2008/10/27

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
(4)その他

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(整数 w, 整数 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. **gSaveWindow(文字列 filepath, 文字列 format)**
   描画したウィンドウを保存する。保存形式 (mode) は JPEG または PNG のどちらかを指定する。

   [使用例] カレントディレクトリ (フォルダ) にウィンドウを png 型式でファイル名を abc.png として保存する。
   gSaveWindow(“./abc.PNG”, “PNG”)

3. **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(整数 r, 整数 g, 整数 b)**
   r, g, b の値の範囲は 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(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(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 3 に変更する。
gSetLineWidth(3)

4. **gSetArrowType(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(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(整数 r, 整数 g, 整数 b)**
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(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(整数 r, 整数 g, 整数 b)**
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(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）

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

[使用例] フォントの種類を Monospaced に変更する。
gSetFont(”Monospaced”)

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

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 $\geqq$ 4 のとき	スタイルを変更しない

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

11. **gSetFontSize(整数 size)**
gDrawText 描画するフォントのサイズを指定する。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(文字列 ”string”, 整数 x, 整数 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(整数 x, 整数 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(整数 x1, 整数 y1, 整数 x2, 整数 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(整数 x, 整数 y, 整数 width, 整数 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(整数 x, 整数 y, 整数 width, 整数 height)**
gDrawBox() の内部を gSetFillColor() で指定した色で塗りつぶした図形を描画する。

[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(整数 x, 整数 y, 整数 width, 整数 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(整数 x, 整数 y, 整数 width, 整数 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(整数 x, 整数 y, 整数 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(整数 x, 整数 y, 整数 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(整数 x, 整数 y, 整数 width, 整数 height, 整数 start, 整数 extent, 整数 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(整数 x, 整数 y, 整数 width, 整数 height, 整数 start, 整数 extent, 整数 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. **gDrawPolygon(整数列 x[], 整数列 y[], 整数 n)**
x, y で指定した点 (座標) を結んだ n 角形を描く。

[使用例] (200, 50), (250, 100), (250, 150), (200, 200),(150, 150),(150, 100)
を結ぶ六角形を描く。
x[0] ← 200 ,x[1] ← 250, x[2] ← 250, x[3] ← 200, x[4] ← 150, x[5] ← 150
y[0] ← 50, y[1] ← 100, y[2] ← 150, y[3] ← 200, y[4] ← 150, y[5] ← 100
gDrawPolygon(x, y, 6)

13. **gFillPolygon(整数列 x[], 整数列 y[], 整数 n)**
x, y で指定した点 (座標) を結んだ n 角形を塗りつぶしたものを描画する。

[使用例] (200, 50), (250, 100), (250, 150), (200, 200),(150, 150),(150, 100)
を結ぶ六角形を描いて塗りつぶす。
x[0] ← 200 ,x[1] ← 250, x[2] ← 250, x[3] ← 200, x[4] ← 150, x[5] ← 150
y[0] ← 50, y[1] ← 100, y[2] ← 150, y[3] ← 200, y[4] ← 150, y[5] ← 100
gDrawPolygon(x, y, 6)

14. **gDrawPolyline(整数列 x[], 整数列 y[], 整数 n)**
x, y で指定した点 (座標) を結ぶ直線を描画する。

[使用例] (10, 60), (10, 20), (60, 20), を結ぶ線を描く。
x[0] ← 10, x[1] ← 10, x[2] ← 60
y[0] ← 60, y[1] ← 20, y[2] ← 20
gDrawPolyline(x, y, 3)

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

(4) その他 (原点移動のための関数)

1. **gSetOrigin(実数 x, 実数 y)**

描画ウィンドウの左上からの右方向に x、下方向に y の点を原点として再定義する。

[使用例]

描画ウィンドウの (100,150) を中心とする円 (r=50) を描いた後に、この座標空間の (300,200) を原点に再定義する。そして、再定義した原点を中心とする円 (r=50) を描く。(図 1)

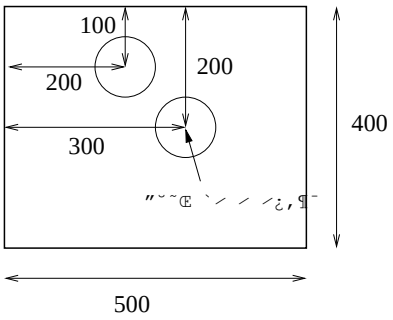

図 1

```
gOpenWindow(500,400)
gSetLineColor(255, 0, 0)
gDrawCircle(200, 100, 50)
gSetOrigin(300, 200)
gSetLineColor(0, 255, 0)
gDrawCircle(0, 0, 50)
```

(補足) グラフ描画のための関数群

このドキュメントは、PEN の描画機能を用いて、主として 2 次元座標平面上でグラフを描画するためのものである。描画ウィンドウを開くためには gOpenWindow() の代わりに gOpenGraphWindow() 関数を用いる。これによって、2 次元仮想平面上の任意の矩形領域を描画ウィンドウにマッピングすることができる。以後の描画命令は、仮想平面に対して『図形描画のための関数群』をそのまま利用することができる。

gSetMap() 関数は、仮想平面上の原点の並行移動もしくは、仮想平面上の任意の矩形領域を描画ウィンドウに再マッピングするためのものである。

グラフ描画のためのウィンドウのオープン/クローズ関数

1. **gOpenGraphWindow(実数 width, 実数 height, 実数 x1, 実数 y1, 実数 x2, 実数 y2, 論理値 axis)**

   幅 width、高さ height の描画ウィンドウを開く。ウィンドウの左下を (x1,y1) 座標とし、右上を (x2, y2) 座標となるような座標平面として定義する。axis の値を true, false に指定することで座標軸の表示、非表示を指定する。

   [使用例]

   幅 500、高さ 500 の描画ウィンドウを用意し、そのウィンドウの左下を (-50, -100)、右上を (350,300) の仮想座標空間として定義する。また、xy 座標軸を表記する。(図 2)

   gOpenGraphWindow(500, 500, -50, -100, 350, 300, true)

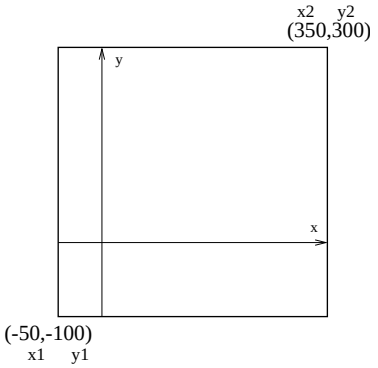

図 2

2. **gSetMap(実数 x1, 実数 y1, 実数 x2, 実数 y2)**

   描画ウィンドウの左下を点 (x1,y1)、右上を点 (x2,y2) として再指定する。(図 3)

   [使用例]

   図 2 のウィンドウを開いた後、左下を (0,-50)、右上を (150,200) の座標空間として再定義する。

   gOpenGraphWindow(500, 500, -50, -100, 350, 300, true)
   gSetMap(0, -50, 150, 200)

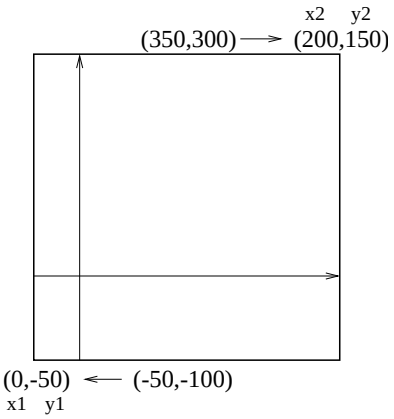

図 3