ブックカバー作成用グラフィックス関数解説（抜粋）

あらまし

C 言語から（文庫本サイズの）ブックカバー用の SVG ファイルを作成するための特定用途専用グラフィックスライブラリです。 Processing (http://processing.org/)という言語とできるだけ同じ名前で描画関数を用意しています。 しかし、いくつかの関数は簡略化されていますし、アニメーションやインタラクション関係の関数はありません。

座標系

初期状態では紙の左上が原点で、x 軸は右向き、y 軸は下向きにのびています。 ふつう数学で使う座標系と y 軸の向きが逆です。 長さの単位は mm（ミリメートル）です。
サイズは A4 紙用に固定になっています。 A4 紙のサイズは 横 297mm × 縦 210mm です。
ブックカバーにしたとき、描いた図形が本の表面に現れるようにするには、 だいたい (43, 31)—(253, 179)の座標の範囲 （横 210mm × 縦 148mm）に図形がおさまるようにして下さい。

関数一覧

初期設定・その他

`void start(void);`
　　描画の開始のときに呼び出します。
`void rulers(void);`
　　折り目の目安となる細い線を描きます。
`void finish(void);`
　　描画の終了のときに呼び出します。

色・属性設定

初期状態は、線なし・塗潰し黒です。
`void stroke(unsigned int color);`
　　線の色を設定します。色は 0xRRGGBB の形式で指定します。
`void strokeWeight(double w);`
　　線の太さを設定します。
`void strokeOpacity(double opacity);`
　　線の透明度を 0（完全透明）～ 1（完全不透明）の値で指定します。
`void noStroke(void);`
　　線を描きません。
`void fill(unsigned int color);`
　　塗潰しの色を設定します。色は 0xRRGGBB の形式で指定します。
`void fillOpacity(double opacity);`
　　塗潰しの透明度を 0（完全透明）～ 1（完全不透明）の値で指定します。
`void noFill(void);`
　　塗潰ししません。

基本図形

```
void line(double x1, double y1, double x2, double y2);
```

(x1, y1)から (x2, y2)へ線分を描きます。

```
void rect(double x, double y, double w, double h);
```

左上の頂点の座標が (x, y)、幅 w、 高さ h の長方形を描きます。

```
void ellipse(double x, double y, double w, double h);
```

中心の座標が (x, y)、幅 w、 高さ h の楕円を描きます。

```
void triangle(double x1, double y1, double x2, double y2,
              double x3, double y3);
```

3 つの頂点の座標が (x1, y1), (x2, y2), (x3, y3)の 三角形を描きます。

```
void quad(double x1, double y1, double x2, double y2,
          double x3, double y3, double x4, double y4);
```

4 つの頂点の座標が (x1, y1), (x2, y2), (x3, y3), (x4, y4)の 四角形を描きます。

```
void arc360(double x, double y, double w, double h,
            double start, double stop);
```

円弧を描きます。

(x, y)	円弧の楕円の中心の座標
(w, h)	円弧の楕円の幅と高さ
start	円弧を開始する角（単位は度）
stop	円弧を終了する角（単位は度）

```
void bezier(double ax0, double ay0, double cx0, double cy0,
            double cx1, double cy1, double ax1, double ay1);
```

ベジエ曲線を描きます。

(ax0, ay0)	1 つめのアンカーポイントの座標
(cx0, cy0)	1 つめのコントロールポイントの座標
(cx1, cy1)	2 つめのコントロールポイントの座標
(ax1, ay1)	2 つめのアンカーポイントの座標

```
void textFont(char fontName[], double size);
```

文字のフォントとサイズ（単位 mm）を設定します。 初期値は "MS-Mincho", 12 です。 Windows 上で SVG を閲覧する場合、フォントとしては "serif", "sans-serif", "monospace", "cursive", "fantasy", "ＭＳ 明朝", "**ＭＳ ゴシック**", "MS P明朝", "**MS P ゴシック**",（注: ＭとＳとＰは全角、空白は半角） "Arial", "Times New Roman", "Verdana", "Courier New", "Andale Mono", "*Comic Sans MS*", "Garamond", "Georgia", "Impact", "Tahoma", "Trebuchet MS" などが使えるはずです。

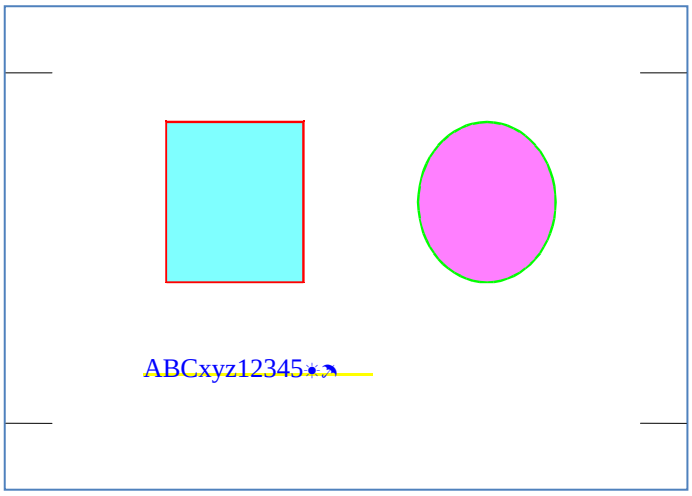
```
void text(char str[], double x, double y);
```

文字列 str を座標 (x, y)に表示します。

サンプルプログラム

```
#include "svg.h"                    /* ライブラリ用のヘッダファイル */

int main(void){
    start();                        /* 最初に必要 */
    rulers();                       /* 四隅の線   */

    strokeWeight(1);
    stroke(hsb360(0, 100, 100)); /* 線の色 */
    fill(hsb360(180, 50, 100));  /* 塗りの色 */
    rect(70, 50, 60, 70);        /* 長方形 */

    stroke(hsb360(120, 100, 100));
    fill(hsb360(300, 50, 100));
    ellipse(210, 85, 60, 70);    /* 楕円 */

    stroke(hsb360(60, 100, 100));
    line(60, 160, 160, 160);     /* 直線 */

    noStroke();
    fill(hsb360(240, 100, 100));
    textFont("Times New Roman", 12);
    text("ABCxyz12345&#x2600;&#x2602;", 60, 160); /* 文字列 */

    finish();                       /* 最後に必要 */
    return 0;
}
```

上記のプログラムが生成する図形



折れ線・曲線・多角形

```
void beginShape(void);
```

図形の定義を開始します。

```
void vertex(double x, double y);
```

beginShape で定義を開始した図形に頂点 (x, y)を追加します。

```
void bezierVertex(double cx0, double cy0, double cx1, double cy1,
                  double ax1, double ay1);
```
beginShape で定義を開始した図形にベジエ曲線を追加します。
(cx0, cy0)　　1 つめのコントロールポイントの座標
(cx1, cy1)　　2 つめのコントロールポイントの座標
(ax1, ay1)　　2 つめのアンカーポイントの座標

```
void endShape(int close);
```
beginShape で定義を開始した図形の定義を終了します。 close が 0 以外のときは、図形を閉じ多角形を描きます。 close が 0 のときは、閉じずに折れ線になります。

注意:

beginShape と endShape の間に vertex と bezierVertex 以外の描画関係の関数を呼び出さないようにして下さい。

色関連のユーティリティ

```
int hsb1(double h, double s, double v);
```
色の値を h（色相）, s（彩度）, v（明度）から計算します。 h（色相）, s（彩度）, v（明度）はそれぞれ 0 から 1 の範囲の数で指定します。

```
int hsb360(double h, double s, double v);
```
色の値を h（色相）, s（彩度）, v（明度）から計算します。 h（色相）は 0 から 360 の範囲、 s（彩度）, v（明度）はそれぞれ 0 から 100 の範囲の数で指定します。

```
int hsl1(double h, double s, double l);
```
色の値を h（色相）, s（彩度）, l（輝度）から計算します。 h（色相）, s（彩度）, l（輝度）はそれぞれ 0 から 1 の範囲の数で指定します。

```
int hsl360(double h, double s, double l);
```
色の値を h（色相）, s（彩度）, l（輝度）から計算します。 h（色相）は 0 から 360 の範囲、 s（彩度）, l（輝度）はそれぞれ 0 から 100 の範囲の数で指定します。

```
int rgb1(double r, double g, double b);
```
色の値を 光の三原色 r（赤）, g（緑）, b（青）から計算します。 r（赤）, g（緑）, b（青）はそれぞれ 0 から 1 の範囲の数で指定します。

```
int rgb255(double r, double g, double b);
```
色の値を 光の三原色 r（赤）, g（緑）, b（青）から計算します。 r（赤）, g（緑）, b（青）はそれぞれ 0 から 255 の範囲の数で指定します。

```
int bw1(double v);
```
無彩色の値を 0（黒）から 1（白）の値で指定します。

```
int bw255(double v);
```
無彩色の値を 0（黒）から 255（白）の値で指定します。

乱数その他のユーティリティ

```
double randomInRange(double min, double max);
```
min から max の範囲の乱数を発生します。

```
double radians(double deg);
```
度をラジアンに変換します。

タートルグラフィックス関数

“亀”は最初ページの真ん中 (148.5, 105)に ペンを下げた状態で 0 度の向き（右）を向いています。

`void forward(double len);`
　　現在の向きに len だけ移動します。

`void backward(double len);`
　　現在の向きと逆向きに線を描画せずに len だけ移動します。

`void turn(double angle);`
　　右方向に angle 度回転します。 （左方向に回転する時は負の数を渡します。）

`void penUp(void);`
　　ペンを上げます。（この状態で移動しても線を描きません。）

`void penDown(void);`
　　ペンを下げます。（この状態で移動すると線を描きます。）

`void direction(double dir);`
　　現在の向きに関係なく、direction 度の方向を向きます。

`void go(double x, double y);`
　　現在の位置に関係なく、(x, y)に移動します。この間、ペンの状態に関係なく、 線は描きません。

`void center(void);`
　　最初の位置（中央）に戻ります。

`double getX(void);`
　　現在の“亀”の現在の x 座標を返します。

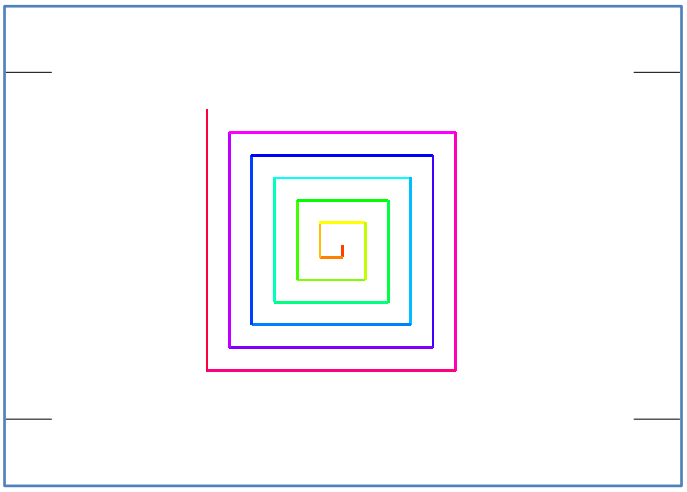
`double getY(void);`
　　現在の“亀”の現在の y 座標を返します。

`double getAngle(void);`
　　現在の“亀”の現在の向き（単位: 度）を返します。

タートルグラフィックスのサンプルプログラム

```
#include "svg.h"                    /* ライブラリ用のヘッダファイル */

int main(void){
    int i;

    start();                        /* 最初に必要 */
    rulers();                       /* 四隅の線   */

    for(i=0; i<24; i++) {
        stroke(hsb360(i*15, 100, 100));
        forward(i*5);
        turn(90);
    }

    finish();                       /* 最後に必要 */
    return 0;
}
```

上記のプログラムが生成する図形



座標系変換関係

`void translate(double x, double y);`
　　以降の描画に使用する座標系を (x, y)だけ平行移動します。
`void rotate(double t);`
　　以降の描画に使用する座標系を原点を中心に t（単位ラジアン）回転します。
`void scale(double sx, double sy);`
　　以降の描画に使用する座標系を縦方向に sx, 横方向に sy だけ拡大します。
`void pushMatrix(void);`
　　現在、使用している座標系を保存します。
`void popMatrix(void);`
　　最近 `pushMatrix` で保存した座標系を取り出します。
`void resetMatrix(void);`
　　座標系を最初のものに戻します。

コンパイルと実行の仕方

Windows 上の Borland C コンパイラを使用して、foo.c というソースファイルをコンパイルする場合

1. カレントディレクトリに 作成した foo.c とライブラリのソースファイル svg.c、ライブラリのヘッダファイル svg.h を置きます
2. `bcc32 foo.c svg.c`
   これで foo.exe というファイルができます
3. `foo > foo.svg`
4. 生成された foo.svg を Firefox バージョン 3 以上で開きます

印刷の仕方

Firefox バージョン 3 以上で印刷します。 「ファイル」－「ページ設定」で「書式とオプション」の「書式」の「印刷方向」を「横」、「余白とヘッダ/フッタ」の「余白」をすべて「0」、 「ヘッダとフッタ」をすべて「なし」に設定して下さい。 文庫版は A4（拡大/縮小 100%）で、印刷します。
プリンタの方の設定でも、印刷の向きを「横」に設定して下さい。