    XtAddCallback(tempW, XmNarmCallback, (XtCallbackProc) ArmMenu, &cw->menu.submenulist[i]);
    XtAddCallback(tempW, XmNdisarmCallback, (XtCallbackProc) DisarmMenu, &cw->menu.submenulist[i]);
    XtAddCallback(tempW, XmNactivateCallback,(XtCallbackProc)ActivateMenu,cw);
    XtAddCallback(tempW, XmNarmCallback, (XtCallbackProc) MenuCall, &cw->menu.armcallbacks[i]);
    XtAddCallback(tempW, XmNdisarmCallback, (XtCallbackProc) MenuCall, &cw->menu.disarmcallbacks[i]);
    XtAddCallback(tempW, XmNactivateCallback, (XtCallbackProc) MenuCall, &cw->menu.activatecallbacks[i]
}
if ((getD(cw->menu.window, XmNwidth)-WINDOW_BORDERS) < (w*horiz)) {
    n = 0;
    XtSetArg(wargs[n], XtNwidth, w*horiz); n++;
    XtSetValues(cw->menu.window, wargs, n);
} else
    w = getD(cw->menu.window, XmNwidth) - WINDOW_BORDERS;
for (i=0; i<horiz; i++) {
    n=0;
    XtSetArg(wargs[n], XmNwidth, w); n++;
    XtSetValues(columns[i], wargs, n);
}
XtManageChild(top);
top->core.height = h;
top->core.width = w*horiz;
cw->menu.buttons = top;
}

static void Initialize( request, new )
    Widget request;/* what the client asked for */
    Widget new;/* what we're going to give him */
{
int i;
Dimension h=0;
WindowPart reg;
    MenuWidget cw = (MenuWidget) new;
    n=0;
    XtSetArg(wargs[n], XjNlockProc, &reg.lockProc); n++;
    XtSetArg(wargs[n], XjNcloseProc, &reg.closeProc); n++;
    XtSetArg(wargs[n], XjNlockable, &reg.lockable); n++;
    XtSetArg(wargs[n], XjNnoTop, &reg.no_top); n++;
    XtSetArg(wargs[n], XjNlockPixmap, &reg.lockpixmap); n++;
    XtSetArg(wargs[n], XjNheaderPixmap, &reg.headerpixmap); n++;
    XtGetValues(cw, wargs, n);
    n=0;
    XtSetArg(wargs[n], XjNlockProc, reg.lockProc); n++;
    XtSetArg(wargs[n], XjNcloseProc, reg.closeProc); n++;
    XtSetArg(wargs[n], XjNlockable, reg.lockable); n++;
```

```
    XtSetArg(wargs[n], XjNnoTop, reg.no_top); n++;
    XtSetArg(wargs[n], XjNlockPixmap, reg.lockpixmap); n++;
    XtSetArg(wargs[n], XjNheaderPixmap, reg.headerpixmap); n++;
    if (cw->menu.topmenu) {
        XtSetArg(wargs[n], XjNnoTop, True); n++;
        XtSetArg(wargs[n], XjNlocked, True); n++;
    }
    cw->menu.window = XtCreateManagedWidget(cw->core.name, windowWidgetClass, cw->core.parent, wargs,
    if (cw->menu.parentmenu != NULL)
        ((WindowWidget)cw->menu.window)->window.toclose = (WindowWidget)cw->menu.parentmenu->menu
    n = 0;
    XtSetArg(wargs[n], XmNleftAttachment, XmATTACH_FORM); n++;
    XtSetArg(wargs[n], XmNrightAttachment, XmATTACH_FORM); n++;
    cw->menu.toplevel=XjWindowAddChild(cw->menu.window,"",xmFormWidgetClass,wargs,n);
    cw->menu.buttons = (Widget)NULL;
    cw->menu.buttonslist=(Widget *)XtMalloc(sizeof(Widget) * cw->menu.nbuttons);
    cw->menu.submenulist=(WindowWidget *)XtMalloc(sizeof(WindowWidget) * cw->menu.nbuttons);
    cw->menu.ylist = (Position *)XtMalloc(sizeof(Position) * cw->menu.nbuttons);
    cw->menu.armcallbacks = (MenuProcS*)XtMalloc(sizeof(MenuProcS)*cw->menu.nbuttons);
    cw->menu.disarmcallbacks =(MenuProcS*)XtMalloc(sizeof(MenuProcS)*cw->menu.nbuttons);
    cw->menu.activatecallbacks =(MenuProcS*)XtMalloc(sizeof(MenuProcS)*cw->menu.nbuttons);
    for (i=0; i<cw->menu.nbuttons; i++) {
        cw->menu.submenulist[i] = (WindowWidget) NULL;
        cw->menu.ylist[i] = (Position) 0;
        cw->menu.armcallbacks[i].function = NULL;
        cw->menu.disarmcallbacks[i].function = NULL;
        cw->menu.activatecallbacks[i].function = NULL;
        cw->menu.armcallbacks[i].data = NULL;
        cw->menu.disarmcallbacks[i].data = NULL;
        cw->menu.activatecallbacks[i].data = NULL;
    }
    for (i=0; i<cw->menu.numcallbacks; i++) {
        if (cw->menu.proclist[i].type == XjARM_CALLBACK) {
            cw->menu.armcallbacks[cw->menu.proclist[i].button].function =
                cw->menu.proclist[i].function;
                cw->menu.armcallbacks[cw->menu.proclist[i].button].data = cw->menu.proclist[i].data;
        }
        if (cw->menu.proclist[i].type == XjDISARM_CALLBACK) {
                cw->menu.disarmcallbacks[cw->menu.proclist[i].button].function = cw->menu.proclist[i].function;
                cw->menu.disarmcallbacks[cw->menu.proclist[i].button].data = cw->menu.proclist[i].data;
        }
        if (cw->menu.proclist[i].type == XjACTIVATE_CALLBACK) {
                cw->menu.activatecallbacks[cw->menu.proclist[i].button].function =  cw->menu.proclist[i].function;
                cw->menu.activatecallbacks[cw->menu.proclist[i].button].data = cw->menu.proclist[i].data;
        }
    }
    cw->menu.numcallbacks = 0;
    createMenuButtons(cw);
    if (getD(cw->menu.buttons, XtNwidth) < getD(cw->menu.window, XtNwidth))
        cw->core.width = getD(cw->menu.window, XtNwidth);
        else cw->core.width = getD(cw->menu.buttons, XtNwidth);
    InstallCallbacks(cw);
}

static void InstallCallbacks (cw)
MenuWidget cw;
{
    if (cw->menu.topmenu == True) return;
```

```
    if (cw->menu.parentmenu == NULL) return;
    n=0;
    XtSetArg(wargs[n], XjNarmButton, cw->menu.parentbutton); n++;
    XtSetArg(wargs[n], XjNarmSubmenu, cw); n++;
    XtSetValues(cw->menu.parentmenu, wargs, n);
}

static void UnrealizeMenu (cw)
    MenuWidget cw;
{
    XtUnrealizeWidget(((WindowWidget)cw->menu.window)->window.shell);
}

static void ArmMenu (button, submenu, Data)
                Widget button;
                Widget *submenu;
                XmAnyCallbackStruct *Data;
{
        if (*submenu) XtRealizeWidget(*submenu);
}

static void DisarmMenu (button, submenu, Data)
                Widget button;
                Widget *submenu;
                XmAnyCallbackStruct *Data;
{
        if (*submenu && !activated) UnrealizeMenu((MenuWidget)*submenu);
    activated = 0;
}

static void ActivateMenu (button, cw, Data)
                Widget button;
                MenuWidget cw;
                XmAnyCallbackStruct *Data;
{
int i;
    for (i=0; i<cw->menu.nbuttons; i++)
        if (cw->menu.buttonslist[i] == button) break;
        if (!cw->menu.submenulist[i]) CloseChildWindow(cw->menu.window);

    MustCloseWindow (cw->menu.window);
    activated = 1;
}

static void MenuCall (button, calldata, Data)
                Widget button;
                MenuProcS *calldata;
                XmAnyCallbackStruct *Data;
{
    if ((calldata->function != NULL) && (*(calldata->function) != NULL)) {
        if (calldata->data != NULL)
            (*(calldata->function))(*(calldata->data));
        else
            (*(calldata->function))();
    }
}
```

PIXMAP

Pixmap.h

```
#ifndef _Pixmap_h
#define _Pixmap_h

#include <X11/Xlib.h>
#include <X11/Intrinsic.h>

typedef struct {
        Pixmap pixmap;
        Dimension width, height;
} PixmapStruct;

typedef Pixmap                  Struct*PixmapP;
typedef PixmapP         *PixmapPList;
typedef String          *StringList;

extern PixmapP XjCreateAllocatedPixmap ();
extern PixmapP XjXpmToPixmap ();

#define XPM2PIXMAP_CMAP(name)  XjXpmToPixmap (              \
                                ##name##_pixels, ##name##_colors,      \
                                ##name##_ncolors, ##name##_width ,     \
                                ##name##_height,                \
                                dpy, screen,DefaultColormap(dpy,screen) )

#define XPM2PIXMAP(name)  XjXpmToPixmap (                \
                                ##name##_pixels, ##name##_colors,      \
                                0 , ##name##_width ,            \
                                ##name##_height,                \
                                dpy, screen,DefaultColormap(dpy,screen) )

#endif /* _Pixmap_h */
```

Pixmap.c

```
#include "Pixmap.h"

PixmapP XjCreateAllocatedPixmap (dpy,win,bits,w,h,fg,bg,depth)
    Display *dpy;
    Drawable win;
    char *bits;
    unsigned int w, h;
    unsigned long fg, bg;
    unsigned int depth;
{
 PixmapP tmp;
        tmp = (PixmapP) XtMalloc(sizeof(PixmapP));
        tmp->pixmap = XCreatePixmap(dpy, win, w, h, depth);
        tmp->pixmap = XCreatePixmapFromBitmapData(dpy,win,bits,w,h,fg,bg,depth);
        tmp->width = w;
```

```
        tmp->height = h;
        return tmp;
}

PixmapP XjXpmToPixmap (data, palette, ncolors, width, height, dpy, screen, cmap)
    char **data;
    char **palette;
    int ncolors;
    unsigned int width;
    unsigned int height;
    Display *dpy;
    int screen;
    Colormap cmap;
{
  PixmapP tmp;
  unsigned int i, j;
  static int colors_table[256];
  unsigned int c, col;
  char str[256];
  char *memo, *buffer;
  unsigned int cnt;
  XColor colors[256];
  XImage *image;
  Pixmap pixmap;
        cnt = width * height;
        tmp = (PixmapP) XtMalloc(sizeof(Pixmap));
        memo = buffer = (char *)XtMalloc(cnt);
        for (i=j=0; i<ncolors; i++, j+=2) {
                sscanf(palette[j], "%s", str);
                c = ((str[0] - 'a') << 4) + (str[1] - 'a');
                sscanf(palette[j+1], "#%6x", &col);
                colors[i].flags = DoRed | DoGreen | DoBlue;
                colors[i].red = (col >> 8) & 0xFF00;
                colors[i].green = (col ) & 0xFF00 ;
                colors[i].blue = (col << 8) & 0xFF00;
                XAllocColor(dpy,cmap,&colors[i]);
                colors_table[c] = colors[i].pixel;
        }
        for (i=0; i<height; i++) {
                sscanf(data[i], "%s", str);
                for (j=0; j<(width*2); j+=2) {
                        c = (str[j] - 'a') << 4;
                        c += str[j+1] - 'a';
                        *buffer = colors_table[c]; buffer++;
                }
        }
        image = XCreateImage(dpy, DefaultVisual(dpy,screen),
        DefaultDepth(dpy,screen), ZPixmap,0,
        (char *)memo,width,height,8, width);
        {
            static cnt=0;
            char str[10];
            sprintf(str, "image%d", cnt);
            cnt++;
            if (!XmInstallImage(image,str)) {
                printf("Nao conseguiu instalar imagem\n");
                exit(0);
            }
```

```
        tmp->pixmap = XmGetPixmap(ScreenOfDisplay(dpy,screen),str,
        WhitePixel(dpy,screen),
        BlackPixel(dpy,screen));
    }
    tmp->width = width;
    tmp->height = height;
    XFree(memo);
    XmInstallImage(image);
    XDestroyImage(image);
    return tmp;
}
```

Referências Bibliográficas

[Mullin, M.] Object Oriented Program Design, With Examples in C++, Addison-Wesley

[Margolis, P.] Software Engineering in C, Springer-Verlag

[Nue, A.] Xlib Programming Manual for version 11, O'Reilly & Associates, Inc.

[Nue, A.] X Toolkit Intrinsics Programming Manual OSF/Motif Edition, O'Reilly & Associates, Inc.

[Rost, R.] X and Motif Quick reference Guide, Digital Press.

OSF/Motif Programmer's Guide, Open Software Foundation.