CURSO DE PROGRAMAÇÃO
EM
X WINDOW (X11 R4)

Álvaro F. M. Azevedo (Eng. Civil U.P. - Docente da FEUP - Porto - Portugal)
Ricardo M. P. Santos (Eng. Civil U.P. - Docente do ISEP - Porto - Portugal)

Julho 1992

Este curso destina-se a programadores de aplicações que pretendam tirar partido das capacidades gráficas das workstations e terminais X, recorrendo à biblioteca de mais baixo nível que nelas se encontra disponível (Xlib). Os programadores de aplicações alfanuméricas multijanela destinadas a correr nesse tipo de equipamento, podem também utilizar a biblioteca Xlib, sendo no entanto mais prático o recurso a um toolkit.

Embora seja possível a chamada das funções da Xlib a partir de outras linguagens (por exemplo FORTRAN), é fortemente aconselhada a sua utilização a partir de programas escritos em ANSI C. Esta afirmação baseia-se nos seguintes factos:

- a própria biblioteca Xlib está escrita em C;
- no computador que se pretende utilizar é mais provável encontrar disponível o C binding;
- os programas que recorrem ao FORTRAN binding não são tão portáteis;
- a linguagem C é mais versátil do que a linguagem FORTRAN;
- é mais provável encontrar bibliografia com programas exemplo escritos em C;
- os toolkits mais recentes apenas possuem C binding;
- os programas que misturam várias linguagens são menos portáteis.

O presente curso foi preparado com base no livro **Xlib Programming Manual** de Adrian Nye (O'Reilly Associates, Inc). Este livro é o mais aconselhado quer para uma leitura sequencial, quer para uma posterior consulta de algum aspecto específico. Foi também consultado o livro **X Window System** de Robert Scheifler e James Gettys (Digital Press). Neste livro quer as funções da Xlib, quer o próprio X Protocol são abordados de uma forma mais aprofundada e exaustiva, sendo aconselhado para esclarecimento de dúvidas que subsistam da leitura do livro de Adrian Nye.

Os presentes apontamentos não se destinam a cobrir a totalidade dos assuntos abordados nesses livros, servindo apenas como texto de apoio a um curso essencialmente prático e estruturado com base numa sequência de programas exemplo. Nestes programas, os diversos aspectos relativos ao X Window System e às funções da biblioteca Xlib vão sendo apresentados de um modo sequencial e

incremental. O formando é incentivado a aplicar todos os aspectos referidos ao longo do curso num programa destinado à representação de gráficos 2D num ambiente multijanela. Este programa ser-lhe-à útil mais tarde como referência para o desenvolvimento de outras aplicações específicas.

Apresentam-se em seguida cópias das tranparências relativas às sessões de formação e, no Anexo A, as listagens dos programas exemplo que foram disponibilizados. Neste Anexo e após cada programa exemplo encontra-se um conjunto de exercícios correspondentes, quer a alterações desses programas, quer a sugestões para o desenvolvimento de um programa de gráficos 2D que inclua todos os aspectos referidos. Nos programas exemplo foram inseridos comentários destinados a facilitar a compreensão dos novos aspectos que foram sendo introduzidos ao longo do curso. É portanto aconselhada a leitura pormenorizada das listagens do Anexo A.

CURSO DE PROGRAMAÇÃO
EM
X WINDOW (X11 R4)

- Alvaro Azevedo (Eng. Civil U.P. - Docente da FEUP)
- Ricardo Santos (Eng. Civil U.P. - Docente do ISEP)

Bibliografia Fundamental:

- *Xlib Programming Manual*
  Adrian Nye
  Vol. I da colecção *"The Definitive Guides to the X Window System"*
  Editora: O'Reilly & Associates, Inc.

- *X Window System (2nd Edition)*
  Robert Scheifler & James Gettys
  Editora: Digital Press

- Manual da implementação da biblioteca Xlib relativo ao computador utilizado.

UM POUCO DE HISTÓRIA

Autores:

Robert W. Sheifler (MIT)
James Gettys (Digital)
Ron Newman (MIT)
... ...

- Nasceu com o propósito de obter um standard gráfico para workstations.
- Baseou-se no **W** de Stanford.
- Actualmente é gerido pelo **MIT X Consortium**

X 11 R 4

Versão ← (é alterada quando há incompatibilidade com as versões anteriores)

Release → (upgrade compatível com as releases anteriores- extensões)

Evolução:

X[1,9] (1984 - 1985)

- fases de desenvolvimento
- pouco divulgadas

X10 (1986)

- primeira versão do X amplamente distribuída em VAXstations
- seguiram-se outros fabricantes (HP, Apollo, Sun, ...)

X11R1 (1987)

- graphics context
- maior portabilidade
- ...

X11R2 (1988)

X11R3 (1989)

X11R4 (1990)

- interclient communication
- ...

X11R5 (1992)

- versão mais recente

DISPLAY

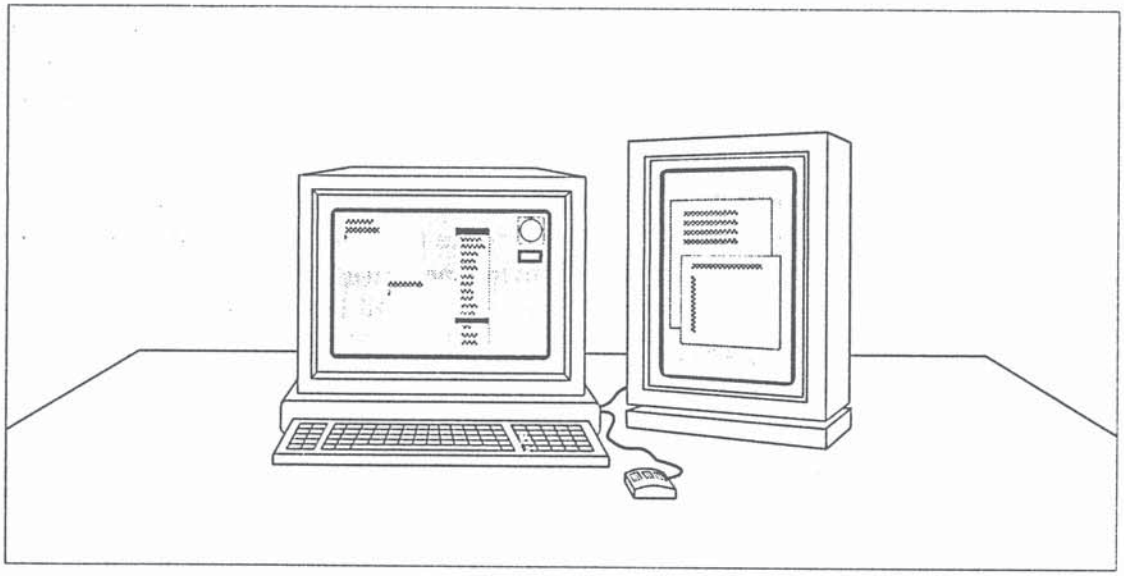


Figure 1-1. A display consisting of more than one screen

Display - um teclado
um rato com um ou mais botões
um ou mais screens

- Em cada screen podem ser representadas uma ou mais janelas
- Os caracteres provenientes do teclado apenas se destinam à janela activa
- O cursor, comandado pelo rato, pode percorrer todo o screen
- Com o rato é possível mudar a janela activa

SERVER - CLIENT MODEL

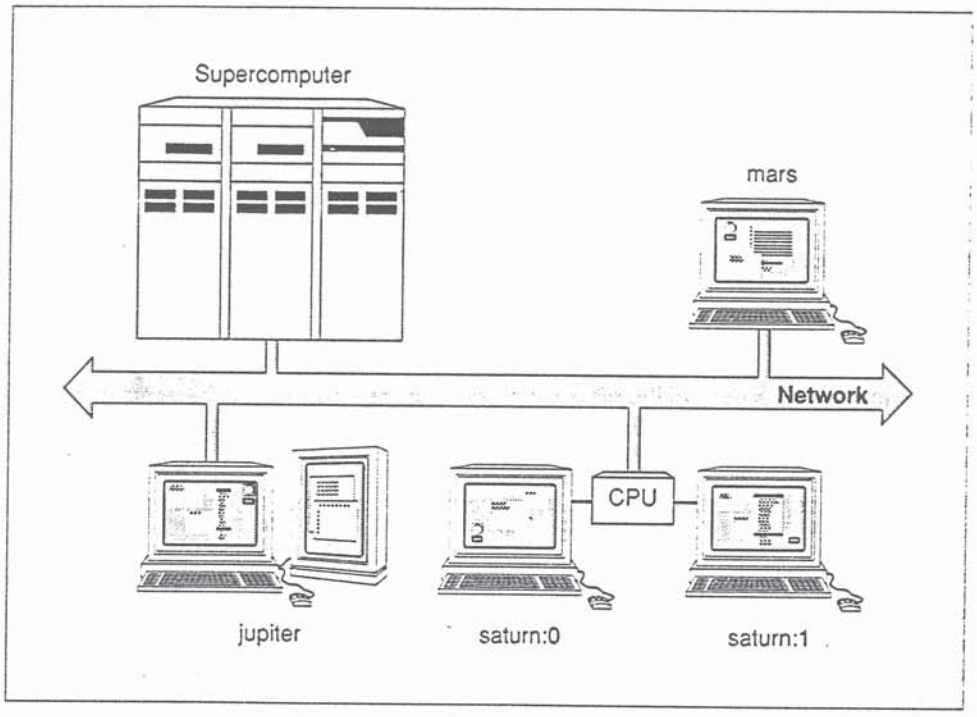


Figure 1-2. Applications can run on any system across the network

2 programas -
- CLIENT -
  - aplicação do utilizador
  - pode correr num nó qualquer
- SERVER -
  - fornecido pelo fabricante do display
  - corre no CPU do display

CLIENT -
- programa linkado com as funções da biblioteca Xlib
- envia requests ao server
- recebe informações sobre events no display

SERVER -
- atende aos requests (ex: abre janelas, representa gráficos na janela, ...)
- comunica ao client os events (ex: movimentos do rato, caracteres provenientes do teclado, ...)

COMUNICAÇÕES

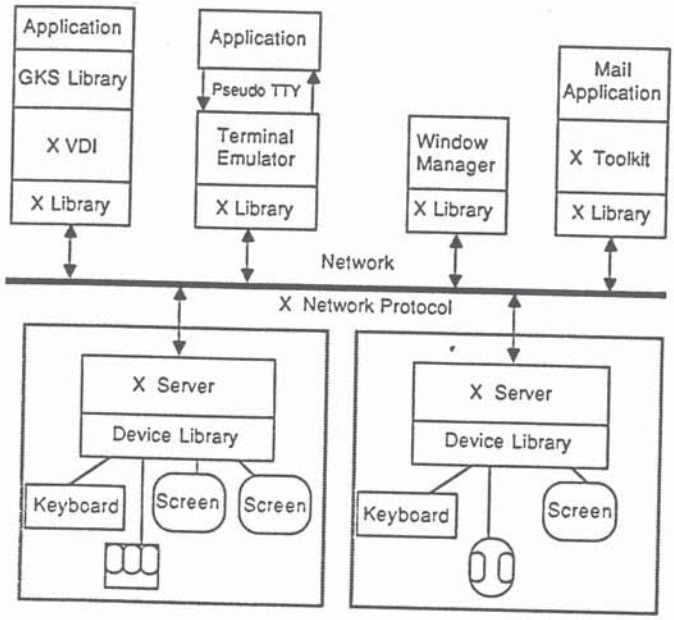


Figure 1. X window system block diagram

Redes Suportadas - **Ethernet - TCP/IP** (mais divulgada)

Ethernet - DECnet (Digital)

- As comunicações pela rede são feitas de um modo standard (**X Protocol**).

- Um display pode representar janelas relativas a clients que estão a correr em nós distintos.

WINDOW MANAGER

- Terceiro programa que corre em simultâneo com o client e o server.

- É um client especial:

  - só pode existir um window manager em cada display.

  - coloca decorações em torno das janelas.

  - permite ao utilizador a realização das seguintes tarefas interactivas:

    - mover janelas com o rato;
    - alterar a dimensão das janelas;
    - alterar a ordem de sobreposição das janelas;
    - transformar as janelas em icons (e vice-versa);
    - mudar a janela activa
    - ...

- Não existe nenhum window manager standard:

Exemplos -	**twm**	- o mais simples (fornecido pelo MIT)
	uwm	- Ultrix (antigamente)
	dxwm	- Dec Windows
	mwm	- Motif
	vuewm	- HP-VUE

EVENTS

Expose	janela total ou parcialmente descoberta (necessidade de redesenhar o conteúdo da janela).
ConfigureNotify	dimensões da janela alteradas (necessidade de alterar as dimensões dos gráficos que são desenhados na janela).
ButtonPress **ButtonRelease**	acção sobre um botão do rato.
MotionNotify	movimento do rato.
KeyPress **(KeyRelease)**	acção sobre o teclado.
EnterNotify **LeaveNotify**	entrada/saída do cursor na janela.
FocusIn **FocusOut**	activação/desactivação da janela relativamente ao keyboard input.

TOOLKITS

- Bibliotecas de utilitários que chamam as funções da Xlib.

- Destinam-se a facilitar o trabalho do programador relativamente ao interface com o utilizador (menus, dialog boxes, radio buttons, scroll bars, ...)

- Não existe nenhum toolkit standard:

- **MOTIF**	Open Software Foundation (HP, IBM, Digital, ...) (Sun - à revelia)
- **DecWindows**	Digital
- **Open Look**	Unix International (Sun, ...)
- **Next Step**	Next, IBM

DESKTOP MANAGERS

- Permitem ao utilizador a execução de tarefas com base na movimentação de icons (drag and drop).

- Não existe nenhum desktop manager standard:

  - **HP-VUE** (Hewlett-Packard)
  - **Open Desktop** (SCO)
  - **Looking Glass** (Intergraph)
  - **X.desktop** (para Motif)

- Outros exemplos (não usam o X Window System):

  - **Windows 3.0 e 3.1** (Microsoft - MSDOS)
  - **Presentation Manager** (IBM - OS/2)
  - **MacIntosh**

SESSION MANAGER

- Realiza de um modo simplificado algumas das funções do Desktop Manager:

  - correr XClients
  - configurar o ambiente
  - ...

CLIENTS FAMOSOS

xterm **dxterm** **hpterm**	emuladores de terminal numa janela
xclock	relógio
xcalc **xhpcalc**	calculadora
xcal	calendário
xeyes	*"big brother is watching you !"*
xwd	captura de todo o screen ou do conteúdo de uma janela
xwud	visualização no monitor do conteúdo do ficheiro com a imagem capturada
xpr	tradução do ficheiro com a imagem capturada em comandos relativos a uma determinada impressora

PRINCIPAIS FUNÇÕES

XOpenDisplay <--> XCloseDisplay

- estabelecer/terminar a conexão entre o client e o server

XCreateSimpleWindow <--> XDestroyWindow

- abrir/eliminar uma janela no display
- a maior parte das características da janela são copiadas da root window

XSelectInput

- indicar os tipos de events que devem ser comunicados ao client
- é necessário especificar uma lista de masks

XMapRaised

- tornar a janela visível (sobre as restantes)

XCreateGC <--> XFreeGC

- criar/libertar um graphics context (conjunto de propriedades relativas às primitivas gráficas)
- todas essas propriedades recebem valores por defeito, podendo ser alteradas com chamadas a outras funções

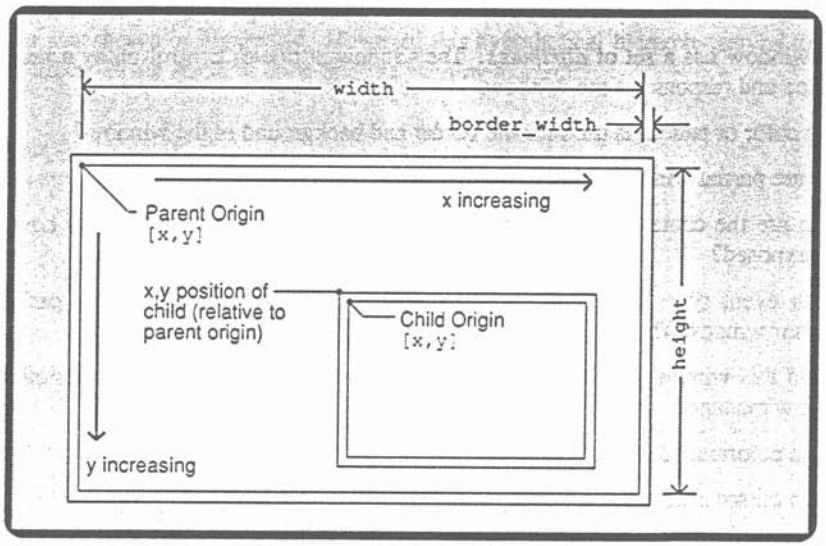
PRINCIPAIS FUNÇÕES

XNextEvent

- ler um event na event queue (o event lido é retirado da queue)

XDrawLine

- desenhar uma recta (é necessário indicar o display, a window, o GC e as coordenadas de início e fim em relação à origem da janela)



- as restantes funções relativas a primitivas gráficas têm argumentos semelhantes

XFlush

- desenhar efectivamente todas as primitivas gráficas que se encontram no buffer
- deve ser sempre chamada após o output de um conjunto de primitivas gráficas

ATRIBUTOS DAS PRIMITIVAS

Funções destinadas a alterar o conteúdo do Graphics Context (GC).

XSetForeground

- redefinição do índice de cor relativo à representação de primitivas

XSetBackground

- redefinição do índice de cor relativo ao fundo do texto em modo gráfico e às parcelas que não são desenhadas numa linha a tracejado

XSetLineAttributes

- redefinição dos atributos das linhas (espessura, tipo de linha, tipo de terminação e tipo de junção entre linhas) (ver as figuras da página seguinte)

XSetFunction

- tipo de operação lógica bitwise entre o pixel que se coloca no monitor e o pixel que lá se encontrava (GXcopy - resultado independente do pixel que lá se encontrava; GXxor - para cursores gráficos; ...)

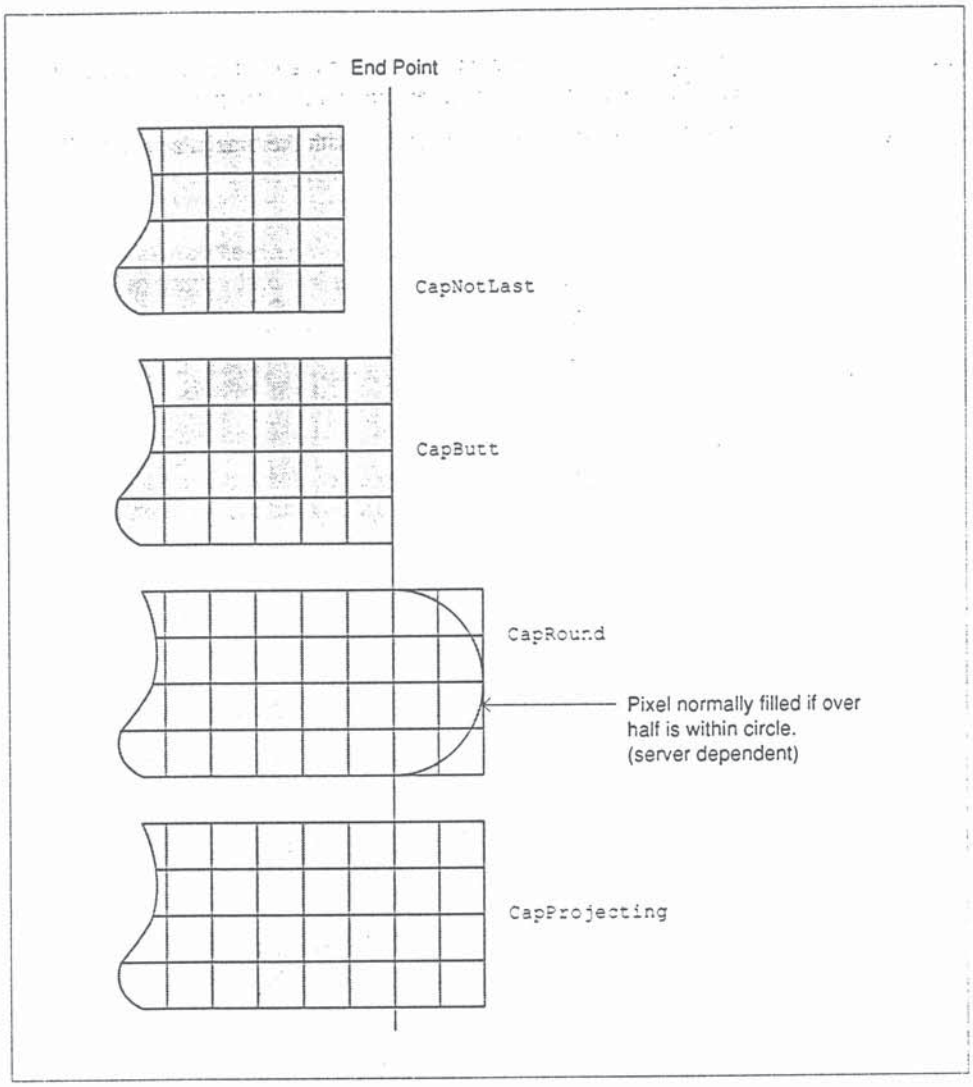


Figure 5-7. The line cap (end) styles

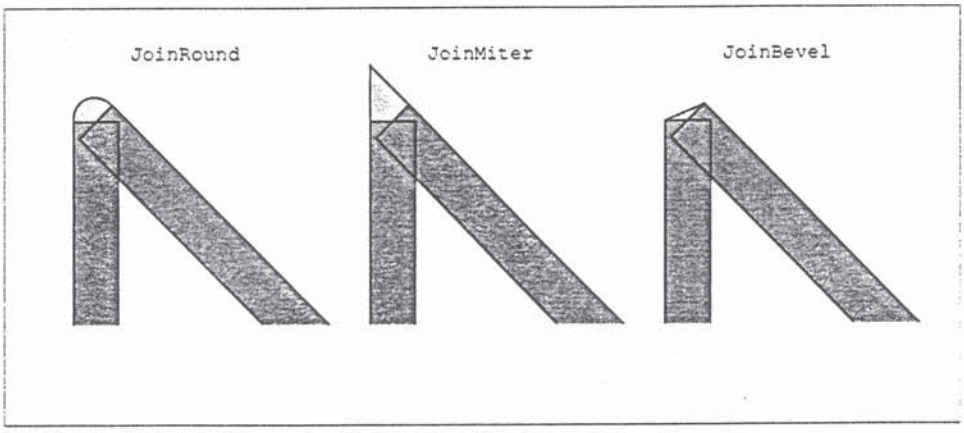


Figure 5-8. The line join styles

NOTAS RELATIVAS AOS ATRIBUTOS DAS PRIMITIVAS

- Se se pretendem linhas de espessura unitária,deve-se atribuir o valor zero à espessura, porque o desenho de rectas com espessura maior que zero é efectuado com um algoritmo menos eficiente (o valor por defeito é zero);

- Não se devem utilizar as cores que aparecem por defeito no default colormap. O utilizador deve acrescentar ao default colormap as cores que vai utilizar;

- O atributo relativo à junção de linhas só é respeitado se se chamarem funções que desenham conjuntos de linhas (XDrawLines, XDrawSegments);

- Alguns atributos relativos à terminação e junção de linhas só têm interesse se estas tiverem uma espessura significativa.

OUTRAS PRIMITIVAS GRÁFICAS

XDrawString, XDrawImageString

- output de texto em modo gráfico (sem ou com rectângulo de fundo)

XDrawArc

- desenho de um arco

XDrawRectangle

- desenho de um rectângulo

XFill Polygon

- desenho de um polígono preenchido
- tem de se fornecer um vector de XPoint's (struct com componentes *x* e *y*)
- pode-se indicar que o polígono é convexo, sendo então utilizado um algoritmo de preenchimento mais eficiente
- é também possível especificar o fill pattern

MANIPULAÇÃO DO MAPA DE CORES

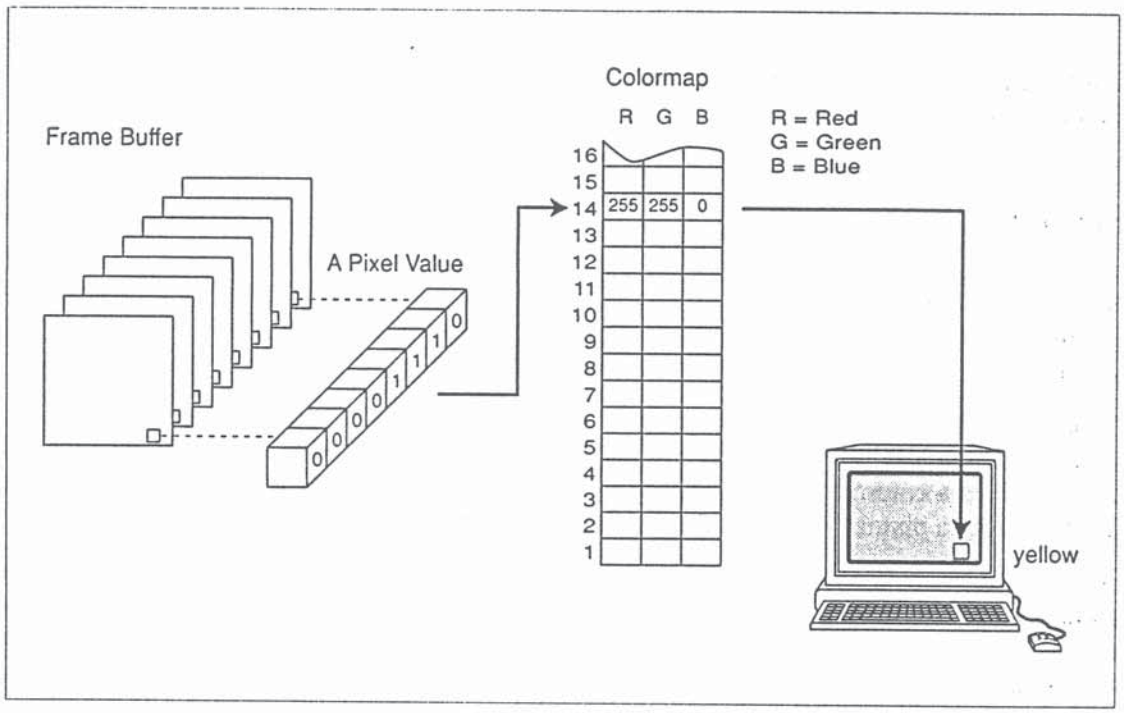
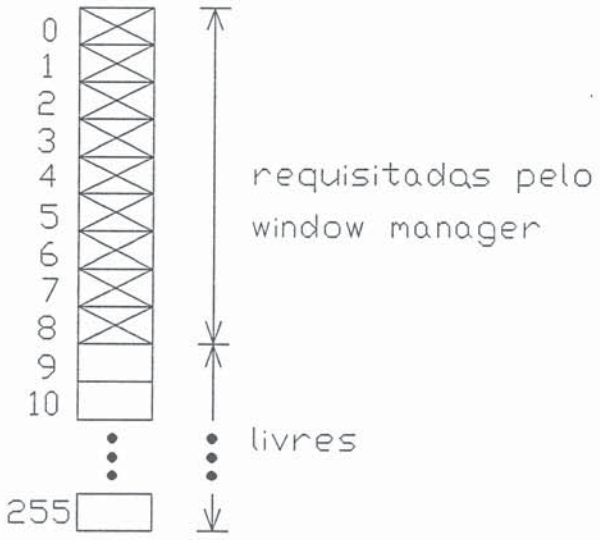


Figure 2-8. Mapping of pixel value into color through colormap

Células do default colormap



MANIPULAÇÃO DO MAPA DE CORES

XParseColor

- conversão de um string que especifica uma determinada cor nas componentes de um struct do tipo XColor
- o string pode ser o nome de uma cor cujas componentes RGB se encontrem no ficheiro **/usr/lib/X11/rgb.txt**
- ou um conjunto de coordenadas RGB em hexadecimal
  Ex: #a3bd**59ac**ef3a

[0, 65535]

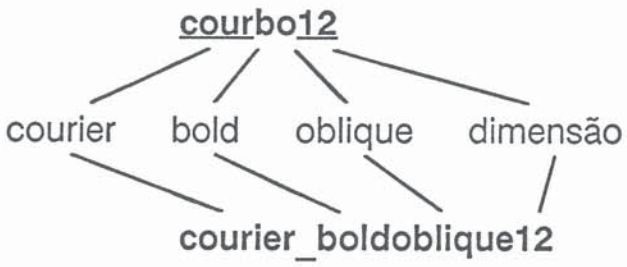
XAllocColor

- coloca as coordenadas RGB numa célula livre do default colormap e indica, via componente pixel do struct XColor, o índice correspondente
- se essas coordenadas RGB já foram colocadas no colormap por outro client, apenas é fornecido o respectivo índice do pixel

FONTS

Estão situadas em **/usr/lib/X11/fonts/...**

Exemplo:



XLoadFont <--> XUnloadFont

- permite o acesso/quebra a associação do client a uma font

XLoadQueryFont <--> XFreeFont

- mais usada
- igual ao anterior mas retorna um struct do tipo **XFontStruct** onde se encontram definidas todas as características da font especificada (ver página seguinte)

XSetFont

- associa uma font a um GC

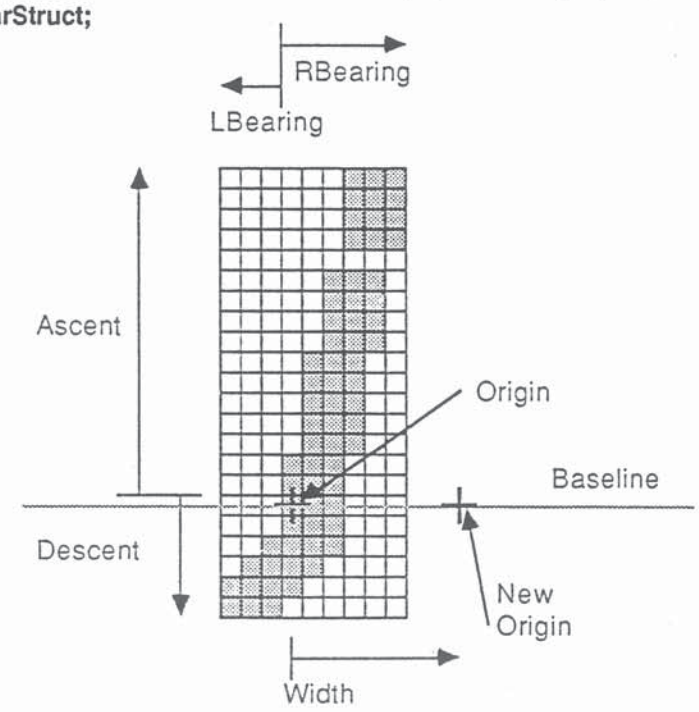
STRUCTS ASSOCIADOS A FONTS

XFontStruct - atributos relativos a uma determinada font

```
typedef struct {
   ...
      Font fid;                        /* Font id for this font */
      unsigned direction;              /* hint - direction the font is painted */
      unsigned min_char_or_byte2;      /* first character */
      unsigned max_char_or_byte2;      /* last character */
      ...
      unsigned default_char;           /* char to print for undefined char. */
      ...
      XCharStruct min_bounds;          /* min. bounds over all char. */
      XCharStruct max_bounds;          /* max. bounds over all char. */
      XCharStruct *per_char;           /* first_char to last_char info */
      int ascent;                      /* extent above baseline (spacing) */
      int descent;                     /* descent below baseline (spacing) */
} XFontStruct;
```

XCharStruct - atributos relativos a cada caracter de uma font

```
typedef struct {
   short lbearing;                     /* origin to left edge of raster */
   short rbearing;                     /* origin to right edge of raster */
   short width;                        /* advance to next char's origin */
   short ascent;                       /* baseline to top edge of raster */
   short descent;                      /* baseline to bottom edge of raster */
   unsigned short attributes;          /* per char flags (not predefined)*/
} XCharStruct;
```



OUTRAS FUNÇÕES ASSOCIADAS A FONTS

XListFonts, XListFontsWithInfo

- permitem obter uma lista das fonts disponíveis no X server
- fornece-se um pattern (que pode conter wildcards à maneira do Unix) e um número máximo de fonts a retornar
- XListFontsWithInfo retorna ainda um vector de **XFontStruct** com os atributos de cada font encontrada

GC CONVENIENCE FUNCTIONS OU MÚLTIPLOS GC

O X Window dispõe de várias funções para alterar os valores correntes do Graphics Context (**XSetForeground, XSetFont, XSetFunction, XChangeGC**, ...)

É possível ter vários GC a coexistirem numa aplicação.

A abordagem mais conveniente dependerá de caso para caso:

- cada GC gasta recursos do terminal (memória);
- a utilização de funções obriga a fazer chamadas ao server (aumento do tráfego na rede);

Uma regra possível será manter um número limitado de GC que configurem os aspectos mais importantes e usar nos outros casos as **Convenience Functions.**

REPRESENTAÇÃO DE TEXTO

XDrawString, XDrawImageString

- ver atrás

XTextWidth

- comprimento em pixels de um determinado string (para a font especificada)
- usa o componente **width** do **XCharStruct**

- exemplo de utilização: cálculo da posição onde desenhar o caracter seguinte para um string dado a partir do teclado

XTextExtents

- comprimento lógico (**bounding box**) de um determinado string (para a font especificada)
- retorna um **XCharStruct** onde cada componente contém o valor máximo verificado no string (**ascent, descent, ...**)

- exemplo de utilização: espaçamento ou posicionamento de linhas de texto

KEYBOARD

Sempre que se carrega (ou liberta) uma tecla o server envia um event (**KeyPress/KeyRelease**).

Por questões de portabilidade de software deve-se evitar a dependência de teclas não-standard e de events do tipo KeyRelease (há servers que não detectam estes events).

Devem ser controladas todas as teclas possíveis para evitar reacções inesperadas da aplicação (ver **kevent_7.c**).

KeyCode --> KeySym (--> String)

KeyCode

- representação (número) de cada tecla [8, 255]
- valor que é retornado pelos events KeyPress e KeyRelease

KeySym

- codificação do símbolo inscrito em cada tecla
  (string que corresponde ao significado de cada tecla)
- corresponde aproximadamente ao CODEPAGE do MS-DOS

exemplo:

Tecla	Keysym
a	XK_a
[SHIFT] + a	XK_A
&	XK_ampersand
Return, Enter	XK_Return
9 (keypad)	XK_KP_9
Enter(keypad)	XK_KP_Enter

KEYBOARD

XLookupString

- função habitualmente utilizada para obter caracteres do teclado
- traduz um key event num keysym (ISO Latin -1)
  fornece um ASCII string associado ao keysym para todas as teclas em que esse valor esteja definido (normalmente o caracter correspondente à tecla)
- fornece NULL se não houver nenhum ASCII associado à tecla (exemplo: XK_Shift_L)

XKeySymToString

- traduz um keysym num string, i.e. fornece o "nome" da tecla

exemplo:

keysym	string	tecla
XK_a	a	a
XK_BackSpace	BackSpace	←
XK_question	question	?

CHILD WINDOWS

São windows descendentes das main windows de uma aplicação.

Contrariamente às main windows não são controladas pelo window manager.

As funções e atributos definidos para as main windows também se aplicam às child windows (**XCreateSimpleWindow, XSelectInput, XDefineCursor**, ...).

Uma child window pode aceitar events diferentes da parent window.

Qualquer child window é sempre truncada pela sua parent window.

XMapSubwindows <--> XDestroySubwindows

- desenha/destrói **todas** as child windows de uma parent window (funções específicas para as child windows)

X FONT CURSORS


XC_fleur


XC_gobbler


XC_gumby


XC_hand1


XC_hand2


XC_heart


XC_icon


XC_iron_cross


XC_left_ptr


XC_left_side


XC_left_tee


XC_leftbutton


XC_ll_angle


XC_lr_angle


XC_man


XC_middlebutton


XC_mouse


XC_pencil


XC_pirate


XC_plus


XC_question_arrow


XC_top_right_corner


XC_top_side


XC_top_tee


XC_trek


XC_ul_angle


XC_right_ptr


XC_right_side


XC_right_tee


XC_rightbutton


XC_rtl_logo


XC_sailboat


XC_sb_down_arrow


XC_sb_h_double_arrow


XC_sb_left_arrow


XC_sb_right_arrow


XC_sb_up_arrow


XC_sb_v_double_arrow


XC_shuttle


XC_sizing

XC_spider

XC_spraycan

XC_star

XC_target

XC_tcross

XC_top_left_arrow

XC_top_left_corner

XC_umbrella

XC_ur_angle

XC_watch

XC_xterm

XC_X_cursor

XC_arrow

XC_based_arrow_down

XC_based_arrow_up

XC_boat

XC_bogosity

XC_bottom_left_corner

XC_bottom_right_corner

XC_bottom_side

XC_bottom_tee

XC_box_spiral

XC_center_ptr

XC_circle

XC_clock

XC_coffee_mug

XC_cross

XC_cross_reverse

XC_crosshair

XC_diamond_cross

XC_dot

XC_dot_box_mask

XC_double_arrow

XC_draft_large

XC_draft_small

XC_draped_box

XC_exchange

```
/*****************************************************************/
/*                                                               */
/* ex9                                                           */
/* Example 9                                                     */
/*                                                               */
/* Alteracao introduzida para poder dispor de um cursor diferente */
/* na janela menu (myopt_win).                                   */
/*                                                               */
/*****************************************************************/

...

#include <X11/cursorfont.h>

...

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   ...
   Cursor hand_cursor;                                  /* <<<<<< */

   ...

/* Create window for options input */

   myopt_win=...

/* Create the cursor for the menu */

   hand_cursor= XCreateFontCursor (mydisplay, XC_hand2);   /* <<<<<< */

   XDefineCursor (mydisplay, myopt_win, hand_cursor);      /* <<<<<< */

   XMapWindow ...

   ...

} /* main - ex9 */
```

FUNÇÕES ASSOCIADAS A EVENTS

O X server envia events para a Xlib como resposta a inputs do teclado ou rato,ou como reacção a pedidos do client.

Os events são enviados para uma event queue onde as funções seguintes as podem ir procurar:

XNextEvent

- processa o **primeiro** event da event queue
- copia o event para a event structure e retira-o da queue
- espera pelo event seguinte se não existir nenhum na queue

XPeekEvent

- igual ao anterior mas não retira o event da event queue

XWindowEvent, XCheckWindowEvent

- procuram um event que satisfaça a window e a event mask
- XWindowEvent espera pelo event seguinte se não existir nenhum na queue

XCheckTypedEvent

- procura e remove um event da queue
- não espera pelo event seguinte

XPutBackEvent

- repõe na queue um event previamente de lá retirado

EVENT LIST

Event Mask	*Event Type*	*Structure*	*Generic Structure*
ButtonMotionMask	MotionNotify	XPointerMovedEvent	XMotionEvent
Button1MotionMask			
Button2MotionMask			
Button3MotionMask			
Button4MotionMask			
Button5MotionMask			
ButtonPressMask	ButtonPress	XButtonPressedEvent	XButtonEvent
ButtonReleaseMask	ButtonRelease	XButtonReleasedEvent	XButtonEvent
ColormapChangeMask	ColormapNotify	XColormapEvent	
EnterWindowMask	EnterNotify	XEnterWindowEvent	XCrossingEvent
LeaveWindowMask	LeaveNotify	XLeaveWindowEvent	XCrossingEvent
ExposureMask	Expose	XExposeEvent	
GCGraphicsExposure in GC	GraphicsExpose	XGraphicsExposeEvent	
	NoExpose	XNoExposeEvent	
FocusChangeMask	FocusIn	XFocusInEvent	XFocusChangeEvent
	FocusOut	XFocusOutEvent	XFocusChangeEvent
KeymapStateMask	KeymapNotify	XKeymapEvent	
KeyPressMask	KeyPress	XKeyPressedEvent	XKeyEvent
KeyReleaseMask	KeyRelease	XKeyReleasedEvent	XKeyEvent
OwnerGrabButtonMask	N.A.	N.A.	
PointerMotionMask	MotionNotify	XPointerMovedEvent	XMotionEvent
PointerMotionHintMask	N.A.	N.A.	
PropertyChangeMask	PropertyNotify	XPropertyEvent	
ResizeRedirectMask	ResizeRequest	XResizeRequestEvent	
StructureNotifyMask	CirculateNotify	XCirculateEvent	
	ConfigureNotify	XConfigureEvent	
	DestroyNotify	XDestroyWindowEvent	
		GravityNotify	XGravityEvent
		MapNotify	XMapEvent
		ReparentNotify	XReparentEvent
		UnmapNotify	XUnmapEvent
	SubstructureNotifyMask	CirculateNotify	XCirculateEvent
		ConfigureNotify	XConfigureEvent
		CreateNotify	XCreateWindowEvent
		DestroyNotify	XDestroyWindowEvent
		GravityNotify	XGravityEvent
		MapNotify	XMapEvent
		ReparentNotify	XReparentEvent
		UnmapNotify	XUnmapEvent
	SubstructureRedirectMask	CirculateRequest	XCirculateRequestEvent
		ConfigureRequest	XConfigureRequestEvent
		MapRequest	XMapRequestEvent
	N.A.	ClientMessage	XClientMessageEvent
	N.A.	MappingNotify	XMappingEvent
	N.A.	SelectionClear	XSelectionClearEvent
	N.A.	SelectionNotify	XSelectionEvent
	N.A.	SelectionRequest	XSelectionRequestEvent
	VisibilityChangeMask	VisibilityNotify	XVisibilityEvent

MOUSE

Dois aspectos a considerar:

- acção sobre os botões do rato;
- movimentos do rato.

As acções sobre os botões do rato são tratadas de modo semelhante ao que é feito para o teclado.

Cada botão retorna um número inteiro de 1 a 5 (**Button1 ... Button5**).

Geram-se events do tipo **ButtonPress** e **ButtonRelease.**

Pressão simultânea sobre vários botões ou usando modifier keys só pode ser identificada usando a função **XQueryPointer.**

MOUSE

Três modos diferentes de detectar movimentos do rato:

- a aplicação recebe e processa todos os **MotionNotify** events
- pode gerar atrasos da aplicação em relação à posição corrente do rato

- usando hints
- obriga a chamar a função **XQueryPointer** para obter a posição corrente do rato mas reduz bastante o número de motion events reportados pelo server (ver **g_out_9.c**)

- usando o **motion history buffer** (se ele estiver definido no server)
- este buffer pode ser consultado para obter um relatório detalhado de todos os movimentos do rato entre dois instantes (**timestamps**)

INTERCLIENT COMMUNICATION (HINTS)

O X Window System funciona em rede.

Todos os clients devem respeitar regras de "boa vizinhança".

Cada client deve estar preparado para trabalhar com os recursos que o window manager lhe puder fornecer; nunca se deve lutar contra o window manager. Exemplo:

> se um client deseja uma window de certa dimensão considerada mínima deve estar preparado para trabalhar com uma window menor (nem que seja para dar uma mensagem do tipo "MAKE ME BIGGER!").

Todos os clients devem comunicar com o window manager através das **properties** sugerindo valores (**hints**). O window manager não é obrigado a respeitar os hints mas o client deve poder esperar que ele tudo fará para que assim aconteça.

INTERCLIENT COMMUNICATION (HINTS)

WM_NAME	Nome da aplicação (semelhante ao **XStoreName**)
WM_ICON_NAME	Nome a colocar no icon da aplicação
WM_NORMAL_HINTS	Sugestões relativas ao tamanho mínimo e máximo, posição da main window, ...
WM_HINTS	Outros hints adicionais (icons relativos às main windows, modo de arranque da aplicação -window/icon-, ...)
CLASS_HINTS	permitem ao window manager obter os recursos relativos à aplicação
WM_PROTOCOLS	lista de protocolos em que a aplicação deseja participar (properties relativas ao session manager)
...	...

X WINDOW SYSTEM: PROGRAMAS EXEMPLO

minix - Desenhar uma recta numa janela com o menor número de chamadas a funções da Xlib.

ex1 - Idem com comentários relativos a cada função e com chamada de funções relativas ao encerramento do diálogo com o X server.

ex2 - Idem com X event loop, com detecção de ButtonPress events e com desenho de texto em modo gráfico.

ex3 - Alteração dos atributos das primitivas gráficas.

ex4 - Redefinição do mapa de cores e chamada de novas primitivas gráficas (arcos e rectângulos).

ex5 - Detecção de ConfigureNotify events, obtenção de window attributes e chamada de novas primitivas gráficas (fill polygon).

ex6 - Alteração da font e obtenção das respectivas características.

ex7 - Leitura do teclado a partir de uma janela.

ex8 - Utilização de child windows.

ex9 - Leitura da posição do rato.

ex10 - Interclient communication (hints) e utilização de várias main windows.

Indice do Anexo A

====== minix.c ======

```
/*****************************************************************/
/*                                                               */
/* minix                                                         */
/* Short X Window application / client.                          */
/*                                                               */
/* Functions from Xlib library (function prototypes in X11/Xlib.h):  */
/*    XOpenDisplay        - start connection with the X server       */
/*    XCreateSimpleWindow - create a window with default options     */
/*    XSelectInput        - select events to be reported             */
/*    XMapRaised          - display window on top of the others      */
/*    XCreateGC           - create a graphics context                */
/*    XNextEvent          - get event from event queue               */
/*    XDrawLine           - draw line from x1,y1 to x2,y2            */
/*    XFlush              - flush the output buffer                  */
/*                                                               */
/* Macros defined in X11/Xlib.h:                                 */
/*    DefaultScreen       - default screen of the specified display  */
/*    DefaultRootWindow   - default root window of the spec. display */
/*    BlackPixel          - index of the color black                 */
/*    WhitePixel          - index of the color white                 */
/*                                                               */
/* typedef's declared in X11/Xlib.h: Display, Window, XEvent and GC  */
/*                                                               */
/* #define's declared in X11/Xlib.h: ExposureMask                */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <X11/Xlib.h>

int main (void) {

   Display   *mydisplay;
   int        myscreen;
   Window     mywindow;
   XEvent     myevent;
   GC         mygc;

   if (   (mydisplay=XOpenDisplay ("")) == NULL   ) {
      fprintf (stderr,"Cannot connect to server\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,600,400,                      /* size */
                    2,                                /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

   XSelectInput (mydisplay,mywindow,ExposureMask);

   XMapRaised (mydisplay,mywindow);

   mygc=XCreateGC (mydisplay,mywindow,0L,NULL);

   XNextEvent (mydisplay,&myevent);

   XDrawLine (mydisplay,mywindow,mygc,
              22,22,222,222);
```

```
    XFlush (mydisplay);

    printf ("\n");
    printf ("Press the Return key ... ");
    getchar ();

    printf ("\n");
    return (0);

} /* main - minix */
```

====== ex1.c ======

```
/*****************************************************************/
/*                                                               */
/* ex1                                                           */
/* Example 1                                                     */
/*                                                               */
/* New features:                                                 */
/*    - comments for every X function                            */
/*    - display selection                                        */
/*    - final clean up                                           */
/*                                                               */
/* New functions:                                                */
/*    XFreeGC        - free graphics context allocated memory    */
/*    XDestroyWindow - eliminate window from display             */
/*    XCloseDisplay  - terminate connection with the X server    */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy,strcat */
#include <X11/Xlib.h>

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display   *mydisplay;
   int        myscreen;
   Window     mywindow;
   XEvent     myevent;
   GC         mygc;

   char       mydispname[82]; /* display name */

/* Variable initialization */

   mydispname[0]=0;

   printf ("\n");
   printf ("Example 1\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
```

```
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

/* Select default screen */

   myscreen=DefaultScreen (mydisplay);

/* Create a window */

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,600,400,                    /* size */
                    2,                              /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Report expose events only */

   XSelectInput (mydisplay,mywindow,ExposureMask);

/* Display window on top of the others */

   XMapRaised (mydisplay,mywindow);

/* Create a graphics context (GC) */

   mygc=XCreateGC (mydisplay,mywindow,0L,NULL);

/* Get event from event queue */

   XNextEvent (mydisplay,&myevent);

/* Do some graphical output */

   XDrawLine (mydisplay,mywindow,mygc,
              22,22,222,222);

/* Flush the output buffer (make all graphics primitives visible) */

   XFlush (mydisplay);

/* Pause in graphics mode */

   printf ("\n");
   printf ("Press the Return key to terminate graphics mode ... ");
   getchar ();

/* Terminate graphical output */

   XFreeGC (mydisplay,mygc);
   XDestroyWindow (mydisplay,mywindow);
   XCloseDisplay (mydisplay);

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to terminate application ... ");
   getchar ();

   printf ("\n");
   printf ("That's all folks !\n");

   printf ("\n");
   return (0);

} /* main - ex1 */
```

====== ex2.c ======

```
/*****************************************************************/
/*                                                               */
/* ex2                                                           */
/* Example 2                                                     */
/*                                                               */
/* New features:                                                 */
/*    - main X event loop                                        */
/*    - window title                                             */
/*    - text graphical output                                    */
/*                                                               */
/* New functions:                                                */
/*    XStoreName       - define window title                     */
/*    XDrawImageString - draw text in the specified location     */
/*                                                               */
/* New #define's: ButtonPressMask, Expose and ButtonPress       */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy,strcat,strlen */
#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display   *mydisplay;
   int        myscreen;
   Window     mywindow;
   XEvent     myevent;
   GC         mygc;

   char       myendflag; /* flag to test the end of the event loop */
   char       mydispname[82]; /* display name */
   char       myhelptext[82]; /* text to be displayed in the window */

/* Variable initialization */

   mydispname[0]=0;
   strcpy (myhelptext,
       "Press a mouse button on this window to terminate application");

   printf ("\n");
   printf ("Example 2\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */
```

```
   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,600,400,                    /* size */
                    2,                              /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay,mywindow,"Hello  X  world !");

   XSelectInput (mydisplay,mywindow,ExposureMask | ButtonPressMask);
/*                                                    ^          */
/* A new mask was added because of ButtonPress events |          */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - +          */

   XMapRaised (mydisplay,mywindow);
   mygc=XCreateGC (mydisplay,mywindow,0L,NULL);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case Expose:

/*          Draw line */

            XDrawLine (mydisplay,mywindow,mygc,
                       22,22,222,222);

/*          Draw text */

            XDrawImageString (mydisplay,mywindow,mygc,
                              44,22,myhelptext,strlen (myhelptext));

            XFlush (mydisplay);

            break;

         case ButtonPress:

/*          Turn on end flag */

            myendflag= TRUE ;

            break;

      } /* switch (myevent.type) */
```

```
   } /* while ( ! myendflag) */

/* Terminate graphical output */

   XFreeGC (mydisplay,mygc);
   XDestroyWindow (mydisplay,mywindow);
   XCloseDisplay (mydisplay);

   printf ("\n");
   printf ("That's all folks !\n");

   printf ("\n");
   return (0);

} /* main - ex2 */
```

====== work2 ======

Exercicios relativos ao programa ex2:

1 - Representar o grafico de uma funcao pre definida.

2 - Acrescentar ao grafico uma quadricula.

3 - Acrescentar a identificacao dos eixos da quadricula.

====== ex3.c ======

```
/*****************************************************************/
/*                                                               */
/* ex3                                                           */
/* Example 3                                                     */
/*                                                               */
/* New features:                                                 */
/*    - event queue is checked for other expose events           */
/*    - all graphical output is in function g_out_3              */
/*    - new primitives are used                                  */
/*    - primitive attributes are changed                         */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy,strcat */
#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

void g_out_3 (Display *,Window,GC);

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display   *mydisplay;
   int        myscreen;
   Window     mywindow;
   XEvent     myevent;
   GC         mygc;
   char       myendflag; /* flag to test the end of the event loop */
   char       mydispname[82]; /* display name */

/* Variable initialization */

   mydispname[0]=0;

   printf ("\n");
   printf ("Example 3\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
```

```
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,600,400,                    /* size */
                    2,                              /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay,mywindow,"Hello  X  world !");
   XSelectInput (mydisplay,mywindow,ExposureMask | ButtonPressMask);
   XMapRaised (mydisplay,mywindow);
   mygc=XCreateGC (mydisplay,mywindow,0L,NULL);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case Expose:

/*          Get rid of all the other Expose events */

            while (   XCheckTypedEvent (mydisplay,Expose,&myevent)   );

/*          Graphical output of a few primitives */

            g_out_3 (mydisplay,mywindow,mygc);

            break;

         case ButtonPress:

/*          Turn on end flag */

            myendflag= TRUE ;

            break;

      } /* switch (myevent.type) */

   } /* while ( ! myendflag) */

/* Terminate graphical output */

   XFreeGC (mydisplay,mygc);
   XDestroyWindow (mydisplay,mywindow);
   XCloseDisplay (mydisplay);

   printf ("\n");
   printf ("That's all folks !\n");

   printf ("\n");
   return (0);

} /* main - ex3 */
```

====== g_out_3.c ======

```
/*****************************************************************/
/*                                                               */
/* g_out_3                                                       */
/* Graphical output of a few primitives.                         */
/* This function is called by the main program ex3.              */
/* Some attributes are changed.                                  */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <string.h> /* strcpy,strlen */
#include <X11/Xlib.h>

void g_out_3 (
   Display   *mydisplay,
   Window     mywindow,
   GC         mygc
) {

/* Variable declaration */

   int ixpos;           /* x position in pixels */
   int iypos;           /* y position in pixels */
   int mylinewidth;     /* line width in pixels */
   char myhelptext[82]; /* text to be displayed in the window */

/* Variable initialization */

   ixpos=100;
   iypos=100;
   mylinewidth=30;
   strcpy (myhelptext,
     " Press a mouse button on this window to terminate application ");

/* Clear window */

   XClearWindow (mydisplay,mywindow);

/* Change some line attributes */

   XSetLineAttributes (mydisplay,mygc,
                       mylinewidth,LineSolid,CapButt,JoinMiter);

/* Set color index 2 as foreground */

   XSetForeground (mydisplay,mygc,2);

   XDrawLine (mydisplay,mywindow,mygc,
              ixpos-60,iypos,ixpos,iypos+100);

   XDrawLine (mydisplay,mywindow,mygc,
              ixpos,iypos+100,ixpos+60,iypos);

   ixpos+=200;

/* Change some line attributes */

   XSetLineAttributes (mydisplay,mygc,
                       mylinewidth,LineSolid,CapRound,JoinMiter);

/* Set color index 4 as foreground */

   XSetForeground (mydisplay,mygc,4);
```

```
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos-60,iypos,ixpos,iypos+100);

    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos+100,ixpos+60,iypos);

    ixpos-=250;
    iypos+=150;

/* Change some line attributes */

    XSetLineAttributes (mydisplay,mygc,
                        2,LineDoubleDash,CapButt,JoinMiter);

    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+300,iypos-30);

    iypos+=100;

/* Set color index 1 as foreground and color index 5 as background */

    XSetForeground (mydisplay,mygc,1);
    XSetBackground (mydisplay,mygc,5);

    XDrawImageString (mydisplay,mywindow,mygc,
                      ixpos,iypos,myhelptext,strlen (myhelptext));

/* Set XOR mode */

    XSetFunction (mydisplay,mygc,GXxor);

/* Change some line attributes */

    XSetLineAttributes (mydisplay,mygc,
                        mylinewidth,LineSolid,CapRound,JoinMiter);

/* Set color index 6 as foreground */

    XSetForeground (mydisplay,mygc,6);

    for (ixpos=50 , iypos=50 ; ixpos <= 450 ; ixpos+=100) {

       XDrawLine (mydisplay,mywindow,mygc,        /* draw */
                  ixpos,iypos,ixpos,iypos+300);

       XFlush (mydisplay);

       sleep (2); /* wait 2 seconds */

       XDrawLine (mydisplay,mywindow,mygc,        /* erase */
                  ixpos,iypos,ixpos,iypos+300);

       XFlush (mydisplay);

    } /* for (ixpos) */

/* Reset attributes to their usual values */

    XSetFunction (mydisplay,mygc,GXcopy);
    XSetForeground (mydisplay,mygc,0);
    XSetBackground (mydisplay,mygc,1);
    XSetLineAttributes (mydisplay,mygc,
                        0,LineSolid,CapButt,JoinMiter);
/*                      |                                          */
/*                      +---> line width is one but output is faster  */
    XFlush (mydisplay);

} /* g_out_3 */
```

====== work3 ======

Exercicios relativos ao programa ex3:

1 - Retirar a chamada da funcao XCheckTypedEvent e verificar como o programa se passou a comportar.

2 - Representar um conjunto de rectas tendo cada uma delas um indice de cor diferente. Percorrer todos os indices disponiveis [0,255] e colocar ao lado de cada recta o numero do indice de cor (utilizar uma cor que seja sempre visivel).

3 - Por a quadricula a tracejado no programa de graficos 2D.

4 - Verifique a diferenca entre as funcoes XDrawImageString e XDrawString (altere o foreground e background).

5 - Verifique a diferenca entre o tipo de linha LineOnOffDash e LineDoubleDash (altere o foreground e background).

====== ex4.c ======

```
/*****************************************************************/
/*                                                               */
/* ex4                                                           */
/* Example 4                                                     */
/*                                                               */
/* New features:                                                 */
/*    - new color map entries are allocated in the default color map */
/*    - all color map changes are done by the subroutine icomap_4 */
/*    - new primitives are used in g_out_4                       */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy,strcat */
#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

void icomap_4 (Display *,
               int *,int *,int *,int *,int *,int *,int *,int *);
void g_out_4 (Display *,Window,GC,int,int,int,int,int,int,int,int);

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display   *mydisplay;
   int        myscreen;
   Window     mywindow;
   XEvent     myevent;
   GC         mygc;

   int        myred;
   int        mygreen;
   int        myblue;
   int        mycyan;
   int        mymagenta;
   int        myyellow;
   int        mylightseagreen;
   int        myrgb;
   char       myendflag; /* flag to test the end of the event loop */
   char       mydispname[82]; /* display name */

/* Variable initialization */

   mydispname[0]=0;

   printf ("\n");
   printf ("Example 4\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */
```

```
/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

   mywindow=XCreateSimpleWindow (mydisplay,
                     DefaultRootWindow (mydisplay),
                     0,0,600,400,                    /* size */
                     2,                              /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay,mywindow,"Hello  X  world !");

   XSelectInput (mydisplay,mywindow,ExposureMask | ButtonPressMask);
   XMapRaised (mydisplay,mywindow);
   mygc=XCreateGC (mydisplay,mywindow,0L,NULL);

/* Initialize new entries in the default color map */

   icomap_4 (mydisplay,
             &myred,           &mygreen,     &myblue,
             &mycyan,          &mymagenta,   &myyellow,
             &mylightseagreen, &myrgb);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case Expose:

/*          Get rid of all the other Expose events */

            while (   XCheckTypedEvent (mydisplay,Expose,&myevent)   );

/*          Graphical output of a few primitives */

            g_out_4 (mydisplay,mywindow,mygc,
                     myred,           mygreen,     myblue,
                     mycyan,          mymagenta,   myyellow,
                     mylightseagreen, myrgb);

            break;

         case ButtonPress:

/*          Turn on end flag */
```

```
            myendflag= TRUE ;

            break;

        } /* switch (myevent.type) */

    } /* while ( ! myendflag) */

/* Terminate graphical output */

    XFreeGC (mydisplay,mygc);
    XDestroyWindow (mydisplay,mywindow);
    XCloseDisplay (mydisplay);

    printf ("\n");
    printf ("Red            color index = %8d\n",myred);
    printf ("Green          color index = %8d\n",mygreen);
    printf ("Blue           color index = %8d\n",myblue);
    printf ("Cyan           color index = %8d\n",mycyan);
    printf ("Magenta        color index = %8d\n",mymagenta);
    printf ("Yellow         color index = %8d\n",myyellow);
    printf ("Lightseagreen color index = %8d\n",mylightseagreen);
    printf ("RGB            color index = %8d\n",myrgb);

    printf ("\n");
    printf ("That's all folks !\n");

    printf ("\n");
    return (0);

} /* main - ex4 */
```

====== g_out_4.c ======

```
/*****************************************************************/
/*                                                               */
/* g_out_4                                                       */
/* Graphical output of a few primitives.                         */
/* This function is called by the main program ex4.              */
/* Some attributes are changed.                                  */
/* New allocated colors are used.                                */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <string.h> /* strcpy,strlen */
#include <X11/Xlib.h>

void g_out_4 (
   Display   *mydisplay,
   Window     mywindow,
   GC         mygc,
   int        myred,
   int        mygreen,
   int        myblue,
   int        mycyan,
   int        mymagenta,
   int        myyellow,
   int        mylightseagreen,
   int        myrgb
) {

/* Variable declaration */

   int ixpos;           /* x position in pixels */
   int iypos;           /* y position in pixels */
   int mylinewidth;     /* line width in pixels */
   int myscreen;
   char myhelptext[82]; /* text to be displayed in the window */
   char myinfotext[82]; /* text to be displayed in the window */

/* Variable initialization */

   ixpos=100;
   iypos=100;
   mylinewidth=30;
   myscreen=DefaultScreen (mydisplay);
   strcpy (myinfotext,
       " Use the window manager to increase window size ");
   strcpy (myhelptext,
     " Press a mouse button on this window to terminate application ");

/* Draw info text */

   XSetForeground (mydisplay,mygc,WhitePixel (mydisplay,myscreen));
   XSetBackground (mydisplay,mygc,BlackPixel (mydisplay,myscreen));

   XDrawImageString (mydisplay,mywindow,mygc,
                     100,20,myinfotext,strlen (myinfotext));

/* Change some line attributes */

   XSetLineAttributes (mydisplay,mygc,
                       mylinewidth,LineSolid,CapRound,JoinMiter);

/* Draw lines each with a different color */
```

```
    XSetForeground (mydisplay,mygc,myred);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,mygreen);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,myblue);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,mycyan);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,mymagenta);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,myyellow);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,mylightseagreen);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

    iypos+=50;
    XSetForeground (mydisplay,mygc,myrgb);
    XDrawLine (mydisplay,mywindow,mygc,
               ixpos,iypos,ixpos+400,iypos);

/* ================================================================== */

    XSetLineAttributes (mydisplay,mygc,
                        5,LineSolid,CapButt,JoinMiter);

    XSetForeground (mydisplay,mygc,mylightseagreen);

/* Draw an arc */

    XDrawArc (mydisplay,mywindow,mygc,
              540,180,          200,150,          45*64,270*64);
/*    (upper left corner)  (width,height) (start and arc angle * 64) */

    XSetForeground (mydisplay,mygc,myred);

/* Draw a rectangle */

    XDrawRectangle (mydisplay,mywindow,mygc,
                    540,180,          200,150);
/*           (upper left corner)  (width,height) */

    XSetForeground (mydisplay,mygc,myblue);

/* Draw help text */

    XDrawString (mydisplay,mywindow,mygc,
                 100,580,myhelptext,strlen (myhelptext));

/* Reset attributes to their usual values */
```

```
   XSetForeground (mydisplay,mygc,0);
   XSetBackground (mydisplay,mygc,1);
   XSetLineAttributes (mydisplay,mygc,
                       0,LineSolid,CapButt,JoinMiter);
/*                     |                                          */
/*                     +---> line width is one but output is faster  */

   XFlush (mydisplay);

} /* g_out_4 */
```

```
====== icomap_4.c ======

/*****************************************************************/
/*                                                               */
/* icomap_4                                                      */
/* Initialize new entries in the default color map.              */
/* Called by the main program ex4.                               */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <X11/Xlib.h>

void icomap_4 (
   Display *mydisplay,
   int *myred,
   int *mygreen,
   int *myblue,
   int *mycyan,
   int *mymagenta,
   int *myyellow,
   int *mylightseagreen,
   int *myrgb
) {

/* Variable declaration */

   int       myscreen;
   Colormap  mycolormap;
   XColor    mycolor;
   Status    mystatus;

   int myredcoord;   /* red   RGB coordinate */
   int mygreencoord; /* green RGB coordinate */
   int mybluecoord;  /* blue  RGB coordinate */
   char mycolorstring[82]; /* color string needed by XParseColor */

/* Variable initialization */

   myscreen=DefaultScreen (mydisplay);
   mycolormap=DefaultColormap (mydisplay,myscreen);

/* ============================================================== */

/* Allocate some named colors */
/* Available color names are in the file /usr/lib/X11/rgb.txt */

/* Red */

   mystatus=XParseColor (mydisplay,mycolormap,"Red",&mycolor);
   if (mystatus == NULL) {
      fprintf (stderr,"XParseColor failed\n");
      exit (1);
   } /* if */

   mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
   if (mystatus == NULL) {
      fprintf (stderr,"XAllocColor failed\n");
      exit (1);
   } /* if */

   (*myred)=mycolor.pixel;

/* Green */
```

```
    mystatus=XParseColor (mydisplay,mycolormap,"Green",&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*mygreen)=mycolor.pixel;

 /* Blue */

    mystatus=XParseColor (mydisplay,mycolormap,"Blue",&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*myblue)=mycolor.pixel;

 /* Cyan */

    mystatus=XParseColor (mydisplay,mycolormap,"Cyan",&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*mycyan)=mycolor.pixel;

 /* Magenta */

    mystatus=XParseColor (mydisplay,mycolormap,"Magenta",&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*mymagenta)=mycolor.pixel;

 /* Yellow */

    mystatus=XParseColor (mydisplay,mycolormap,"Yellow",&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
```

```
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*myyellow)=mycolor.pixel;

/* LightSeaGreen */

    mystatus=XParseColor (mydisplay,mycolormap,"LightSeaGreen",
                          &mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*mylightseagreen)=mycolor.pixel;

/* ================================================================== */

/* Allocate RGB colors */

    myredcoord   = (43.67/100.0)*65535;
    mygreencoord = (12.77/100.0)*65535;
    mybluecoord  = (74.14/100.0)*65535;

/* Put in a string 3 hexadecimal RGB coordinates */

    sprintf (mycolorstring,"#%4.4x%4.4x%4.4x",
                           myredcoord,mygreencoord,mybluecoord);

    mystatus=XParseColor (mydisplay,mycolormap,mycolorstring,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XParseColor failed\n");
       exit (1);
    } /* if */

    mystatus=XAllocColor (mydisplay,mycolormap,&mycolor);
    if (mystatus == NULL) {
       fprintf (stderr,"XAllocColor failed\n");
       exit (1);
    } /* if */

    (*myrgb)=mycolor.pixel;

} /* icomap_4 */
```

====== work4 ======

Exercicios relativos ao programa ex4:

1 - Chamar em alternativa 2 default colormaps ligeiramente diferentes, correr o programa a partir de 2 janelas distintas seleccionando colormaps diferentes e verificar as diferencas nos indices de cor.

2 - Criar um colormap com 100 tons de azul e colocar no background um degrade desde o azul escuro ate' ao azul claro.

3 - Criar um colormap com as cores do arcoiris e desenhar no monitor o contour fill relativo 'a funcao z=cos (3*x^2 + 5*y^3).

4 - Passar a usar cores no programa de graficos 2D.

5 - Visualizar o mapa de cores corrente com o programa desenvolvido no exercicio 2 do work3.

====== ex5.c ======

```
/*****************************************************************/
/*                                                               */
/* ex5                                                           */
/* Example 5                                                     */
/*                                                               */
/* New features:                                                 */
/*    - get window attributes                                    */
/*    - fill polygon primitives are used in g_out_5              */
/*    - picture is scaled when a configure notify event is reported */
/*    - a new random polygon is generated when an expose event is */
/*      reported                                                 */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy,strcat */
#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

void g_out_5 (Display *,Window,GC,int,int);

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display   *mydisplay;
   int        myscreen;
   Window     mywindow;
   XEvent     myevent;
   GC         mygc;

   int        myeventcount; /* event count */
   int        mywinwidth;  /* window width */
   int        mywinheight; /* window height */
   char       myendflag; /* flag to test the end of the event loop */
   char       mydispname[82]; /* display name */
   XWindowAttributes myattributes; /* width, height, etc. */

/* Variable initialization */

   myeventcount=0;
   mydispname[0]=0;

   mywinwidth  = 600;
   mywinheight = 400;

   printf ("\n");
   printf ("Example 5\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */
```

```
/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,mywinwidth,mywinheight,
                    2,                          /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay,mywindow,"Hello  X  world !");

   XSelectInput (mydisplay,mywindow,
                 ExposureMask | ButtonPressMask | StructureNotifyMask);
/*                                                          ^       */
/* A new mask was added because of ConfigureNotify events   |       */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +       */

   XMapRaised (mydisplay,mywindow);

   mygc=XCreateGC (mydisplay,mywindow,0L,NULL);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case Expose:

            myeventcount++;
            printf ("%3d === Expose event\n",myeventcount);

/*          Get rid of all the other Expose events */

            while (   XCheckTypedEvent (mydisplay,Expose,&myevent)   );

/*          Graphical output of a few primitives */

            g_out_5 (mydisplay,mywindow,mygc,
                     mywinwidth,mywinheight);

            break;

         case ConfigureNotify:

            myeventcount++;
            printf ("%3d === Configure notify event\n",myeventcount);
```

```
/*          Inquire window attributes */

            XGetWindowAttributes (mydisplay,mywindow,&myattributes);

            mywinwidth  = myattributes.width;
            mywinheight = myattributes.height;

            break;

         case ButtonPress:

            myeventcount++;
            printf ("%3d === Button press event\n",myeventcount);

/*          Turn on end flag */

            myendflag= TRUE ;

            break;

      } /* switch (myevent.type) */

   } /* while ( ! myendflag) */

/* Terminate graphical output */

   XFreeGC (mydisplay,mygc);
   XDestroyWindow (mydisplay,mywindow);
   XCloseDisplay (mydisplay);

   printf ("\n");
   printf ("That's all folks !\n");

   printf ("\n");
   return (0);

} /* main - ex5 */
```

====== g_out_5.c ======

```
/*****************************************************************/
/*                                                               */
/* g_out_5                                                       */
/* Graphical output of a few primitives.                         */
/* This function is called by the main program ex5.              */
/* Fill polygon functions are used.                              */
/* Random polygons are generated.                                */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-10                                           */
/* Author : Alvaro Azevedo                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* malloc,free,rand,RAND_MAX */
#include <string.h> /* strcpy,strlen */
#include <X11/Xlib.h>

void g_out_5 (
   Display   *mydisplay,
   Window     mywindow,
   GC         mygc,
   int        mywinwidth, /* current window width */
   int        mywinheight /* current window height */
) {

/* Variable declaration */

   int iiaux;           /* auxiliary integer variable */
   int ipoin;           /* current point */
   int npoin;           /* number of points */

   double rxmin,rxmax,rymin,rymax; /* real world min. and max.values */
   double wxmin,wxmax,wymin,wymax; /* window     min. and max.values */
   double scalx,scaly,origx,origy; /* scales and origins */

   char myhelptext[82]; /* text to be displayed in the window */
   char myinfotext[82]; /* text to be displayed in the window */
   XPoint *mypoints;   /* struct with x and y coor.of several points */

/* Variable initialization */

   npoin=10;
   strcpy (myinfotext,
       " Use the window manager to change window size ");
   strcpy (myhelptext,
     " Press a mouse button on this window to terminate application ");

/* Memory allocation */

   mypoints=(XPoint *)malloc (   npoin*sizeof (XPoint)   );

/* Clear window */

   XClearWindow (mydisplay,mywindow);

/* Draw borders */

   XDrawRectangle (mydisplay,mywindow,mygc,
                   10,10,mywinwidth-20,50);

   XDrawRectangle (mydisplay,mywindow,mygc,
                   10,70,mywinwidth-20,mywinheight-80);

/* Calculate scales and origins in x and y */
```

```
    rxmin=0.0;
    rxmax= RAND_MAX ;

    rymin=0.0;
    rymax= RAND_MAX ;

    wxmin=10.0;
    wxmax=(double)mywinwidth-10.0;

    wymin=70.0;
    wymax=(double)mywinheight-10.0;

    scalx=(wxmax-wxmin)/(rxmax-rxmin);
    scaly=(wymax-wymin)/(rymax-rymin);

    if (scalx > scaly) {
       scalx=scaly;
    } else {
       scaly=scalx;
    } /* if */

    origx=(wxmin+wxmax)/2.0-scalx*(rxmin+rxmax)/2.0;
    origy=(wymin+wymax)/2.0-scaly*(rymin+rymax)/2.0;

/* Generate a random polygon */

    for (ipoin=0 ; ipoin < npoin ; ipoin++) {

       iiaux=rand (); /* random values in the range [0,RAND_MAX] */
       mypoints[ipoin].x=(short)(scalx*iiaux+origx+0.5);

       iiaux=rand (); /* random values in the range [0,RAND_MAX] */
       mypoints[ipoin].y=(short)(scaly*iiaux+origy+0.5);

    } /* for (ipoin) */

/* Draw polygon */

    XFillPolygon (mydisplay,mywindow,mygc,
                  mypoints,npoin,Complex,CoordModeOrigin);

/* Draw info text */

    XDrawImageString (mydisplay,mywindow,mygc,
                      20,30,myinfotext,strlen (myinfotext));

/* Draw help text */

    XDrawImageString (mydisplay,mywindow,mygc,
                      20,50,myhelptext,strlen (myhelptext));

/* Free allocated memory */

    free (   (void *)mypoints   );

    XFlush (mydisplay);

} /* g_out_5 */
```

====== work5 ======

Exercicios relativos ao programa ex5:

1 - Substituir a funcao XFillPolygon pela funcao XDrawLines.
Experimentar as terminacoes de linhas (CapButt, CapRound, etc.)
e os modos de juncao de linhas (JoinRound, JoinMiter e JoinBevel).
Nao esquecer definir uma espessura suficientemente grande para se
visualizar as diferentes terminacoes das linhas.

2 - Por o programa de graficos 2D a detectar a alteracao do tamanho
da janela e a ajustar as respectivas escalas.

====== ex6.c ======

```
/*****************************************************************/
/*                                                               */
/* ex6                                                           */
/* Example 6                                                     */
/*                                                               */
/* New features:                                                 */
/*    - Font characteristics.                                    */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-23                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy, strcat */

#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

#define MAXCHR  82 /* maximum lenght of one string line */

void icomap_4 (Display *,
               int *,int *,int *,int *,int *,int *,int *,int *);
void g_out_6 (Display *, Window, GC, GC, XFontStruct *, int,
              int, int, int, int, char *, int, int);

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display     *mydisplay;
   int          myscreen;
   Window       mywindow;
   XEvent       myevent;
   GC           mygc_1;
   GC           mygc_2;
   XFontStruct *myfont;

   int          mywinwidth;        /* window width */
   int          mywinheight;       /* window height */
   char         myendflag;         /* test the end of the event loop */
   char         myfontname[MAXCHR]; /* font name */
   char         mydispname[MAXCHR]; /* display name */
   char         ttext[MAXCHR];      /* input text line */

   int          ixtor;       /* text origin x coordinate */
   int          iytor;       /* text origin y coordinate */

   int          myred;
   int          mygreen;
   int          myblue;
   int          mycyan;
   int          mymagenta;
   int          myyellow;
   int          mylightseagreen;
   int          myrgb;

/* Variable initialization */

   mydispname[0]=0;
```

```
   ixtor=100;
   iytor=100;

   printf ("\n");
   printf ("Example 6\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Text line to be used as demo */

   printf ("\n");
   printf ("Type in some text (80 character) ... ",ttext);
   gets (ttext);

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

/* Main window */

   mywinwidth  = DisplayWidth (mydisplay,myscreen);
   mywinheight = 300;

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,mywinwidth,mywinheight,
                    2,                              /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay, mywindow, "Hello  X  world !");

   XSelectInput (mydisplay, mywindow, ExposureMask | ButtonPressMask);

   XMapWindow (mydisplay, mywindow);

/* Create two Graphics Contexts (one for each font) */

   mygc_1=XCreateGC (mydisplay ,mywindow, 0L, NULL);
   mygc_2=XCreateGC (mydisplay ,mywindow, 0L, NULL);

/* Set color map */

   icomap_4 (mydisplay,
             &myred,           &mygreen,     &myblue,
             &mycyan,          &mymagenta,   &myyellow,
             &mylightseagreen, &myrgb);
```

```
/* Load the best possible font */

   strcpy (myfontname,"helvbo24");

   if ( (myfont= XLoadQueryFont (mydisplay, myfontname)) == NULL) {
      fprintf (stderr,"Cannot load font: %s\n", myfontname);

/*    If font could not be loaded try a second one */

      strcpy (myfontname,"courbo24");

      if ( (myfont= XLoadQueryFont (mydisplay, myfontname)) == NULL) {
         fprintf (stderr,"Cannot load font: %s\n", myfontname);

/*       If 2nd font could not also be loaded terminate application */

         fprintf (stderr,"At least one big font must be found.\n");
         fprintf (stderr,"Cannot proceed.\n");
         exit (1);

      } /* if (myfont) - 2nd try */

   } /* if (myfont) - 1st try */

/* Load chosen font on second Graphics Context */

   XSetFont (mydisplay, mygc_2, myfont->fid);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case ButtonPress:

/*          Press a mouse button to terminate application */

            myendflag= TRUE ;

            break;

         case Expose:

/*          Discard all other Expose events */

            while (   XCheckTypedEvent (mydisplay,Expose,&myevent)   );

            g_out_6 (mydisplay, mywindow, mygc_1, mygc_2, myfont,
                     myscreen, myred, mygreen, myblue, mymagenta,
                     ttext, ixtor, iytor);

            break;

      } /* switch (myevent.type) */

   } /* while ( ! myendflag) */

/* Terminate graphical output */

   XFreeFont (mydisplay, myfont);
   XFreeGC (mydisplay, mygc_1);
   XFreeGC (mydisplay, mygc_2);
   XDestroyWindow (mydisplay, mywindow);
```

```
    XCloseDisplay (mydisplay);

    printf ("\n");
    printf ("That's all folks !\n");

    printf ("\n");
    return (0);

} /* main - ex6 */
```

====== g_out_6.c ======

```
/*****************************************************************/
/*                                                               */
/* g_out_6                                                       */
/* Graphical output of user given text line.                     */
/* Display font characteristics.                                 */
/* This function is called by the main program ex6.              */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-23                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <string.h> /* strcpy, strlen */

#include <X11/Xlib.h>

#define MAXCHR  82 /* maximum lenght of one string line */

void g_out_6 (
   Display     *mydisplay,
   Window       mywindow,
   GC           mygc_1,
   GC           mygc_2,
   XFontStruct *myfont,
   int          myscreen,
   int          myred,
   int          mygreen,
   int          myblue,
   int          mymagenta,
   char        *ttext,
   int          ixtor,
   int          iytor
) {

/* Variable declaration */

   XCharStruct  myoverall;

   int         iiaux;
   int         ixdor;       /* text bounding box width */
   int         ixupl;       /* x upper coord. - text bounding box */
   int         iydor;       /* text bounding box height */
   int         iyupl;       /* y upper coord. - text bounding box */
   int         ilast;
   int         ileng;

   int         myascent;
   int         mydescent;
   int         mydirection;
   char        ttaux[MAXCHR];      /* auxiliary text line */

/* Reset dialog window */

/* Display help message */

   XSetForeground (mydisplay, mygc_1, mymagenta);
   strcpy (ttaux,"Press a mouse button to terminate application");
   XDrawString (mydisplay, mywindow, mygc_1, 40, 270,
                ttaux, strlen (ttaux) );

/* Display user user defined text line */

   XDrawString (mydisplay, mywindow, mygc_2, ixtor, iytor,
                ttext, strlen (ttext) );
```

```
/* Mark origin of string */

   XSetForeground (mydisplay, mygc_1, mygreen);
   XDrawLine (mydisplay, mywindow, mygc_1,  70, iytor, ixtor, iytor);
   XDrawLine (mydisplay, mywindow, mygc_1, ixtor, 130, ixtor, iytor);
   XDrawLine (mydisplay, mywindow, mygc_1, 95, 105, 50, 150);
   XDrawString (mydisplay, mywindow, mygc_1, 40, 170,"Origin",
                strlen ("Origin") );

/* Text line width dimension */

   XSetForeground (mydisplay, mygc_1, myblue);
   ileng= XTextWidth (myfont, ttext, strlen (ttext) );
   ilast=ileng+ixtor;
   XDrawLine (mydisplay, mywindow, mygc_1, 98, 110,
              ilast+2, 110);
   XDrawLine (mydisplay, mywindow, mygc_1, ilast, 105, ilast, 115);
   iiaux= (ixtor+ileng/2-30);
   strcpy (ttaux,"XTextWidth");
   XDrawString (mydisplay, mywindow, mygc_1, iiaux, 125,
                ttaux, strlen (ttaux) );

/* Text line logical bounding box */

   XSetForeground (mydisplay, mygc_1, myred);
   XTextExtents (myfont, ttext, strlen (ttext), &mydirection,
                 &myascent, &mydescent, &myoverall);
   ixupl=ixtor- (myoverall.lbearing);
   iyupl=iytor-myascent;
   ixdor=(myoverall.lbearing) + (myoverall.rbearing);
   iydor=myascent+mydescent;
   XDrawRectangle (mydisplay, mywindow, mygc_1,
                   ixupl, iyupl, ixdor, iydor);
   strcpy (ttaux,"Logical Bounding Box");
   XDrawString (mydisplay, mywindow, mygc_1, ixupl+10,
                iyupl-10, ttaux, strlen (ttaux) );

/* Additional information */

   XSetForeground (mydisplay, mygc_1,
                   BlackPixel (mydisplay, myscreen) );
   strcpy (ttaux,
           "Direction of text is       ... 'FontLeftToRight'");
   XDrawString (mydisplay, mywindow, mygc_1, 500, 150,
                ttaux, strlen (ttaux) );
   sprintf (ttaux,"First defined character is ... '%c'",
            myfont->min_char_or_byte2);
   XDrawString (mydisplay, mywindow, mygc_1, 500, 170,
                ttaux, strlen (ttaux) );
   sprintf (ttaux,"Last defined character is  ... '%c'",
            myfont->max_char_or_byte2);
   XDrawString (mydisplay, mywindow, mygc_1, 500, 190,
                ttaux, strlen (ttaux) );
   sprintf (ttaux,"Default character is       ... '%c'",
            myfont->max_char_or_byte2);
   XDrawString (mydisplay, mywindow, mygc_1, 500, 210,
                ttaux, strlen (ttaux) );
   strcpy (ttaux,"Minimum bounds are");
   XDrawString (mydisplay, mywindow, mygc_1, 500, 240,
                ttaux, strlen (ttaux) );
   XDrawRectangle (mydisplay, mywindow, mygc_1,
                   690,240-(myfont->min_bounds.ascent),
                   (myfont->min_bounds.width),
                   (myfont->min_bounds.ascent) +
                   (myfont->min_bounds.descent) );
   strcpy (ttaux,"Maximum bounds are");
   XDrawString (mydisplay, mywindow, mygc_1, 500, 270,
                ttaux, strlen (ttaux) );
   XDrawRectangle (mydisplay, mywindow, mygc_1,
```

```
                    690,270-(myfont->max_bounds.ascent),
                    (myfont->max_bounds.width),
                    (myfont->max_bounds.ascent) +
                    (myfont->max_bounds.descent) );

} /* g_out_6 */
```

====== work6 ======

Exercicios relativos ao programa ex6:

1 - Usar XListFonts ou XListFontsWithInfo para obter uma lista das fonts disponiveis.

2 - No programa de graficos 2D permitir a possibilidade de utilizar fonts alternativas.

====== ex7.c ======

```
/*****************************************************************/
/*                                                               */
/* ex7                                                           */
/* Example 7                                                     */
/*                                                               */
/* New features:                                                 */
/*    - keyboard input handling - function kevent_7              */
/*    - "roll-up" menu          - function rollup_7              */
/*    - circular string vector                                   */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-18                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* malloc,exit */
#include <string.h> /* strcpy,strcat */

#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

#define MAXVCT 200
#define MAXCHR  82

void kevent_7 (Display *, Window, GC, XEvent, XFontStruct *,
               int, int, int *, int *, char *, char *, char **);
void rollup_7 (Display *, Window, GC, int, int, int, int, int,
               char *, char **);

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display           *mydisplay;
   int                myscreen;
   Window             mywindow;
   XEvent             myevent;
   GC                 mygc;
   XWindowAttributes  myattributes; /* width, height, etc. */
   XFontStruct       *myfont;
   GContext           mygcontx;

   int        mywinwidth;         /* window width */
   int        mywinheight;        /* window height */
   char       myendflag;          /* test the end of the event loop */
   char       myfontname[MAXCHR]; /* font name */
   char       mydispname[MAXCHR]; /* display name */
   char       mykey[MAXCHR];     /* string with tha last key pressed */
   char       tline[MAXCHR];     /* current text line */
   char       ttext[MAXCHR];     /* help message */
   char     **tvect;

   int         iposi;       /* auxiliary integer */
   int         fdnlf;       /* flag - did not load font */
   int         iline;       /* current line of string vector */
   int         ixcur;       /* text x current coordinate */
   int         ixtor;       /* text x origin coordinate  */
   int         iytor;       /* text y origin coordinate  */
   int         nlina;       /* n. of lines for text in wndow */
   int         mytextheight;
```

```
/* Variable initialization */

   mydispname[0]=0;

   mywinwidth  = 600;
   mywinheight = 400;
   iline = 1;
   tline[0] = ' ';              /* initialize text line with cursor */
   tline[1] = '\0';

/* Allocate and initialize circular string vector */

   tvect=(char **)malloc ((MAXVCT+1)*sizeof (char *));

   for (iposi=0 ; iposi <= MAXVCT ; iposi++) {
      tvect[iposi]=(char *)malloc ((MAXCHR+1)*sizeof (char));
      tvect[iposi][0]='\0';
   } /* for (iposi) */

   printf ("\n");
   printf ("Example 7\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

/* Main window */

   mywindow=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,mywinwidth,mywinheight,
                    2,                          /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay,mywindow,"Hello  X  world !");

   XSelectInput (mydisplay,mywindow,
                 ExposureMask | StructureNotifyMask | KeyPressMask);
/*                                                    ^                */
/* A new mask was added because of KeyPress events    |                */
/* - - - - - - - - - - - - - - - - - - - - - - - - - -+                */

   XMapWindow (mydisplay,mywindow);
```

```
/* Create a Graphics Context */

   mygc=XCreateGC (mydisplay ,mywindow, 0L, NULL);

/* Load the best possible font */

   strcpy (myfontname, "courr14");

   fdnlf = FALSE ;

   if ( (myfont= XLoadQueryFont (mydisplay, myfontname)) == NULL) {
      fprintf (stderr,"Cannot load font: %s\n", myfontname);

/*    If font could not be loaded try a second one */

      strcpy (myfontname, "courr12");

      if ( (myfont= XLoadQueryFont (mydisplay, myfontname)) == NULL) {
         fprintf (stderr,"Cannot load font: %s\n", myfontname);

         fdnlf = TRUE ;

/*       If 2nd font also could not be loaded do not try again; */
/*       accept the default font and get its attributes         */

         mygcontx= XGContextFromGC (mygc);      /*   get GContext ID */
/*                                 ^                               */
/*       A GC must have been created __|                           */

         myfont= XQueryFont (mydisplay, mygcontx); /* font attrib.s */
         fprintf (stderr,"Using default font.\n");

      } /* if (myfont) - 2nd try */

   } /* if (myfont) - 1st try */

/* Load chosen font if it is not the default font */

   if ( ! fdnlf ) XSetFont (mydisplay, mygc, myfont->fid);

/* Set text default values */

   mytextheight = myfont->ascent + myfont->descent;
   ixtor = 4;              /* text orig. x coord. */
   iytor = mywinheight-4; /* text orig. y coord.;depends on win.size */
   ixcur = ixtor;

/* N. of lines on window; set a margin of 3 lines for header */
   nlina = (mywinheight/mytextheight)-3;

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case Expose:

/*          Discard all other Expose events */

            while (   XCheckTypedEvent (mydisplay,Expose,&myevent)   );

/*          Reset help message */
```

```
            strcpy (ttext,
            "Type in...  and see text roll up! ('quit' to terminate)");
            XDrawString (mydisplay, mywindow, mygc, 4,14,
                         ttext, strlen (ttext) );
            XDrawLine (mydisplay, mywindow, mygc, 4,18, 500, 18);

/*          Reset dialog lines */

            rollup_7 (mydisplay, mywindow, mygc,
                      ixtor, iytor, iline, nlina, mytextheight,
                      tline, tvect);

            break;

         case ConfigureNotify:

/*          Inquire new window attributes */

            XGetWindowAttributes (mydisplay,mywindow,&myattributes);

            mywinwidth  = myattributes.width;
            mywinheight = myattributes.height;

/*          Reset values for text orig. y coord. and number of lines */

            iytor = mywinheight-4;
            nlina = mywinheight/mytextheight-3; /* as before... */

            break;

         case KeyPress:

/*          React accordingly to the specific key pressed */

            kevent_7 (mydisplay, mywindow, mygc,
                      myevent, myfont,
                      ixtor, iytor, &iline, &ixcur,
                      mykey, tline, tvect);

/*          If last key is CR react accordingly */

            if (   ( ! strcmp (mykey, "Return")    ) ||
                   ( ! strcmp (mykey, "Enter")     ) ||
                   ( ! strcmp (mykey, "Linefeed")  )   ) {

/*             Line is completed; roll up text */

               rollup_7 (mydisplay, mywindow, mygc,
                         ixtor, iytor, iline, nlina, mytextheight,
                         tline, tvect);

            } /* if */

/*          Verify last completed input line against exit condition */

            if ( ! strcmp (tvect[iline-1],"quit") )
                                                myendflag = TRUE ;

            break;

      } /* switch (myevent.type) */

   } /* while ( ! myendflag) */

/* Terminate graphical output */

   if ( ! fdnlf) XFreeFont (mydisplay, myfont);
   XFreeGC (mydisplay,mygc);
   XDestroyWindow (mydisplay,mywindow);
```

```
    XCloseDisplay (mydisplay);

    printf ("\n");
    printf ("That's all folks !\n");

    printf ("\n");
    return (0);

} /* main - ex7 */
```

====== kevent_7.c ======

```
/*****************************************************************/
/*                                                               */
/* kevent_7                                                      */
/* Process KeyPress type events.                                 */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-18                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

typedef char *caddr_t;

#include <stdio.h>
#include <string.h> /* strcat, strcpy, strlen */

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/Xatom.h>
#include <X11/keysym.h>

#define MAXVCT 200   /* maximum length of commands vector */
#define MAXCHR  82   /* maximum length of string */
#define MAXKEY  20   /* maximum length of one keysym */

void kevent_7 (
   Display      *mydisplay,
   Window        mywindow,
   GC            mygc,
   XEvent        myevent,
   XFontStruct  *myfont,
   int           ixtor,   /* original text x coordinate      */
   int           iytor,   /* original text y coordinate      */
   int          *iline,   /* curr. line in vector of strings */
   int          *ixcur,   /* current  text x coordinate      */
   char         *mykey,   /* "name" of last key pressed      */
   char         *tline,   /* current text line               */
   char        **tvect    /* circular vector of strings      */
) {

   KeySym        mykeysym;

   int ileng;
   int kount;
   int mybufsize = MAXKEY;
   char *tprov;
   char clr_char[MAXKEY];
   char adv_curs[MAXKEY];
   char mybuffer[MAXKEY];

   strcpy (clr_char,"_ ");
   strcpy (adv_curs," _");

/* Get the keysym of the key */

   kount= XLookupString ( (XKeyEvent *) &myevent, mybuffer, mybufsize,
                          &mykeysym, NULL);

/* Translate the key symbol into a string */
/* (strip it from the leading "XK_")      */

   tprov= XKeysymToString (mykeysym);
   strcpy (mykey, tprov);

/* Do the right thing with as many keysyms as possible */
```

```
/*****************************************************************/

   if (   (mykeysym == XK_Return) || (mykeysym == XK_KP_Enter) ||
                                     (mykeysym == XK_Linefeed)   ) {

/*    Key is CR or equivalent */

      if ( ((*ixcur) == ixtor) && (strlen (tline) == 1) ) {

/*       Text line is cursor and must be ignored */
         tline[0]='\0';
      } /* if (*ixcur) */

/*    Put current line on circular string vector */
      strcpy (tvect[*iline],tline);

/*    Reset position of last string on string vector */
      (*iline)++;
      if ( (*iline) > MAXVCT ) (*iline)=1;

/*    Reset text x current coordinate and text line */
      (*ixcur)=ixtor;
      tline[0]=' ';
      tline[1]='\0';

/*****************************************************************/

   } else if ( ((mykeysym >= XK_KP_Space)  &&
                  (mykeysym <= XK_KP_9)) ||
               ((mykeysym >= XK_space)  &&
                (mykeysym <= XK_asciitilde))  ) {

/*    Key is ASCII code between 39 and 126 */

      if (   (strlen (tline) + strlen (mybuffer)) >= MAXCHR   ) {

         XBell (mydisplay,100); /* does not fit on line */

      } else {

/*       Erase cursor and redraw it one position forward */

         if (  (strlen (tline) + strlen (mybuffer) + 1) < MAXCHR  ) {
            XDrawImageString (mydisplay, mywindow, mygc, *ixcur,
                              iytor, adv_curs, strlen (adv_curs) );

         } else {

/*          Just erase cursor - no space to redraw it forward */

            XDrawImageString (mydisplay, mywindow, mygc, *ixcur,
                              iytor, " ", strlen (" ") );
         } /* if */

/*       Add text to current text line */

         if ( ((*ixcur) == ixtor) && (strlen (tline) == 1) ) {
/*          First character is cursor and must be ignored */
            strcpy (tline, mybuffer);

         } else {

            strcat (tline, mybuffer);

         } /* else */

/*       Redraw current text line */
/*       XDrawString is faster but does not paint text cell */
```

```
            XDrawString (mydisplay, mywindow, mygc, *ixcur, iytor,
                         mybuffer, strlen (mybuffer) );
            (*ixcur)+= XTextWidth (myfont, mybuffer, strlen (mybuffer) );

         } /* if (strlen... */

/*****************************************************************/

      } else if ( (mykeysym >= XK_Shift_L) && (mykeysym <= XK_Hyper_R) ) {

         ; /* do nothing because it is a modifier key */

/*****************************************************************/

      } else if (  (mykeysym >= XK_F35)  ) {

         if (   strlen (mybuffer) == 0) {
            printf ("Unmapped function key\n");

         } else if (  (strlen (tline) + strlen (mybuffer)) >= MAXCHR  ) {
            XBell (mydisplay,100);

         } else {

/*          Proceed as in the previous case */

            if (  (strlen (tline) + strlen (mybuffer) + 1) < MAXCHR  ) {
               XDrawImageString (mydisplay, mywindow, mygc, *ixcur,
                                 iytor, " _", strlen (" _") );
            } else {
               XDrawImageString (mydisplay, mywindow, mygc, *ixcur,
                                 iytor, " ", strlen (" ") );
            } /* if */

/*          Add text to current text line */

            if ( ((*ixcur) == ixtor) && (strlen (tline) == 1) ) {
/*             First character is cursor and must be ignored */
               strcpy (tline, mybuffer);

            } else {

               strcat (tline, mybuffer);

            } /* else */

            XDrawString (mydisplay, mywindow, mygc, *ixcur, iytor,
                         mybuffer, strlen (mybuffer) );
            (*ixcur)+= XTextWidth (myfont, mybuffer, strlen (mybuffer) );

         } /* if (mybuffer) else */

/*****************************************************************/

      } else if ( (mykeysym == XK_BackSpace) ||
                  (mykeysym == XK_Delete) ) {

/*       Erase previous character if ... */

         if (  (ileng = strlen (tline)) > 0  ) {
            tline[ileng-1] = '\0';
            (*ixcur)=ixtor;
            (*ixcur)+= XTextWidth (myfont, tline, strlen (tline) );
            XDrawImageString (mydisplay, mywindow, mygc, *ixcur,
                              iytor, clr_char, strlen (clr_char) );

         } else {

/*          Do not erase if at beggining of text line */
```

```
            XBell (mydisplay, 100);
         } /* if */
/*****************************************************************/
      } else {
/*    Ignore other unpredicted situations */
         printf ("keysym %s is not handled\n",
                  XKeysymToString (mykeysym));
         XBell (mydisplay, 100);
      } /* if (mykeysym) else */
} /* kevent_7 */
```

```
====== rollup_7.c ======

/*********************************************************************/
/*                                                                   */
/* rollup_7                                                          */
/* Roll up text or reset text dialog window.                         */
/*                                                                   */
/* Version: 1.0                                                      */
/* Date   : 1992-05-13                                               */
/* Author : Ricardo Santos                                           */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h> /* malloc, free */
#include <X11/Xlib.h>

#define MAXVCT 200   /* maximum length of commands vector */
#define MAXCHR  82   /* maximum length of string          */

void rollup_7 (
   Display      *mydisplay,
   Window        mywindow,
   GC            mygc,
   int           ixtor,        /* original text x coordinate */
   int           iytor,        /* original text y coordinate */
   int           iline,        /* current position on text vector */
   int           nlina,        /* n. of text lines on window */
   int           mytextheight, /* height of a line           */
   char         *tline,        /* current text line          */
   char        **tvect         /* circular vector of strings */
) {

   int iiaux;
   int kline;
   int maxln;
   int iyaux;
   char *clr_line;

/* Set up cleaning line */

   clr_line=(char *) malloc ( MAXCHR *sizeof (char));

   for (iiaux=0 ; iiaux < MAXCHR ; iiaux++)
                          clr_line[iiaux] = ' ';

   clr_line[MAXCHR] = '\0'; /* NULL terminated string */

/* Reset current input text line; 2 cases are possible:              */
/*   - user in middle of an input --> reset unfinished text line     */
/*   - input line terminated (with CR) --> reset cursor              */

   iyaux=iytor;

   XDrawImageString (mydisplay, mywindow, mygc, ixtor, iyaux,
                     clr_line, strlen (clr_line) );
   XDrawImageString (mydisplay, mywindow, mygc, ixtor, iyaux,
                     tline, strlen (tline) );

/* Reset previous text lines up to minimum of:                       */
/*   - maximum number of lines already entered                       */
/*   - maximum number of lines available on window                   */

   maxln=nlina;
   if (nlina > MAXVCT ) maxln= MAXVCT ;

   for (iiaux=1 ; iiaux <= maxln ; iiaux++) {
      kline=( MAXVCT + (iline)-iiaux) % MAXVCT ;
```

```
        if (kline == 0) kline = MAXVCT ;
        iyaux-=mytextheight;
        XDrawImageString (mydisplay, mywindow, mygc, ixtor, iyaux,
                          clr_line, strlen (clr_line) );
        XDrawImageString (mydisplay, mywindow, mygc, ixtor, iyaux,
                          tvect[kline],strlen (tvect[kline])   );
    } /* for (iiaux) */

    free ((void *) clr_line);

} /* rollup_7 */
```

====== work7 ======

Exercicios relativos ao programa ex7:

1 - Acrescentar ao programa de graficos 2D a possibilidade de alterar as cores com comandos alfanumericos.

====== ex8.c ======

```
/*****************************************************************/
/*                                                               */
/* ex8                                                           */
/* Example 8                                                     */
/*                                                               */
/* New features:                                                 */
/*    - multi-window environment (child windows)                 */
/*    - draw text, lines, rectangles or arcs according to window */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-19                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy */

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

#define TRUE  1
#define FALSE 0

void icomap_4 (Display *,
               int *,int *,int *,int *,int *,int *,int *,int *);

int main (
   int argc,
   char **argv
) {

   Display        *mydisplay;
   int             myscreen;
   Window          mymain_win;
   Window          mycur_win;
   Window          myprev_win;
   Window          mywin_ul;
   Window          mywin_ur;
   Window          mywin_bl;
   Window          mywin_br;
   XEvent          myevent;
   GC              mygc;
   KeySym          mykeysym;
   XFontStruct    *myfont;

   int             ixini;
   int             ixend;
   int             iyini;
   int             iyend;
   int             itroc;
   int             mywinnum;
   char            myendflag; /* test the end of the event loop */

   int             iikey;
   char            mybuffer[82];
   char            tline[82];
   char            ttext[82];

   int             myred;
   int             mygreen;
   int             myblue;
   int             mycyan;
   int             mymagenta;
   int             myyellow;
   int             mylightseagreen;
```

```
   int             myrgb;

   int             mydispwidth;
   int             mydispheight;
   unsigned int    mywinwidth;
   unsigned int    mywinheight;

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display */

   if ( (mydisplay=XOpenDisplay ("")) == NULL ) {
      fprintf (stderr,"Cannot connect to server\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

/* Main window */

   mydispwidth=DisplayWidth (mydisplay,myscreen);
   mydispheight=DisplayHeight (mydisplay,myscreen);

   mywinwidth = mydispwidth;
   mywinheight= mydispheight;

   mymain_win=XCreateSimpleWindow (mydisplay,
                    DefaultRootWindow (mydisplay),
                    0,0,mywinwidth,mywinheight,
                    2,                          /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

/* Create Graphics Context */

   mygc=XCreateGC (mydisplay,mymain_win,0L,NULL);

/* Set color map */

   icomap_4 (mydisplay,
             &myred,          &mygreen,     &myblue,
             &mycyan,         &mymagenta,   &myyellow,
             &mylightseagreen, &myrgb);

   XSelectInput (mydisplay,mymain_win,ExposureMask | EnterWindowMask);
   XMapWindow (mydisplay,mymain_win);

/* Child windows (4) */

   mywinwidth = mywinwidth/2-20;
   mywinheight= mywinheight/2-40;

   mywin_ul=XCreateSimpleWindow (mydisplay,
                    mymain_win,
                    5, 5, mywinwidth, mywinheight,
                    2,                          /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */

   mywin_ur=XCreateSimpleWindow (mydisplay,
                    mymain_win,
                    mywinwidth+10, 5, mywinwidth, mywinheight,
                    2,                          /* border width */
                    BlackPixel (mydisplay,myscreen),  /* border */
                    WhitePixel (mydisplay,myscreen)); /* background */
```

```
   mywin_bl=XCreateSimpleWindow (mydisplay,
                     mymain_win,
                     5, mywinheight+10, mywinwidth, mywinheight,
                     2,                                  /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

   mywin_br=XCreateSimpleWindow (mydisplay,
                     mymain_win,
                     mywinwidth+10, mywinheight+10, mywinwidth,
                     mywinheight,
                     2,                                  /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

   XSelectInput (mydisplay,mywin_ul,EnterWindowMask | KeyPressMask   );
   XSelectInput (mydisplay,mywin_ur,EnterWindowMask | ButtonPressMask);
   XSelectInput (mydisplay,mywin_bl,EnterWindowMask | ButtonPressMask);
   XSelectInput (mydisplay,mywin_br,EnterWindowMask | ButtonPressMask);

/* Map all subwindows of main window in one single command */

   XMapSubwindows (mydisplay, mymain_win);

/* Set new (big) font */

   if ( (myfont= XLoadQueryFont (mydisplay, "courb24")) != NULL) {
      XSetFont (mydisplay, mygc, myfont->fid);
   } else {
      fprintf (stderr,"Cannot load font\n");
      exit (1);
   } /* if (myfont) */

/* Main X event loop */

   myprev_win=mywin_ur;
   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay, &myevent);

      switch (myevent.type) {

         case EnterNotify:

/*          Make current window the one returned by the struct  */
/*          XButtonEvent, i.e. the window the pointer is in     */

            mycur_win=myevent.xcrossing.window;
            ixini=myevent.xcrossing.x_root;
            iyini=myevent.xcrossing.y_root;

            mywinnum=0;
            if (mycur_win == mywin_ul) mywinnum=1;
            if (mycur_win == mywin_ur) mywinnum=2;
            if (mycur_win == mywin_bl) mywinnum=3;
            if (mycur_win == mywin_br) mywinnum=4;

            if (mywinnum >= 1 || mywinnum <= 4)  {
               XSetWindowBorder (mydisplay, myprev_win,
                                 BlackPixel (mydisplay, myscreen));
               XSetWindowBorder (mydisplay, mycur_win, mygreen);
               myprev_win=mycur_win;
            } /* if mywinnum */

            break;
```

```
         case ButtonPress:

            switch (mywinnum) {

               case 2:                                    /* Lines */
                  ixini=myevent.xbutton.x;
                  iyini=myevent.xbutton.y;

/*                Wait for the correct event on the current window */

                  do
                     XNextEvent (mydisplay, &myevent);
                  while (myevent.type != ButtonPress ||
                         myevent.xbutton.window != mycur_win) ;

                  ixend=myevent.xbutton.x;
                  iyend=myevent.xbutton.y;
                  XDrawLine (mydisplay, mywin_ur, mygc,
                             ixini, iyini, ixend, iyend);
                  break;

               case 3:                                    /* Rectangles */
                  ixini=myevent.xbutton.x;
                  iyini=myevent.xbutton.y;

/*                Wait for the correct event on the current window ! */
/*                One might think it would be better to use          */
/*                XWindowEvent or                                    */
/*                XCheckWindowEvent/XCheckTypedWindowEvent           */
/*                but these functions don't discard events in the    */
/*                queue; in the end (when the correct event was got) */
/*                all other unwanted events would be processed.      */

                  do
                     XNextEvent (mydisplay, &myevent);
                  while (myevent.type != ButtonPress ||
                         myevent.xbutton.window != mycur_win) ;

/*                Swap coord.s if needed; rectangles are allways     */
/*                defined by upper_left_x, upper_left_y,             */
/*                width, height                                      */

                  ixend=myevent.xbutton.x;
                  iyend=myevent.xbutton.y;
                  if (ixend < ixini) {
                     itroc=ixini;
                     ixini=ixend;
                     ixend=itroc;
                  } /* if ixend */
                  if (iyend < iyini) {
                     itroc=iyini;
                     iyini=iyend;
                     iyend=itroc;
                  } /* if iyend */

                  ixend-=ixini;
                  iyend-=iyini;

                  XDrawRectangle (mydisplay, mywin_bl, mygc,
                                  ixini, iyini, ixend, iyend);
                  break;

               case 4:                                    /* Arcs */
                  ixini=myevent.xbutton.x;
                  iyini=myevent.xbutton.y;

/*                Wait for the correct event on the current window ! */
/*                ... ... same as before ... ...                     */
```

```
                    do
                       XNextEvent (mydisplay, &myevent);
                    while (myevent.type != ButtonPress ||
                           myevent.xbutton.window != mycur_win) ;

/*                  Swap coord.s if needed; arcs are allways defined */
/*                  in terms of the enclosing rectangle              */
/*                  (upper_left_x, upper_left_y, width, height)      */

                    ixend=myevent.xbutton.x;
                    iyend=myevent.xbutton.y;
                    if (ixend < ixini) {
                       itroc=ixini;
                       ixini=ixend;
                       ixend=itroc;
                    } /* if ixend */
                    if (iyend < iyini) {
                       itroc=iyini;
                       iyini=iyend;
                       iyend=itroc;
                    } /* if iyend */

                    ixend-=ixini;
                    iyend-=iyini;

                    XDrawArc (mydisplay, mywin_br, mygc,
                              ixini, iyini, ixend, iyend, 0*64, 360*64);
                    break;

                 default:
                    XBell (mydisplay, 50);

              } /* switch (mywinnum) */

              break;

           case Expose:

/*            Reset all messages */

              strcpy (ttext,"*** TEXT *** - type 'q' here to end");
              XDrawImageString (mydisplay, mywin_ul, mygc,
                                30, 30, ttext, strlen (ttext) );

              strcpy (ttext,"*** LINES *** - 2 coordinates");
              XDrawImageString (mydisplay, mywin_ur, mygc,
                                30, 30, ttext, strlen (ttext) );

              strcpy (ttext,"*** RECTANGLES *** - 2 coordinates");
              XDrawImageString (mydisplay, mywin_bl, mygc,
                                30, 30, ttext, strlen (ttext) );

              strcpy (ttext,"*** ARCS *** - 2 coordinates");
              XDrawImageString (mydisplay, mywin_br, mygc,
                                30, 30, ttext, strlen (ttext) );

              break;

           case KeyPress:

/*            KeyPress events are only available on this window */
/*            so the confirmation might not be needed           */

              if (mywinnum == 1) {                          /* Text */

                 iikey=XLookupString (&myevent, tline, 20,
                                      &mykeysym, NULL);
                 if (iikey == 1 && tline[0] == 'q')
                                      myendflag = TRUE ;
```

```
                sprintf (mybuffer,"Key is ...%s...",
                         XKeysymToString (mykeysym) );

                XClearWindow (mydisplay, mywin_ul);
                strcpy (ttext,"*** TEXT *** - type 'q' here to end");
                XDrawImageString (mydisplay, mywin_ul, mygc,
                                  30, 30, ttext, strlen (ttext) );
                XDrawString (mydisplay,mywin_ul,mygc,
                             100,250,mybuffer,strlen (mybuffer));

              } /* if (mywinnum) */

              break;

        } /* switch (myevent.type) */

    } /* while (! myendflag) */

/* Terminate graphical output */

    XFreeFont (mydisplay, myfont);
    XFreeGC (mydisplay,mygc);
    XDestroySubwindows (mydisplay,mymain_win);
    XDestroyWindow (mydisplay,mymain_win);
    XCloseDisplay (mydisplay);

    printf ("\n");
    printf ("That's all folks !\n");

    printf ("\n");
    return (0);

} /* main - ex8 */
```

====== work8 ======

Exercicios relativos ao programa ex8:

1 - Fazer a manutencao das child windows de modo a que elas se mantenham sempre totalmente visiveis qualquer que seja o estado da main window (ConfigureNotify na main window).

2 - No programa de graficos 2D, colocar o dialogo com o utilizador numa child window.

====== ex9.c ======

```
/*****************************************************************/
/*                                                               */
/* ex9                                                           */
/* Example 9                                                     */
/*                                                               */
/* New features:                                                 */
/*    - pointer input handling:                                  */
/*      - use of a window as a menu                              */
/*      - ButtonPress - decision according to specified button   */
/*      - XQueryPointer - rubberbanding (lines, rectangles and arcs) */
/*    - recursive call of g_out_9 function for opening child windows */
/*    - XSetFunction as a GC attribute (drawing in Xor mode)     */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-21                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* exit */
#include <string.h> /* strcpy, strcat, strlen */
#include <aa/cutil.h>

#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

#define MAXVCT 200
#define MAXCHR  82

void icomap_4 (Display *,
               int *,int *,int *,int *,int *,int *,int *,int *);
void g_out_9 (Display *, Window, GC, int, Window, int, int, int, int);


int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display   *mydisplay;
   int        myscreen;
   Window     mymain_win;
   Window     myopt_win;
   XEvent     myevent;
   GC         mygc;

   int        mywinwidth;         /* window width */
   int        mywinheight;        /* window height */
   char       mydispname[MAXCHR]; /* display name */

   char       myendflag; /* test the end of the event loop */
   char       ttext[82];

   int        myred;
   int        mygreen;
   int        myblue;
   int        mycyan;
   int        mymagenta;
   int        myyellow;
   int        mylightseagreen;
   int        myrgb;
```

```
/* Variable initialization */

   mydispname[0]=0;

   mywinwidth  = 600;
   mywinheight = 400;

   printf ("\n");
   printf ("Example 9\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

/* Main window */

   mymain_win=XCreateSimpleWindow (mydisplay,
                     DefaultRootWindow (mydisplay),
                     0, 0, mywinwidth, mywinheight,
                     2,                          /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

/* Specify a title to be displayed on top of the window */

   XStoreName (mydisplay, mymain_win, "Hello  X  world !");

   XSelectInput (mydisplay,mymain_win,
                 ExposureMask | StructureNotifyMask | ButtonPressMask);

   XMapWindow (mydisplay,mymain_win);

/* Create a Graphics Context */

   mygc=XCreateGC (mydisplay, mymain_win, 0L, NULL);

/* Set color map */

   icomap_4 (mydisplay,
             &myred,           &mygreen,     &myblue,
             &mycyan,          &mymagenta,   &myyellow,
             &mylightseagreen, &myrgb);

/* Create window for options input */

   myopt_win=XCreateSimpleWindow (mydisplay,
                                  mymain_win,
```

```
                                    mywinwidth*3/4, mywinheight/8,
                                    110, 80,
                                    2,                /* border width */
                                    myblue,           /* border */
                                    myyellow);        /* background */

   XSelectInput (mydisplay,mymain_win, ButtonPressMask | ExposureMask);

   XMapWindow (mydisplay, myopt_win);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay, &myevent);

      switch (myevent.type) {

         case Expose:

/*          Reset option_window */

            XSetForeground (mydisplay, mygc, myblue);

            strcpy (ttext, "Left  button to");
            XDrawString (mydisplay, myopt_win, mygc, 10, 20,
                         ttext, strlen (ttext) );
            strcpy (ttext, "create  window!");
            XDrawString (mydisplay, myopt_win, mygc, 10, 30,
                         ttext, strlen (ttext) );

            strcpy (ttext, "Right button to");
            XDrawString (mydisplay, myopt_win, mygc, 10, 50,
                         ttext, strlen (ttext) );
            strcpy (ttext, "destroy window!");
            XDrawString (mydisplay, myopt_win, mygc, 10, 60,
                         ttext, strlen (ttext) );

/*          Reset message on main window */

            strcpy (ttext,
"Right Button to terminate application !                                    "
            );
            XDrawImageString (mydisplay, mymain_win, mygc,20,380,
                              ttext, strlen (ttext) );

            break;

         case ButtonPress:

            if (myevent.xbutton.subwindow == myopt_win) {

/*             Redraw border of option_window on leaving main */

               XSetWindowBorder (mydisplay, myopt_win, mymagenta);

               if (myevent.xbutton.button == 1) {

/*                Reset message on main window */

                  strcpy (ttext,
"Left Button - LINES; Middle Button - RECTANGLES; Right Button - ARCS"
                  );
                  XDrawImageString (mydisplay, mymain_win, mygc,20,380,
                                    ttext, strlen (ttext) );

/*                Create a window (child of main) */
```

```
/*                  Call event function */

                    g_out_9 (mydisplay, mymain_win, mygc, myscreen,
                             myopt_win,
                             myred, mygreen, myblue, mymagenta);

/*                  Reset message on main window */

                    strcpy (ttext,
"Right Button to terminate application !                                    "
                    );
                    XDrawImageString (mydisplay, mymain_win, mygc,20,380,
                                      ttext, strlen (ttext) );

/*              Redraw border of option_window on entering main */

                    XSetWindowBorder (mydisplay, myopt_win, myblue);

                  } else if (myevent.xbutton.button == 3) {

/*                     Right button to terminate application */

                     myendflag= TRUE ;

                  } /* else */

              } /* if */

              break;

        } /* switch (myevent.type) */

    } /* while (! myendflag) */

    XFlush (mydisplay);

/* Last window was abandoned; destruction of main window */
/* implies the termination of program                    */

/* Terminate graphical output */

    XFreeGC (mydisplay, mygc);
    XDestroyWindow (mydisplay, myopt_win);
    XDestroyWindow (mydisplay, mymain_win);
    XCloseDisplay (mydisplay);

    printf ("\n");
    printf ("That's all folks !\n");

    printf ("\n");
    return (0);

} /* main - ex9 */
```

====== g_out_9.c ======

```
/*****************************************************************/
/*                                                               */
/* g_out_9                                                       */
/* Graphical output of primitives (lines, rectangles and arcs).  */
/* Generation of child windows by recursive call of function.    */
/* This function is called by the main program ex9 and by itself.*/
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-21                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h> /* rand, RAND_MAX */
#include <X11/Xlib.h>

#define TRUE  1
#define FALSE 0

#define LEFTBUTTON     256
#define MIDBUTTON      512
#define RIGHTBUTTON   1024

void g_out_9 (
   Display   *mydisplay,
   Window     mymain_win,
   GC         mygc,
   int        myscreen,
   Window     myopt_win,
   int        myred,
   int        mygreen,
   int        myblue,
   int        mymagenta
) {

/* Variable declaration */

   Window       mydraw_win;
   Window       myroot_win;
   Window       mychild_win;
   XEvent       myevent;

   char         myendflag; /* test the end of the event loop */
   int          inipx;
   int          inipy;
   int          intpx;
   int          intpy;
   int          itroc;
   int          ixaux;
   int          iyaux;
   int          ixpre;
   int          iypre;
   int          minxx;
   int          minyy;
   int          maxxx;
   int          maxyy;
   int          root_x;
   int          root_y;
   unsigned int iwidt;
   unsigned int iheig;
   unsigned int mousebut;
   double       xrand;

/* Variable initialization */

/* Randomly generates coordinates for child windows */
```

```
      xrand = rand ();
      minxx = (xrand/RAND_MAX)*20+20;
      xrand = rand ();
      minyy = (xrand/RAND_MAX)*20+20;
      xrand = rand ();
      maxxx = (xrand/RAND_MAX)*200+150;
      xrand = rand ();
      maxyy = (xrand/RAND_MAX)*200+130;

   /* New child window */

      mydraw_win=XCreateSimpleWindow (mydisplay,
                        mymain_win,
                        minxx, minyy, maxxx, maxyy,
                        2,                                  /* border width */
                        mymagenta,                           /* border */
                        WhitePixel (mydisplay,myscreen)); /* background */

      XSelectInput (mydisplay,mydraw_win,
                    ButtonPressMask | ButtonReleaseMask |
                    ButtonMotionMask | PointerMotionHintMask);

      XMapWindow (mydisplay, mydraw_win);

   /* Allways reset border of option_window on entering function */

      XSetWindowBorder (mydisplay, myopt_win, myblue);

   /* Main X event loop */

      myendflag = FALSE ;

      while ( ! myendflag) {

         XNextEvent (mydisplay,&myevent);

         switch (myevent.type) {

            case Expose:

   /*          Discard all other Expose events */
   /*          Expose events should be dealt more consistently */

               while (   XCheckTypedEvent (mydisplay,Expose,&myevent)   );

               break;

            case ButtonPress:

               if (myevent.xbutton.subwindow == myopt_win) {

                  XSetFunction (mydisplay, mygc, GXcopy);

   /*             Allways reset border of option_window on */
   /*             leaving function */

                  XSetWindowBorder (mydisplay, myopt_win, mymagenta);

                  switch (myevent.xbutton.button) {

                     case 1: /* left button */

   /*                   Reset (deactivated) border of draw_window */

                        XSetWindowBorder (mydisplay, mydraw_win,
                                          BlackPixel (mydisplay,myscreen) );

   /*                   Go for another round(call function recursively) */
```

```
                    g_out_9 (mydisplay, mymain_win, mygc, myscreen,
                             myopt_win,
                             myred, mygreen, myblue, mymagenta);

/*                  Allways reset windows borders on entering func. */

                    XSetWindowBorder (mydisplay,myopt_win,myblue);

                    XSetWindowBorder (mydisplay,mydraw_win,mymagenta);

                    break;

                 case 3: /* right button */

                    myendflag= TRUE ;
                    break;

                 } /* switch (myevent.xbutton.button) */

              } else if (myevent.xbutton.window == mydraw_win) {

                 inipx= myevent.xbutton.x;
                 inipy= myevent.xbutton.y;
                 ixpre= inipx;
                 iypre= inipy;
                 iwidt= 0;
                 iheig= 0;

/*               Start rubberbanding in Xor mode */
/*               In Xor mode "white on white" is black! */

                 XSetFunction (mydisplay, mygc, GXxor);
                 XSetForeground (mydisplay, mygc,
                                 WhitePixel (mydisplay,myscreen));

              } /* if (myevent.xbutton.window) */

              break;

           case ButtonRelease:

/*            Final position of primitive; switch back to Copy mode */
/*            and draw primitive with final attributes               */

              XSetFunction (mydisplay, mygc, GXcopy);
              XSetLineAttributes (mydisplay, mygc, 3, LineSolid,
                                  CapButt, JoinMiter);


              switch (myevent.xbutton.button) {

/*               Draw primitive according to button pressed/released */

                 case 1:
                    XSetForeground (mydisplay, mygc, myred);
                    XDrawLine (mydisplay, mydraw_win, mygc,
                               inipx,inipy,ixpre,iypre);
                    break;

                 case 2:
                    XSetForeground (mydisplay, mygc, mygreen);
                    XDrawRectangle (mydisplay, mydraw_win, mygc,
                                    ixpre, iypre,
                                    iwidt, iheig);
                    break;

                 case 3:
                    XSetForeground (mydisplay, mygc, myblue);
```

```
                XDrawArc (mydisplay, mydraw_win, mygc,
                          ixpre, iypre,
                          iwidt, iheig,0*64,360*64);
                break;

            } /* switch (myevent.xbutton.button) */

/*          Reset line attributes; most important is line width */
/*          (reset to 0 - zero), otherwise server might not perform */
/*          rubberbanding */

            XSetLineAttributes (mydisplay, mygc, 0, LineSolid,
                                CapButt, JoinMiter);
            break;

         case MotionNotify:

/*          Rubberbanding drawing while final posit. is not reached */

            while (XCheckMaskEvent (mydisplay,
                                    ButtonMotionMask, &myevent) )
               ;   /* do nothing; wait until the mouse stays put */

            if (myevent.xmotion.window == mydraw_win) {

               if (! XQueryPointer (mydisplay, mydraw_win, &myroot_win,
                                    &mychild_win,
                                    &root_x, &root_y, &intpx, &intpy,
                                    &mousebut) )
                  break; /* exit if pointer is not on the same scr. */

/*             Act according to code returned by XQueryPointer for */
/*             each button hold down (pressed)                     */

               switch (mousebut) {

                  case  LEFTBUTTON:

/*                   Redraw (erase) previous line */

                     XDrawLine (mydisplay, mydraw_win, mygc,
                                inipx, inipy, ixpre, iypre);

/*                   Draw new line */

                     XDrawLine (mydisplay, mydraw_win, mygc,
                                inipx, inipy, intpx, intpy);

                     ixpre=intpx;
                     iypre=intpy;

                     break;

                  case  MIDBUTTON:

/*                   Swap coord.s if needed */

                     ixaux=inipx;
                     iyaux=inipy;
                     if (intpx < ixaux) {
                        itroc=ixaux;
                        ixaux=intpx;
                        intpx=itroc;
                     } /* if intpx */
                     if (intpy < iyaux) {
                        itroc=iyaux;
                        iyaux=intpy;
                        intpy=itroc;
                     } /* if intpy */
```

```
/*                      Redraw (erase) previous rectangle */

                        XDrawRectangle (mydisplay, mydraw_win, mygc,
                                        ixpre, iypre,
                                        iwidt,iheig);

                        ixpre=ixaux;
                        iypre=iyaux;
                        iwidt=intpx-ixaux;
                        iheig=intpy-iyaux;

/*                      Draw new rectangle */

                        XDrawRectangle (mydisplay, mydraw_win, mygc,
                                        ixpre, iypre,
                                        iwidt,iheig);
                        break;

                    case RIGHTBUTTON:

/*                      Swap coord.s if needed */

                        ixaux=inipx;
                        iyaux=inipy;
                        if (intpx < ixaux) {
                           itroc=ixaux;
                           ixaux=intpx;
                           intpx=itroc;
                        } /* if intpx */
                        if (intpy < iyaux) {
                           itroc=iyaux;
                           iyaux=intpy;
                           intpy=itroc;
                        } /* if intpy */

/*                      Redraw (erase) previous arc */

                        XDrawArc (mydisplay, mydraw_win, mygc,
                                  ixpre, iypre,
                                  iwidt,iheig,0*64,360*64);

                        ixpre=ixaux;
                        iypre=iyaux;
                        iwidt=intpx-ixaux;
                        iheig=intpy-iyaux;

/*                      Draw new arc */

                        XDrawArc (mydisplay, mydraw_win, mygc,
                                  ixpre, iypre,
                                  iwidt,iheig,0*64,360*64);
                        break;

                } /* switch (mousebut) */

            } /* if (myevent.xmotion.subwindow) */

            break;

        } /* switch (myevent.type) */

    } /* while ( ! myendflag) */

/* Destroy current level window */

    XDestroyWindow (mydisplay, mydraw_win);

} /* g_out_9 */
```

====== work9 ======

Exercicios relativos ao programa ex9:

1 - Acrescentar ao programa de graficos 2D a possiblidade de colocar comentarios e legendas numa posicao seleccionada com o rato.

2 - Acrescentar ao programa de graficos 2D a possibilidade de seleccionar certas funcoes com menus do tipo push button.

====== ex10.c ======

```
/*****************************************************************/
/*                                                               */
/* ex10                                                          */
/* Example 10                                                    */
/*                                                               */
/* New features:                                                 */
/*    - Open several main windows.                               */
/*    - Set hints for the window manager.                        */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-24                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

typedef char *caddr_t;

#include <stdio.h>
#include <stdlib.h> /* exit, malloc */
#include <string.h> /* strcpy, strcat */

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/Xatom.h>
#include <X11/keysym.h>

#define TRUE  1
#define FALSE 0

#define MAXCHR  82

void sethints (Display *, Window, Window, Window, int, int, int,
               char *, XClassHint *, XSizeHints *, XWMHints *, Atom *);
void singra (Display *, Window, GC);
void cosgra (Display *, Window, GC);

int main (
   int argc,
   char **argv
) {

/* Variable declaration */

   Display          *mydisplay;
   int               myscreen;
   Window            a_window;
   Window            b_window;
   Window            bigwindow;
   XEvent            myevent;
   GC                mygc;
   KeySym            mykeysym;
   XClassHint        myclass_hints;
   XSizeHints        mysize_hints;
   XWMHints          mywm_hints;
   Atom              myprot_atom;
   XWindowAttributes myattributes; /* width, height, etc. */
   static char      *myprogname;

   int          mybufsize = MAXCHR;
   int          mydispwidth;
   int          mydispheight;
   int          mywinwidth;        /* window width */
   int          mywinheight;       /* window height */
   char        *mybuffer;
   char         myendflag;         /* test the end of the event loop */
   char         mydispname[MAXCHR]; /* display name */
```

```
   char         ttext[MAXCHR];      /* current text line */

/* Variable initialization */

   myprogname=argv[0];
   mydispname[0]=0;
   mybuffer=(char *) malloc ( MAXCHR *sizeof (char));
   mybuffer[0]='\0';

   printf ("\n");
   printf ("Example 10\n");

/* Use command line argument if present */

   if (argc > 1) {
      strcpy (mydispname,argv[1]);
      strcat (mydispname,":0");
      printf ("\n");
      printf ("%s will be used as a display\n",mydispname);
   } /* if */

/* Pause in text mode */

   printf ("\n");
   printf ("Press the Return key to start graphics mode ... ");
   getchar ();

/* Open display mydispname */

   if (   (mydisplay=XOpenDisplay (mydispname)) == NULL   ) {
      fprintf (stderr,"\n");
      fprintf (stderr,"Cannot connect to server %s\n",mydispname);
      fprintf (stderr,"\n");
      exit (1);
   } /* if */

   myscreen=DefaultScreen (mydisplay);

   mydispwidth= DisplayWidth (mydisplay, myscreen);
   mydispheight= DisplayHeight (mydisplay, myscreen);

   mywinwidth  = (mydispwidth*0.50) * .90;
   mywinheight = (mydispheight*0.75);

/* Open three (3) main windows */

   bigwindow=XCreateSimpleWindow (mydisplay,
                     DefaultRootWindow (mydisplay),
                     0, 0, mydispwidth, mydispheight,
                     2,                          /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

   a_window=XCreateSimpleWindow (mydisplay,
                     DefaultRootWindow (mydisplay),
                     20,50,mywinwidth,mywinheight,
                     2,                          /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

   b_window=XCreateSimpleWindow (mydisplay,
                     DefaultRootWindow (mydisplay),
                     (mydispwidth*0.50),50,mywinwidth,mywinheight,
                     2,                          /* border width */
                     BlackPixel (mydisplay,myscreen),  /* border */
                     WhitePixel (mydisplay,myscreen)); /* background */

   XSelectInput (mydisplay,bigwindow,
                  ExposureMask | StructureNotifyMask | KeyPressMask);
```

```
   XSelectInput (mydisplay,a_window,
                 ExposureMask | StructureNotifyMask | KeyPressMask);
   XSelectInput (mydisplay,b_window,
                 ExposureMask | StructureNotifyMask | KeyPressMask);
/* Set hints for main window */

   sethints (mydisplay, bigwindow, a_window, b_window, myscreen,
             mydispwidth, mydispheight, myprogname,
             &myclass_hints, &mysize_hints, &mywm_hints, &myprot_atom);

/* Map windows in the correct order or bigwindow covers the other */
/* two windows                                                    */

   XMapWindow (mydisplay,bigwindow);
   XMapWindow (mydisplay,a_window);
   XMapWindow (mydisplay,b_window);

/* Create a Graphics Context */

   mygc=XCreateGC (mydisplay ,a_window, 0L, NULL);

/* Main X event loop */

   myendflag= FALSE ;

   while ( ! myendflag) {

      XNextEvent (mydisplay,&myevent);

      switch (myevent.type) {

         case Expose:

/*          Cannot discard any Expose events or some windows */
/*          might not be redrawned                           */
/*          while ( XCheckTypedEvent (mydisplay,Expose,&myevent) ); */

            if (myevent.xexpose.window == bigwindow) {
               strcpy (ttext,
                 "Press the Return key to terminate application ... ");
               XDrawImageString (mydisplay, bigwindow, mygc,
                                 15, 15, ttext, strlen (ttext) );
            } /* if */

            if (myevent.xexpose.window == a_window)
                     singra (mydisplay, a_window, mygc);

            if (myevent.xexpose.window == b_window)
                     cosgra (mydisplay, b_window, mygc);

            break;

         case KeyPress:

            XLookupString ( (XKeyEvent *) &myevent, mybuffer,
                            mybufsize, &mykeysym, NULL);

            if (   (mykeysym == XK_Return) ||
                   (mykeysym == XK_KP_Enter) ||
                   (mykeysym == XK_Linefeed)   ) myendflag= TRUE ;

            break;

         case ClientMessage:

/*          This event checks possible attempts to destroy a window  */
/*          This might have been allowed, but should allways be      */
/*          controlled otherwise the application terminates with     */
/*          an X Server IO error.                                    */
```

```
/*          Does not allow destruction of windows; */
/*          display message instead                 */

            XBell (mydisplay, 100);
            strcpy (ttext,"User not allowed to destroy windows!");
            XDrawImageString (mydisplay, myevent.xclient.window, mygc,
                              15, 15, ttext, strlen (ttext) );

            break;

      } /* switch (myevent.type) */

   } /* while ( ! myendflag) */

/* Terminate graphical output */

   XFreeGC (mydisplay,mygc);
   XDestroyWindow (mydisplay,bigwindow);
   XDestroyWindow (mydisplay,a_window);
   XDestroyWindow (mydisplay,b_window);
   XCloseDisplay (mydisplay);

   printf ("\n");
   printf ("That's all folks !\n");

   printf ("\n");
   return (0);

} /* main - ex10 */
```

====== sethints.c ======

```
/*****************************************************************/
/*                                                               */
/* sethints                                                      */
/* Set hints for ex10 top-level windows.                         */
/* Hints are svailable for windows children of RootWindow only.  */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-24                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

typedef char *caddr_t;

#include <stdio.h>
#include <stdlib.h> /* exit */

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/Xatom.h>

#include "main.h"        /* icon file */
#include "sine.h"        /* icon file */
#include "cossine.h"     /* icon file */

void sethints (
   Display     *mydisplay,
   Window       bigwindow,
   Window       a_window,
   Window       b_window,
   int          myscreen,
   int          mydispwidth,
   int          mydispheight,
   char        *myprogname,
   XClassHint  *myclass_hints,
   XSizeHints  *mysize_hints,
   XWMHints    *mywm_hints,
   Atom        *myprot_atom
) {

   char *icon_bw_name= "BigWindow";
   char *icon_sine_name= "Sine function";
   char *icon_cossine_name= "Cossine function";

   char *win_bw_name= "Hello  X  world !";
   char *win_sine_name= "This is the window of sine!";
   char *win_cossine_name= "This is the window of cossine!";

   XTextProperty iconName;
   XTextProperty winName;
   Pixmap        icon_bw_pixmap;
   Pixmap        icon_sine_pixmap;
   Pixmap        icon_cossine_pixmap;
   XIconSize    *size_list;    /* list of available sizes for icons */

   int count;

/* End of variable declarations                                  */

/* Icon size list                                                */
   if ( (size_list= XAllocIconSize () ) == NULL) {
      fprintf (stderr,
               "%s: structure allocation for icon size list failed.\n",
               myprogname);
      exit (1);
   } /* if (size_list) */
```

```
/* Get available icon sizes from window manager                          */
   if ( (XGetIconSizes (mydisplay, RootWindow (mydisplay,myscreen),
                        &size_list, &count) )                          == 0)
      fprintf (stderr,
        "%s: Window manager didn't set icon sizes - using default.\n",
             myprogname);
   else

/*    Application should search through size_list here to find an      */
/*    acceptable icon size and then create a pixmap of that size;      */
/*    this requires that the application have data for several         */
/*    sizes of icons                                                   */

      XFree ( (char *) size_list);     /* free allocated storage       */
                                       /* returned by XGetIconSizes    */

/* Create pixmap of depth 1 (bitmap) for icons                           */

   icon_bw_pixmap= XCreateBitmapFromData (mydisplay, bigwindow,
                                          main_bits,
                                          main_width,
                                          main_height);
   icon_sine_pixmap= XCreateBitmapFromData (mydisplay, a_window,
                                            sine_bits,
                                            sine_width,
                                            sine_height);
   icon_cossine_pixmap= XCreateBitmapFromData (mydisplay, b_window,
                                               cossine_bits,
                                               cossine_width,
                                               cossine_height);

/* Application names  (name of windows)                                  */

   if (XStringListToTextProperty (&win_bw_name, 1, &winName) == 0) {
      fprintf (stderr,
               "%s: structure allocation for winName failed.\n",
               myprogname);
      exit (1);
   } /*  if (XStringListToTextProperty */

   XSetWMName (mydisplay, bigwindow, &winName);

   if (XStringListToTextProperty (&win_sine_name, 1, &winName) == 0) {
      fprintf (stderr,
               "%s: structure allocation for winName failed.\n",
               myprogname);
      exit (1);
   } /*  if (XStringListToTextProperty */
   XSetWMName (mydisplay, a_window, &winName);

   if (XStringListToTextProperty (&win_cossine_name,
                                  1, &winName) == 0) {
      fprintf (stderr,
               "%s: structure allocation for winName failed.\n",
               myprogname);
      exit (1);
   } /*  if (XStringListToTextProperty */
   XSetWMName (mydisplay, b_window, &winName);

/* Icon names                                                            */

   if (XStringListToTextProperty (&icon_bw_name, 1, &iconName) == 0) {
      fprintf (stderr,
               "%s: structure allocation for iconName failed.\n",
               myprogname);
      exit (1);
   } /*  if (XStringListToTextProperty */
   XSetWMIconName (mydisplay, bigwindow, &iconName);
```

```
    if (XStringListToTextProperty (&icon_sine_name,
                                   1, &iconName) == 0) {
       fprintf (stderr,
                "%s: structure allocation for iconName failed.\n",
                 myprogname);
       exit (1);
    } /*  if (XStringListToTextProperty */
    XSetWMIconName (mydisplay, a_window, &iconName);

    if (XStringListToTextProperty (&icon_cossine_name,
                                   1, &iconName) == 0) {
       fprintf (stderr,
                "%s: structure allocation for iconName failed.\n",
                 myprogname);
       exit (1);
    } /*  if (XStringListToTextProperty */
    XSetWMIconName (mydisplay, b_window, &iconName);

/* SIZE HINTS                                                        */
/* Set geom. preferences for top level windows before mapping them. */

/* Struct allocation must be made first for future compatibility    */
/* Size_hints struct returned is zeroed                             */

    if ( (mysize_hints= XAllocSizeHints () ) == NULL) {
       fprintf (stderr,
                "%s: structure allocation for size hints failed.\n",
                myprogname);
       exit (1);
    } /* if (mysize_hints) */

    mysize_hints->flags= USPosition | USSize | PBaseSize;

/* Big window size - Set minimae to 25% of disp_size                */
    mysize_hints->base_width= (int) (mydispwidth*0.40);
    mysize_hints->base_height= (int) (mydispheight*0.25);

    XSetWMNormalHints (mydisplay, bigwindow, mysize_hints);

/* a and b windows size - Set minimae to 10% of disp_size           */
    mysize_hints->base_width= (int) (mydispwidth*0.10);
    mysize_hints->base_height= (int) (mydispheight*0.10);

    XSetWMNormalHints (mydisplay, a_window, mysize_hints);
    XSetWMNormalHints (mydisplay, b_window, mysize_hints);

/* WM HINTS                                                         */
/* Assortment of information to the window manager.                 */

/* Struct allocation must be made first for future compatibility    */
/* wm_hints struct returned is zeroed                               */

    if ( (mywm_hints= XAllocWMHints () ) == NULL) {
       fprintf (stderr,
                "%s: structure allocation for wm hints failed.\n",
                myprogname);
       exit (1);
    } /* if (mywm_hints) */

    mywm_hints->flags= StateHint | InputHint | IconPixmapHint;
    mywm_hints->initial_state= NormalState;
    mywm_hints->input= True;                    /* Passive Input */
    mywm_hints->icon_pixmap= icon_bw_pixmap;

    XSetWMHints (mydisplay, bigwindow, mywm_hints);

    mywm_hints->icon_pixmap= icon_sine_pixmap;
```

```
   XSetWMHints (mydisplay, a_window, mywm_hints);

   mywm_hints->icon_pixmap= icon_cossine_pixmap;

   XSetWMHints (mydisplay, b_window, mywm_hints);

/* CLASS HINTS                                                          */
/* Elements to be used by the wm to look up resources applicable        */
/* to this application or identification information.                   */

/* Struct allocation must be made first for future compatibility        */
/* class_hint struct returned is zeroed                                 */

   if ( (myclass_hints= XAllocClassHint () ) == NULL) {
      fprintf (stderr,
               "%s: structure allocation for class hints failed.\n",
               myprogname);
      exit (1);
   } /* if (myclass_hints) */

   myclass_hints->res_name= myprogname;
   myclass_hints->res_class= "ex10_win";

   XSetClassHint (mydisplay, bigwindow, myclass_hints);
   XSetClassHint (mydisplay, a_window, myclass_hints);
   XSetClassHint (mydisplay, b_window, myclass_hints);

/* Xlib provides a func. to set the properties for a given window       */
/* wich might have been used instead of setting each type of hints      */
/* separately:                                                          */

/* XSetWMProperties (display, mywindow,                                 */
/*                   &windowName, &iconName,                            */
/*                   argv, argc,                                        */
/*                   mysize_hints, mywm_hints, myclass_hints);          */
/* This function needs all the following properties to be defined:      */
/*      windowName, iconName, size_hints, wm_hints and class_hints.     */

/**********************************************************************/

/* Prevention of abnormal destruction of window by the wm.              */
/* XInternAtom must be called to define the atom - there is no          */
/* predefined atom for protocols                                        */

   (*myprot_atom)= XInternAtom (mydisplay, "WM_DELETE_WINDOW", False);
   if (myprot_atom == None) {
      fprintf (stderr,
               "%s: did not create WM_DELETE_WINDOW atom.\n",
               myprogname);
      exit (1);
   } /* if (myprot_atom) */

   XSetWMProtocols (mydisplay, bigwindow, myprot_atom, 1);
   XSetWMProtocols (mydisplay, a_window, myprot_atom, 1);
   XSetWMProtocols (mydisplay, b_window, myprot_atom, 1);

} /* sethints */
```

====== singra.c ======

```
/*****************************************************************/
/*                                                               */
/* singra                                                        */
/* Graph of sine function [-PI, PI]                              */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-27                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <math.h> /* sin */

#include <X11/Xlib.h>

void singra (
   Display   *mydisplay,
   Window     a_window,
   GC         mygc
) {

/* Variable declaration */

   XWindowAttributes mywin_attr;

   int        iterm;
   int        ixgra;
   int        ixori;
   int        ixpre;
   int        iygra;
   int        iyori;
   int        iypre;
   int        nterm;
   int        mywinheight;
   int        mywinwidth;

   double     valpi = 3.1415926535;
   double     rincr;
   double     scale;
   double     scalx;
   double     scaly;
   double     xxval;
   double     yyval;

/* Get current window size for scaling */

   XGetWindowAttributes (mydisplay, a_window, &mywin_attr);
   mywinwidth=mywin_attr.width;
   mywinheight=mywin_attr.height;

/* Set origin of axes */

   ixori=mywinwidth/2;
   iyori=mywinheight/2;

/* Set scale */

   scalx=(mywinwidth*0.85)/(valpi*4);
   scaly=(mywinheight*0.85)/2.0;

   scale=scalx;
   if (scalx > scaly) scale=scaly;
   nterm= ( mywinwidth*0.10 > mywinheight*0.10 ) ? mywinwidth*0.10 :
                                                   mywinheight*0.10;

   rincr= (double) (valpi*4/nterm);
```

```
/* Draw axes */

   XDrawLine (mydisplay, a_window, mygc,
              mywinwidth*0.05, iyori,
              mywinwidth*0.95, iyori);
   XDrawLine (mydisplay, a_window, mygc,
              ixori, mywinheight*0.05,
              ixori, mywinheight*0.95);

   xxval=(-valpi*2);
   yyval=sin (xxval);
   ixpre=(int) (xxval*scale);
   iypre=(int) (yyval*scale);
   ixpre+=ixori;
   iypre+=iyori;

/* Draw sine graph */

   for (iterm=1 ; iterm <= nterm ; iterm++) {
      xxval+=rincr;
      yyval=sin (xxval);
      ixgra=(int) (xxval*scale);
      iygra=(int) (yyval*scale);
      ixgra+=ixori;
      iygra+=iyori;
      XDrawLine (mydisplay, a_window, mygc, ixpre, iypre,
                 ixgra, iygra);
      ixpre=ixgra;
      iypre=iygra;
   } /* for (iterm) */

} /* singra */
```

====== cosgra.c ======

```
/*****************************************************************/
/*                                                               */
/* cosgra                                                        */
/* Graph of cossine function [-PI, PI]                           */
/*                                                               */
/* Version: 1.0                                                  */
/* Date   : 1992-06-27                                           */
/* Author : Ricardo Santos                                       */
/*                                                               */
/*****************************************************************/

#include <stdio.h>
#include <math.h> /* cos */

#include <X11/Xlib.h>

void cosgra (
   Display   *mydisplay,
   Window     b_window,
   GC         mygc
) {

/* Variable declaration */

   XWindowAttributes mywin_attr;

   int        iterm;
   int        ixgra;
   int        ixori;
   int        ixpre;
   int        iygra;
   int        iyori;
   int        iypre;
   int        nterm;
   int        mywinheight;
   int        mywinwidth;

   double     valpi = 3.1415926535;
   double     rincr;
   double     scale;
   double     scalx;
   double     scaly;
   double     xxval;
   double     yyval;

/* Get current window size for scaling */

   XGetWindowAttributes (mydisplay, b_window, &mywin_attr);
   mywinwidth=mywin_attr.width;
   mywinheight=mywin_attr.height;

/* Set origin of axes */

   ixori=mywinwidth/2;
   iyori=mywinheight/2;

/* Set scale */

   scalx=(mywinwidth*0.85)/(valpi*4);
   scaly=(mywinheight*0.85)/2.0;

   scale=scalx;
   if (scalx > scaly) scale=scaly;
   nterm= ( mywinwidth*0.10 > mywinheight*0.10 ) ? mywinwidth*0.10 :
                                                   mywinheight*0.10;

   rincr= (double) (valpi*4/nterm);
```

```
/* Draw axes */

   XDrawLine (mydisplay, b_window, mygc,
              mywinwidth*0.05, iyori,
              mywinwidth*0.95, iyori);
   XDrawLine (mydisplay, b_window, mygc,
              ixori, mywinheight*0.05,
              ixori, mywinheight*0.95);

   xxval=(-valpi*2);
   yyval=cos (xxval);
   ixpre=(int) (xxval*scale);
   iypre=(int) (yyval*scale);
   ixpre+=ixori;
   iypre+=iyori;

/* Draw cossine graph */

   for (iterm=1 ; iterm <= nterm ; iterm++) {
      xxval+=rincr;
      yyval=cos (xxval);
      ixgra=(int) (xxval*scale);
      iygra=(int) (yyval*scale);
      ixgra+=ixori;
      iygra+=iyori;
      XDrawLine (mydisplay, b_window, mygc, ixpre, iypre,
                 ixgra, iygra);
      ixpre=ixgra;
      iypre=iygra;
   } /* for (iterm) */

} /* cosgra */
```

====== work10 ======

Exercicios relativos ao programa ex10:

1 - Por o programa de graficos 2D a dialogar com o window manager (hints):
    - posicao inicial da janela
    - bitmap associado ao icon
    - dimensoes minimas das janelas