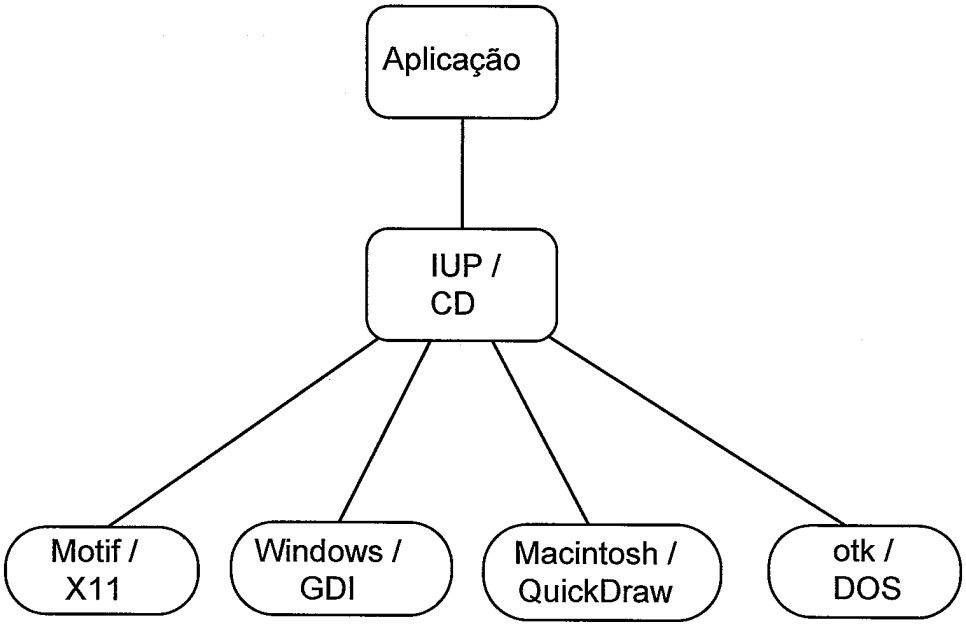
Canvas Draw (CD)

- Sistema gráfico projetado para ser usado em conjunto com o Sitema IUP/LED com o elemento de interface *canvas*.

- Objetivo
  - Portabilidade dos programas que utilizam primitivas de desenho (linhas, áreas preenchidas, textos e *pixmaps*).

Arquitetura



Classificação Funcional

As funções do CD podem ser agrupadas segundo suas características funcionais:

- Controle da Superfície de Visualização
- Sistema de Coordenadas e *Clipping*
- Primitivas
- Sistema de Cor
- Atributos
- Imagens

Controle da Superfície de Visualização

- **Ativação do sistema gráfico:**

int cdActivate (cdContext driver, void *data)

- Ativa um *driver* para a área de desenho, que é chamada genericamente de superfície de visualização (Visualization Surface - VS). As VS's disponíveis são: o elemento *canvas* do sistema IUP, impressora, *off-screen drawing*, *clipboard*, Windows Metafile, CGM, CD Metafile e Postscript.
- Parâmetros : CD_IUP / (Ihandle*) do *canvas*
  CD_PRINTER / (char *) nome_arq
  CD_IMAGE / (*) p/ cdImage
- Retorno : CD_OK / CD_ERROR
- Notas : 1) Desenha-se somente na VS ativa,
  2) Não existe função explícita para desativar uma VS,
  3) Deve-se chamar esta função antes de qualquer função do CD.

Controle da Superfície de Visualização (cont.)

- **int cdFlush** (void)
  - ⇨ Função dependente do *driver:*
    - ✔ IUP-CANVAS (X-Windows) : envia as primitivas armazenadas na *workstation* cliente para o servidor X (*canvas*).
    - ✔ Impressora : avança a página
    - ✔ Outros : não faz nada

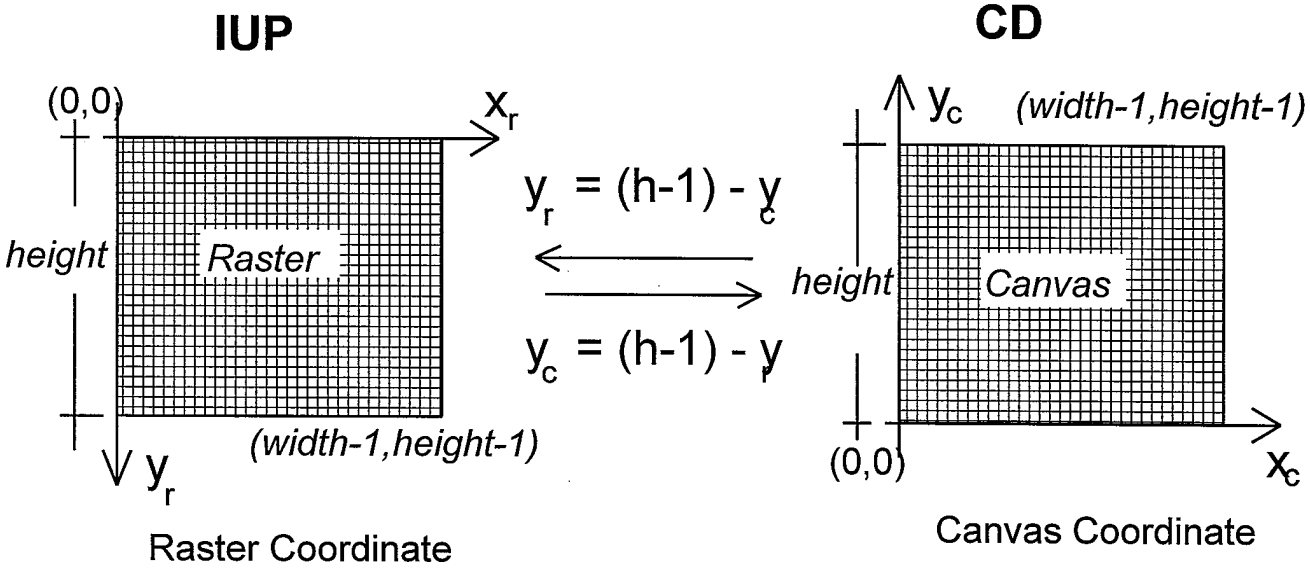
- **Limpeza da VS :**

void cdClear (void)

Sistema de Coordenadas e *Clipping*

As coordenadas do CD são em raster (*pixels*).

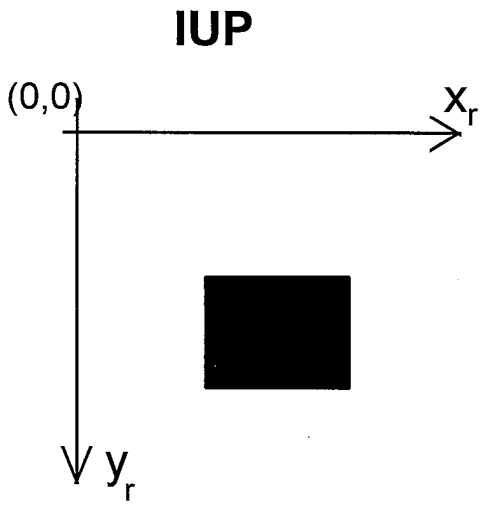
Relação entre as coordenadas do IUP e do CD:



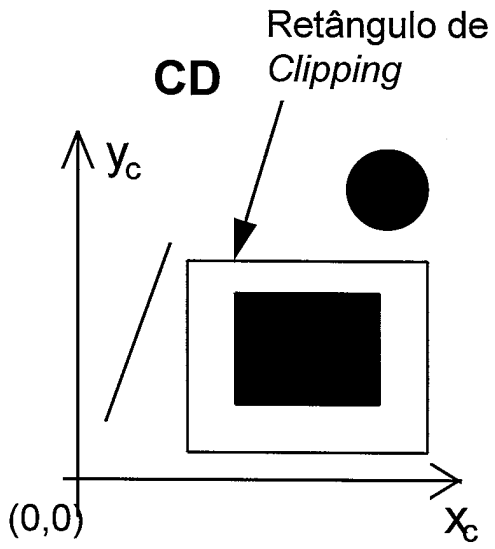
TeCGraf

Sistema de Coordenadas e *Clipping* (cont.)

Cerceamento (*clipping*) de primitivas.



Raster Coordinate



Canvas Coordinate

TeCGraf

Sistema de Coordenadas e *Clipping* (cont.)

- **Tamanho do *Canvas* :**

void cdGetCanvasSize (int *width, int *height
double *mm_width, double *mm_height)

- **Indicador de *Clipping***

int cdClip (int mode)

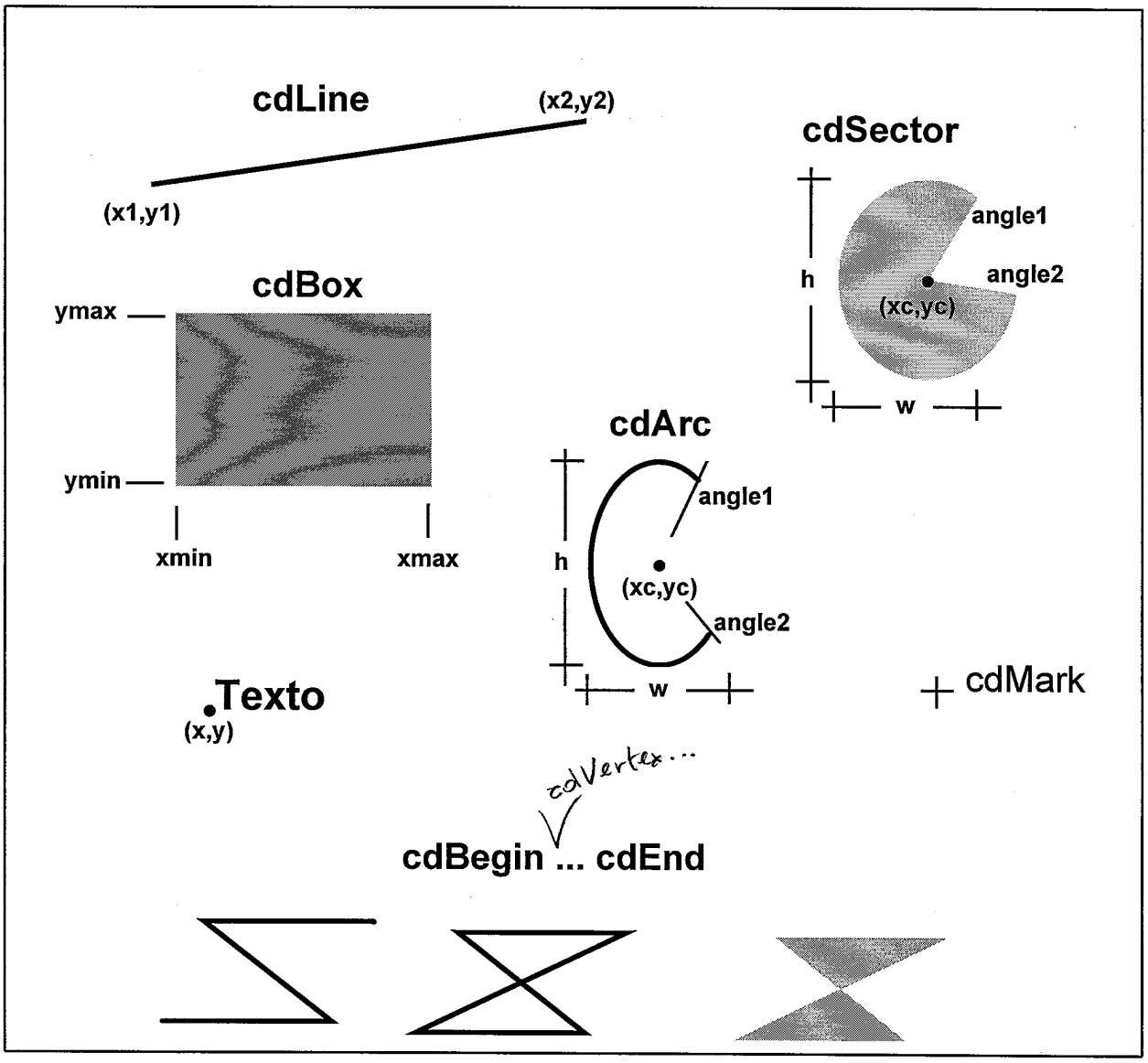
- "Liga" ou "desliga" o indicador de *clipping*.
- Parâmetros : CD_CLIPON / CD_CLIPOFF
- Retorno : estado anterior do indicador de *clipping*

- **Definição do "Retângulo de *Clipping*"**

void cdClipArea (int xmin, int xmax, int ymin,
int ymax)

Primitivas

Primitivas de desenho do CD:



TeCGraf

Primitivas (cont.)

- **Linha**

void cdLine (int x1, int y1, int x2, int y2)

  - ⇨ Desenha uma linha de (x1,y1) até (x2,y2)

- **Arco**

void cdArc (int xc, int yc, int w, int h, double angle1, double angle2)

  - ⇨ Desenha um arco de elipse alinh. com o eixo
  - ⇨ Ângulo é dado em graus

- **Marca**

void cdMark (int x, int y)

  - ⇨ Desenha uma marca na posição (x,y)

- **Preenchimento de um retângulo**

void cdBox (int xmin, int xmax, int ymin, int ymax)

Primitivas (cont.)

- **Texto**

void cdText (int x, int y, char *text)

  - ⇨ Coloca um texto na posição (x,y)

- **Arco**

void cdSector (int xc, int yc, int w, int h, double angle1, double angle2)

  - ⇨ Preenche um arco de elipse alinh. com o eixo
  - ⇨ Ângulo é dado em graus

- **void cdBegin** (int mode)

  - ⇨ Marca o início da definição de um poligono/ poligonal que será desenhada de acordo com o modo fornecido.
  - ⇨ Modos disponíveis : CD_CLOSED_LINES, CD_OPEN_LINES, CD_FILL

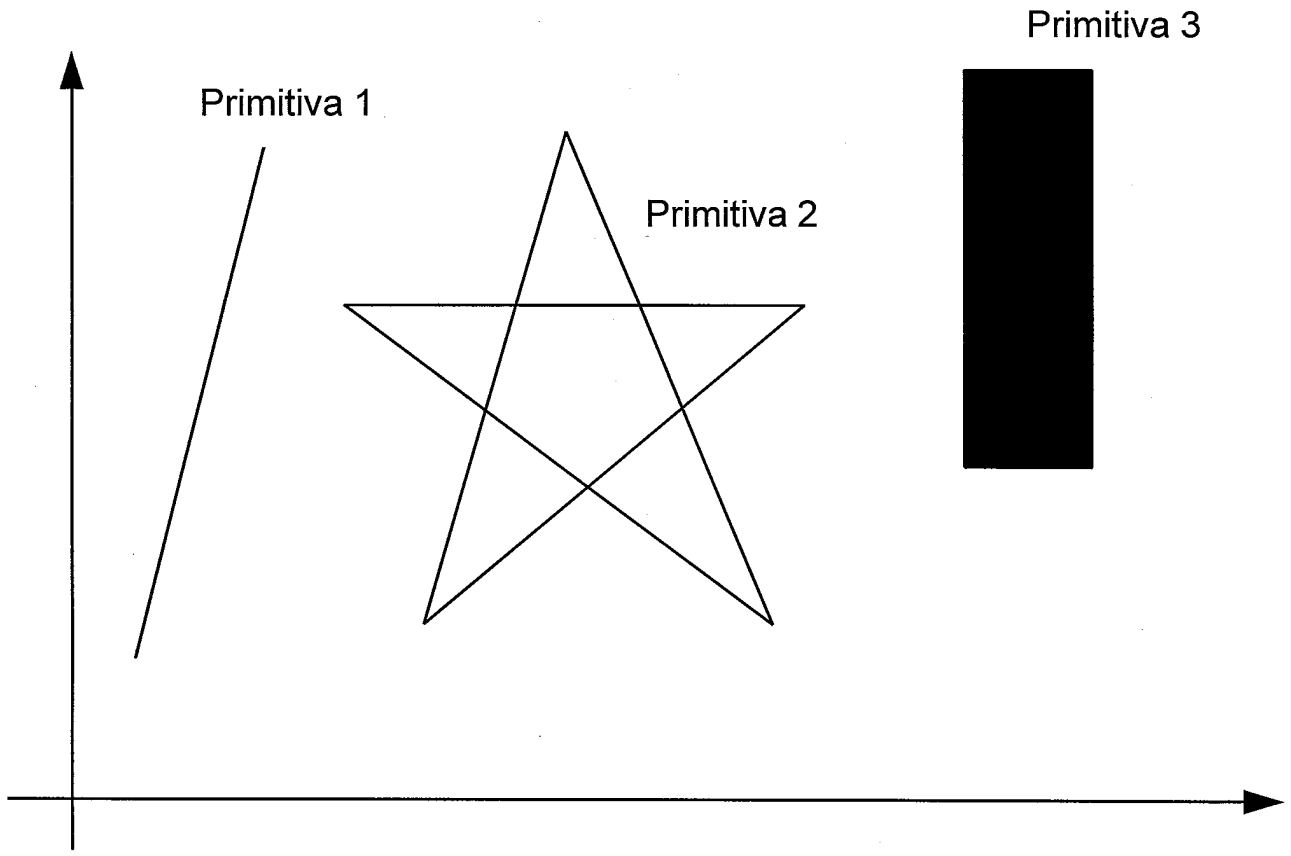
Primitivas (cont.)

- **void cdVertex** (int x, int y)
  - ⇨ Adiciona um vértice na definição de um poligono/poligonal.

- **void cdEnd** (void)
  - ⇨ Marca o término da definição de um poligono/ poligonal, desenhando-o

Nota : Todas as coordenadas das primitivas são no Sistema de Coordenadas do Canvas.

Exemplo 1



LED: Criação de um diálogo que tenha um menu (opção Saída) e um canvas.

```
  :
canv = CANVAS (acao_redesenha)
dial = DIALOG[ ... ] (canv)
```

```
#include "iup.h"
#include "cd.h"
#include "cdiup.h"

void main (void)
{
        :
/* Depois do IupShow e antes do IupMain Loop desenha-se no
   canvas as primitivas anteriores                          */
   Desenha_primitivas( );
}

void Desenha_primitivas (void)
{
/* Captura handle do canvas                                 */

/* Ativa VS                                                 */

/* Desenha primitivas                                       */


}
```

Notas 1) Preenchimento do poligono ?
2) Uso da ClipArea
3) Eventos e interações

Sistema de Cores

Cores são definidas em RGB.

- **long int cdEncode Color** (unsigned char red, unsigned char green, unsigned char blue)
  - ⇨ Define uma cor no CD
  - ⇨ Retorna um inteiro c/ 4 bytes (0x00RRGGBB)
- **void cdDecode Color** (long int color, unsigned char *red, unsigned char *green, unsigned char *blue)
  - ⇨ Retorna as componentes RGB de uma cor.
- **long int cdForeground** (long int color)
  - ⇨ Define a nova cor *foreground* que será usada
  - ⇨ Retorna a cor *foreground* que estava definida
- **long int cdBackground** (long int color)
  - ⇨ Define a nova cor *background* do canvas
  - ⇨ Retorna a cor *background* que estava definida

Atributos

Atributos Específicos

❑ Atributos de Linha

⇨ Estilo: **int cdLineStyle** (int style)

style = CD_CONTINUOS*, CD_DASHED, CD_DOTTED,CD_DASH_DOTTED, CD_DASH_DOT_DOT

⇨ Espessura: **int cdLineWidth** (int width)

width dado em pixels (*default* = 1)

❑ Atributos de Marca

⇨ Tipo: **int cdMarkerType** (int type)

type = CD_PLUS, CD_STAR*, CD_CIRCLE, CD_X, CD_BOX, CD_DIAMOND, CD_HOLLOW_CIRCLE, CD_HOLLOW_BOX, CD_HOLLOW_DIAMOND.

⇨ Tamanho: **int cdMarkerSize** (int size)

size dado em pixels (*default* = 10)

Atributos (cont.)

- **Atributos de Áreas preenchidas**
  - ⇨ Estilo de preenchimento:
    int cdInteriorStyle (int style)
    style = CD_SOLID*, CD_HATCH, CD_STIPPLE,CD_PATTERN
  - ⇨ Estilo de hachura: **int cdHatch** (int style)
    style = CD_HORIZONTAL, CD_VERTICAL, CD_FDIAGONAL, CD_BDIAGONAL, CD_CROSS, ou CD_DIAGCROSS
    - ✔ Além de definir o tipo de hachura a ser usada, define automaticamente o estilo de preenchimento como send CD_HATCH.
  - ⇨ Existem também a definição para o preenchimento CD_STIPPLE e CD_PATTERN que são dados através de matrizes.

Atributos (cont.)

- **Atributos de Texto**
  - ⇨ Fonte, estilo e tamanho:

    void cdFont (int typeface, int style, int size)

    typeface = CD_SYSTEM*, CD_COURIER, CD_TIMES_ROMAN, CD_HELVETICA

    style = CD_PLAIN*, CD_BOLD, CD_ITALIC

    size (points) = CD_SMALL(=10), CD_STANDARD*(=12) , CD_LARGE(=18)

  - ⇨ Alinhamento:

    int cdTextAlignment (int alignment)

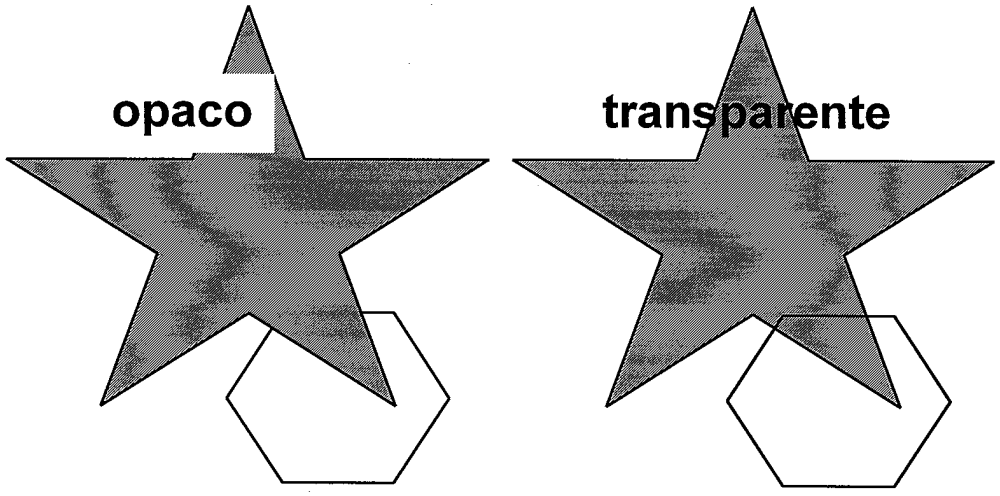
    alignment = CD_NORTH, CD_SOUTH, CD_EAST, CD_WEST, CD_NORTH_EAST, CD_NORTH_WEST, CD_SOUTH_EAST, CD_SOUTH_WEST, CD_CENTER, CD_BASE_LEFT*, CD_BASE_CENTER, CD_BASE_RIGHT

Atributos (cont.)

Atributos Globais

- **Opacidade e cor**
  - ⇨ **int cdBackOPacity** (int opacity)

    opacity = CD_OPAQUE*, CD_TRANSPARENT

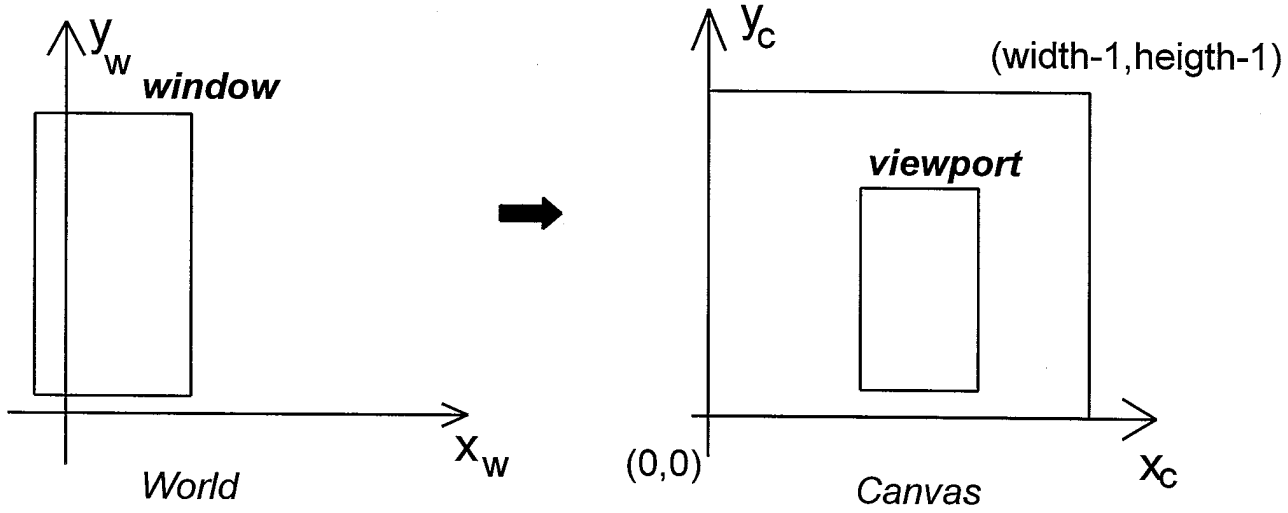


Transformação Window-Viewport

❑ Motivação

"As coordenadas dos pontos que descrevem o objeto sejam definidas por um sistema de coordenadas especificado pelo programador da aplicação".

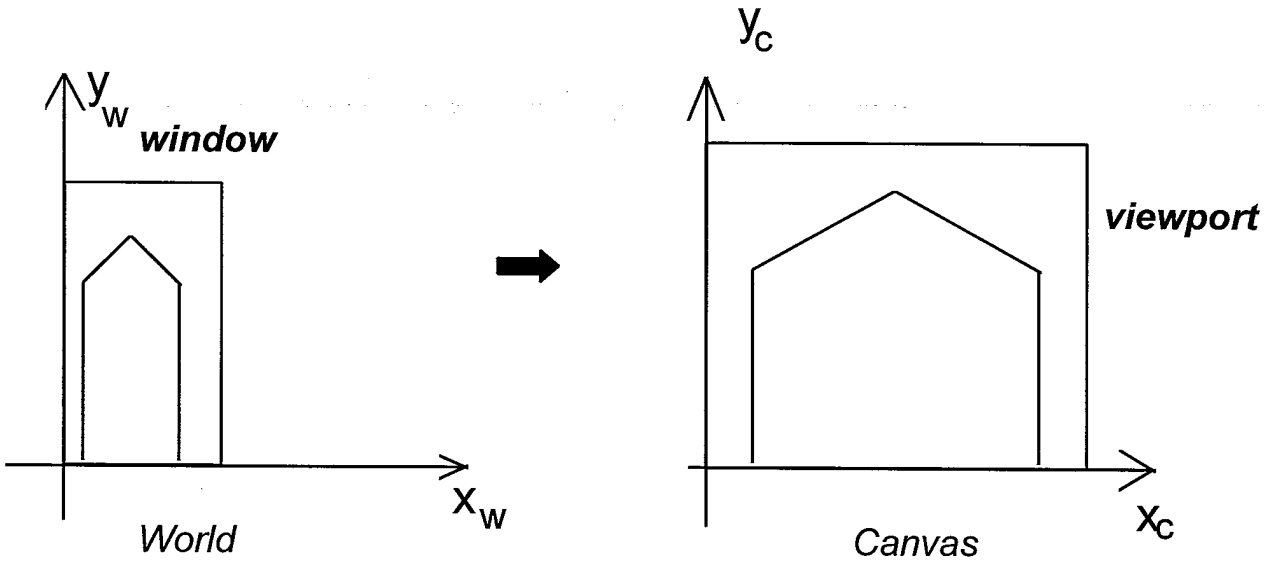
Desta forma, foi criado um sistema cartesiano independente do dispositivo, definido pelo programador para descrever e manipular suas imagens gráficas.



TeCGraf

Transformação Window-Viewport (cont.)

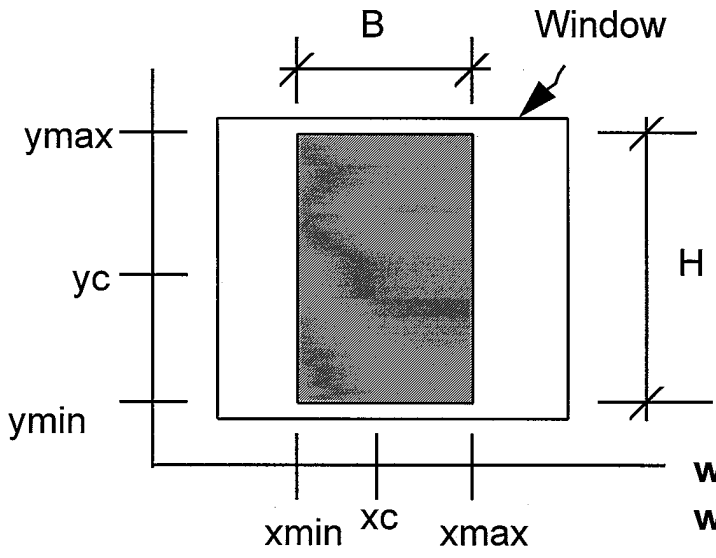
- **Distorção**



Para evitar distorção, é necessário que os lados da *window* sejam proporcionais aos da *viewport*.

Transformação Window-Viewport (cont.)

Usando a função cdGetCanvasSize, obtem-se a altura (h) e a largura (w) do canvas, podendo-se desta forma, calcular a razão do canvas (cr = h / w)



H/B > cr	H/B < cr
Ly = H	Lx = B
Lx = Ly/cr	Ly = Lx*cr

Definição da Window

wxmin = xc - 1.05 (Lx / 2.0)
wxmax = xc + 1.05 (Lx / 2.0)
wymin = yc - 1.05 (Ly / 2.0)
wymax = yc + 1.05 (Ly / 2.0)

TeCGraf

Funções WD

Para facilitar as aplicações onde os objetos são descritos em valores reais, foi construida uma camada denominada WD (*World Draw*).

❑ Definição da Window e Viewport

- ⇨ **void wdWindow** (double xmin, double xmax, double ymin, double ymax)
- ⇨ **void wdViewport** (int xmin, int xmax, int ymin, int ymax)

❑ Conversão de coordenadas

- ⇨ **void wdWorld2Canvas** (double xw, double yw, int *xv, int *yv)
- ⇨ **void wdCanvas2World** (int xv, int yv, double *xw, double *yw)

Funções WD - Primitivas

- **Linha**

void wdLine (double x1,double y1,double x2,double y2)

- **Arco**

void wdArc (double xc, double yc, double w, double h, double angle1, double angle2)

- **Marca**

void wdMark (double x, double y)

- **Preenchimento de um retângulo**

void wdBox (double xmin, double xmax, double ymin, double ymax)

- **Texto**

void wdText (double x, double y, char *text)

- **Arco**

void wdSector (double xc, double yc, double w, double h, double angle1, double angle2)

- **Vértices**

void wdVertex (double x, double y)

Funções vetoriais p/ texto

- **void wdVectorTextDirection** (double x1, double y1, double x2, double y2)
  - ⇨ Define a direção do texto.
- **void wdVectorTextSize** (const double size_x, const double size_y,const char *s)
  - ⇨ Enquadra o texto no retângulo definido por size_x e size_y.
- **void wdVectorCharSize** (double size)
  - ⇨ Define o tamanho dos caracteres.
- **void wdVectorText** (double x, double y, const char *s)
  - ⇨ Desenha o texto (vetorial) na posição (x,y). Respeita o alinham.dado por cdTextAlignment.
- **void wdMultiLineVectorText** (double x, double y, const char *s)
  - ⇨ Desenha o texto (várias linhas) na posição (x,y). Respeita o alinhamento dado por cdTextAlignment. Pula linha no \n.

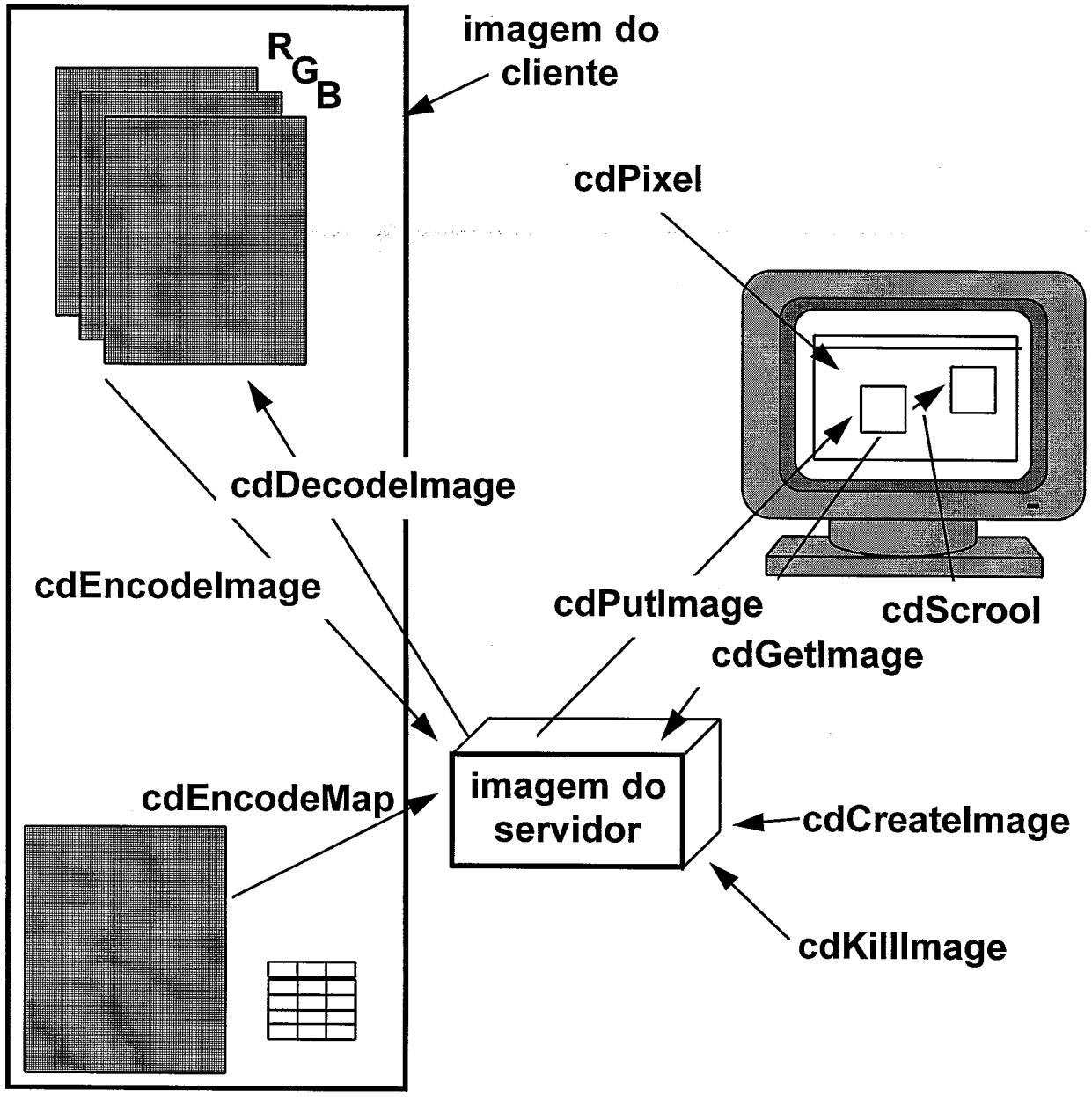
Imagens

No CD existem dois tipos de imagem: do cliente e do servidor.

A primeira possui um formato simples e pode ser modificada pelo programa de aplicação.

A segunda tem o formato do equipamento que é eficiente para tirar e colocar imagens na tela.

Tratamento de Imagens



TeCGraf

❑ Imagem do Cliente

- **void cdDecodeImage** (void *image, unsigned char *r,unsigned char *g, unsigned char *b)
  - ✔ Retorna os componentes RGB de cada pixel.
  - ✔ Três matrizes de *bytes.*

- **void EncodeImage** (void *image, unsigned char *r,unsigned char *g, unsigned char *b)
  - ✔ Retorna a imagem do servidor que é composta por três matrizes RGB de cada pixel.

- **void EncodeMap** (void *image, unsigned char *map,long int *palette)
  - ✔ Análogo à função **EncodeImage** com a excessão que as cores são dadas através de índices.

❑ Imagem do Servidor

- **void cdPixel** (int x, int y, long int color)
  - ✔ Define a cor de um pixel.
- **void cdCreateImage** (int w, int h)
  - ✔ Cria uma imagem na área de memória do servidor com **w**x**h** *pixels*.
- **void cdGetImage** (void *image, int x, int y)
  - ✔ Copia uma região retangular da tela p/ a memória.
  - ✔ (**x**,**y**) são as coordenadas do canto esquerdo inferior.
- **void cdPutImage** (void *image, int x, int y)
  - ✔ Copia uma imagem em uma região retangular do *canvas.*
  - ✔ (**x**,**y**) são as coordenadas do canto esquerdo inferior.
- **void cdKillImage** (void *image)
  - ✔ Libera memória alocada p/ a imagem.
- **void cdScroll** (int x1, inty1, int x2, int y2, int dx, int dy)
  - ✔ Translada um retângulo definido por (x1,y1) e (x2,y2) de dx na direção x e dy nadireção y.
  - ✔ Esta função não limpa a região.

IUP - CANVAS

- Elemento primitivo de interface que faz a ligação entre a parte gráfica da aplicação e o sistema gráfico.

Interação : os eventos ocorridos no CANVAS são passados para a aplicação, que deve tratá-los convenientemente.

❑ LED

CANVAS [ATRIBUTOS] (ação_redesenha)

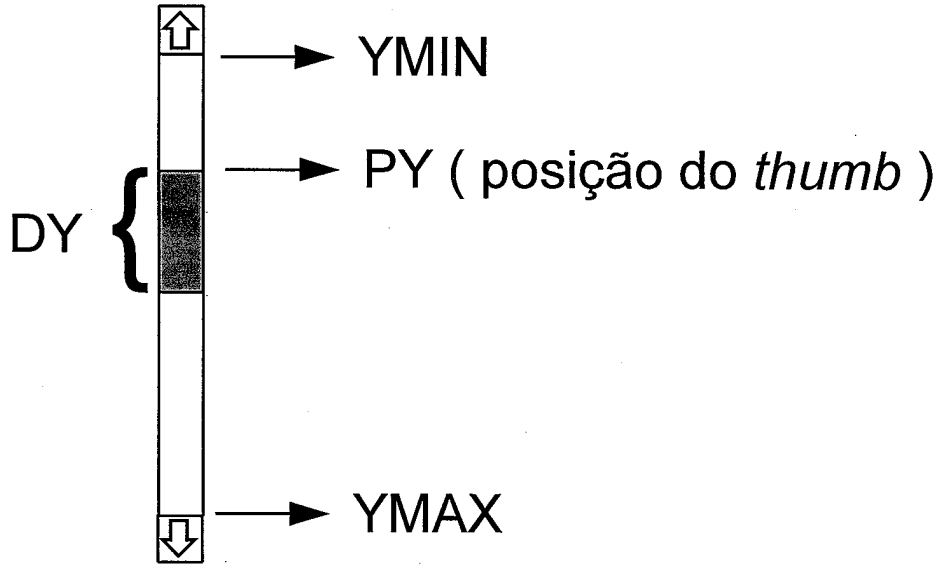
❑ IUP

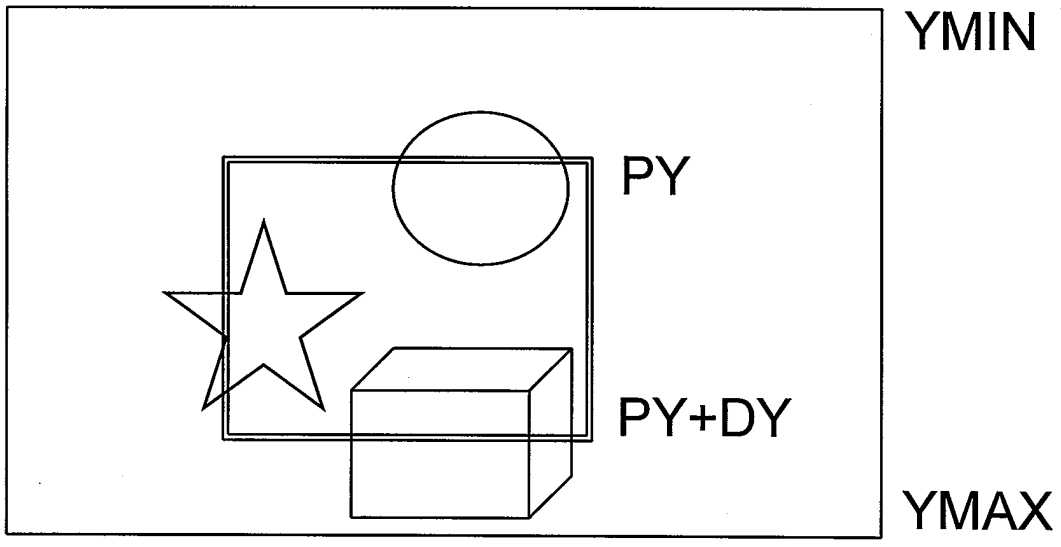
```
Ihandle *e = IupCanvas ("ação_redesenha")
```

❑ Principais Atributos

- ⇨ **SCROLLBAR**
- ⇨ **XMIN,XMAX,DX,POSX**
- ⇨ **YMIN,YMAX,DY,POSY**
- ⇨ **CURSOR** (NONE, ARROW, BUSY, CROSS, HAND, PEN, etc)

SCROLLBAR





❑ Ações de *repaint* e *resize*

⇨ CANVAS precisa ser redesenhado.

✔ `int função_nome (Ihandle *h, double x, double y)`

`h : id. do canvas`

`x : posição do thumb no scrollbar horiz.`

`y : posição do thumb no scrollbar vertical.`

⇨ CANVAS tem o ser tamanho alterado.

✔ `int função_nome (Ihandle *h, int width, int height)`

`h : id. do canvas`

`width : Nova largura do canvas.`

`height : Nova altura do canvas.`

⇨ Exemplo:

```
canv = CANVAS
  [RESIZE_CB = acao_tamanho]
  (acao_redesenha)
```

IUP/CANVAS - Ações (cont.)

O elemento de interface *canvas* além de tratar dos eventos de *RESIZE* e *REPAINT*, também deve tratar dois eventos relacionados com o *mouse*, que são: movimento e botões do *mouse.*

❑ Ações relacionadas com o *mouse*

- Botões do *mouse*
  - ✔ int função_nome (Ihandle *h, int botao, int mode, int x, int y, char *r)

    h : id. do *canvas*

    botao : # do botão (1/2/3) pressionado ou solto

    mode : estado (0-solto / 1- pressionado)

    x,y : posição do *mouse* (pixels)

    r : combinação de teclas com os botões.

- Movimento do *mouse*
  - ✔ int função_nome (Ihandle *h, int x, int y, char *r)

    h : id. do *canvas*

    x,y : posição do *mouse* (pixels)

    r : combinação de teclas com os botões.

- Exemplo:

```
canv = CANVAS
[ BUTTON_CB = acao_botao,
  MOTION_CB = acao_move]
(acao_redesenha)
```

Tipos de *prompt / eco*

Na programação sequencial, utilizando um sistema gráfico (por exemplo o GKS), ao utilizar-se uma função de entrada, pode-se determinar qual o tipo de *prompt / eco* desejado, bem como avaliar o *status* da informação.

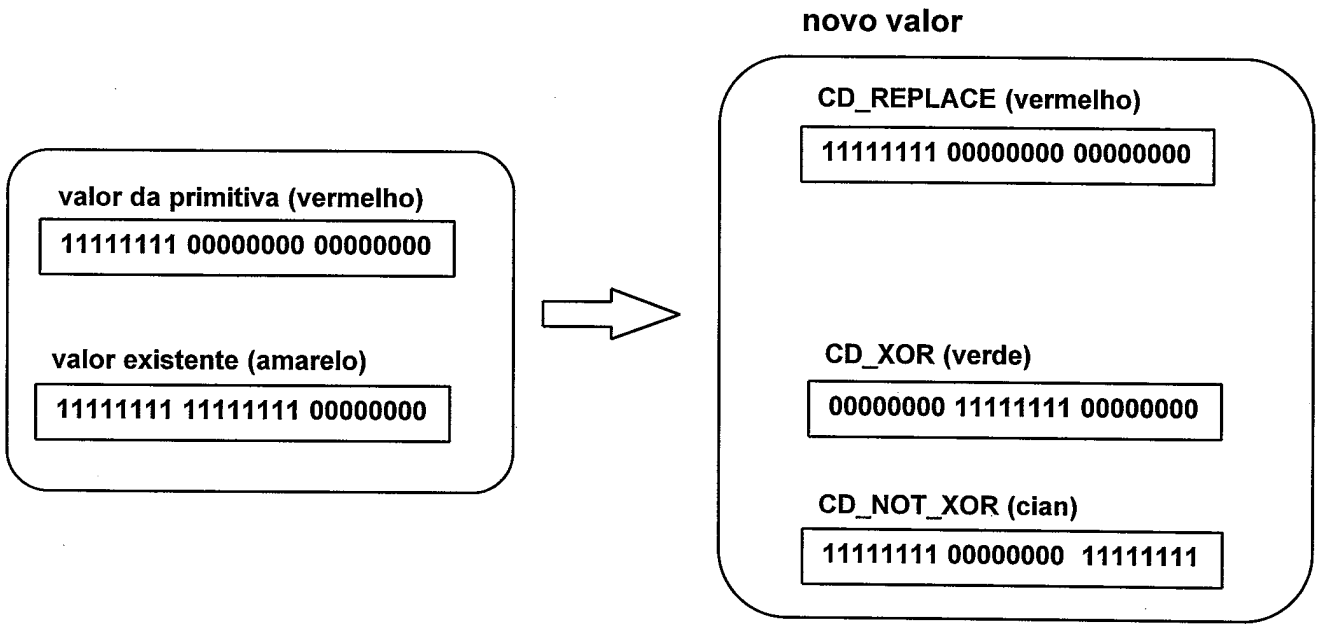
Na programação orientada à eventos este tipo de *feedback* não é "dado de graça", tem que ser programado dependendo do *status* corrente da aplicação.

Será visto o tipo Rubber Band Line.

Modo de Escrita

O modo de escrita diz respeito ao valor que será atribuido a um *pixel* afetado por uma primitiva de desenho. Este valor pode ser simplesmente o valor determinado pela primitiva (CD_REPLACE) ou pode ser uma combinação deste valor com o anteriormente existente na VS (CD_XOR ou CD_NOT_XOR). Estes últimos existem para desenhar temporariamente primitivas. Duas chamadas de desenho de uma mesma primitiva nestes modos tem o efeito de desenhar e apagar, respectivamente.

int cdWrite (int mode);



TeCGraf

Rubber Band Line

- Algoritmo

O algoritmo será dividido em duas partes, uma com relação ao movimento do mouse e o outro com relação aos botões do mouse.

Movimento do mouse

Posições : primeiro_pto, pto_anterior, pto_corrente

```
Se estado corrente solicita Rubber Band Line então
    Enquanto houver movimento do mouse e o
      primeiro ponto já foi dado faça
        Ativa o modo XOR
        Desenha uma linha do primeiro ponto até o
        ponto anterior
        Desenha uma linha do primeiro ponto até o
        ponto corrente
        Desativa o modo XOR ( Replace )
    Fim_enquanto
Fim_se
```

Botões do mouse

```
Se estado corrente solicita Rubber Band Line então
    Se for pressionado o botão da esquerda então
        Se será dado o primeiro ponto então
            Define-se que será dado o próximo ponto
            Marca-se o primeiro ponto
        Senão
            Ativa o modo XOR
            "Apaga" a linha gerada pelo mov. do mouse
             Desativa o modo XOR
             Desenha a linha final
        Fim-se
    Senão
        Se for pressionado o botão da direita então
            Ativa o modo XOR
            "Apaga" a linha gerada pelo mov. do mouse
             Desativa o modo XOR
        Fim_se
    Fim_se
    Atualiza estado corrente
Fim_se
```

Implementação :

```
/* Definicao dos estados do sitema */
typedef enum  {
 NADA,
 RUBBER,
} Tarefa;

/* Definicao do estado corrente */
static Tarefa TarefaCorrente = NADA;

/* Funcoes do Sistema */
int CB_Fim ( void )
{ return IUP_CLOSE; }


int CB_Rubber ( void )
{
 TarefaCorrente = RUBBER;
 PrimeiroPonto = TRUE;
 return IUP_DEFAULT;
}
```

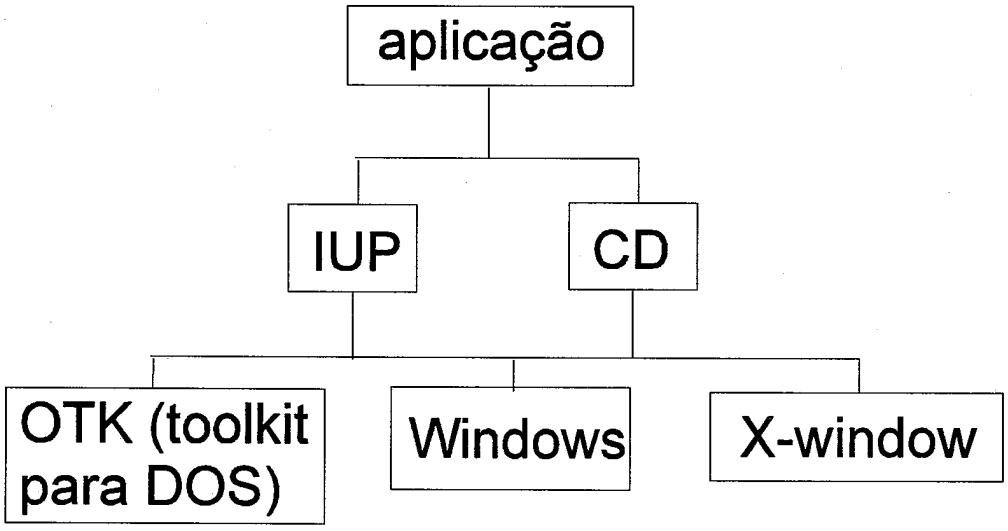
```
/* b -> botao pressionado
** mode -> 1 - pressionado  0 - solto  ---- x,y  -> posicao no canvas  */
int CB_Botao ( Ihandle *h, int botao, int mode, int x, int y, char *r )
{
/* No caso de nao se ter nenhuma primitiva escolhida qualquer botao
   pressionado nao executura nenhuma acao, retornando p/ interacao
   com o usuario.                                                    */
/* Determina qual o "status" do botao, se pressionado ou solto       */
/* Calcula as coordenadas em WC ( uma vez que os parametros x e y sao
   dados em pixels                                                   */
 /* No caso de se ter selecionado  "RubberBand".                     */
 if (TarefaCorrente == RUBBER) {
/* No caso de se ter sido pressionado o botao da esquerda            */
   if ((botao_status == ButtonDown) && (botao == IUP_BUTTON1)) {
/* Testa se sera o primeiro ponto a ser dado pelo usuario. Se for o caso,
   so e necessario "setar" os "flags" corretamente e guardar a posicao.
   Caso contrario, tem-se que ativar o "modo Xor" para "apagar" a linha
   do "RubberBand", desenhar a linha gerada , "setar" flags e guardar
   posicoes.                                                         */
    if (PrimeiroPonto == TRUE) {
    }
    else {
    }
   }
/* No caso de se ter pressionado o botao da direita,  ESCAPE         */
   if (botao == IUP_BUTTON3) {
/* Desenhar no modo Xor (replace) a ult. linha e atualizar estado    */
   }
 }
return IUP_DEFAULT;
}
```

```
int CB_Mouse ( Ihandle *h, int x, int y, char *r )
{
/* No caso de nao se ter nenhuma primitiva escolhida o movimento do mouse
   nao executura nenhuma acao, retornando p/ interacao com o usuario.  */
/* Calcula as coordenadas em WC ( uma vez que os parametros x e y sao
   dados em pixels                                                      */

/* No caso de se ter selecionado  "RubberBand".                         *
 if (TarefaCorrente == RUBBER) {
/* Testa-se para saber se e o primeiro ponto dado, se for nao se desenha
   no modo Xor (replace) a linha anterior, uma vez que nao ha linha
   anterior para ser desenhada                                          */
   if (PrimeiroPonto == FALSE)  {
    if ( Rubberflag == TRUE )  {
      Rubberflag = FALSE;
    }
    else {
    }
   }
   return IUP_DEFAULT;
 }
```

CanvasDraw – CD

Arquitetura de um programa que usa IUP e CD



Funções Gerais

`int cdCanvas(Ihandle *h)`

Esta função estabelece que as próximas primitivas devem ser desenhadas no canvas cujo *handle* do IUP é `h`. Ela é a única função que faz a ligação do canvas do IUP com as funções do CD.
Note que, entre uma e outra *callback*, o IUP pode ter selecionado outro *canvas*; esta função deve ser chamada toda vez que se pretende desenhar no *canvas* identificado pelo *handle* fornecido.
Retorno: CD_OK, *canvas* válido para desenho.
CD_ERROR, *canvas* inválido para desenho.

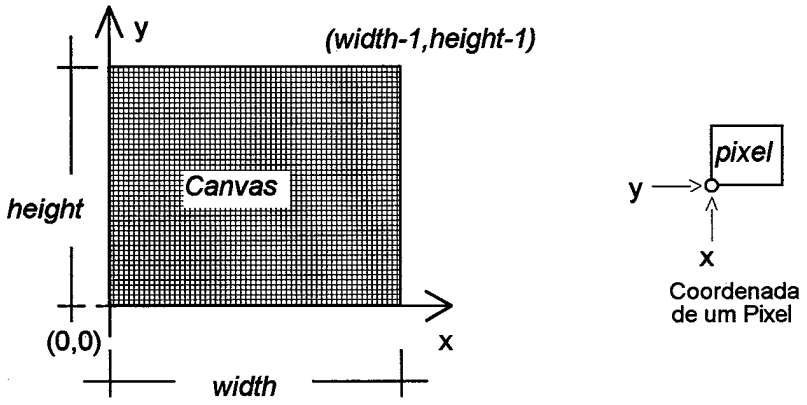
`void cdFlush(void)`

Garante o envio das primitivas do *canvas*.

`void cdClear(void)`

Esta função limpa a área do *canvas* com a cor de fundo corrente.

Sistema de Coordenadas




```
void cdGetCanvasSize(int *width, int *height,
                                 double *mm_width, double *mm_height)
```

Retorna a largura e altura do canvas em *pixels* (**width**, **height**) e em milímetros (**mm_width**, **mm_height**). Devido a indefinições de tamanho dos monitores, os valores retornados em milímetros são aproximados.
Para facilitar aplicações que não desejem todos os parâmetros, os valores só são retornados se os ponteiros não forem NULL. Assim sendo, uma aplicação que não desejar saber os valores em *pixels* pode fazer uma chamada do tipo:
cdGetCanvasSize(NULL, NULL, &w, &h)

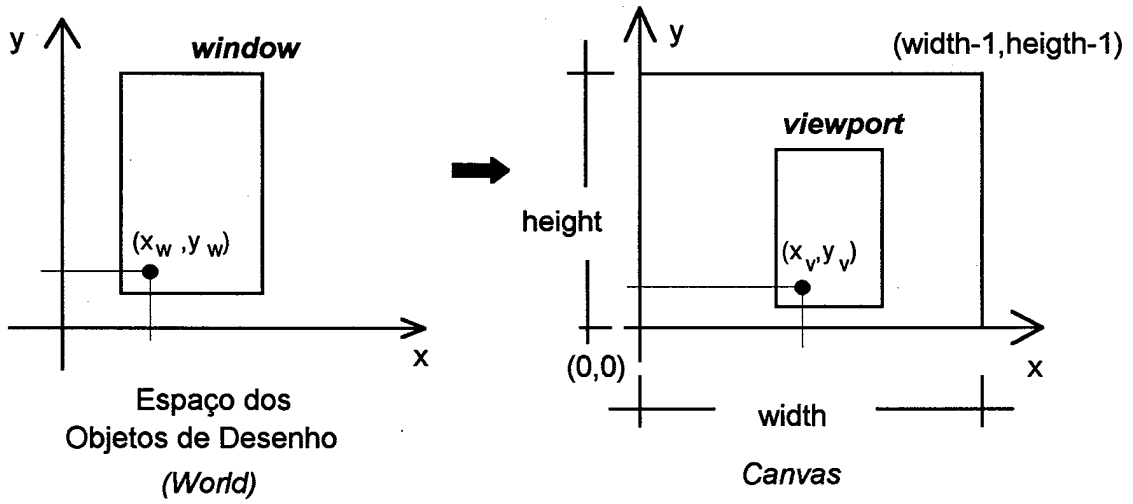
```
void cdClip(int mode)
```

Esta função ativa ou desativa o efeito de *clipping* dentro do *canvas*. Os valores possíveis do parâmetro **mode** são CD_OFF e CD_ON. O valor inicial (*default*) é CD_OFF, isto é, *clipping* desativado. Neste caso as primitivas de desenho só são cerceadas quando saem fora do *canvas*. Retorna o valor corrente anterior.

```
void cdClipArea(int xmin, int xmax, int ymin, int ymax)
```

Esta função define um retângulo de *clipping* dentro do *canvas*. A área fora deste retângulo permanece inalterada sob ação de qualquer das outras funções do CD.
Valores de **xmin** ≥ **xmax** ou **ymin** ≥ **ymax** definem regiões inválidas e neste caso a chamada da função é ignorada.
O retângulo de *clipping* inicial (*default*) de um *canvas* corresponde ao seu tamanho total.

Transformação *window-viewport*



`void wdWindow(double xmin, double xmax, double ymin, double ymax)`

Estabelece a janela para o par *window-viewport* corrente.
Valores de **`xmin`** > **`xmax`** ou de **`ymin`** > **`ymax`** invertem o sentido do mapeamento e não são considerados pelo CD como um erro.
Valores de **`xmin`** = **`xmax`** ou **`ymin`** = **`ymax`** são considerados erro e a janela corrente fica inalterada.
A *window* inicial (*default*) é o *canvas* em milímetros, ou seja (0, **`mm_width`**, 0, **`mm_height`**).

`void wdViewport(int xmin, int xmax, int ymin, int ymax)`

Valores de **`xmin`** ≥ **`xmax`** ou de **`ymin`** ≥ **`ymax`** invertem o sentido do mapeamento e não são considerados pelo CD como um erro. As igualdades reduzem o desenho a uma linha.
Estabelece a *viewport* para o par *window-viewport* corrente. O *viewport* inicial (*default*) de um *canvas* corresponde ao seu tamanho total.

Transformação *window-viewport* (cont.)

```
void wdWorld2Raster(double xw, double yw, int *xv, int *yv)
void wdRaster2World(int xv, int yv, double *xw, double *yw)

void wdMM2Raster(double mm_dx,double mm_dy,int *dx,int *dy)
void wdRaster2MM(int dx, int dy, double *dx, double *dy)
```

As duas primeiras transformam de coordenadas de objetos (*world*) para *raster* e vice-versa.
As duas últimas transformam de milímetros para *raster* e vice-versa.

```
void cdCanvas2Raster(int *x, int *y)
```

Converte as coordenadas *raster* fornecidas no sistema de coordenadas do IUP para o sistema de coordenadas *raster* do CD.

```
void cdGetClipArea(int *xmin,int *xmax,int *ymin,int *ymax)
```

Obtém o retângulo de *clipping* corrente.

```
void wdGetWindow(double *xmin, double *xmax,
                                        double *ymin, double *ymax)
```

Obtém a janela (*window*) corrente.

```
void wdGetViewport(int *xmin, int *xmax, int *ymin, int *ymax)
```

Obtém a *viewport* corrente.

Primitivas de Desenho

Funções de linhas

```
void cdLine(int x1, int y1, int x2, int y2)
void wdLine(double x1, double y1, double x2, double y2)
```

Desenha o segmento de reta (**x1**,**y1**) e (**x2**,**y2**).

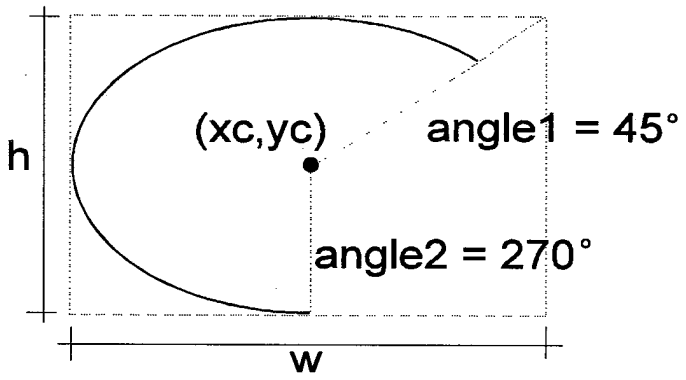
```
void cdArc(int xc, int yc, int w, int h,
                                    double angle1, double angle2)
void wdArc(double xc, double yc, double w, double h,
                                    double angle1, double angle2)
```

Esta função desenha um arco de elipse centrado em (**xc**,**yc**) e inscrito em um retângulo de largura **w** e altura **h**. Os limites do arco são dados por dois ângulos, ligados no sentido trigonométrico. Note-se que os pontos inicial e final do arco são:

xi = (w/2)*sin(angle1) e **yi = (h/2)*cos(angle1)**

xf = (w/2)*sin(angle2) e **yf = (h/2)*cos(angle2)**

Ou seja, estes ângulos não são os que aparecem no desenho, mas sim os correspondentes ao arco de círculo unitário que é mapeado na elipse.



Primitivas de Desenho (cont.)

Funções de marcas

```
void cdMark(int x, int y)
void wdMark(double x, double y)
```

Coloca uma marca na posição (**x**,**y**).

Funções de áreas

```
void cdBox(int x1, int x2, int y1, int y2)
void wdBox(double x1, double x2, double y1, double y2)
```

Esta função desenha um retângulo preenchido com a cor atual.
Argumentos: As coordenadas dos vertices inferior esquerdo e superior direito, na ordem **x1**, **x2**, **y1**, **y2**.

```
void cdSector(int xc, int yc, int w, int h,
                                    double angle1, double angle2)
void wdSector(double xc, double yc, double w, double h,
                                    double angle1, double angle2)
```

Esta função desenha um setor de elipse centrado em (**xc**,**yc**) e inscrito em um retângulo de largura **w** e altura **h**. Os limites do setor são dados por dois ângulos, ligados no sentido trigonométrico. Note-se que os pontos inicial e final do arco são:

xi = (w/2)*sin(angle1) e **yi = (h/2)*cos(angle1)**

xf = (w/2)*sin(angle2) e **yf = (h/2)*cos(angle2)**

Ou seja, estes ângulos não são os que aparecem no desenho, mas sim os correspondentes ao arco de círculo unitário que é mapeado na elipse.

Primitivas de Desenho (cont.)

Funções de texto

```
void cdText(int x, int y, char *s)
void wdText(int x, int y, char *s)
```

Esta função coloca um texto na tela.

```
void cdTextSize(char *s, int *width, int *height)
```

Esta função retorna o tamanho da caixa envolvente para um string em *pixels*.

```
char *cdFontSize(int *w, int *h)
```

Esta função retorna o nome da fonte atualmente em uso, sua largura máxima e sua altura máxima.
Retorno: Nome da fonte em uso.

Primitivas de Desenho (cont.)

Funções de vários vértices (de linhas e/ou de área)

```
void cdBegin(int mode);
void cdVertex(int x, int y);
void wdVertex(double x, double y);
void cdEnd(void);
```

Exemplo:

```
cdBegin(mode);
    cdVertex(x0, y0);
    cdVertex(x1, y1);
...
cdEnd();
```

ou

```
cdBegin(mode);
    wdVertex(x0, y0);
    wdVertex(x1, y1);
...
cdEnd();
```

Desenha uma linha poligonal aberta (CD_OPEN_LINES), fechada (CD_CLOSED_LINES) ou uma área poligonal (CD_FILL) definida pelos vértices (`x0`,`y0`), (`x1`,`y1`), ...

Atributos

Atributos globais:

1. **cor de desenho**: define a cor do desenho (*foreground*).
2. **cor do fundo**: define a cor de fundo (*background*). Quando o atributo de opacidade for opaco (CD_OPAQUE), este atributo afeta todas a primitivas de desenho e imagens.
3. **aproximação de cor**: define a como uma cor não existente nos dispositivos baseados em palhetas deve ser aproximada pelo CD. Valores CD_NEAREST ou CD_DITHER.
4. **opacidade da cor do fundo**: define se a cor de fundo deve ou não ser atribuída às primitivas correspondentes. Afeta todas a primitivas de desenho e imagens. Valores: CD_TRANSPARENT ou CD_OPAQUE.
5. **modo de escrita**: define como a cor do *pixel* da primitiva c_{prim} deve ser combinada com a cor anterior do *pixel*, c_{old}, de forma a obter a nova cor. Ou seja:

   $c_{new} = c_{prim}$ (*modo de escrita*) c_{old}.

   Afeta todas a primitivas de desenho e imagens. Valores: CD_REPLACE ou CD_XOR.
6. **padrão de *bits* (*stipple*)**: define um padrão de retangular de *bits* que forma um mosaico a partir da origem do sistema de coordenadas (0 representa a cor de fundo e 1 a cor de desenho). Afeta todas a primitivas de desenho e imagens tipo *bitmaps*. O CD fornece alguns padrões básicos simples: CD_SOLID, CD_BRICK, CD_DIAMOND, CD_HATCH_H, CD_HATCH_V, CD_HATCH_D45 e CD_HATCH_D135.
7. **padrão de cores (*pattern*)**: Matriz retangular de índices de cores (*bytes*). Define um mosaico de cores a partir da origem do sistema de coordenadas.
8. **estilo de interior**: Define se as cores geradas no interior das primitivas devem utilizar a cor de *foreground* (CD_FOREGROUND), ou com o padrão de *bits* (CD_STIPPLE) ou com o padrão de cores (CD_PATTERN).

Atributos (cont.)

Atributos que afetam a aparência das primitivas tipo linha:

1. **estilo traço da linha**: padrão de 16 *bits* que define se o traço de uma linha deve ser contínuo, tracejado, traço-ponto, etc. O CD fornece os seguintes padrões básicos: CD_CONTINUOUS, CD_DASHED, CD_DOTTED, CD_DASH_DOT e CD_DASH_DOT_DOT.
2. **espessura de linha**: número de *bits* que define o lado do quadrado que é usado para gerar a linha.
3. **junta de linha**: define o estilo de junta de linhas grossas.
4. **extremidade de linha**: define o estilo de extremidade de linha.

Atributos que afetam a aparência das primitivas tipo marcas:

1. **tipo de marcas:** tipo de marca. Valores: CD_POINT, CD_PLUS, CD_STAR, CD_CIRCLE e CD_X.
2. **tamanho das marcas**: tamanho, em *pixels* das marcas.

Atributos que afetam a aparência das primitivas tipo *fill*:

1. **estilo das arestas e/ou interior**: determina se as primitivas de área devem desenhar as bordas (CD_BORDER) ou preencher apenas o interior (CD_INTERIOR), ou ambos (CD_BORDER&CD_INTERIOR).

Atributos que afetam a aparência das primitivas tipo texto:

1. **fonte e tamanho do texto**: define a fonte e o tamanho do texto do sistema.
2. **alinhamento do texto**: define a posição de um texto com relação ao ponto de referência. Valores: CD_NORTH, CD_SOUTH, CD_EAST, CD_WEST, CD_NORTH_EAST, CD_NORTH_WEST, CD_SOUTH_EAST, CD_SOUTH_WEST e CD_CENTER.

Atributos (cont.)

Funções para definição dos valores dos atributos (retornam o valor corrente anterior):

```
int  cdColorAprox(int mode)
int  cdBackOpacity(int opacity)
int  cdWriteMode(int mode)
int  *cdSimpleStipple(int stipple[16])
void cdStipple(int n, int m, unsigned char *stipple)
void cdPattern(int n, int m, unsigned char *pattern)
int  cdInteriorStyle(int style)
int  cdLineStyle(int style)
int  cdLineWidth(int width)
int  cdMarkerType(int type)
int  cdMarkerSize(int size)
int  cdFillStyle(int style)
char *cdFont(char *name)
int  cdTextAlignment(int alignment)
```

Modelo de Cor

```
long int cdCodeColor(double red, double green, double blue)
double cdRed(long int color)
double cdGreen(long int color)
double cdBlue(long int color)
```

```
int cdGetColorPlanes(int *cdcolors)
```

Esta função retorna quantos planos de cor (*bit/pixel*) o dispositivo possui e retorna em **cdcolors** quantas cores estão disponíveis para o CD (as demais cores estão sendo usadas pelo sistema gerenciador de janelas, por outros clientes e pelo IUP). Quando o CD não tem condições de saber quantas cores a aplicação pode definir (é indefinida pelo sistema nativo), o CD retorna em **cdcolors** uma aproximação dependente da implementação.

```
long int cdBackground(long int color)
```

Esta função define a cor de fundo. A função retorna a cor de fundo corrente definida pela aplicação anteriormente.
Valor *default* CD_BLACK.

```
long int cdForeground(long int color)
```

Esta função define a cor do desenho. A função retorna a cor de desenho corrente definida pela aplicação anteriormente.
Valor *default* CD_WHITE.

```
long int cdGetBackground(long int *system_color)
long int cdGetForeground(long int *system_color)
```

Retornam as cores correntes de fundo e de desenho, respectivamente. No valor de retorno encontra-se o valor solicitado pela aplicação e no parâmetro **system_color** a cor aproximada pelo sistema.

Modelo de Cor (cont.)

`int cdPalette(unsigned char index, long int color)`

Coloca a cor **`color`** na posição **`index`** da tabela de cores do CD. Esta tabela é limitada nesta implementação de 1 a 255. O índice 0 (zero) está reservado para ter um significado especial para várias funções.
A função retorna CD_OK se a cor especificada for alocada na palheta do dispositivo e CD_NEAREST caso ela tenha sido aproximada por uma existente.

`unsigned char cdForegroundIndex(unsigned char index)`
`unsigned char cdBackgroundIndex(unsigned char index)`

Estabelecem que a cor corrente para o *foreground* (ou *background*) é a cor cujos RGB estão na tabela de cores do CD na posição **`index`**. Retorna o índice da cor corrente anterior. Caso a cor de *foreground* (ou *background*) corrente tenha sido definida com a função **`cdForeground`** (ou **`cdBackground`**), o valor de retorno é 0 (zero).

Funções do Imagens

```
void cdPutPixel(int x, int y)
```

Esta função coloca a cor corrente de *foreground* no *pixel* de coordenadas (**x**,**y**). Ela é afetada pelo modo de escrita corrente (CD_REPLACE ou CD_XOR).

```
long int cdGetPixel(int x, int y)
```

Esta retorna a cor (R,G,B) do *pixel* de coordenadas (**x**,**y**).

```
void cdScanLine(int x1, int x2, int y)
```

Esta função coloca uma linha de *scan* no *canvas*. Ela é afetada pela estilo de interior (CD_SOLID, CD_STIPLE ou CD_PATTERN) e pelo modo de escrita (CD_REPLACE ou CD_XOR).

Funções do Imagens (cont.)

`void *cdGetImage(int x1, int y1, int x2, int y2)`

Esta função captura a imagem contida no retângulo definido por (**`x1`**,**`x2`**,**`y1`**,**`y2`**). Ela armazena memoria para isto e retorna um ponteiro para a área criada. É responsabilidade do usuario liberar esta área através da função **`cdFreeImage`**.

`void cdPutImage(int x, int y, void *image)`

Esta função coloca uma imagem guardada com **`cdGetImage`** de volta na tela. Ela recebe a coordenada do ponto que deve ser o novo canto inferior esquerdo da imagem. Ela é afetada pelo modo de escrita corrente (CD_REPLACE ou CD_XOR).

`void cdMoveImage(int x1, int y1, int x2, int y2, int dx, int dy)`

Esta função pega a imagem contida no retângulo definido por (**`x1`**,**`x2`**,**`y1`**,**`y2`**) e translada no canvas de (**`dx`**,**`dy`**).

`int cdFreeImage(void *image)`

Esta função libera a imagem.
Retorna CD_OK caso tenha sido bem sucedida e CD_ERROR caso contrário.

Funções do Imagens (cont.)

```
void *cdCreateImage(int w, int h, unsigned char *index_pixmap)
```

Esta função recebe um *pixmap* de índices de cor armazenados em um vetor e o converte para o formato usado por **cdPutImage**. Neste vetor, as primeiras **w** posições são a linha inferior e as demais as linhas subseqüentes superiores.
Ela armazena memoria para isto e retorna um ponteiro para a area criada. E responsabilidade do usuario liberar esta area através da função **cdFreeImage**.
Argumentos: Largura e altura da imagem e o *pixmap*.
Retorno: Ponteiro para vetor a ser usado em **cdPutimage**.

```
void cdPutPixmap(int x, int y, int i, int j,
                         int n, int m, unsigned char *index_pixmap)
```

Esta função coloca um *pixmap* retangular no *canvas* de tal forma que o ponto (**i**,**j**) do *pixmap* esteja na posição (**x**,**y**) do *canvas*. O pixmap é definido por **n** linhas e **m** colunas de índices de cor. Note-se que se o atributo da cor de fundo for CD_TRANSPARENT os índices com valor 0 (zero) não são colocados na tela. Os *pixels* colocados são ainda afetados pelas modo de escrita CD_REPLACE ou CD_XOR.