2006.12.04

図形描画のための関数群

図形描画のための関数は以下の関数群から構成される。

(1)描画のためのウィンドウのオープン/クローズ関数
(2)図形の属性を指定するための関数
(3)図形を描画するための関数

図形は、一つのウィンドウ上に描画する。このウィンドウの大きさを指定して開くための関数として gOpenWindow() がある。 gCloseWindow() は閉じるための関数であるが、この関数を呼び出すと描画ウィンドウが画面から消滅するので、通常この関数を使用することはない。
図形は、線分、折れ線などの**線図形**、長方形、楕円、多角形などの**面図形**、および、**文字図形**からなる。 図形を描くには、原則として先に描画属性を指定しなければな らない。描画属性としては、線の太さ、線の種類、線の色等の線図形に適応されるもの、塗りつぶしの色等の面図形に適応されるもの、および、フォントの種類、フォントサイズ等の文字図形に適応されるものがある。描画属性を指定しない場合は、予め定められた属性値が用いられ、一旦、設定した属性値は、同じ属性に対して新たに設定されるまで有効である。図形を描画するための関数としては、線分の描画、長方形の描画、楕円の描画、円の描画、文字列の描画等の関数がある。

以下、各関数について説明する。

(1) 描画のためのウィンドウのオープン/クローズ関数

1. **gOpenWindow(w, h)**
   整数 w, h
   幅 w、高さ h の描画のためのウィンドウを開く。ウィンドウの位置は、画面の左上に固定される。

   [使用例] 幅 400、高さ 400 のウィンドウを開く。
   gOpenWindow(400, 400)

2. **gCloseWindow()**
   描画のためのウィンドウを閉じる。

(2) 図形の属性を指定するための関数

1. **gSetLineColor(r, g, b)**
   整数 r, g, b (値の範囲 0-255)
   線の色属性を、赤の要素 r、緑の要素 g、青の要素 b で指定する。
   初期値は黒 (0, 0, 0)。

   [使用例] 線の色を黄色に設定する。
   gSetLineColor(255, 255, 0)

2. **gSetLineShape(type)**
   整数 type
   描画する線のタイプを変更。線を使用する全ての描画に影響。

   type=0 のとき　実線 (初期値)
   type=1 のとき　破線
   type=2 のとき　間隔の狭い破線
   type=3 のとき　一点鎖線
   type≥4 のとき　線の種類を変更しない

   [使用例] 線のタイプを一点鎖線に変更する。
   gSetLineShape(3)

3. **gSetLineWidth(width)**
   整数 width
   描画する線の太さを選択した種類に変更。線を使用する全ての描画に影響。
   初期値は 1。
   [使用例] 線の太さを 10 に変更する。
   gSetLineWidth(10)

4. **gSetArrowType(type)**
   整数 type
   type で矢じりの形状を指定する。
   (現在 1 種類のみの実装であるためどのような数値でも同じ)

   [使用例] gDrawLine の矢印の種類を設定する。
   gSetArrowType(1)

5. **gSetArrowDir(edge)**
   整数 edge
   gDrawLine で描画する線分の両端に矢印を付加する関数。
   edge で矢じりの箇所を指定する。

   edge=0 のとき　矢じりなし (初期値)
   edge=1 のとき　始点のみ矢じり描画
   edge=2 のとき　終点のみ矢じり描画
   edge=3 のとき　両端に矢じり描画
   edge≥4 のとき　指定箇所を変更しない

   [使用例] gDrawLine の両端に矢印を設定する。
   gSetArrowDir(3)

6. **gSetFillColor(r, g, b)**
   整数 r, g, b (値の範囲 0-255)
   図形内部の色を、赤の要素 r、緑の要素 g、青の要素 b で指定する。使用方法は gSetLineColor と同じ。
   初期値は黒 (0, 0, 0)。

7. **gSetDotShape(type)**
   整数 type
   gDrawPoint で描く点の種類を指定する。

   type=0 のとき　小さな丸い点 (初期値)
   type=1 のとき　少し大きな
   type=2 のとき　大きな丸い点
   type≥3 のとき　点の種類を変更しない

   [使用例] ドットの種類を大きな丸い点に変更する。
   gSetDotShape(2)

8. **gSetTextColor(r, g, b)**
   整数 r, g, b (値の範囲 0-255)
   文字列の色属性を、赤の要素 r、緑の要素 g、青の要素 b で指定する。使用方法は gSetLineColor と同じ。
   初期値は (0, 0, 0)。

9. **gSetFont(”String”)**
   文字列 String
   gDrawText 描画するフォントの種類を指定変更する。
   初期値はシステムで定義されているディフォールトフォント。
   指定できるフォントは以下の通り。
   （ただし、計算機環境に依存する）
   [使用例] フォントの種類を Monospaced に変更する。
   gSetFont(”Monospaced”)

10. **gSetFontType(style)**
    整数 type
    gDrawText 描画するフォントのスタイルを指定する。

    [使用例] フォントスタイルをイタリックに設定。
    gSetFontType(2)

11. **gSetFontSize(size)**
    整数 size
    gDrawText 描画するフォントのサイズを指定する。

フォント名	解説
明朝	明朝体フォント
ゴシック	ゴシック体フォント
TimesRoman	セリフ付きのフォント
Courier	等幅フォント
Helvetica	セリフのないフォント
Dialog	ダイアログ表示用フォント
DialogInput	ダイアログ入力用フォント
ZapfDingbats	記号フォント
Serif	セリフ付きフォント (明朝体)
SansSerif	セリフのないフォント (ゴシック)
Monospaced	等幅フォント

style=0 のとき	プレイン (初期値)
style=1 のとき	ボールド
style=2 のとき	イタリック
style=3 のとき	ボールド + イタリック
style ≧ 4 のとき	スタイルを変更しない

初期値は 10。

[使用例] フォントのサイズを 20 に変更する。
gSetFontSize(20)

(3) 図形を描画するための関数

1. **gDrawText(”string”, x, y)**
   文字列 string, 整数 x, y
   座標 (x, y) を一文字目の左下にあわせ、文字列を描く。

   [使用例] (20, 30) から「文字列」を表示。
   gDrawText(”文字列”, 20, 30)

2. **gDrawPoint(x, y)**
   整数 x, y
   座標 (x, y) に点を描く。

   [使用例] (20, 30) に点を描く。
   gDrawPoint(20, 30)

3. **gDrawLine(x1, y1, x2, y2)**

整数 x1, y1, x2, y2
座標 (x1 ,y1) から座標 (x2, y2) まで線分を描く。

[使用例] (50, 60) から (100, 120) まで線分を描画。
gDrawLine(50, 60, 100, 120)

4. **gDrawBox(x, y, width, height)**
整数 x, y, width, height
座標 (x, y) から幅 width 高さ height の矩形を描く。

[使用例] (20, 30) から幅 40、高さ 50 の矩形を描く。
gDrawBox(20, 30, 40, 50)

5. **gFillBox(x, y, width, height)**
整数 x, y, width, height
gDrawBox() の内部を gSetFillColor() で指定した色で塗りつぶしたものを描画する。

[使用例] (20, 30) から右下に幅 40、高さ 50 の内部を塗りつぶした矩形を描く。
gFillBox(20, 30, 40, 50)

6. **gDrawOval(x, y, width, height)**
整数 x, y, width, height
座標 (x, y) から幅 width 高さ height で矩形を定義し、内接する楕円を描く。

[使用例] (20, 30) から右下に幅 40、高さ 50 の矩形に内接する楕円を描く。
gDrawOval(20, 30, 40, 50)

7. **gFillOval(x, y, width, height)**
整数 x, y, width, height
gDrawOval() の内部を gSetFillColor() で指定した色で塗りつぶしたものを描画する。

[使用例] (20, 30) から右下に幅 40、高さ 50 の矩形に内接する、内部を塗りつぶした楕円を描く。
gFillOval(20, 30, 40, 50)

8. **gDrawCircle(x, y, r)**
整数 x, y, r
座標 (x, y) を中心に半径 r の円を描く。

[使用例] (60, 70) を中心に半径 50 の円を描く。
gDrawCircle(60, 70, 50)

9. **gFillCircle(x, y, r)**

整数 x, y, r
gDrawCircle() の内部を gSetFillColor() で指定した色で塗りつぶしたものを描画する。

[使用例] (60, 70) を中心に半径 50 の、内部を塗りつぶした円を描く。
gFillCircle(60, 70, 50)

10. **gDrawArc(x, y, width, height, start, extent, type)**
整数 x, y, width, height, start(度), extent(度), type
座標 (x, y) から幅 width 高さ height で矩形を定義し、内接する楕円において (中心から真右を 0 度とし左回りに) start 度 から (左回りに)extent 度の弧を描画する。
type　により弧の閉じ方の種類 (OPEN=0,CHORD=1,PIE=2) のいづれかを指定する。

[使用例] (20, 30) から右下に幅 40、高さ 50 の矩形に内接する楕円の、60 度から、90 度分の孤を閉じ方 OPEN で描く。
gDrawArc(20, 30, 40, 50, 60, 90, 0)

11. **gFillArc(x, y, width, height, start, extent, type)**
整数 x, y, width, height, start(度), extent(度), type
gDrawArc() の内部を gSetFillColor() で指定した色で塗りつぶしたものを描画する。

[使用例] (20, 30) から右下に幅 40、高さ 50 の矩形に内接する楕円の、60 度から、90 度分の扇を閉じ方 OPEN で描く。
gFillArc(20, 30, 40, 50, 60, 90, 0)

12. **gClearWindow()**
描画 Window をクリアする。

ファイル入出力のための関数群

プログラムの中から、ファイル内のデータを読んだり、ファイルに書いたりするための関数群を紹介する。

ファイルを読み書きするためには、まず、当該ファイルを開く (オープンする) 必要がある。また、ファイルの読み書きが終了したとき、ファイルを閉じなければならない (クローズ)。

以下、各関数について説明する。

(1) ファイルのオープン／クローズ関数

1. **fd ← openr(filepath)**
   整数 fd, 文字列 filepath
   filepath で指定したファイルを、データの読み込みのために開く。
   fd はファイル識別子であり、以後、fd を指定して、ファイルからの読み込み等を行う。
2. **fd ← openw(filepath)**
   整数 fd, 文字列 filepath
   filepath で指定したファイルを、データの書き込みのために開く。
   指定したファイルが既に存在しているならば、そのファイルはこれから書き出すデータで置き換えられる。
3. **fd ← opena(filepath)**
   整数 fd, 文字列 filepath
   filepath で指定したファイルを、データの追記書込みのために開く。
   指定したファイルが既に存在しているならば、そのファイルの続きとして、これから書き出すデータを追記する。
4. **close(fd)**
   整数 fd
   fd で指定したファイルを閉じる。

(2) ファイルの読み書きのための関数

1. **str ← getstr(fd, n)**
   文字列 str, 整数 fd, n
   fd で指定したファイルから、n 文字読み込んで その文字列を返す。ただし、改行コード (Return, Linefeed) もそれぞれ 1 文字と数える。
2. **str ← getline(fd)**
   文字列 str, 整数 fd
   fd で指定したファイルから、1 行読み込んで その文字列を返す。
3. **putstr(fd, str)**
   整数 fd, 文字列 str
   fd で指定したファイルに、str の文字列を書き出す。

4. **putline(fd, str)**
   整数 fd, 文字列 str
   fd で指定したファイルに、str の文字列を書き出す。
   putstr() は、文字列だけを書き出すのに比べて、putline() は、文字列の後に改行コードも書き出す。

(3) ファイル処理のための補助関数

1. **str ← isfile(filepath)**
   文字列 str, 文字列 filepath
   filepath で指定したファイルが存在するかどうかを調べる。
   ファイルが存在する場合 ”tree”、存在しない場合”false”の文字列が返される。
2. **rename(filepath1, filepath2)**
   文字列 filepath1, filepath2
   filepath1 で指定したファイルの名前を、filepath2 に置き換える。
3. **flush(fd)**
   整数 fd
   fd で指定されるファイルに書き込み途中のデータを強制的に書込む。(ファイルは通常バッファリングされるので、この関数を呼び出すことで、バッファ内容を吐き出す)
4. **remove(filepath)**
   文字列 filepath
   filepath で指定したファイルを削除する。