EGOLib — Manual de Referência

Eduardo Aguiar Patrocínio
Pedro Jussieu de Rezende

Relatório Técnico DCC–29/93

Dezembro de 1993

EGOLib — Manual de Referência

Eduardo Aguiar Patrocínio*
Pedro Jussieu de Rezende†

Departamento de Ciência da Computação
Universidade Estadual de Campinas
13081-970 Campinas, SP

Sumário

Apresentamos a descrição detalhada de uma biblioteca de funções (**EGOLib**) construída sobre o sistema *X* para manipulação de objetos gráficos, que provê facilidades para atualização de tais objetos enquanto modificações de atributos são realizadas. Embora a biblioteca *Xlib* proveja funções para esta finalidade, é desejável que se possa permitir ao usuário um acesso de mais alto nível a modificações de tais atributos e de maneira uniforme.

O objetivo da **EGOLib** é prover tais funções ao usuário de forma homogênea, permitindo a criação de código muito mais elegante do que é possível usando-se puramente funções de *Xlib*.

Neste documento é apresentado um manual de referência desta biblioteca, constando de uma descrição detalhada dos objetos gráficos e de seus atributos. Uma descrição das características básicas de seu funcionamento pode ser encontrado em [PR93].

*Pesquisa desenvolvida com suporte financeiro parcial do CNPq — Conselho Nacional de Desenvolvimento Científico e Tecnológico

†Pesquisa desenvolvida com suporte financeiro parcial do CNPq através dos auxílios 300157/90-8 e 500787/91-3

Conteúdo

Lista de Figuras

1 Introdução

EGOLib (**E**lementary **G**raphical **O**bjects **Lib**rary) é uma biblioteca de funções para a manipulação de objetos gráficos cujo objetivo é o de criar uma camada acima à do *Xlib*.

Uma descrição dos fundamentos utilizados, de sua motivação, suas vantagens e uma descrição de seus objetos podem ser encontrados em [PR93]. Alguns exemplos destes objetos são apresentados na figura 1.

Organização

O restante deste documento consiste de, primeiramente, uma descrição da classe **EGOView**, que implementa um sistema de coordenadas virtuais, na seção 2. Esta classe encapsula as funções de desenho, provendo um sistema virtual de coordenadas.

Em seguida é apresentado na seção 3 um detalhamento dos atributos dos objetos gráficos como cor de borda e de fundo, largura de linha, nível de transparência para os objetos que são conhecidos pela biblioteca. Todos os atributos são descritos, tanto os comuns a todos os objetos quanto aos específicos a alguns objetos e uma listagem dos tipos enumerados a serem utilizados pelo programador.

Na seção 4, apresentamos uma descrição de cada um dos 12 tipos de objetos gráficos existentes, os atributos definidos, as definições de ponto de referência, do modo de destaque, dos parâmetros de criação, assim como as funções definidas para cada classe.

Finalmente, na seção 5, descrevemos detalhadamente cada uma das 85 funções que constituem a biblioteca **EGOLib**, especificando uma sinopse, seus argumentos, uma descrição, as estruturas utilizadas e as condições de erro.

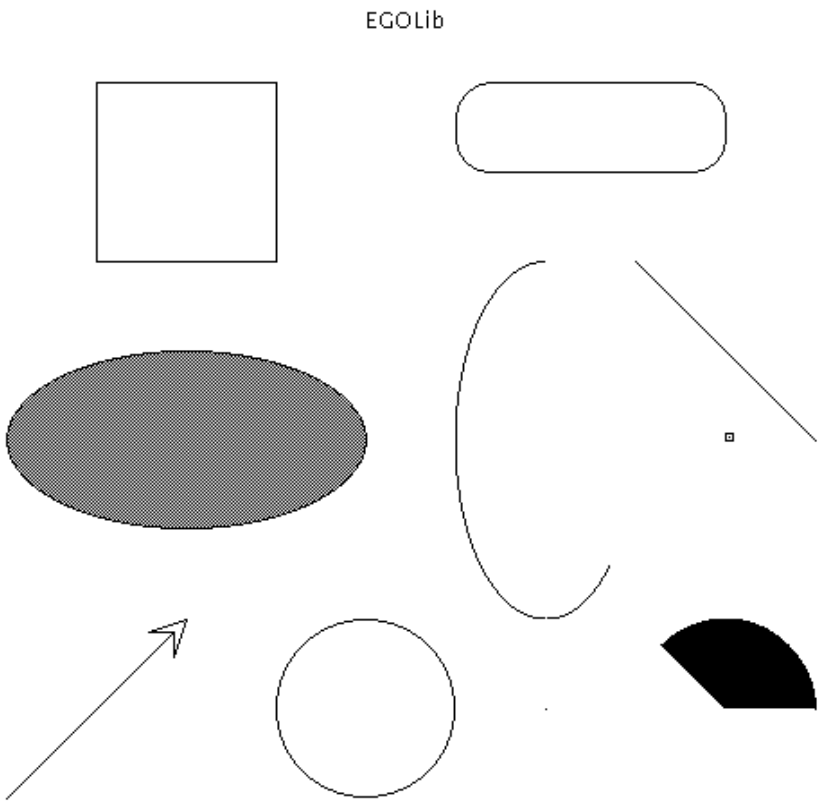

Figura 1: Exemplo de alguns objetos gráficos elementares

2 EGOView

2.1 Introdução

A classe **EGOView** implementa um sistema de coordenadas virtuais e encapsula as funções de desenho providas pelo sistema *Xlib*, convertendo automaticamente as coordenadas virtuais para coordenadas reais.

Para isto, é necessário fazer um processo de regularização que converte as coordenadas virtuais em coordenadas reais, além de tratar dos casos em que certos parâmetros passados recebem valores negativos, os quais precisam ser devidamente acertados para as respectivas chamadas de *Xlib*.

Para o construtor da classe **EGOView**, é necessário passar um Canvas, portanto, para cada Canvas, deve ser declarado um **EGOView**. Como existe uma relação natural entre um Canvas, um Display e um Window, nas chamadas das funções dos métodos da classe, não é necessário passar nenhum destes parâmetros.

Esta classe foi desenvolvida para ser utilizada com a **EGOLib**, que provê uma camada acima de **EGOView**, no que tange às funções de desenho. Contudo, a biblioteca **EGOLib** não incorpora todas as funções descritas na classe **EGOView**, sendo portanto ainda necessária a sua utilização para algumas funções, conforme descritas abaixo. Além disso, a classe aqui apresentada interage com a **EGOLib**, armazenando uma lista de todos os objetos gráficos associados a este **EGOView**.

A **EGOView** provê métodos para: realizar zoom, scroll; obter o canvas, paint window, display, e a window a ele associada; obter as coordenadas limitantes do canvas, do view, e da paint window; redefinir as coordenadas virtuais do **EGOView**; transformar coordenadas virtuais em reais e vice-versa.

Além disso, todas as funções de desenho do módulo *Xlib*, que

são encapsuladas pela **EGOView** são declaradas como *private*, já que, a priori, esta classe foi feita para ser utilizado com a biblioteca **EGOLib**, e as funções de **EGOLib** provêm uma camada acima desta, para o desenho.

Esta classe foi inspirada na classe *World*, implementada por Welson R. Jacometti, como parte do ambiente **GeoLab** [Jac92]. A classe *World*, além da definição de um sistema de coordenadas virtuais, realiza o suporte para a manipulação de eventos.

2.2 Descrição dos métodos

Este módulo apresenta basicamente os seguintes métodos:

zoom: Estes métodos permitem a realização de zoom, através das funções de *zoom_in* e *zoom_out*. Além disto, é possível a realização de zoom, armazenando a configuração anterior em uma pilha, através das funções *zoom_in_stack*, *zoom_out_stack*, sendo possível retornar à configuração anterior através do método *zoom_previous*. Podemos, ainda, retornar à configuração inicial, através do método *zoom*.

scroll: É possível a realização de scroll através de duas formas: pela manipulação de pixmaps, ou pelo ajuste dos parâmetros. A primeira forma é feita através das funções *pscroll_begin* e *pscroll_end*. Sendo o tamanho da paint window o mesmo da view window, deve-se utilizar a função *pscroll_same_size*. Caso contrário, deve-se utilizar a função *pscroll_paint_bigger*. A segunda forma, é feita diretamente através da função *scroll*.

obtenção dos parâmetros: A uma **EGOView**, estão associados um canvas, um display, uma view e uma paint window. Estes valores são obtidos através dos métodos: *get_canvas*, *get_paint*, *get_display*, *get_window*.

obtenção dos limites das janelas: O tamanho real do canvas pode ser obtido através da função *get_values*. Além disso, o tamanho da view e da paint windows podem ser obtidos através dos métodos *get_view* e *get_paint*.

redefinição das coordenadas virtuais: É possível redefinir o sistema de coordenadas virtuais de uma **EGOView**. Para isto, deve-se utilizar a função *define_EgoView*.

transformação de coordenadas: Dadas coordenadas virtuais, podemos obter as coordenadas reais correspondentes, utilizando os métodos *xcoord* e *ycoord*. A obtenção das coordenadas virtuais a partir de coordenadas reais é feito através de *virtual_xcoord* e *virtual_ycoord*.

redesenho da tela: Através da lista de objetos gráficos associados à **EGOView**, é possível redesenhar toda a janela, através da função *redraw_screen*.

obtenção da profundidade do display: A profundidade do display é obtida através do método *depth*.

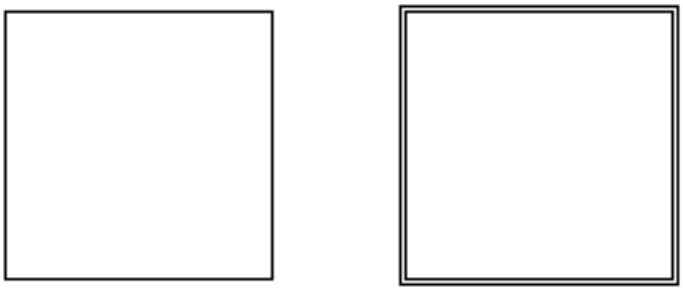

Figura 2: Exemplo do atributo highlight em um retângulo: à esquerda, o objeto com o atributo highlight desligado; à direita, o objeto com este atributo ligado

3 Descrição dos atributos

A descri,c ao dos atributos existentes nesta biblioteca pode ser encontrada em [PR93]. A seguir, damos alguns exemplos destes atributos.

3.1 Alguns atributos comuns a todos os objetos

Na figura 2, apresentamos um exemplo de highlight para um retângulo.

Na figura 3, apresentamos um exemplo dos níveis de transparência para um círculo.

Na figura 4, apresentamos um exemplo de linha sólida e de linha tracejada

Na figura 5, apresentamos um mesmo objeto, com diferentes fatores de zoom.

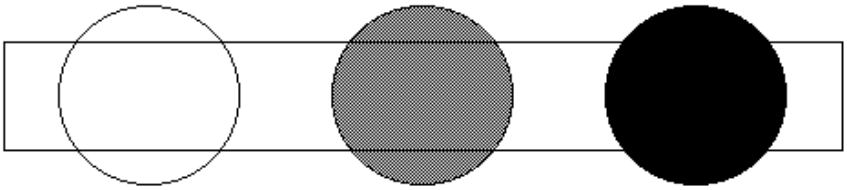

Figura 3: Os três níveis de transparência existente aplicados ao objeto círculo, sobreposto a um retângulo

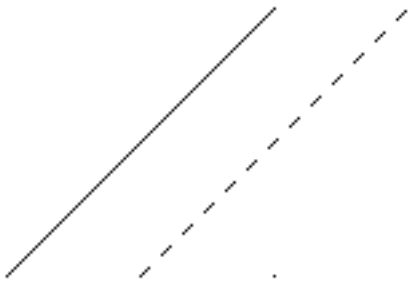

Figura 4: Dois segmentos, um com linha sólida e o outro com linha tracejada

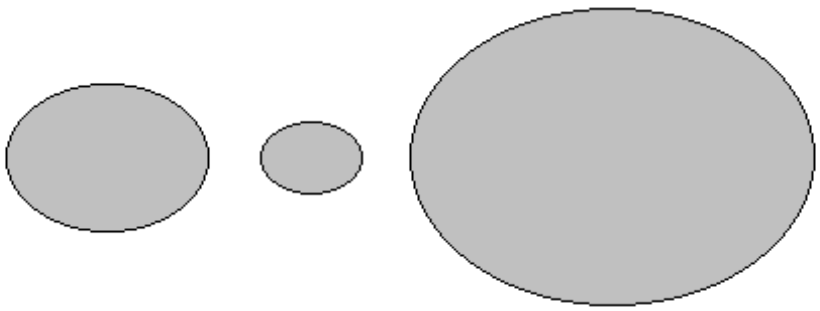

Figura 5: Uma elipse apresentada com diferentes fatores de zoom: à esquerda, em seu tamanho normal; ao centro, com zoom = 0.5; à direita, com zoom = 2

Na figura 6, apresentamos um objeto com sombra.

Na figura 7, apresentamos um exemplo da mudança de camada entre dois objetos.

3.2 Tipos Enumerados Definidos

Nos módulos da **EGOLib** existem diversos tipos definidos para os enumerados. A seguir, descrevemos estes tipos:

```
ArrowPosition =
    INITIAL_ARROW,
    FINAL_ARROW;

FunctionOption =
    PARALLEL_LINE,
    AXES_FIXED_LINE,
    TRIANGLE,
    ELIPSIS,
    PARABOLA,
    SIN;

SetOption =
    END_OF_SET_OPTION,
    ANGLE,
    ARROW_PRESENCE,
    ARROW_HEIGHT,
    ARROW_REWIND,
```

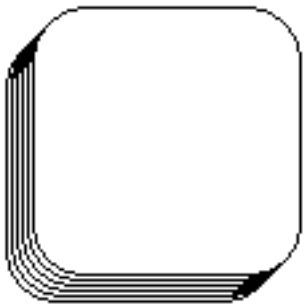

Figura 6: Um retângulo com as bordas ovaladas, com atributos ShadowX e ShadowY definidos como 10 pixels

```
    ARROW_WIDTH,
    BORDERCOLOR,
    FAMILY,
    FILLCOLOR,
    HEIGHT,
    HIGHLIGHT,
    INITIAL_ANGLE,
    LAYER,
    LINE_STYLE,
    LINE_WIDTH,
    RADIUS,
    SELECTED,
    SHADOW_X,
    SHADOW_Y,
    SIZE,
    SOUND,
    STRING,
    STRING_FONT_SIZE,
    STYLE,
    SYNCHRONIZED,
    TRANSPARENCY,
    VISIBLE,
    VOLUME,
    WIDTH,
    X,
    X_RADIUS,
    Y,
    Y_RADIUS,
    ZOOM;

StringPosition =
    HORIZONTAL,
    VERTICAL;

Transparency =
```

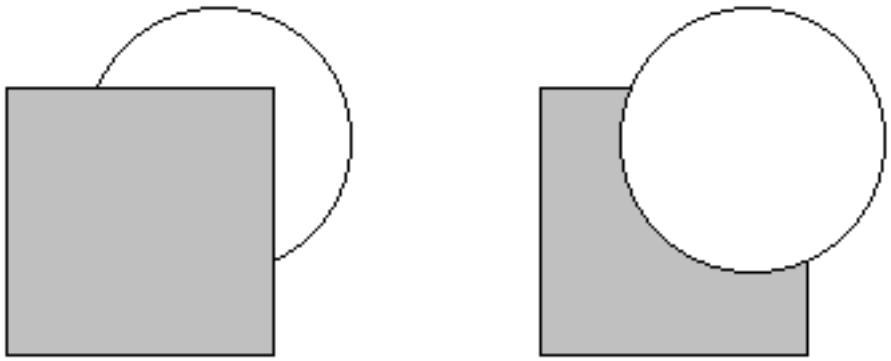

Figura 7: Exemplo de alteração de camada entre objetos: à esquerda, o retângulo está em uma camada superior ao círculo, enquanto que na direita, o retângulo está numa camada inferior

```
    TRANSPARENT,
    OPAQUE,
    TRANSLUCID;

CreateOvalType =
    GIVEN_RADIUS,
    GIVEN_RECTANGLE;

GroupMode =
```

```
    ONLY_THIS,
    EVERYONE,
    ONLY_THAT_MATCHES;

TreePosition =
    LEFT,
    RIGHT;
```

Figura 8: Diversos tipos de arcos

4 Descrição dos objetos

4.1 Arc

Descrição
Este objeto é um arco de circunferência;

Atributos

- Angle,
- InitialAngle,
- Radius,
- Size;

Ponto de referência
Centro da circunferência;

Highlight

Se (Angle = 0)
(Círculo Completo)
Círculo envolvendo o objeto
senão,
Arco envolvendo o objeto e os dois segmentos, ligando as extremidades do arco ao centro da circunferência;

Parâmetros de criação

1. Coordenadas X e Y do centro da circunferência,
   Radius,
   Angle,
   InitialAngle;
2. Coordenadas X e Y do centro da circunferência,
   GIVEN_RADIUS,

Figura 9: Diversos objetos do tipo arrow

Radius,
Angle,
InitialAngle;

3. Coordenadas X e Y do ponto de referência do retângulo,
GIVEN_RECTANGLE,
Size,
Angle,
InitialAngle;

Funções definidas

- Distance,
- GetAngle,
- GetInitialAngle,
- GetRadius,
- GetSize;

Exemplo

```
/* Cria um arco de circunferencia com
centro em (100, 100), raio 50, angulo
de 90 graus, comecando a 180 graus */

Arc *arc1 = new Arc (egoview, 100,
100, 50, 90, 180);

/* Cria um circulo de raio 60, usando
o objeto arc */

Arc *arc2 = new Arc (egoview, 100,
100, GIVEN_RADIUS, 60, 0, 0);

/* Cria um arco dentro do quadrado de
tamanho 40, posicionado na coordenada
(150, 50); */

Arc *arc3 = new Arc (egoview, 150,
50, GIVEN_RECTANGLE, 40, 90, 180);
```

Observações

O atributo Size é sempre igual ao dobro do atributo Radius;

4.2 Arrow

Descrição

Este objeto é um segmento, que pode ter uma seta em suas pontas;

Atributos

- ArrowHeight,
- ArrowWidth,
- ArrowRewind,
- Height,
- Width;

Ponto de referência

Segue a mesma regra do segmento (ver Segment - Ponto de referência)

Highlight

Retângulo, envolvendo o objeto;

Parâmetros de criação

1. Coordenadas X e Y do ponto de referência,
   Height,
   Width,
   ArrowHeight,
   ArrowWidth,
   ArrowRewind;
2. Coordenadas X e Y do ponto de referência,
   Height,
   Width,
   InitialArrowPresence,
   FinalArrowPresence,
   ArrowHeight,
   ArrowWidth,
   ArrowRewind;

Funções definidas

- GetArrow,
- GetArrowHeight,
- GetArrowWidth,
- GetArrowRewind,
- SetArrow,
- SetArrowHeight,
- SetArrowWidth,
- SetArrowRewind,
- SetHeight,
- SetWidth;

Exemplo

```
/* Cria um segmento comecando na
coordenada (100, 100), de altura e
comprimento 70, com uma seta no final
deste segmento, ocupando 10% do
tamanho do segmento, com largura
perpendicular ao segmento de 15
pontos e um recuo de 10 pontos */

Arrow *arrow1 = new Arrow (egoview,
100, 100, 70, 70, 0.10, 15, 10);

/* Cria um segmento, utilizando o
objeto Arrow */

Arrow *arrow2 = new Arrow (egoview,
100, 100, 70, 70, 0, 0, 0);

/* Cria um triangulo isosceles,
utilizando Arrow */

Arrow *arrow3 = new Arrow (egoview,
100, 100, 70, 70, 1, 0, 0);
```

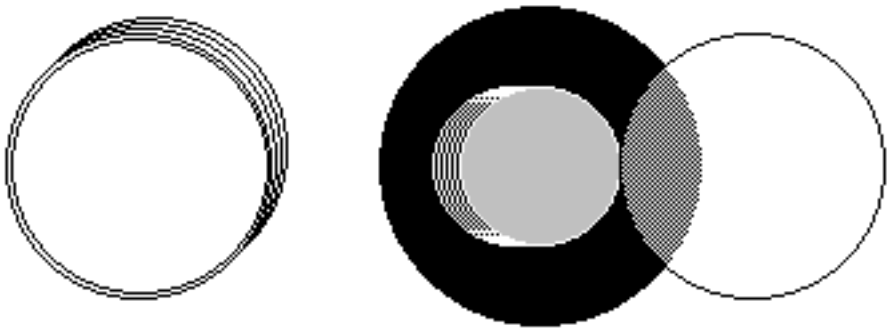

Figura 10: Diversos círculos com diferentes atributos

```
/* Cria um segmento com uma seta em
cada extremidade (ambas apresentam o
mesmo formato) */

Arrow *arrow4 = new Arrow (egoview,
100, 100, 70, 70, True, True, 0.10,
15, 10);
```

Observações

Se ArrowHeight = 0,
objeto é um segmento;
Se ArrowRewind = 0 e
ArrowHeight = 1,
objeto é um triângulo;

4.3 Circle

Descrição
Este objeto é um círculo;

Atributos

- Radius,
- Size;

Ponto de referência

Centro do círculo;

Highlight
Um outro círculo, envolvendo
este círculo;

Parâmetros de criação

1. Coordenadas X e Y do ponto
   de referência,
   Radius;

2. Coordenadas X e Y do ponto de referência,
   GIVEN_RADIUS,
   Radius;
3. Coordenadas X e Y do ponto de referência do retângulo que encapsula o círculo,
   GIVEN_RECTANGLE,
   Size;

Funções definidas

- Distance,
- GetRadius,
- GetSize,
- SetRadius,
- SetSize;

Exemplo

```
/* Cria um circulo de raio 50
centrado na coordenada (200, 200); */
Circle *circle1 = new Circle
(egoview, 200, 200, 50);

/* Cria um outro circulo de raio 70
com centro em (100, 100); */

Circle *circle2 = new Circle
(egoview, 100, 100, GIVEN_RADIUS,
70);

/* Cria um terceiro circulo dentro de
um quadrado cujo canto superior
esquerdo está posicionado na
coordenada (50,50) e possui tamanho
100 */

Circle *circle3 = new Circle
(egoview, 50, 50, GIVEN_RECTANGLE,
100);
```

Observações

O atributo Size é sempre igual ao dobro do atributo Radius;
Ao se criar um objeto, dando o retângulo, o ponto de referência exigido é o do retângulo, e não mais o centro do círculo.

4.4 Dot

Descrição

Este objeto é apenas um pixel na tela;

Atributos

Não possui nenhum atributo próprio;

Ponto de referência

É a própria localização do pixel;

Highlight

Quadrado de 5 pixels de lado, envolvendo o ponto;

Parâmetros de criação

- Coordenadas X e Y do pixel;

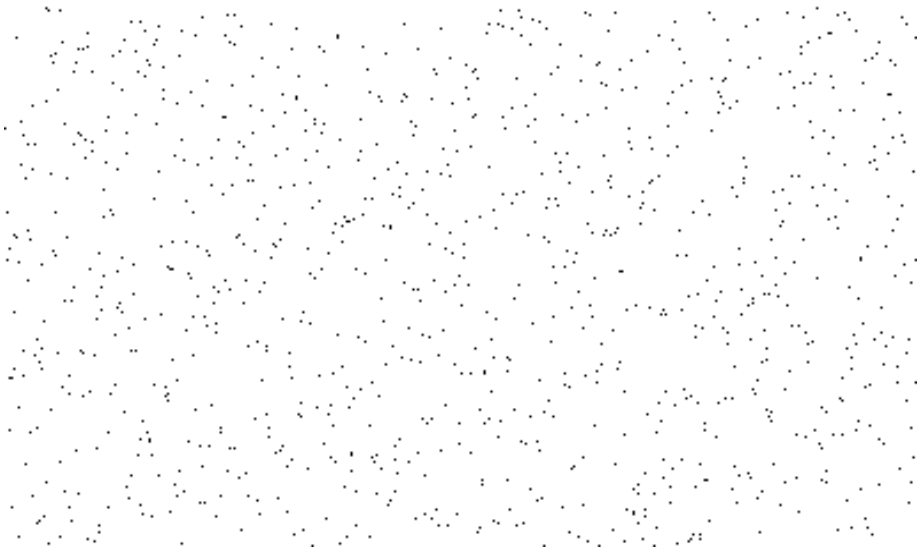

Figura 11: Um conjunto de muitos objetos do tipo dot

Funções definidas

- Distance;

Exemplo

```
EgoView *egoview = new EgoView
(canvas);
/* "Cria" um pixel na coordenada
virtual (100, 100) */
Dot *dot = new Dot (egoview, 100,
100);
```

Observações

Não é afetado por zoom's e alterações em FillColor;

4.5 Oval

Descrição

Este objeto é uma elipse;

Atributos

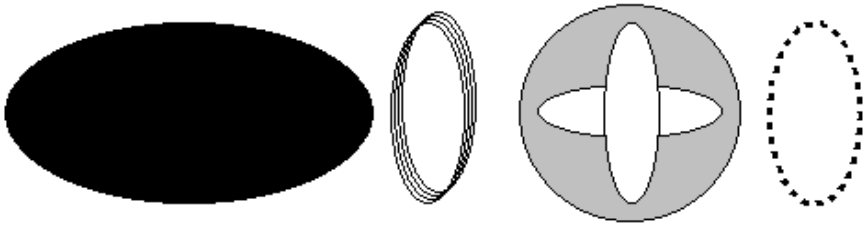

Figura 12: Exemplo de diversas elipses

- XRadius,
- YRadius;

Ponto de referência
Centro da elipse;

Highlight
Elipse maior, envolvendo o objeto;

Parâmetros de criação

1. Coordenadas X e Y do ponto de referência,
   XRadius,
   YRadius;
2. Coordenadas X e Y do ponto de referência,
   GIVEN_RADIUS,
   XRadius,
   YRadius,
3. Coordenadas X e Y do ponto de referência do retângulo,
   GIVEN_RECTANGLE,
   Height,
   Width;

Funções definidas

- Distance,
- GetXRadius,
- GetYRadius,
- GetHeight,
- GetWidth,
- SetXRadius,
- SetYRadius,
- SetHeight,
- SetWidth;

Exemplo

```
/* Cria uma elipse na coordenada
(100, 100), com raios 50 e 200 */

Oval *oval1 = new Oval (egoview, 100,
100, 50, 200);

/* Cria um circulo de raio 150,
atraves do objeto Oval */
```

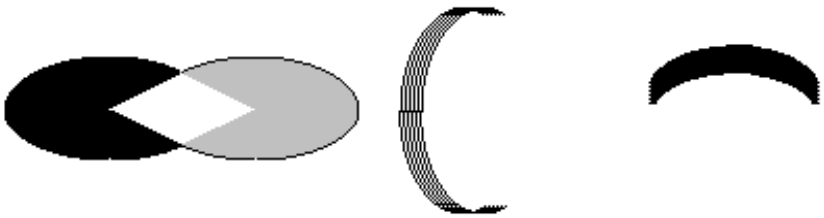

Figura 13: Alguns arcos de elipses interessantes

```
Oval *oval2 = new Oval (egoview, 100,
100, GIVEN_RADIUS, 150, 150);

/* Cria uma elipse encapsulada dentro
de um retangulo, cuja extremidade
superior esquerda esta colocada em
(50, 50) e que possui altura 100 e
comprimento 20 */

Oval *oval3 = new Oval (egoview, 50,
50, GIVEN_RECTANGLE, 100, 20);
```

Observações
O atributo Height é sempre igual ao dobro de YRadius;
O atributo Width é sempre igual ao dobro de XRadius;

4.6 OvalArc

Descrição
Este objeto é um arco de elipse;

Atributos

- Angle,
- InitialAngle,
- XRadius,
- YRadius,
- Height,
- Width;

Ponto de referência
Centro da elipse;

Highlight

Se Angle = 0 (Elipse completa)
Elipse maior envolvendo o objeto
senão
Arco maior envolvendo o objeto e dois segmentos, ligando os pontos extremos do arco ao centro da elipse;

Parâmetros de criação

1. Coordenadas X e Y do centro da elipse,
   XRadius,
   YRadius,
   Angle,
   InitialAngle;
2. Coordenadas X e Y do centro da elipse,
   GIVEN_RADIUS,
   XRadius,
   YRadius,
   Angle,
   InitialAngle;
3. Coordenadas X e Y do centro da elipse,
   GIVEN_RECTANGLE,
   Width,
   Height,
   Angle,
   InitialAngle;

Funções definidas

- Distance,
- GetAngle,
- GetInitialAngle,
- GetXRadius,
- GetYRadius,
- GetHeight,
- GetWidth,
- SetAngle,
- SetInitialAngle,
- SetXRadius,
- SetYRadius,
- SetHeight,
- SetWidth;

Exemplo

```
/* Cria um arco de elipse centrado no
ponto (100, 100), com raios 200 e 50,
angulo de 90 graus, comecando a 180
graus */

OvalArc *ovalarc1 = new OvalArc
(egoview, 100, 100, 200, 50, 90,
180);

/* Cria uma elipse de raios 100 e 70,
utilizando OvalArc */

OvalArc *ovalarc2 = new OvalArc
(egoview, 100, 100, GIVEN_RADIUS,
100, 70, 0, 0);

/* Cria um arco de circunferencia,
dentro de um quadrado de tamanho 75,
posicionado em (35, 45); */

OvalArc *ovalarc3 = new OvalArc
(egoview, 35, 45, GIVEN_RECTANGLE,
75, 75, 90, 180);
```

Observações

O atributo Height é sempre igual ao dobro de YRadius;
O atributo Width é sempre igual ao dobro de XRadius;

Figura 14: Alguns objetos do tipo point

4.7 Point

Descrição
Este objeto, em tamanho normal, é formado por um quadrado de 5 pixels, com um ponto no centro;

Atributos
Não possui nenhum atributo próprio;

Ponto de referência
Canto superior esquerdo do quadrado;

Highlight
Quadrado maior, envolvendo o objeto;

Parâmetros de criação

- Coordenadas X e Y do ponto de referência;

Funções definidas

- Distance;

Exemplo

```
/* Cria um ponto na coordenada (100,
100) */
Point *point = new Point (100, 100);
```

Observações
O fator de zoom altera o tamanho do quadrado; no entanto, o ponto no centro permanece sempre do mesmo tamanho;

4.8 Rectangle

Descrição
Este objeto é um retângulo;

Atributos

- Height,
- Width;

Ponto de referência

Se (Height >= 0)

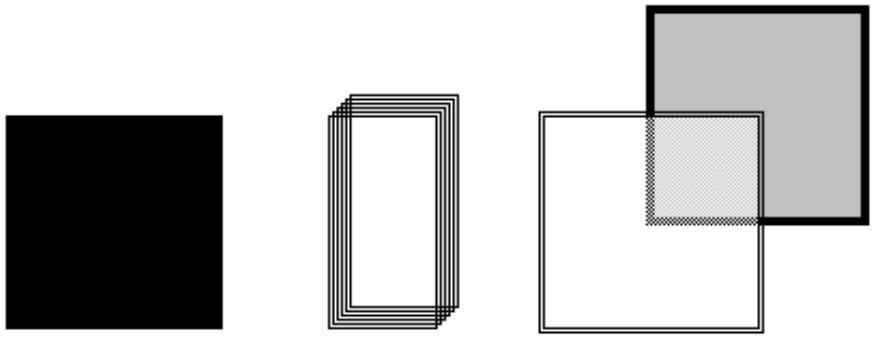

Figura 15: Exemplo de retângulo

Canto superior
senão
Canto inferior
Se (Width >= 0)
Canto esquerdo
senão
Canto direito;

Highlight
retângulo, envolvendo o objeto, com dimensões um pouco maiores;

Parâmetros de criação

- Coordenadas X e Y do ponto de referência,
- Height,
- Width;

Funções definidas

- Distance,
- GetHeight,
- GetWidth,
- SetHeight,
- SetWidth;

Exemplo

```
/* Cria um retangulo na coordenada
(100, 100) de altura 50 e comprimento
70 */

Rectangle *rectangle = new Rectangle
(egoview, 100, 100, 50, 70);
```

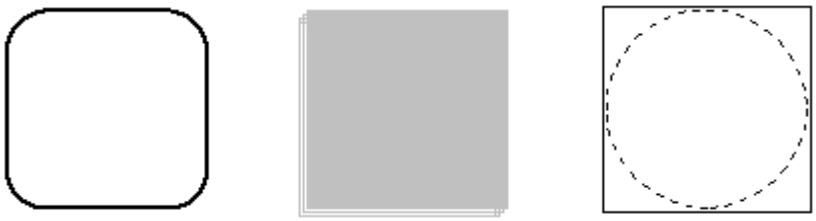

Figura 16: Alguns objetos do tipo RoundRectangle

4.9 RoundRectangle

Descrição
Este objeto é um retângulo, com as bordas ovaladas. O atributo Radius é utilizado para saber quão ovaladas serão as bordas;

Atributos

- Height,
- Width;
- Radius,

Ponto de referência
Segue a mesma regra do retângulo (Ver Rectangle - Ponto de referência)

Highlight
Retângulo, envolvendo o objeto;

Parâmetros de criação

- Coordenadas X e Y do ponto de referência,
- Width,
- Height,
- Radius;

Funções definidas

- Distance,
- GetHeight,
- GetWidth,
- GetRadius,
- SetHeight,
- SetWidth,
- SetRadius;

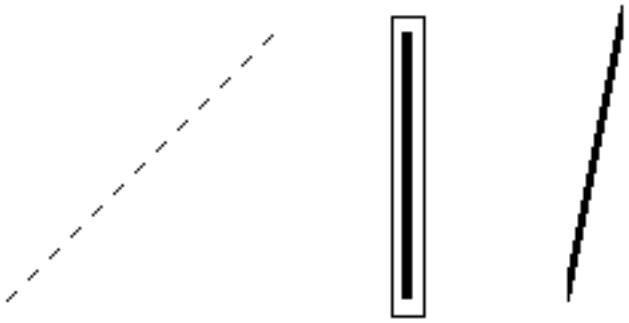

Figura 17: Alguns segmentos

Exemplo

```
/* Cria um retangulo de tamanho 100,
50, com as bordas ovaladas, com raio
de curvatura de 30, posicionado em
(120, 120); */

RoundRectangle *roundrectangle1 = new
RoundRectangle (120, 120, 100, 50,
30);

/* Cria um retangulo, sem as bordas
ovaladas a partir de um objeto
RoundRectangle */

RoundRectangle *roundrectangle2 = new
RoundRectangle (120, 120, 100, 50,
0);

/* Cria um circulo, usando
RoundRectangle */

RoundRectangle *roundrectangle3 = new
RoundRectangle (120, 120, 100, 100,
50);
```

Observações

Se Radius = 0,
o objeto é um retângulo;
Se Height = Width e
Radius = Height/2,
o objeto é um círculo;

4.10 Segment

Descrição

Este objeto é um segmento, ligando dois pontos da tela;

Atributos

- Height,
- Width;

Ponto de referência

Um dos cantos do menor retângulo que englobe o segmento; a determinação deste canto segue a mesma regra que no retângulo (ver Retângulo - Ponto de Referência)

Highlight
Outro segmento, desenhado um pouco abaixo e à esquerda do segmento;

Parâmetros de criação

- Coordenadas X e Y do ponto de referência,
- Height,
- Width;

Funções definidas

- Distance,
- GetHeight,
- GetWidth,
- SetHeight,
- SetWidth;

Exemplo

```
/* Cria um segmento na coordenada
(50, 50) de altura 100 e comprimento
200 */

Segment *segment = new Segment
(egoview, 50, 50, 100, 200);
```

Observações
Este objeto não é afetado pelo atributo FillColor.

4.11 Sound

Descrição
Este é um falso objeto gráfico. A ele está associado um arquivo, o qual é dado um play, quando este objeto é "desenhado".

Atributos

- Sound,
- Synchronized,
- Volume;

Ponto de referência
Não está definido para este objeto;

Highlight
Não está definido para este objeto;

Parâmetros de criação

- Sound,
- Volume,

- Synchronized;

Funções definidas

- GetSound,
- GetVolume,
- GetSynchronized,
- SetSound,
- SetVolume;
- SetSynchronized,

Exemplo

```
/* Cria o objeto Sound, associado ao
arquivo music.au, com volume 40 e sem
ser sincronizado */

Sound *sound = new Sound (egoview,
"music.au", 40, False);
```

Observações

Não é feita nenhuma verificação se o arquivo de som existe.

4.12 String

Descrição

Este objeto é uma cadeia de caracteres;

Atributos

- FontFamily,
- FontSize,
- FontStyle,
- String;

Ponto de referência

Coordenada a partir da qual serão desenhadas as letras;

Highlight

Retângulo, envolvendo a cadeia;

Parâmetros de criação

- Coordenadas X e Y do ponto de referência,
- String,
- FontFamily,
- FontStyle,
- FontSize;

Funções definidas

- GetFamily,
- GetFontSize,
- GetString,
- GetStyle,
- SetFamily,
- SetFontSize,
- SetString,
- SetStyle;

Exemplo

```
/* Cria uma cadeia de caracteres, na
posicao (100,100), com a seguinte
mensagem Ëgoview", utilizando a
familia de fontes e o estilo
defaults, com tamanho 50 */

String *string = new String (egoview,
100, 100, FONT_FAMILY_DEFAULT,
FONT_STYLE_DEFAULT, 50);
```

Observações

Não é feito nenhum teste se existe a configuração de fonte, estilo e tamanho especificados.

5 Descrição Detalhada das Funções

5.1 Distance

Name

Distance - calcula a distância de um ponto ao objeto;

Synopsis

virtual double Distance (double x, double y);

Description

Esta função está definida para todas as classes. Caso o ponto esteja dentro do objeto, a sua distância é igual a zero. No caso de Arc e OvalArc, a distância é calculada, considerando-se a parte da elipse que não faz parte do objeto. No caso de Oval e OvalArc, a distância do ponto à elipse é calculada através de métodos numéricos.

Example

```
/* Calcula a distancia do objeto
roundrectangle2 ao ponto (0,0); */

int distance =
roundrectangle2->Distance (0,0);
```

5.2 GetAngle

Name

GetAngle - obtém o ângulo do objeto;

Synopsis

virtual int GetAngle (void);

Description

Esta função é definida para as classes Arc e OvalArc. Através dela, é possível obter o valor do ângulo do objeto;

Example

```
/* Obtem o ^angulo do objeto oval1 */

int angle = oval1->GetAngle ();
```

Errors

Se o objeto não for Arc
ou OvalArc
success = false;

Related Commands

GetInitialAngle, SetAngle.

5.3 GetArrow

Name

GetArrow - obtém a presença da ponta da seta em um dos extremos;

Synopsis
virtual Boolean GetArrow
(ArrowPosition position);

Arguments

position - define qual dos extremos para o qual está se obtendo a presença;

Description
Esta função só está definida para a classe Arrow.

Structures

```
enum ArrowPosition
{
    INITIAL_ARROW,
    FINAL_ARROW
  };
```

Example

```
/* Obtem a presenca de uma seta no
inicio do segmento definido por
arrow1 (i.e., na coordenada (100,
100)); */

Boolean presence = arrow1->GetArrow
(INITIAL_ARROW);
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrowHeight, GetArrowRewind, GetArrowWidth, SetArrow.

5.4 GetArrowHeight

Name
GetArrowHeight - obtém a altura da ponta da seta;

Synopsis
virtual double GetArrowHeight
(void);

Description
Esta função só está definida para a classe Arrow.

Example

```
/* Obtem a altura da ponta da seta do
objeto arrow2 */

double arrowheight =
arrow2->GetArrowHeight ();
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrow, GetArrowRewind, GetArrowWidth, SetArrowHeight.

5.5 GetArrowRewind

Name
GetArrowRewind - obtém o recuo da ponta da seta;

Synopsis
virtual double GetArrowRewind (void);

Description
Esta função só está definida para a classe Arrow.

Example
```
/* Obtem o recuo da ponta da seta do
triangulo arrow3 */

double arrowrewind =
arrow3->GetArrowRewind ();
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrow, GetArrowHeight, GetArrowWidth, SetArrowRewind.

5.6 GetArrowWidth

Name
GetArrowWidth - obtém o tamanho da ponta da seta;

Synopsis
virtual double GetArrowWidth (void);

Description
Esta função só está definida para a classe Arrow.

Example
```
/* Obtem o tamanho da ponta da seta
do objeto arrow4 */

double arrowwidth =
arrow4->GetArrowWidth ();
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrow, GetArrowHeight, GetArrowRewind, SetArrowWidth.

5.7 GetAutoFlush

Name
GetAutoFlush - obtém valor do indicador de flush automático;

Synopsis
static Boolean GetAutoFlush (void);

Description
Esta função está definida para todas as classes. Ela obtém o valor da variável que indica se as atualizações são refletidas automaticamente, sem a necessidade de invocar um comando explicitamente;

Example
```
/* Obtem o valor do indicador de
flush automatico */

Boolean autoflush =
graphobj::GetAutoFlush ();
```

Related Commands
SetAutoFlush.

5.8 GetBorderColor

Name
GetBorderColor - obtém cor da borda;

Synopsis
int GetBorderColor (void);

Description
Esta função está definida para todos os objetos, embora não faça sentido para a classe Sound.

Example
```
/* Obtem a cor da borda do objeto */

int bordercolor =
point->GetBorderColor ();
```

Related Commands
GetFillColor, SetBorderColor.

5.9 GetFamily

Name
GetFamily - obtém a família de fontes a ser utilizada nos caracteres;

Synopsis
virtual void GetFamily (char *Family);

Arguments

family - família de fontes. Neste argumento, será colocado o nome do fonte, após a chamada desta função;

Description
Esta função é definida apenas para a classe String.

Example
```
/* Obtem o nome da familia utilizada
no string acima */

char family [100];

string->GetFamily (family);
```

Errors

Se objeto não for um String, success = false.

Related Commands
GetFontSize, GetString, GetStyle, SetFamily.

5.10 GetFillColor

Name
GetFillColor - obtém cor do fundo;

Synopsis
virtual int GetFillColor (void);

Description
Esta função está definida para todas as classes, embora não seja utilizada nas classes Dot, Segment e Sound;

Example
```
/* Obtem a cor de preenchimento do
objeto circle1 */

int fillcolor = circle1->GetFillColor
();
```

Related Commands
GetBorderColor, SetFillColor.

5.11 GetFontSize

Name
GetFontSize - obtém o tamanho do fonte a ser utilizado nos caracteres;

Synopsis
virtual int GetFontSize (void);

Description
Esta função é definida apenas para a classe String.

Example
```
/* Obtem o tamanho do font utilizado
em string */

int fontsize = string->GetFontSize
();
```

Errors

Se objeto não for um String, success = false.

Related Commands
GetFamily, GetString, GetStyle, SetFontSize.

5.12 GetGroupMode

Name
GetGroupMode - obtém o modo como os atributos serão propagados para o grupo;

Synopsis
virtual GroupMode GetGroupMode (void);

Description
Esta função é definida apenas para a classe Group e as classes que herdam esta classe (BinaryTree etc.).

- Se mode = ONLY_THIS, as chamadas das funções serão aplicadas apenas ao objeto Group, e não serão propagadas para os objetos do grupo.
- Se mode = EVERYONE, as chamadas das funções serão propagadas para todos os objetos pertencentes ao grupo.
- Se mode = ONLY_THAT_MATCHES, as chamadas das funções serão propagadas apenas aos objetos cujo atributo que está sendo alterado é igual ao atributo do grupo; no caso das outras funções, ela é propagada a todos os objetos.

Structures

```
enum GroupMode
{
    ONLY_THIS,
    EVERYONE,
    ONLY_THAT_MATCHES
  };
```

Errors

Se objeto não é um Group,
success = false;

Related Commands
SetGroupMode.

5.13 GetHeight

Name
GetHeight - obtém a altura de um objeto;

Synopsis
virtual double GetHeight (void);

Description
Esta função está definida para as classes Arrow, Oval, OvalArc, Rectangle, RoundRectangle, Segment. No caso de Oval e OvalArc, este atributo é sempre o dobro do atributo YRadius;

Example

```
/* Obtem a altura de um retangulo */

double height = rectangle->GetHeight
();
```

Errors

Se objeto for uma das classes definidas acima, success = false.

Related Commands
GetWidth, SetHeight.

5.14 GetHighlight

Name
GetHighlight - obtém o highlight do objeto;

Synopsis
Boolean GetHighlight (void);

Description
Esta função está definida para todas as classes, embora ela não seja utilizada na classe Sound.

Example
```
/* Obtem o indicativo se existe um
destaque em um pixel ou nao */

Boolean highlight = dot->GetHighlight
();
```

Related Commands
SetHighlight.

5.15 GetInitialAngle

Name
GetInitialAngle - Obtém o ângulo inicial dos objetos;

Synopsis
virtual int GetInitialAngle (void);

Description
Esta função é definida para as classes Arc e OvalArc. Ela obtém o ângulo a partir do qual o arco do objeto será desenhado;

Example
```
/* Obtem o angulo inicial de um arco
*/

int initialangle =
arc1->GetInitialAngle ();
```

Errors

Se objeto não for Arc ou OvalArc, Success = false

Related Commands
GetAngle, SetInitialAngle.

5.16 GetLayer

Name
GetLayer - obtém a camada em que o objeto é desenhado;

Synopsis
unsigned int GetLayer (void);

Description
Esta função está definida para todos os objetos. Cada objeto pertence a uma camada diferente. Ao se executar este comando, a lista dos objetos é percorrida, a partir do objeto na camada mais abaixo, até se encontrar o objeto atual, retornando a sua posição nesta lista.

Example
```
/* Obtem a camada de um segmento */
int layer = segment->GetLayer ();
```

Related Commands
MoveLayer, SetLayer.

5.17 GetLeftSegment

Name
GetLeftSegment - obtém o segmento associado ao filho esquerdo do nó atual;

Synopsis
virtual Segment
*GetLeftSegment (void);

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um
BinaryTree,
success = false;

Related Commands
GetLeftTree, GetObject, SetRightSegment, GetRightTree, SetLeftSegment.

5.18 GetLeftTree

Name
GetLeftTree - obtém o apontador do objeto que é o filho esquerdo do nó atual;

Synopsis
virtual BinaryTree
*GetLeftTree (void);

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um
BinaryTree,
success = false;

Related Commands
GetLeftSegment, GetObject,
GetRightSegment,
GetRightTree, SetLeftTree.

5.19 GetLineStyle

Name
GetLineStyle - obtém o estilo da linha;

Synopsis
int GetLineStyle (void);

Description
Esta função está definida para todas as classes, embora não seja utilizada em Dot e Sound;

Example
```
/* Obtem o estilo da linha utilizado
em um retangulo com as bordas

ovaladas */

int linestyle =
roundrectangle1->GetLineStyle ();
```

Related Commands
GetLineWidth, SetLineStyle.

5.20 GetLineWidth

Name
GetLineWidth - obtém o tamanho da linha;

Synopsis
unsigned int GetLineWidth
(void);

Description
Esta função está definida para todas as classes, embora não seja utilizada em Dot e Sound;

Example
```
/* obtem a espessura da linha
utilizada em um segmento */

unsigned int linewidth =
segment->GetLineWidth ();
```

Related Commands
GetLineStyle, SetLineWidth.

5.21 GetObject

Name
GetObject - obtém o apontador do objeto associado ao nó da árvore;

Synopsis
virtual GraphObj *GetObject
(void);

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um BinaryTree,
success = false;

Related Commands
GetLeftSegment, GetLeftTree, GetRightSegment, GetRightTree, SetObject.

5.22 GetRadius

Name
GetRadius - obtém o raio do círculo;

Synopsis
virtual double GetRadius (void);

Description
Esta função só está definida para as classes Circle e Arc. O valor deste atributo é sempre igual à metade do valor do atributo Size.

Example

```
/* Obtem o raio de um circulo */
double radius = circle2->GetRadius
();
```

Errors

Se objeto não for um Circle ou Arc,
success = false.

Related Commands
GetSize, GetXRadius, GetYRadius, SetRadius.

5.23 GetRightSegment

Name
GetRightSegment - obtém o segmento associado ao filho direito do nó atual;

Synopsis
virtual Segment *GetRightSegment (void);

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um BinaryTree,
success = false;

Related Commands
GetRightSegment, GetLeftTree, GetObject, GetRightTree, SetRightSegment.

5.24 GetRightTree

Name
GetRightTree - obtém o apontador do objeto que é o filho direito do nó atual;

Synopsis
virtual BinaryTree *GetRightTree (void);

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um BinaryTree, success = false;

Related Commands
GetLeftSegment, GetLeftTree, GetObject, GetRightSegment, SetRightTree.

5.25 GetSelected

Name
GetSelected - obtém um valor indicando se o objeto está selecionado;

Synopsis
Boolean GetSelected (void);

Description
Esta função está definida para todas as classes. Este atributo não é visível graficamente.

Example
```
/* Verifica se um objeto esta
selecionado */

Boolean selected = sound->GetSelected
();
```

Related Commands
SetSelected.

5.26 GetShadowX

Name
GetShadowX - obtém o tamanho da sombra no eixo x;

Synopsis
int GetShadowX (void);

Description
Esta função está definida para todas as classes. Se shadowx = 0, não existe sombra na direção horizontal. Se shadowx < 0, a sombra fica acima do objeto; senão, a sombra fica abaixo do objeto.

Example
```
/* Obtem a sombra no eixo x de um
pixel */

int shadowx = dot->GetShadowX ();
```

Related Commands
GetShadowY, SetShadowX.

5.27 GetShadowY

Name
GetShadowY - obtém o tamanho da sombra no eixo y;

Synopsis
int GetShadowY (void);

Description
Esta função está definida para todas as classes.
Se shadowy = 0, não existe sombra na direção vertical.
Se shadowy < 0, a sombra fica acima do objeto; senão, a sombra fica abaixo do objeto.

Example
```
/* Obtem o tamanho da sombra no eixo
y de um ponto */

int shadowy = point->GetShadowY ();
```

Related Commands
GetShadowX, SetShadowY.

5.28 GetSize

Name
GetSize - obtém o tamanho do objeto;

Synopsis
virtual double GetSize (void);

Description
Esta função só está defina para as classes Circle e Arc. O valor deste atributo é sempre igual ao dobro do atributo Radius.

Example
```
/* Obtem o tamanho de um arco */

double size = arc3->GetSize ();
```

Errors

Se objeto não for um Circle
ou Arc,
success = false;

Related Commands
GetRadius, SetSize.

5.29 GetSound

Name
GetSound - obtém o nome do arquivo de som;

Synopsis
virtual void GetSound (char *sound);

Arguments

sound - nome do arquivo de som;

Description
Esta função só está definida para a classe Sound.

Example
```
/* Obtem o arquivo de som associado
ao objeto sound */
char soundname[100];
sound->GetSound (soundname);
```

Errors

Se objeto não for Sound,
success = false.

Não é feita nenhuma verificação se o arquivo de som realmente existe.

Related Commands
GetSynchronized, GetVolume, SetSound.

5.30 GetString

Name
GetString - obtém a cadeia de caracteres a ser exibida;

Synopsis
virtual void GetString (char *string);

Arguments

string - cadeia de caracteres; neste argumento será colocado a cadeia de caracteres a ser exibida;

Description
Esta função é definida apenas para a classe String.

Example
```
/* Obtem a cadeia de caracteres
associada ao objeto string */
char stringname [100];
string->GetString (stringname);
```

Errors

Se objeto não for um String,
success = false.

Related Commands
GetFamily, GetFontSize, GetStyle, SetString.

5.31 GetStyle

Name

GetStyle - obtém o estilo do fonte a ser utilizado na cadeia de caracteres a ser exibida;

Synopsis

virtual void GetStyle (char *style);

Description

Esta função é definida apenas para a classe String. Os estilos aceitos dependem da configuração dos arquivos de fonte existentes. Não são todas as famílias e tamanhos de letra que existem para um dado estilo.

Example

```
/* Obtem o estilo do fonte do objeto
string */

char stringstyle [100];

string->GetStyle (stringstyle);
```

Errors

Se objeto não for um String, success = false.

Related Commands

GetFamily, GetFontSize, GetString, SetStyle.

5.32 GetSynchronized

Name

GetSynchronized - obtém um valor indicando se deve haver sincronismo quando se toca um som ou não;

Synopsis

virtual Boolean GetSynchronized (void);

Description

Esta função só está definida para a classe Sound.

Example

```
/* Obtem o indicativo de
sincronizacao do objeto sound */

Boolean sync = sound->GetSynchronized
();
```

Errors

Se objeto não for Sound, success = false.

Related Commands

GetSound, GetVolume, SetSynchronized.

5.33 GetTransparency

Name
GetTransparency - obtém a transparência de um objeto;

Synopsis
Transparency
GetTransparency (void);

Description
Esta função está definida para todos os objetos.

- Se transparency = OPAQUE, todos os objetos que estão abaixo deste objeto não são visíveis.
- Se transparency = TRANSPARENCY, então apenas o borda do objeto é desenhada.
- Se transparency = TRANSLUCENT, embora o objeto seja totalmente mostrado, é possível ver partes dos objetos que estão em camadas inferiores. Para as classes Dot, Segment e Sound, este atributo não tem utilidade.

Structures

```
enum Transparency
{
     TRANSPARENT,
     OPAQUE,
     TRANSLUCENT};
```

Example

```
/* Obtem a transparencia de uma
elipse oval2 */

Transparency transparency =
oval->GetTransparency ();
```

Related Commands
SetTransparency.

5.34 GetVisible

Name
GetVisible - obtém a visibilidade do objeto;

Synopsis
Boolean GetVisible (void);

Description
Esta função está definida para todos os objetos. No caso de Sound, sendo o objeto não vísivel, o seu som não é tocado.

Example

```
/* Obtem a visibilidade do objeto
ovalarc1 */

Boolean visible =
ovalarc1->GetVisible ();
```

Related Commands
SetVisible.

5.35 GetVolume

Name
GetVolume - obtém o volume em que será tocado o som;

Synopsis
virtual int GetVolume (void);

Description
Esta função só está definida para a classe Sound.

Example
```
/* Obtem o volume de sound */
int volume = sound->GetVolume ();
```

Errors

Se objeto não for Sound, success = false.

Related Commands
GetSound, GetSynchronized, SetVolume.

5.36 GetWidth

Name
GetWidth - obtém o comprimento de um objeto;

Synopsis
virtual double GetWidth (void);

Description
Esta função está definida para as classes Arrow, Oval, OvalArc, Rectangle, RoundRectangle, Segment. No caso de Oval e OvalArc, o valor deste atributo é igual ao dobro do atributo XRadius.

Example
```
/* Obtem o comprimento do objeto
ovalarc2 */

double width = ovalarc2->GetWidth ();
```

Errors

Se objeto não for uma das classes definidas acima, success = false.

Related Commands
GetHeight, SetWidth.

5.37 GetX

Name
GetX - obtém a coordenada X do ponto de referência do objeto;

Synopsis
double GetX (void);

Description
Esta função está definida para todas as classes, embora ele não seja utilizado na classe Sound. Cada objeto define o ponto de referência de uma forma. Para maiores detalhes, veja a descrição dos objetos.

Example
```
/* Obtem a coordenada X do ponto de
referencia de um retangulo */

double x = rectangle->GetX ();
```

Related Commands
Move, SetX.

5.38 GetXRadius

Name
GetXRadius - obtém o tamanho do raio no eixo x da elipse;

Synopsis
virtual double GetXRadius (void);

Description
Esta função só está definida para as classes Oval e OvalArc. O valor deste atributo é sempre a metade do atributo Width.

Example
```
/* Obtem o raio no eixo x para o
objeto oval3 */

double xradius = oval3->GetXRadius
();
```

Errors

Se objeto não for um Oval
ou OvalArc
success = false.

Related Commands
GetRadius, GetWidth, GetYRadius, SetXRadius.

5.39 GetY

Name
GetY - obtém a coordenada Y do ponto de referência do objeto;

Synopsis
double GetY (void);

Description
Esta função está definida para todas as classes, embora ela não seja utilizado na classe Sound. Cada objeto define o ponto de referência de uma forma. Para maiores detalhes, veja descrição dos objetos.

Example
```
/* Obtem a coordenada Y de um
retangulo comas bordas ovaladas */

double y = roundrectangle2->GetY ();
```

Related Commands
Move, SetY.

5.40 GetYRadius

Name
GetYRadius - obtém o tamanho do raio no eixo y da elipse;

Synopsis
virtual double GetYRadius (void);

Description
Esta função só está definida para as classes Oval e OvalArc. O valor deste atributo é sempre igual à metade do atributo Height.

Example
```
/* Obtem o tamanho do raio no eixo y
para o objeto ovalarc3 */

double yradius = ovalarc3->GetYRadius
();
```

Errors

Se objeto não for um Oval
ou OvalArc
success = false.

Related Commands
SetYRadius, GetRadius, GetWidth, GetXRadius.

5.41 GetZoom

Name
GetZoom - obtém o fator de zoom de um objeto;

Synopsis
float GetZoom (void);

Description
Esta função está definida para todas as classes, embora não seja utilizada pelo objeto Dot. Sendo este fator igual a 1, o objeto passa a ter o seu tamanho normal. Sendo $|zoom| < 1$, o objeto fica menor que o seu tamanho

normal, e |zoom| > 1, o objeto fica maior.

Example

```
/* Obtem o valor do fator de zoom
para o circulo circle3 */

float zoom = circle3->GetZoom ();
```

Related Commands

SetZoom, Zoom.

5.42 Move

Name

Move - move o ponto de referência de um objeto, relativamente à posição atual;

Synopsis

virtual void Move (double movex, double movey);
virtual void Move (double movex, double movey, double nstep, unsigned twait);

Arguments

movex - deslocamento no eixo x do objeto;

movey - deslocamento no eixo y do objeto;

nstep - número de passos que deve ser dado no deslocamento do objeto;

twait - tempo, em centésimos de segundos, que deve ser dado em cada passo;

Description

Estas funções servem para a movimentação do objeto. No caso da primeira sinopse, o objeto é retirado da posição antiga e colocado na nova posição. No caso da segunda sinopse, este deslocamento é feito através do número de passos indicados no argumento nstep, sendo que em cada passo é esperado um tempo twait. A função SetX e SetY podem ser implementadas através da função Move;

Example

```
/* Move o objeto segment 10 posicoes
nos eixos x e y */

segment->Move (10, 10);

/* Move o objeto arc3 -100 posicoes
no eixo x, em 10 passos, com */

/* tempo de espera de 1 segundo em
cada passo */

arc3->Move (-100, 0, 10, 100);
```

Related Commands

GetX, GetY, SetX, SetY.

5.43 MoveLayer

Name

MoveLayer - desloca o objeto algumas camadas acima ou abaixo da camada atual;

Synopsis

virtual void MoveLayer (int layer);

Arguments

layer - deslocamento da camada em relação à camada atual;

Description

Esta função serve para deslocar o objeto camadas acima ou abaixo.

- Se layer < 0, o objeto é deslocado algumas camadas abaixo;
- Se layer > 0, o objeto é deslocado algumas camadas acima;

A função SetLayer podem ser implementadas através da função Movelayer.

Example

```
/* Move o objeto circle 5 camada
abaixo */

circle->MoveLayer (5);
```

Related Commands

GetLayer, SetLayer.

5.44 Set

Name

Set - seta diversos atributos do objeto.

Synopsis

virtual void Set (va_tdcl (SetOption));

Arguments

O número de argumentos desta função é variável. Os argumentos deve ser uma lista de SetOption, seguida dos parâmetros necessários. O primeiro argumento deve ser do tipo SetOption. A lista de argumentos deve terminar com 0.

Description

Esta função define os atributos de todos os tipos de objeto. A vantagem de se chamar esta função, ao invés das funções específicas (Set*) é que,

quando o objeto for visível, para cada chamada específica, ele é apagado e desenhado novamente, com o novo atributo. Ao se chamar esta função, o objeto é apagado e desenhado apenas uma vez. Para as funções que são virtuais, ele faz uma chamada a estas funções.

Structures

```
enum SetOption
{
    END_OF_SET_OPTION,
    ANGLE,
    ARROW_PRESENCE,
    ARROW_HEIGHT,
    ARROW_REWIND,
    ARROW_WIDTH,
    BORDERCOLOR,
    FAMILY,
    FILLCOLOR,
    HEIGHT,
    HIGHLIGHT,
    INITIAL_ANGLE,
    LAYER,
    LINE_STYLE,
    LINE_WIDTH,
    RADIUS,
    SELECTED,
    SHADOW_X,
    SHADOW_Y,
    SIZE,
    SOUND,
    STRING,
    STRING_FONT_SIZE,
    STYLE,
    SYNCHRONIZED,
    TRANSPARENCY,
    VISIBLE,
    VOLUME,
    WIDTH,
    X,
    X_RADIUS,
    Y,
    Y_RADIUS,
    ZOOM,
  };
```

Example

```
/* Define diversos parametros para o
objeto dot */

dot->Set (BORDERCOLOR, GREEN,

HIGHLIGHT, True,

LAYER, 1,

SELECTED, False,

SHADOW_X, 5

SHADOW_Y, 5,

VISIBLE, True,

X, 100,

Y, 100,

END_OF_OPTION)
```

Errors

Se algum dos atributos testados não for definido para a classe, Success = false.

Related Commands
SetAngle, SetArrow, SetArrowHeight, SetArrowRewind, SetArrowWidth, SetAutoFlush, SetBorderColor, SetFillColor, SetFamily, SetFontSize, SetGroupMode, SetHeight, SetHighlight, SetInitialAngle, SetLayer, SetLeftSegment, SetLeftTree, SetLineStyle, SetLineWidth, SetObject, SetRadius, SetRightSegment, SetRightTree, SetSelected, SetShadowX, SetShadowY, SetSize, SetSynchronized, SetTransparency, SetVisible, SetVolume, SetWidth, SetX, SetXRadius, SetY, SetYRadius, SetZoom.

5.45 SetAngle

Name
SetAngle - Define o ângulo dos objetos;

Synopsis
virtual void SetAngle (int angle);

Arguments

angle - ângulo do objeto, em graus;

Description
Esta função é definida para as classes Arc e OvalArc. Ela define o ângulo total do objeto;

Errors

Se objeto não for Arc ou OvalArc, success = false

Example
```
/* Define o angulo do objeto arc1 em
180 graus */

arc1->SetAngle (180);
```

Related Commands
GetAngle, Set, SetInitialAngle

5.46 SetArrow

Name
SetArrow - define a presença da ponta da seta em um dos extremos;

Synopsis
virtual void SetArrow
(ArrowPosition position,
Boolean presence);

Arguments

position - define qual dos extremos para o qual está se estabelecendo a presença da seta;

presence - define a presença da seta;

Description
Esta função só está definida para a classe Arrow;

Structures

```
enum ArrowPosition
{
    INITIAL_ARROW,
    FINAL_ARROW
  };
```

Example

```
/* Define uma seta no inicio do
objeto arrow1 */
arrow1->SetArrow (INITIAL_ARROW,
True);
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrow, Set,
SetArrowHeight,
SetArrowRewind,
SetArrowWidth.

5.47 SetArrowHeight

Name
SetArrowHeight - define a altura da ponta da seta;

Synopsis
virtual void SetArrowHeight
(double height);

Arguments

height - altura da seta.
Define o tamanho da ponta da seta perpendicular ao segmento.

- Se |height| <= 1, este valor é proporcional ao comprimento da seta, senão este valor é absoluto.
- Se height = 0, a ponta da seta é imperceptível.

Description
Esta função só está definida para a classe Arrow.

Example
```
/* Define a altura da ponta da seta
de arrow2 como sendo 20 pixels */

arrow2->SetArrowHeight (20);
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrowHeight, Set, SetArrow, SetArrowRewind, SetArrowWidth.

5.48 SetArrowRewind

Name
SetArrowRewind - define o recuo da ponta da seta;

Synopsis
virtual void SetArrowRewind (double rewind);

Arguments

rewind - recuo da seta.

- Se recuo = 0, a ponta da seta é um triângulo.
- Se |recuo| > 0, define de quanto as extremidades da ponta da seta vai ser recuada, paralelamente à orientação da seta.
- Se |recuo| <= 1, este valor é proporcional ao comprimento da seta, senão este valor é absoluto.

Description
Esta função só está definida para a classe Arrow.

Example
```
/* Define o recuo da seta arrow3 como
sendo 20% de seu tamanho */

arrow3->SetArrowRewind (0.20);
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrowRewind, Set, SetArrow, SetArrowHeight, SetArrowWidth.

5.49 SetArrowWidth

Name
SetArrowWidth - define o tamanho da ponta da seta;

Synopsis
virtual void SetArrowWidth (double width);

Arguments

width - tamanho da ponta da seta. Se |width| < 1, este tamanho é proporcional ao comprimento da seta, senão este valor é absoluto.

Description
Esta função só está definida para a classe Arrow.

Example

```
/* Define o tamanho da ponta da seta
arrow4 como sendo 50% do */
/* comprimento da mesma */
arrow4->SetArrowWidth (0.50);
```

Errors

Se o objeto não for um Arrow, success = false;

Related Commands
GetArrowWidth, Set, SetArrow, SetArrowHeight, SetArrowRewind.

5.50 SetAutoFlush

Name
SetAutoFlush - define se as alterações são refletidas automaticamente, ou deve se dar um comando explicitamente;

Synopsis
static void SetAutoFlush (Boolean flush);

Arguments

flush - se verdadeiro, será dado um flush automaticamente depois de cada alteração nos objetos. Senão, este flush terá que ser dado pelo programador.

Description
Esta função está definida para todas as classes. Ela manipula um atributo que é comum a todos os objetos. Ao se chamar esta função para um objeto, todos os demais objetos terão este atributo alterado. A vantagem de se ter AutoFLush

ligado é que o programador não precisa se preocupar em fazer chamadas de flush em seu programa. Contudo, pode-se tornar algo indesejável uma chamada de flush após cada alteração.

Example

```
/* Define o autoflush como sendo
False */

graphobj::SetAutoFlush (False);
```

Related Commands

Set, GetAutoFlush

5.51 SetBorderColor

Name

SetBorderColor - define a cor da borda do objeto;

Synopsis

virtual void SetBorderColor (int color);

Arguments

color - cor da borda do objeto. As cores que podem ser assumidas por um objeto devem ser definidas previamente. Este argumento se refere ao índice da cor no vetor de cores definido;

Description

Esta função está definida para todos os objetos, embora não faça sentido para a classe Sound;

Example

```
/* Define a cor da borda de um ponto
como sendo preta */

point->SetBorderColor (BLACK);
```

Related Commands

GetBorderColor, Set, SetFillColor.

5.52 SetFillColor

Name

SetFillColor - define a cor de preenchimento do objeto;

Synopsis

virtual void SetFillColor (int color);

Arguments

color - cor de preenchimento do objeto. As cores que podem ser assumidas por um objeto devem ser definidas previamente. Este argumento se refere ao índice da cor no vetor de cores definido.

Description
Esta função está definida para todas as classes, embora não seja utilizada nas classes Dot, Segment e Sound;

Example
```
/* Define a cor de preenchimento do
objeto rectangle como sendo

amarela */

rectangle->SetFillColor (YELLOW);
```

Related Commands
GetFillColor, Set, SetBorderColor;

5.53 SetFamily

Name
SetFamily - define a família de fontes a ser utilizado nos caracteres.

Synopsis
virtual void SetFamily (char *family);

Arguments

family - família de fontes. Algumas das famílias aceitas são:

- FONT_FAMILY_DEFAULT,
- FONT_DEFAULT_FIXED_WIDTH,
- FONT_LUCIDA,
- FONT_LUCIDA_FIXED_WIDTH,
- FONT_ROMAN,
- FONT_SERIF,
- FONT_COUR.

Description
Esta função é definida apenas para a classe String. As famílias aceitas dependem da configuração dos arquivos de fonte existentes. Não são todos os estilos e tamanhos de letra que existem para uma dada família.

Example
```
/* Define FONT_SERIF como a fonte do
objeto string */

string->SetFamily (FONT_SERIF);
```

Errors

Se objeto não for um String, success = false.

Se o programa não conseguir encontrar a família da fonte, com o tamanho e estilo especificados, ele carrega a fonte default, com tamanho e estilo defaults.

Related Commands
GetFamily, Set, SetFontSize, SetString, SetStyle.

5.54 SetFontSize

Name
SetFontSize - define o tamanho do fonte a ser utilizado nos caracteres;

Synopsis
virtual void SetFontSize (int size);

Arguments

size - tamanho do fonte. A priori, este tamanho pode ser qualquer valor inteiro. Contudo a existência do fonte neste tamanho vai depender da configuração do sistema local.

Description
Esta função é definida apenas para a classe String. Os tamanhos aceitos dependem da configuração dos arquivos de fonte existentes. Não são todos os estilos e famílias de letra que existem para um dado tamanho;

Errors

Se objeto não for um String, success = false.

Se o programa não conseguir encontrar o tamanho da fonte, com a família e estilo especificados, ele carrega a fonte default, com tamanho e estilo defaults.

Example

```
/* Definicao do tamanho do fonte do
objeto string como 20 */

string->SetFontSize (20);
```

Related Commands
GetFontSize, Set, SetFamily, SetString, SetStyle.

5.55 SetGroupMode

Name
SetGroupMode - define o modo como os atributos serão propagados para o grupo;

Synopsis
virtual void SetGroupMode (GroupMode mode);

Arguments

mode - modo como as chamadas das funções serão propagadas para os elementos do grupo.

Description
Esta função é definida apenas para a classe Group e as classes que herdam esta classe (BinaryTree etc.).

- Se mode = ONLY_THIS, as chamadas das funções serão aplicadas apenas ao objeto Group, e não serão propagadas para os objetos do grupo.
- Se mode = EVERYONE, as chamadas da funções serão propagadas para todos os objetos pertencentes ao grupo.
- Se mode = ONLY_THAT_MATCHES, as chamadas das funções de Set serão propagadas apenas aos objetos cujo atributo sendo alterado é igual ao atributo do grupo; no caso das outras funções, ela é propagada a todos os objetos.

Structures

```
enum GroupMode
{
    ONLY_THIS,
    EVERYONE,
    ONLY_THAT_MATCHES
  };
```

Errors

Se objeto não é um Group,
success = false;

Related Commands
GetGroupMode, Set.

5.56 SetHeight

Name
SetHeight - define a altura de um objeto;

Synopsis
virtual void SetHeight (double height);

Arguments

height - altura do objeto.

Description
Esta função está definida para as classes Arrow, Oval, OvalArc, Rectangle, RoundRectangle, Segment. No caso de Oval e OvalArc, a alteração deste atributo provoca uma alteração do atributo YRadius.

Example
```
/* Alteracao da altura do objewto
roundrectangle3 para 45 */

roundrectangle3->SetHeight (45);
```

Errors

Se objeto não for uma das classes definidas acima, success = false.

Related Commands
GetHeight, Set, SetWidth.

5.57 SetHighlight

Name
SetHighlight - define o highlight do objeto;

Synopsis
virtual void SetHighlight (Boolean highlight);

Arguments

highlight - se verdadeiro, o objeto está em highlight; senão, o objeto não está em highlight.

Description
Esta função está definida para todas as classes, embora ela não seja utilizada na classe Sound;

Example
```
/* Colocacao do objeto segment em
highlight */

segment->SetHighlight (True);
```

Related Commands
GetHighlight, Set.

5.58 SetInitialAngle

Name
SetInitialAngle - Define o ângulo inicial dos objetos;

Synopsis
virtual void SetInitialAngle (int angle);

Arguments

angle - ângulo inicial do objeto, em graus;

Description
Esta função é definida para as classes Arc e OvalArc. Ela define o ângulo a partir do qual o arco será desenhado do objeto;

Example
```
/* Definicao do angulo inicial de
ovalarc1 como sendo 0 grau */

ovalarc1->setInitialAngle (0);
```

Errors

Se objeto não for Arc
ou OvalArc,
Success = false

Related Commands
GetInitialAngle, Set, SetAngle

5.59 SetLayer

Name
SetLayer - define a camada em que o objeto é desenhado;

Synopsis
virtual void SetLayer
(unsigned int layer);

Arguments

layer - camada do objeto.

Description
Esta função está definida para todos os objetos. Cada objeto pertence a uma camada diferente. Ao se executar este comando, todos os objetos entre a antiga camada do objeto e a nova são deslocados de um na lista.

Example

```
/* Colocacao do objeto circle1 sobre
todos os objetos */
circle1->SetLayer (0);
```

Related Commands
MoveLayer, GetLayer, Set.

5.60 SetLeftSegment

Name
SetLeftSegment - define o objeto do tipo Segment que ligará o nó atual a seu filho esquerdo;

Synopsis
virtual void SetLeftSegment
(Segment *segment);

Arguments

segment - apontador do segmento associado ao filho esquerdo deste nó;

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um
BinaryTree,
success = false;

Related Commands
GetLeftSegment, Set, SetLeftTree, SetObject, SetRightSegment, SetRightTree.

5.61 SetLeftTree

Name
SetLeftTree - define o objeto do tipo BinaryTree que será o filho esquerdo do nó atual;

Synopsis
virtual void SetLeftTree (BinaryTree *tree);

Arguments

tree - apontador da árvore binária que será utilizada como o filho esquerdo do nó atual.

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um BinaryTree, success = false;

Related Commands
GetLeftTree, Set, SetLeftSegment, SetObject, SetRightSegment, SetRightTree.

5.62 SetLineStyle

Name
SetLineStyle - define o estilo da linha;

Synopsis
virtual void SetLineStyle (int style);

Arguments

style - estilo da linha. O estilo da linha pode ser LineSolid, LineOnOffDash, LineDoubleDash;

Description
Esta função está definida para todas as classes, embora não seja utilizada em Dot e Sound;

Example
```
/* Definicao do estilo de linha do
objeto oval1 como sendo

pontilhada */

oval1->SetLineStyle (LineOnOffDash);
```

Related Commands
GetLineStyle, Set,
SetLineWidth.

5.63 SetLineWidth

Name
SetLineWidth - define o tamanho da linha;

Synopsis
virtual void SetLineWidth (unsigned int width);

Arguments

width - tamanho da linha. O tamanho normal da linha é 1. O tamanho da linha define o número de pixels que serão utilizados quando o objeto for desenhado;

Description
Esta função está definida para todas as classes, embora não seja utilizada em Dot e Sound;

Example

```
/* Definicao da espessura da linha do
objeto rectangle como 5

pixels */

rectangle->setLineWidth (5);
```

Related Commands
GetLineWidth, Set,
SetLineStyle.

5.64 SetObject

Name
SetObject - define o objeto que será tratado como nó atual;

Synopsis
virtual void SetObject (GraphObj *object);

Arguments

object - apontador para objeto gráfico que será utilizado como nó atual.

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um
BinaryTree,
success = false;

Related Commands
GetObject, Set,
SetLeftSegment, SetLeftTree,
SetRightSegment,
SetRightTree.

5.65 SetRadius

Name
SetRadius - define o raio do círculo;

Synopsis
virtual void SetRadius (double radius);

Arguments

radius - raio do círculo;

Description
Esta função só está definida para as classes Circle e Arc. A alteração deste atributo provoca uma alteração do atributo Size.

Example
```
/* Alteracao do raio do arco para 30
pixels */
arc2->SetRadius (30);
```

Errors

Se objeto não for um Circle
ou Arc,
success = false.

Related Commands
GetRadius, Set, SetSize, SetXRadius, SetYRadius.

5.66 SetRightSegment

Name
SetRightSegment - define o objeto que ligará o nó atual ao seu filho direito;

Synopsis
virtual void SetRightSegment (Segment *segment);

Arguments

segment - apontador do segmento associado ao filho direito deste nó;

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um
BinaryTree,
success = false;

Related Commands
GetRightSegment, Set, SetRightSegment, SetLeftTree, SetObject, SetRightTree.

5.67 SetRightTree

Name
SetRightTree - define o objeto do tipo BinaryTree que será tratado como filho direito do nó atual.

Synopsis
virtual void SetRightTree (BinaryTree *tree);

Arguments

tree - apontador da árvore binária que será utilizada como o filho direito do nó atual.

Description
Esta função só está definida para a classe BinaryTree;

Errors

Se objeto não for um BinaryTree, success = false;

Related Commands
GetRightTree, Set, SetLeftSegment, SetLeftTree, SetObject, SetRightSegment.

5.68 SetSelected

Name
SetSelected - define se o objeto está selecionado;

Synopsis
virtual void SetSelected (Boolean selected);

Arguments

selected - se verdadeiro, o objeto está selecionado; senão, o objeto não está selecionado.

Description
Esta função está definida para todas as classes. Esta atributo não é visível graficamente.

Example
```
/* Define o objeto dot como nao
selecionado */

dot->SetSelected (False);
```

Related Commands
GetSelected, Set.

5.69 SetShadowX

Name
SetShadowX - define o tamanho da sombra no eixo x;

Synopsis
virtual void SetShadowX (int shadowx);

Arguments

shadowx - tamanho da sombra no eixo x.

Description
Esta função está definida para todas as classes.

- Se shadowx = 0, não existe sombra na direção horizontal.
- Se shadowx < 0, a sombra fica acima do objeto; senão, a sombra fica abaixo do objeto.

Example
```
/* Definicao da sombra no eixo x do
objeto point como 5 pixels */

point->SetShadowX (5);
```

Related Commands
GetShadowX, Set, SetShadowY.

5.70 SetShadowY

Name
SetShadowY - define o tamanho da sombra no eixo y;

Synopsis
virtual void SetShadowY (int shadowy);

Arguments

shadowy - tamanho da sombra no eixo y.

Description
Esta função está definida para todas as classes.

- Se shadowy = 0, não existe sombra na direção vertical.
- Se shadowy < 0, a sombra fica acima do objeto; senão, a sombra fica abaixo do objeto.

Example
```
/* Definicao da sombra no eixo y do
objeto roundrectangle1 como */

sendo 10

roundrectangle1->SetShadowY (10);
```

Related Commands
GetShadowY, Set, SetShadowX.

5.71 SetSize

Name
SetSize - define o tamanho do objeto;

Synopsis
virtual void SetSize (double size);

Arguments

size - tamanho do objeto.

Description
Esta função só está defina para as classes Circle e Arc. A alteração neste atributo provoca uma alteração no atributo Radius.

Example
```
/* Alteracao no tamanho (diametro) do
circulo circle2 para 100 */

circle2->setSize (100);
```

Errors

Se objeto não for um Circle ou Arc,
success = false;

Related Commands
GetSize, Set, SetRadius.

5.72 SetSound

Name
SetSound - define o arquivo de som;

Synopsis
virtual void SetSound (char *sound);

Arguments

sound - nome do arquivo de som;

Description
Esta função só está definida para a classe Sound.

Example
```
/* Define o arquivo "sound.au" como o
som associado ao objeto

sound */

sound->SetSound("sound.au");
```

Errors

Se objeto não for Sound,
success = false.

Não é feita nenhuma verificação se o arquivo de som existe.

Related Commands
GetSound, Set, SetSynchronized, SetVolume.

5.73 SetString

Name
SetString - define a cadeia de caracteres a ser exibida;

Synopsis
virtual void SetString (char *string);

Arguments

string - cadeia de caracteres;

Description
Esta função é definida apenas para a classe String;

Example
```
/* Associa a cadeia "Novo String" ao
objeto string */

string->SetString ("Novo String");
```

Errors

Se objeto não for um String, success = false.

Related Commands
GetString, Set, SetFamily, SetFontSize, SetStyle.

5.74 SetStyle

Name
SetStyle - define o estilo do fonte a ser utilizado na cadeia de caracteres a ser exibida;

Synopsis
virtual void SetStyle (char *style);

Arguments

style - estilo do fonte. Alguns dos tipos aceitos são:

- FONT_STYLE_DEFAULT,
- FONT_STYLE_PLAIN,
- FONT_STYLE_BOLD,
- FONT_STYLE_ITALIC,
- FONT_STYLE_BOLD_ITALIC

Description
Esta função é definida apenas para a classe String. Os estilos aceitos dependem da configuração dos arquivos de fonte existentes. Não são todas as famílias e tamanhos de letra que existem para um dado estilo;

Example
```
/* Altera o estilo do fonte do objeto
string para

FONT_STYLE_ITALIC */

string->SetStyle (FONT-STYLE_ITALIC);
```

Errors

Se objeto não for um String, success = false.

Related Commands
GetStyle, Set, SetFamily, SetFontSize, SetString.

5.75 SetSynchronized

Name
SetSynchronized - define se deve haver sincronismo quando se toca um som ou não;

Synopsis
virtual void SetSynchronized (Boolean synchronized);

Arguments

synchronized - flag indicando se, ao tocar um som, deve-se esperar acabar o som para retornar o controle (verdadeiro) ou deve-se retornar imediatamente após a chamada do comando, sem esperar tocar o som (falso);

Description
Esta função só está definida para a classe Sound.

Example

```
/* Define o objeto sound como sendo
sincronizado */

sound->setSynchronized (True);
```

Errors

Se objeto não for Sound, success = false.

Related Commands
GetSynchronized, Set, SetSound, SetVolume.

5.76 SetTransparency

Name
SetTransparency - define a transparência de um objeto;

Synopsis
virtual void SetTransparency (Transparency);

Arguments

transparency - transparência do objeto.

Description
Esta função está definida para todos os objetos.

- Se transparency = OPAQUE, todos os objetos que estão abaixo deste objeto não são visíveis.
- Se transparency = TRANSPARENT, então apenas o borda do objeto é desenhada.
- Se transparency = TRANSLUCENT, embora o objeto seja totalmente mostrado, é possível ver partes dos objetos que estão em camadas inferiores.

Para as classes Dot, Segment e Sound, este atributo não tem utilidade.

Structures

```
enum Transparency
{
    TRANSPARENT,
    OPAQUE,
    TRANSLUCENT};
```

Example

```
/* Define o objeto oval2 como sendo
translucido */
oval2->SetTransparency (TRANSLUCENT);
```

Related Commands

GetTransparency, Set.

5.77 SetVisible

Name

SetVisible - define a visibilidade do objeto;

Synopsis

virtual void SetVisible (Boolean visible);

Arguments

visible - visibilidade do objeto; se verdadeiro, o objeto é visível; senão, o objeto não é visível;

Description

Esta função está definida para todos os objetos. No caso de Sound, sendo o objeto não vísivel, o seu som não é tocado.

Example

```
/* Define o objeto segment como nao
visivel */

segment->setVisible (False);
```

Related Commands

GetVisible, Set.

5.78 SetVolume

Name
SetVolume - define o volume em que será tocado o som;

Synopsis
virtual void SetVolume (int volume);

Arguments

volume - volume em que será tocado o som. Se volume é 0, o som é inaudível.

Description
Esta função só está definida para a classe Sound.

Example
```
/* Altera o volume do objeto sound
para 0 (inaudivel) */

sound->SetVolume (0);
```

Errors

Se objeto não for Sound, success = false.

Related Commands
GetVolume, Set, SetSound, SetSynchronized.

5.79 SetWidth

Name
SetWidth - define o comprimento de um objeto;

Synopsis
virtual void SetWidth (double width);

Arguments

width - altura do objeto;

Description
Esta função está definida para as classes Arrow, Oval, OvalArc, Rectangle, RoundRectangle, Segment. No caso de Oval e OvalArc, a alteração deste atributo provoca uma alteração do atributo XRadius.

Example
```
/* Define o comprimento do arco de
elipse ovalarc2 como sendo 145 */

ovalarc2->SetWidth (145);
```

Errors

Se objeto não for uma das classes definidas acima, success = false.

Related Commands
GetWidth, Set, SetHeight.

5.80 SetX

Name
SetX - define a coordenada X do ponto de referência do objeto.

Synopsis
virtual void SetX (double coordx);

Arguments

coordx - coordenada X do ponto de referência do objeto.

Description
Esta função está definida para todas as classes, embora ele não seja utilizado na classe Sound. Cada objeto define o ponto de referência de uma forma. Para maiores detalhes, veja descrição dos objetos.

Example
```
/* Move o ponto point para a
coordenada (100, y_atual) */

point->SetX (100);
```

Related Commands
GetX, Move, Set, SetX.

5.81 SetXRadius

Name
SetXRadius - define o tamanho do raio no eixo x da elipse;

Synopsis
virtual void SetXRadius (double xradius);

Arguments

xradius - tamanho do raio no eixo x da elipse;

Description
Esta função só está definida para as classes Oval e OvalArc. A alteração deste atributo provoca uma alteração do atributo Width;

Example
```
/* Define o tamanho do raio no eixo x
do objeto oval3 como sendo
100 */
oval3->SetXRadius (100);
```

Errors

Se objeto não for um Oval
ou OvalArc,
success = false.

Related Commands
GetXRadius, Set, SetRadius, SetWidth, SetYRadius.

5.82 SetY

Name
SetY - define a coordenada Y do ponto de referência do objeto;

Synopsis
virtual void SetY (double coordy);

Arguments

coordy - coordenada Y do ponto de referência do objeto.

Description
Esta função está definida para todas as classes, embora ele não seja utilizado na classe Sound. Cada objeto define o ponto de referência de uma forma. Para maiores detalhes, veja descrição dos objetos.

Example
```
/* Define a ordenada do objeto dot
como sendo 200 */

dot->SetY (200);
```

Related Commands
GetY, Move, Set, SetX.

5.83 SetYRadius

Name
SetYRadius - define o tamanho do raio no eixo y da elipse;

Synopsis
virtual void SetYRadius (double yradius);

Arguments

yradius - tamanho do raio no eixo y da elipse;

Description
Esta função só está definida para as classes Oval e OvalArc. A alteração deste atributo provoca uma alteração do atributo Height;

Example
```
/* Define o tamanho do raio no eixo y
do objeto ovalarc3 como

sendo 300 */

ovalarc3->SetYRadius (300);
```

Errors

Se objeto não for um Oval
ou OvalArc,
success = false.

Related Commands
GetYRadius, Set, SetRadius, SetWidth, SetXRadius.

5.84 SetZoom

Name
SetZoom - define o fator de zoom de um objeto;

Synopsis
virtual void SetZoom (float zoom);

Arguments

zoom - fator de zoom do objeto;

Description
Esta função está definida para todas as classes, embora não seja utilizada pelo objeto Dot.

- Sendo este fator igual a 1, o objeto passa a ter o seu tamanho normal.
- Sendo $|zoom| < 1$, o objeto fica menor que o seu tamanho normal, e $|zoom| > 1$, o objeto fica maior.

Example
```
/* Amplia o o objeto rectangle para o
dobro de seu tamanho normal */

rectangle->SetZoom (2);

/* Retorna o objeto acima ao seu
tamanho normal */

rectangle->Setzoom (1);
```

Related Commands
GetZoom, Set, Zoom.

5.85 Zoom

Name
Zoom - realiza um zoom em um objeto;

Synopsis
virtual void Zoom (float zoom);

Arguments

zoom - fator de zoom do objeto;

Description
Esta função está definida para todas as classes, embora não seja utilizada pelo objeto Dot.

- Sendo este fator igual a 1, esta função não altera o fator de zoom corrente.
- Sendo $|zoom| < 1$, objeto fica menor, e sendo maior que 1, o objeto fica maior.

Example
```
/* Amplia o objeto rectangle para o
dobro de seu tamanho atual */

rectangle->Zoom (2);
```

```
/* Retorna ao tamanho anterior */

rectangle->Zoom (0.5);
```

Related Commands
GetZoom, Set, SetZoom.

A Exemplo da definição de uma classe

A seguir, apresentamos as definições das classes Rectangle, bastante simples, e da classe BinaryTree, mais complexa. Além disso, apresentamos a definição da classe GraphObj, que é a classe base para todas as demais

A.1 Definição da classe Rectangle

```
/*************************************************
 *
 * Modulo:       Rectangle
 *
 * Arquivo:      rectangle.h
 *
 * Funcao:       Implementacao da classe Rectangle.
 *
 * Estrutura:  Como atributo proprio, esta classe so' possui
 *               o tamanho e a largura do retangulo.
 *
 *
 * Operacoes:
 *
 * Obs:
 *
 * Autor:        Eduardo Aguiar Patrocinio
 *
 * Versao :                    1.6
 * Data de Criacao:            17/12/91
 *
 * Data da Ultima Alteracao: 04/08/92
 *
```

```
 * Ultimas Alteracoes: - Utilizacao da classe EgoView, ao inves
 *                       de World (1.6)
 *
 ************************************************/

#ifndef _RECTANGLE_DEFINED
#define _RECTANGLE_DEFINED

#include "graphobj.h"

class Rectangle : public GraphObj
{
    public :
      Rectangle (EgoView *, double, double, double, double);
      Rectangle (EgoView *,
                 double,
                 double,
                 double,
                 double,
                 va_tdcl (SetOption));
      ~Rectangle (void);

      virtual double Distance (double, double);
      Boolean Visible (void);
      void SetHeight (double);
      void SetWidth (double);
      virtual double GetHeight (void);
      virtual double GetWidth (void);
    private :
      virtual void Border (double, double);
      virtual void Fill (void);
```

```
        virtual void GetArea (double &, double &, double &,
double &);
        virtual void Highlight (double, double);
        virtual void MagnifyZoomFactor (double);

        double height;
        double width;
};                                              /* Rectangle */

typedef Rectangle *RectanglePointer;

#endif
```

A.2 Definição da classe BinaryTree

```
/*****************************************************
 *
 * Modulo:       BinaryTree
 *
 * Arquivo:      binarytree.h
 *
 * Funcao:       Implementacao da classe BinaryTree.
 *
 * Estrutura:  Como atributo proprio, esta classe possui um
 *               retangulo, dois segmentos e dois apontadores
 *               para os filhos.
 *
 *
 * Operacoes:
 *
 * Obs:
 *
```

```
 * Autor:        Eduardo Aguiar Patrocinio
 *
 * Versao :                    1.3
 * Data de Criacao:            06/03/92
 *
 * Data da Ultima Alteracao: 04/08/92
 *
 * Ultimas Alteracoes: -
 Utilizacao de EgoView, ao inves de World (1.3)
 *
 ******************************************************/

#ifndef _BINARYTREE_DEFINED
#define _BINARYTREE_DEFINED

#include "group.h"
#include "segment.h"

class BinaryTree : public Group
{
    public :
      BinaryTree (EgoView *, double, double, GraphObj *);
      BinaryTree (EgoView *,
                  double,
                  double,
                  GraphObj *,
                  Segment *,
                  Segment *,
                  BinaryTree *,
                  BinaryTree *);
      ~BinaryTree (void);
```

```
    virtual void SetInfo (InfoPointer);
    virtual void SetLeftSegment (Segment *);
    virtual void SetLeftTree (BinaryTree *);
    virtual void SetObject (GraphObj *);
    virtual void SetRightSegment (Segment *);
    virtual void SetRightTree (BinaryTree *);

    virtual BinaryTree *GetFatherTree (void);
    virtual InfoPointer GetInfo (void);
    virtual Segment *GetLeftSegment (void);
    virtual BinaryTree *GetLeftTree (void);
    virtual GraphObj *GetObject (void);
    virtual Segment *GetRightSegment (void);
    virtual BinaryTree *GetRightTree (void);

    virtual Boolean Delete (GraphObjPointer);
    virtual void Insert (GraphObjPointer);

  private :

    GraphObj *Object;
    Segment *LeftSegment;
    Segment *RightSegment;
    BinaryTree *LeftTree;
    BinaryTree *RightTree;
    BinaryTree *FatherTree;
    InfoPointer info;

};                                          /* BinaryTree */

typedef BinaryTree *BinaryTreePointer;
```

```
#endif
```

A.3 Definição da meta-classe GraphObj

```
/*********************************************************
 *
 * Modulo:       Graphical Objetcs
 *
 * Arquivo:      graphobj.C
 *
 * Funcao:       Criacao da classe GraphObj. A classe GraphObj
 *               e' a classe basica, e todas as demais classes
 *               herdam-na.
 *
 * Estrutura:  A classe GraphObj contem as coordenadas do
 *               ponto de referencia, o GC e o world associado
 *               ao objeto.
 *
 *
 * Operacoes:  inline GraphObj& operator = (GraphObj &);
 *               inline Boolean operator == (const GraphObj &);
 *
 * Obs:
 *
 * Autor:        Eduardo Aguiar Patrocinio
 *
 * Versao :                    1.9
 *
 * Data de Criacao:            08/11/91
 *
 * Data da Ultima Alteracao: 17/09/92
```

```
 *
 * Ultimas Alteracoes: -
 Declaracao do atributo PointMode (1.9)
 *
 ********************************************************/


#ifndef _GRAPH_OBJ_DEFINED
#define _GRAPH_OBJ_DEFINED

#include "global.h"
#include "linkedlist.h"

class Segment;
class BinaryTree;

enum ArrowPosition
{
    INITIAL_ARROW,
    FINAL_ARROW
  };

enum FunctionOption
{
    PARALLEL_LINE,
    AXES_PARALLEL_LINE,
    TRIANGLE,
    ELIPSIS,
    PARABOLA,
    SIN
  };
enum SetOption
```

```
{
    END_OF_SET_OPTION,
    ANGLE,
    ARROW_PRESENCE,
    ARROW_HEIGHT,
    ARROW_REWIND,
    ARROW_WIDTH,
    BORDERCOLOR,
    FAMILY,
    FILLCOLOR,
    HEIGHT,
    HIGHLIGHT,
    INITIAL_ANGLE,
    LAYER,
    LINE_STYLE,
    LINE_WIDTH,
    POINT_MODE,
    RADIUS,
    SELECTED,
    SHADOW_X,
    SHADOW_Y,
    SQUARE_SIZE,
    SOUND,
    STRING,
    STRING_FONT_SIZE,
    STYLE,
    SYNCHRONIZED,
    TRANSPARENCY,
    VISIBLE,
    VOLUME,
    WIDTH,
    X,
```

```
    X_RADIUS,
    Y,
    Y_RADIUS,
    ZOOM,
  };

enum StringPosition
{
    HORIZONTAL,
    VERTICAL
  };

enum Transparency
{
    TRANSPARENT,
    OPAQUE,
    TRANSLUCENT};

enum CreateOvalType
{
    GIVEN_RADIUS,
    GIVEN_RECTANGLE
  };

enum GroupMode
{
    ONLY_THIS,
    EVERYONE,
    ONLY_THAT_MATCHES
  };

enum PointMode
```

```
{
    DOT,
    BOX,
    CROSS,
    ROUND
  };                                        /* PointMode */

enum TreePosition
{
    LEFT,
    RIGHT
  };

typedef double (*FunctionPtr)(double);

#define NUM_TRANSPARENCY_SIZE 3

#define HIGHLIGHT_OFFSET 2

#define NUMBER_DEFAULT 0
#define BOOLEAN_DEFAULT false
#define STRING_DEFAULT ""
#define GROUP_MODE_DEFAULT EVERYONE
#define POINTER_DEFAULT 0

class Info
{
};                                          /* Info */

typedef Info *InfoPointer;

class GraphObj
```

```
{
    friend class Group;
    friend class FindList;
    friend class EgoView;

    public :
      GraphObj (EgoView *, double, double);
      virtual ~GraphObj (void);

      GraphObj& operator = (const GraphObj &);
      Boolean operator == (const GraphObj &);

      virtual void Set (va_tdcl (SetOption));
      virtual void SetAngle (int);
      virtual void SetArrow (ArrowPosition, Boolean);
      virtual void SetArrowHeight (double);
      virtual void SetArrowRewind (double);
      virtual void SetArrowWidth (double);
      static void SetAutoFlush (Boolean);
      virtual void SetBorderColor (int);
      virtual void SetDelta (double);
      virtual void SetFamily (char *);
      virtual void SetFillColor (int);
      virtual void SetFinalPoint (double);
      virtual void SetFontSize (int);
      virtual void SetGroupMode (GroupMode);
      virtual void SetHeight (double);
      virtual void SetHighlight (Boolean);
      virtual void SetInfo (InfoPointer);
      virtual void SetInitialAngle (int);
      virtual void SetInitialPoint (double);
      virtual void SetLayer (unsigned int);
```

virtual void SetLeftSegment (Segment *);
virtual void SetLeftTree (BinaryTree *);
virtual void SetLineStyle (**int**);
virtual void SetLineWidth (**unsigned int**);
virtual void SetObject (GraphObj *);
virtual void SetPointMode (PointMode);
virtual void SetRadius (**double**);
virtual void SetRightSegment (Segment *);
virtual void SetRightTree (BinaryTree *);
virtual void SetSelected (Boolean);
virtual void SetShadowX (**int**);
virtual void SetShadowY (**int**);
virtual void SetSize (**double**);
virtual void SetSound (**char** *);
virtual void SetString (**char** *);
virtual void SetStyle (**char** *);
virtual void SetSynchronized (Boolean);
virtual void SetTransparency (Transparency);
virtual void SetVisible (Boolean);
virtual void SetVolume (**int**);
virtual void SetWidth (**double**);
virtual void SetX (**double**);
virtual void SetXFunction (FunctionPtr xfunction);
virtual void SetXRadius (**double**);
virtual void SetY (**double**);
virtual void SetYFunction (FunctionPtr yfunction);
virtual void SetYRadius (**double**);
virtual void SetZoom (**float**);

virtual int GetAngle (**void**);
virtual Boolean GetArrow (ArrowPosition);
virtual double GetArrowHeight (**void**);

```
virtual double GetArrowRewind (void);
virtual double GetArrowWidth (void);
static Boolean GetAutoFlush (void);
int GetBorderColor (void);
virtual double GetDelta (void);
int GetFillColor (void);
virtual int GetFontSize (void);
virtual void GetFamily (char *);
virtual BinaryTree *GetFatherTree (void);
virtual GroupMode GetGroupMode (void);
virtual double GetHeight (void);
Boolean GetHighlight (void);
virtual int GetInitialAngle (void);
virtual double GetInitialPoint (void);
virtual InfoPointer GetInfo (void);
virtual double GetFinalPoint (void);
unsigned int GetLayer (void);
virtual Segment *GetLeftSegment (void);
virtual BinaryTree *GetLeftTree (void);
virtual int GetLength (void);
int GetLineStyle (void);
unsigned int GetLineWidth (void);
virtual GraphObj *GetObject (void);
virtual PointMode GetPointMode (void);
virtual double GetRadius (void);
virtual Segment *GetRightSegment (void);
virtual BinaryTree *GetRightTree (void);
Boolean GetSelected (void);
int GetShadowX (void);
int GetShadowY (void);
virtual double GetSize (void);
virtual void GetSound (char *);
```

```
        virtual void GetString (char *);
        virtual void GetStyle (char *);
        virtual Boolean GetSynchronized (void);
        Transparency GetTransparency (void);
        Boolean GetVisible (void);
        virtual int GetVolume (void);
        virtual double GetWidth (void);
        double GetX (void);
        virtual FunctionPtr GetXFunction (void);
        virtual double GetXRadius (void);
        double GetY (void);
        virtual FunctionPtr GetYFunction (void);
        virtual double GetYRadius (void);
        float GetZoom (void);

        virtual double Distance (double, double);
        virtual void Move (double, double);
        virtual void Move (double, double, double, unsigned);
        virtual void Move (FunctionPtr,
                           FunctionPtr,
                           double,
                           double,
                           double,
                           int);
        virtual void Move (int, double *, double *, unsigned);
        virtual void Move (int, double *, double *, double,
unsigned);
        virtual void Move (FunctionOption,
                           double,
                           double,
                           double,
                           double,
```

```
                                unsigned);
        virtual void MoveLayer (int);
        virtual void Rotate (float);
        virtual void Zoom (float);

        GraphObj *GetInsideObject (double, double, double);
        void GetObjectArea (double &, double &, double &,
double &);
    protected :
        void AppendDrawPointer (NodePointer);
        void ClearArea (void);
        virtual void Draw (GraphObj *);
        void DrawShadow (void);
        void Redraw (void);
        virtual void Remove (GraphObj *);
        void Sleep (double);
        void Undraw (void);
        virtual Boolean Search (GraphObj*);
        Region SetArea (void);
        virtual void SetDrawPointer (GraphObj *);

        virtual void Border (double, double);
        virtual void Fill (void);
        virtual void GetArea (double &, double &, double &,
double &);
        virtual void Highlight (double, double);
        virtual void MagnifyZoomFactor (double);

        void ParallelLine (double, double, double, double,
unsigned);
        void AxesParallelLine (double, double, double, double,
unsigned);
```

```
        void Triangle (double, double, double, double,
unsigned);
        void Elipsis (double, double, double, double,
unsigned);
        void Parabola (double, double, double, double,
unsigned);
        void Sin (double, double, double, double, unsigned);

        // coordenadas do ponto de referencia
        double x1;
        double y1;

        // gc associado ao objeto
        static GC gc;

        // EgoView associado ao objeto
        EgoView *egoview;

        // cor do objeto
        unsigned long bordercolor;
        unsigned long fillcolor;
        int fgcolor, bgcolor;

        // flush automatico
        static Boolean autoflush;

        // objeto visivel
        Boolean visible;

        // transparencia
        Transparency transparency;
```

```
        // angulo de rotacao
        float angle;

        // fator de zoom
        double zoom;

        // estilo da linha (solida/tracejada)
        int linestyle;

        // tamanho da linha
        unsigned int linewidth;

        // objeto em highlight/nao
        Boolean highlight;

        // tamanho da sombra
        int shadowx;
        int shadowy;

        // objeto no qual este objeto deve ser desenhado
        NodePointer drawpointer;

        // objeto selecionado/nao
        Boolean selected;

};                                              /* GraphObj */

typedef GraphObj *GraphObjPointer;

#endif
```

B Exemplo da declaração de uma classe

A seguir, apresentamos o código-fonte para a classe Rectangle, definida acima

```
/*********************************************************
 *
 * Modulo:       Rectangle
 *
 * Arquivo:      rectangle.h
 *
 * Funcao:       Implementacao da classe Rectangle.
 *
 * Estrutura:  Como atributo proprio, esta classe so' possui o
 *               tamanho e a largura do retangulo.
 *
 *
 * Operacoes:
 *
 * Obs:
 *
 * Autor:        Eduardo Aguiar Patrocinio
 *
 * Versao :                    1.6
 * Data de Criacao:            17/12/91
 *
 * Data da Ultima Alteracao: 04/08/92
 *
 * Ultimas Alteracoes: - Utilizacao de EgoView, ao inves de
 *                             World (1.6)
```

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

#include "egoview.h"
#include "rectangle.h"

 Rectangle::Rectangle (EgoView *egoview,
                       double x,
                       double y,
                       double w,
                       double h) : (egoview, x, y)
{
    width = w;
    height = h;
}  /* Rectangle */

 Rectangle::Rectangle (EgoView *egoview,
                       double x,
                       double y,
                       double w,
                       double h,
                       va_tdcl (SetOption)) : (egoview, x, y)
{
    width = w;
    height = h;
    Set (va_alist1);
}  /* Rectangle */

 Rectangle::~Rectangle (void)
{
    SetVisible (false);
}  /* ~Rectangle */
```

```
double Rectangle::Distance (double x, double y)
{
    return rectangledistance (x1, y1, width, height, x, y);
}  /* Distance */

Boolean Rectangle::Visible (void)
{
    int view_width, view_height;

    egoview→get_view (view_width, view_height);

    return (Boolean)(((x1 > 0) && (x1 < view_width) &&
                      (y1 > 0) && (y1 < view_height)) ||
                     ((x1+width-1 > 0) && (x1+width-1 <
view_width) &&
                      (y1+height-1 > 0) && (y1+height-1 <
view_height)));
}  /* Visible */

void Rectangle::SetHeight (double h)
{
//     cerr << "Rectangle::SetHeight\n";
    if (height ≠ h)
      if (visible)
        {
            Undraw();
            height = h;
            Redraw ();
        }
      else
        height = h;
```

```
}  /* SetHeight */

void Rectangle::SetWidth (double w)
{
    if (width ≠ w)
      if (visible)
        {
            Undraw ();
            width = w;
            Redraw ();
        }
      else
        width = w;
}  /* SetWidth */

double Rectangle::GetHeight (void)
{
    return height;
}  /* GetHeight */

double Rectangle::GetWidth (void)
{
    return width;
}  /* GetWidth */

void Rectangle::Fill (void)
{
    egoview→FillRectangle (gc, x1, y1, x1+width-1,
y1+height-1);
}  /* Fill */

void Rectangle::Border (double x1, double y1)
```

```
{
    egoview→DrawRectangle (gc, x1, y1, x1+width-1,
y1+height-1);
}  /* Border */

void Rectangle::GetArea (double &xmin,
                         double &ymin,
                         double &xmax,
                         double &ymax)
{
//    cerr << "Rectangle::GetArea\n";
    if (width ≥ 0)
      {
          xmin = 0;
          xmax = width;
      }
    else
      {
          xmin = width;
          xmax = 0;
      }
    if (height ≥ 0)
      {
          ymin = 0;
          ymax = height;
      }
    else
      {
          ymin = height;
          ymax = 0;
      }
}  /* GetArea  */
```

```
void Rectangle::Highlight (double offset_x, double offset_y)
{
    egoview→DrawRectangle (gc,
                           x1 - offset_x,
                           y1 - offset_y,
                           x1 + width - 1 + offset_x,
                           y1 + height - 1 + offset_y);
}  /* Highlight */

void Rectangle::MagnifyZoomFactor (double factor)
{
    height * = factor;
    width * = factor;
}  /* MagnifyZoomFactor */
```

C Exemplo da utilização de uma classe

A seguir, apresentamos o código de um procedimento que recebe como parâmetro o egoview a ser utilizado um vetor e o número de elementos deste e cria um "vetor" de retângulo, com o tamanho proporcional ao valor da respectiva posição no vetor.

```
void showelements(Egoview *egoview, int *x, int len)
{
    double window_min_x;
    double window_min_y;
    double window_max_x;
    double window_max_y;

    length = len;
```

```
    egoview→get_values(window_min_x, window_max_y,
window_max_x, window_min_y);
    height = 0.8*((window_max_y-window_min_y) / length);

    int i;

    rect = new RectanglePointer [length];
    p = new int [length];

    width = (window_max_x - window_min_x) / length;

    for (i = 0; i < length; i++)
      {
          p[i] = i;
          rect[i] = new Rectangle (egoview, i*width, 0, width,
x[i]*height);
          rect[i]→SetBorderColor (BORDER_COLOR);
          rect[i]→SetFillColor (FILL_COLOR);
          rect[i]→SetVisible (true);
      }
}
```

Referências

[Hel90] Dan Heller. *XView Programming Manual*, volume seven. O'Reilly & Associates, Inc., second edition, July 1990.

[Jac92] W. R. Jacometti. GeoLab – um ambiente para desenvolvimento de algoritmos em geometria computacional. Master's thesis, DCC - IMECC - UNICAMP, 1992.

[Nye90a] Adrian Nye. *Xlib Programming Manual (for Version 11)*, volume one. O'Reilly & Associates, Inc., Sebastapol, CA, second edition, Abril 1990.

[Nye90b] Adrian Nye. *Xlib Reference Manual (for Version 11)*, volume one. O'Reilly & Associates, Inc., Sebastapol, CA, second edition, Abril 1990.

[PR93] E. A. Patrocínio and P. J. Rezende. EGOLib — uma biblioteca orientada a objetos gráficos. In *XX Seminário Integrado de Software e Hardware*, 1993.

[Str87] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley Publishing Company, 1st. edition, 1987.

[SW85] Daniel F. Stubbs and Neil W. Webre. *Data Structures with Abstract Data Types and Pascal.* Pacific Grove, 2nd. edition, 1985.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,**
J. L. de Oliveira, R. de O. Anido

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

04/93 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

05/93 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

06/93 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

07/93 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

08/93 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

09/93 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

10/93 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

11/93 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

12/93 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

13/93 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

14/93 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

15/93 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

16/93 $\mathcal{LL}$ **– An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

17/93 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

18/93 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

19/93 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

20/93 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

21/93 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

22/93 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

23/93 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

24/93 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

25/93 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

26/93 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

27/93 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

28/93 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br