

Easy and Gratifying Graphics library for X11

## EGGX / ProCALL version 0.78

Chisato Yamauchi \*

Dec.19,2002

### 1 はじめに

#### 1.1 EGGX / ProCALL とは

EGGX / ProCALL は「究極の簡単さ」を目指して作り上げた簡素なグラフィックスライブラリです。C 言語か FORTRAN から呼出す事ができ、C の関数群を “EGGX” (§2.1), FORTRAN のサブルーチン群を “ProCALL” (§3.1) と呼び、機能的にはほぼ N88-BASIC 相当のグラフィックス機能を持ちます。とにかく「簡単」に使えて、「8 ビットマシンの BASIC のような楽しさ」をこれからプログラミングを始める方々に伝えることができれば、と考えて作成したもので、初等プログラミング教育に良い道具となる事でしょう。また、EGGX / ProCALL は X11 関連のライブラリは Xlib しか使っていませんから、インストールも容易です。

ユーザ関数の簡単さは常識破りです。画面サイズだけを引数とする「たった 1 つの関数」を呼ぶだけで、X Window 上でウィンドウを開いて、即グラフィックス関数を扱うことができます。ウィンドウが隠れた場合の再描画はライブラリ側がやってくれますから、ユーザはめんどろなイベントを気にせず、描画関数を並べるだけで簡単に絵を描くことができます。数値計算のコードに若干の描画関数を挟み込んで、デバッグ時など、必要な時だけ計算させながらその様子をモニターする、といった事も容易です。任意のサイズで複数のグラフィックス用ウィンドウをオープンし、24 ビットでのカラー指定が可能な線、点、多角形、円などの描画、フォントセットの描画（半角・全角混在の文字列もそのまま描画可能）、数値計算の可視化に必要な種々の矢印の描画、カラーバーの生成（約 50 種類）、24 ビット画像の転送、関数 `ggetch()` を使ったキー入力読み込み（N88-BASIC の `INKEY$` に相当）、関数 `ggetxpress()` を使ったマウス・キー入力読み込み、レイヤ機能（PC-9801 の V-RAM の 2 プレーンに相当）によるスムーズなアニメーションが可能です。

もうおわかりかと思いますが、24 ビットカラーが使えて、アニメーションができるとなれば、数値計算の可視化に必要な基本要素は十分に整っているというわけです。あとは、ユーザのプログラミング次第で、いくらかでも派手な画像を作る事が可能です。

また、`netpbm`<sup>1)</sup>、`ImageMagick`<sup>2)</sup> の `convert` コマンドを通して多くのフォーマットに画像を連続して保存するための関数を用意しているため、大量のコマ数の画像ファイルの生成が瞬時に行え、`gif` アニメーションや `mpeg` ムービーの作成も簡単です。

商用の X サーバ（例えば `Xsun` や `SGI` の X 端末）のように複数の `Visual` を持つ X サーバにも対応し、デフォルトが `PseudoColor` でも EGGX/ProCALL では `TrueColor Visual` でウィンドウを開くようになっています。Sun のワークステーションのように商用 X サーバ+`Pseudo Visual` なアプリケーションをベースとした環境でもスムーズに導入できます。

\*) <http://phe.phyas.aichi-edu.ac.jp/~cyamauch/>

1) <http://sourceforge.net/projects/netpbm/>

2) <http://www.imagemagick.org/>

## 1.2 C 言語を使おう

このライブラリを作り始めた当初は、NEC のパーソナルコンピュータ・PC-9801 上で動作する「Pro-FORTRAN」という GKS、カルコンパチブルのグラフィックスサブルーチンが扱える FORTRAN コンパイラと互換となるように作成していました。格好悪い「ProCALL」という名前はこの Pro-FORTRAN が由来です。しかし現在では、FORTRAN は廃れた言語となってしまいました。ソフトウェア開発の主流は、柔軟性に富む C、C++ です。また、数値計算の分野では、いまだに FORTRAN が多く使われているようですが、コーディングレベルでのアルゴリズムの考案を伴った新しいスキームの開発は、FORTRAN の言語仕様では難しいと予測されます (FORTRAN9x という新しい FORTRAN がありますが、大半の利用者の頭脳は FORTRAN77 から進化していないので、FORTRAN77 が実質の言語仕様になっていると思われます)。数値計算技術の未来は、C、C++ が支えていくことでしょう。

EGGX / ProCALL も C から使った方が、いくつかの点において好都合です (もちろん、EGGX / ProCALL は C で書いています)。EGGX の関数には、ProCALL では使えないいくつかの便利な機能があります。今、プログラミングを始めたばかりの方々は、C 言語をターゲットにする事をお勧めします。

EGGX では、有名なウィンドウマネージャである AfterStep、WindowMaker の本格的なアプレットがきわめて簡単に作成できます。EGGX の拡張関数で、アプレット用に初期化するように設定すると、ドックに入れるためのウィンドウ属性をライブラリ側で設定します。例えば、デジタル時計程度のものであれば、わずか数十行で作れてしまうほどの簡単さを提供しています。

## 2 C 編

### 2.1 EGGX の標準関数一覧

ggetdisplayinfo	X サーバの情報 (depth, 画面サイズ) を取得する	§2.5.1
gopen	任意のサイズのグラフィックス用ウィンドウを開く	§2.5.2
gclose	任意のグラフィックス用ウィンドウを閉じる	§2.5.3
gcloseall	すべてのグラフィックス用ウィンドウを閉じ, X サーバと断線する	§2.5.4
gsetbgcolor	ウィンドウのバックグラウンドカラー (gclr での色) を指定する	§2.5.5
gsetborder	ウィンドウのホーダーと色を指定する	§2.5.6
winname	ウィンドウのタイトルを変更する	§2.5.7
window	座標系の変更	§2.5.8
layer	レイヤの設定を行う	§2.5.9
copylayer	レイヤのコピーを行う	§2.5.10
newpen	描画色 (16 色) の変更	§2.5.11
newcolor	描画色の変更 (X サーバの持つ色を直接指定)	§2.5.12
newrgbcolor	描画色の変更 (Red,Green,Blue の輝度を指定)	§2.5.13
makecolor	変数からカラーを生成する (カラーバー生成)	§2.5.14
tclr	端末画面の消去	§2.5.15
gclr	グラフィックス用ウィンドウの消去	§2.5.16
pset	点の描画	§2.5.17
line	直線の描画	§2.5.18
circle	中心座標, 半径を与えて円を描く	§2.5.19
drawarc	円の中心, 半径, 始点, 終点の角度を与えて円弧を描く	§2.5.20
fillarc	円の中心, 半径, 始点, 終点の角度を与えて円弧を塗り潰す	§2.5.21
drawrect	長方形の描画	§2.5.22
fillrect	長方形の領域を塗り潰す	§2.5.23
fillpoly	多角形を塗り潰す	§2.5.24
drawarrow	種々の矢印の描画	§2.5.25
drawstr	文字列の描画	§2.5.26
gsetfontset	フォントセット (日本語フォント) の指定	§2.5.27
drawsym	1 個のシンボルの描画	§2.5.28
putimg24	バッファに用意した画像をウィンドウに一括転送する	§2.5.29
saveimg	画像をコンバータ (netpbm,ImageMagick) を通してファイルに保存する	§2.5.30
ggetch	キーボードから 1 文字入力があるまで待ち, 入力された文字を返す	§2.5.31
ggetxpress	マウスのボタンクリック, キーボード入力があるまで待ち, その情報を返す	§2.5.32
msleep	ミリ秒単位で実行を延期する	§2.5.33

## 2.2 EGGX の拡張関数一覧

<code>gsetinitialattributes</code>	<code>gopen</code> での各ウィンドウ属性を指定する	§2.6.1
<code>gsetinitialbgcolor</code>	<code>gopen</code> での背景カラーを指定する	§2.6.2
<code>gsetinitialborder</code>	<code>gopen</code> でのウィンドウのボーダーを指定する	§2.6.3
<code>gsetinitialparsegeometry</code>	<code>x,y</code> の値を含む文字列から <code>gopen</code> でのウィンドウ出現座標を設定する	§2.6.4
<code>gsetinitialwinname</code>	<code>gopen</code> でのウィンドウ名, アイコン名, リソース名, クラス名を指定する	§2.6.5
<code>generatecolor</code>	変数からカラーを生成する (コントラスト, ブライトネス, 補正等が可能)	§2.6.6

## 2.3 基本的な使用方法

ソースの冒頭で,

```
#include <eggx.h>
```

と宣言します。C 標準関数と同じように EGGX の関数を使ってソースを書き, コンパイルは `egg` コマンドを用います。

例 `egg program.c -o program`

## 2.4 描画速度をアップする方法

`layer()` 関数 (§2.5.9) と `copylayer()` 関数 (§2.5.10) とを使って, 描画関数では常に裏画面に描きましよう。描き終わったら, 裏画面を表画面にコピーする (`copylayer()` 関数) か, 裏画面と表画面とを切り替えます (`layer()` 関数)。このようにする事で, かなり描画速度が改善されます。

## 2.5 EGGX の標準関数リファレンス

### 2.5.1 int ggetdisplayinfo( int \*depth, int \*root\_width, int \*root\_height )

**機能** X サーバの情報 (depth, 画面サイズ) を取得する

X サーバに接続し, depth, 画面サイズを調べます. \*depth は EGGX でウィンドウをオープンした時に使われる depth 値 (ピクセルに何ビット使っているか), すなわち 8 (PseudoColor: 256 色カラー), 16 (TrueColor: 65536 色カラー), 24 (TrueColor: 1677 万色カラー) といった値が代入されます. \*root\_width, \*root\_height はそれぞれ X サーバの画面のピクセルサイズが代入されます.

3 つの引数 depth, root\_width, root\_height それぞれについて値を取り出す必要がない場合は, NULL を与えてもかまいません.

関数の戻り値は, X サーバへの接続が成功した場合は 0, 失敗した場合は戻り値は -1 となります.

putimg24() 関数 (§2.5.29) を使用する時は, 必ず depth 値を調べるようにします.

**使用例** status = ggetdisplayinfo(&depth, NULL, NULL);

### 2.5.2 int gopen( int xsize, int ysize )

**機能** 任意のサイズのグラフィックス画面を開く

xsize, ysize にはそれぞれ横方向, 縦方向のドット数を指定します. 戻り値には EGGX で使用する, ウィンドウ番号が返ってきます.

**使用例** win = gopen(800,600);

### 2.5.3 void gclose( int wn )

**機能** グラフィックス用ウィンドウを閉じる

wn で指定されたウィンドウを閉じます.

**使用例** gclose(win);

### 2.5.4 void gcloseall( void )

**機能** すべてのグラフィックス用ウィンドウを閉じ, X サーバと断つ

すべてのウィンドウを閉じ, X サーバと断線します.

**使用例** gcloseall();

### 2.5.5 void gsetbgcolor( int wn, const char \*argsformat, ... )

**機能** ウィンドウの背景色を変更する

wn で指定されたウィンドウの背景色 (gclr() 関数 (§2.5.16) で初期化される色) を変更します. argsformat には X サーバの rgb.txt<sup>3)</sup> に設定されている色か, "#c0c0ff" のように, 16 進数の Red, Green, Blue で指定します.

**使用例** gsetbgcolor(win, "white");

<sup>3)</sup> rgb.txt は UNIX 系の OS なら /usr/X11R6/lib/X11/ などにあります.

## 2.5.6 void gsetborder( int wn, int width, const char \*argsformat, ... )

**機能** ウィンドウのホーダーと色を指定する

wn で指定されたウィンドウのボーダー幅とボーダーカラーを変更します。この関数でのボーダーの指定は、gsetinitialattributes() 関数 (§2.6.1) で OVERRIDE のウィンドウ属性を指定した時だけ機能します。

**使用例** gsetborder(win,1,"white") ;

## 2.5.7 int winname( int wn, const char \*argsformat, ... )

**機能** ウィンドウのタイトルを変更する

gopen() で開いたウィンドウはデフォルトではユーザの実行ファイル名がウィンドウのタイトルとして設定されますが、このタイトルは自由に変更できます。最初の引数 wn はウィンドウ番号で、2 つめの引数 argsformat 以降は printf() 関数の引数と同じ形式のフォーマットです。使用例のように、変数の値を表示するような使い方もあります。

戻り値は設定したウィンドウタイトルの文字列の長さです。

**使用例** winname(win,"ギコ x=%f y=%f",x,y) ;

## 2.5.8 void window( int wn, float xs, float ys, float xe, float ye )

**機能** 座標系の変更

wn で指定したウィンドウの座標系を変更します (実際のグラフィックスエリアの大きさが変わるわけではありません)。グラフィックス画面の座標は、デフォルトでは左下が (0.0, 0.0) で右上が (xsize-1.0, ysize-1.0) になっています。window() 関数でこの座標系を左下を (xs, ys)、右上を (xe, ye) に変更できます。

下の使用例では、左上を (0.0, 0.0)、右下を (799.0, 599.0) に変更します。

**使用例** window(win, 0.0, 599.0, 799.0, 0.0) ;

## 2.5.9 void layer( int wn, int lys, int lyw )

**機能** レイヤの設定をする

EGGX ではグラフィックス用ウィンドウ毎に 2 枚のレイヤを持ち、表示するレイヤと書き込むレイヤを独立に指定できます。wn にはウィンドウ番号を指定し、lys には表示するレイヤ番号、lyw には書き込むレイヤ番号を 0 か 1 で指定します。

現在表示しているレイヤに対して (lys == lyw の場合に) 連続して描画関数を実行すると、良い描画パフォーマンスが得られません。高速な描画が必要な場合には、現在表示していないレイヤに対して描画し、最後に表示レイヤを layer() 関数で切り替えるか、copylayer() 関数 (§2.5.10) で描画レイヤの画像を表示レイヤにコピーするようにします。

デフォルトでは layer(wn,0,0) の状態となっています。

**使用例** layer(win,0,1) ;

## 2.5.10 void copylayer( int wn, int lysrc, int lydest )

**機能** レイヤのコピーをする

wn のウィンドウ番号の、レイヤ lysrc の画像をレイヤ lydest にそのままコピーします。このコピーは瞬時に行われるため、アニメーションの再生に使うことができます。

**使用例** copylayer(win,1,0) ;

## 2.5.11 void newpen( int wn, int cn )

**機能** 描画色の変更

wn で指定したウィンドウでの描画色を変更します。cn と色との関係は以下の通りです。

0:黒 1:白 2:赤 3:緑 4:青 5:シアン 6:マゼンタ 7:黄

8:DimGray 9:Gray 10:red4 11:green4 12:blue4 13:cyan4 14:magenta4 15:yellow4  
red4, green4... の“4”のつく色は、暗い赤、暗い緑...となっています。

デフォルトでは、白が指定されています。

**使用例** newpen(win,2) ;

## 2.5.12 void newcolor( int wn, const char \*argsformat, ... )

**機能** 描画色の変更

wn で指定したウィンドウでの描画色を変更します。argsformat には X サーバの rgb.txt<sup>4)</sup> に設定されている色か、“#c0c0ff”のように、16 進数の Red,Green,Blue で指定します。

**使用例** newcolor(win,"Violet") ;

## 2.5.13 void newrgbcolor( int wn, int r, int g, int b )

**機能** 描画色の変更

wn で指定したウィンドウでの描画色を変更します。r, g, b にはそれぞれ Red, Green, Blue の輝度を 256 段階の整数 (0 ~ 255) で指定します。

**使用例** newrgbcolor(win,255,127,0) ;

## 2.5.14 int makecolor( int cmode, float dmin, float dmax, float data, int \*r, int \*g, int \*b )

**機能** 変数からカラーを生成する (カラーバー生成)

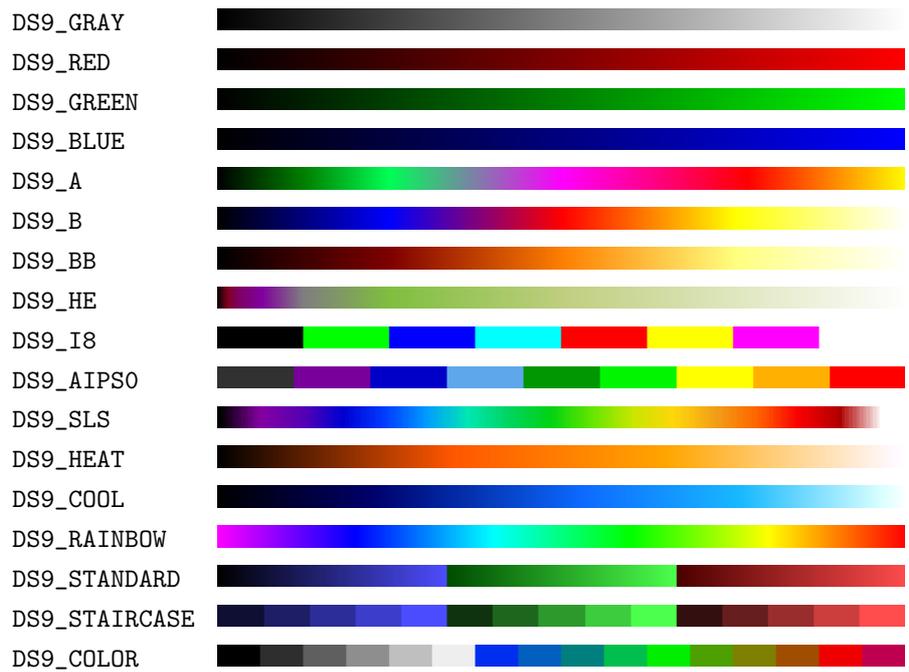
変数 data の値を使って、r, g, b それぞれに 256 段階の Red, Green, Blue のカラーを生成します。変数の最小、最大は dmin, dmax で指定します。この関数で得た、r, g, b の値は newrgbcolor() 関数 (§2.5.13) にそのまま与えて使う事ができます。

返り値は data が dmin, dmax の範囲内にあった場合は 0 を返し、dmin 未満であれば -1 を、dmax を越えていれば、1 を返します。

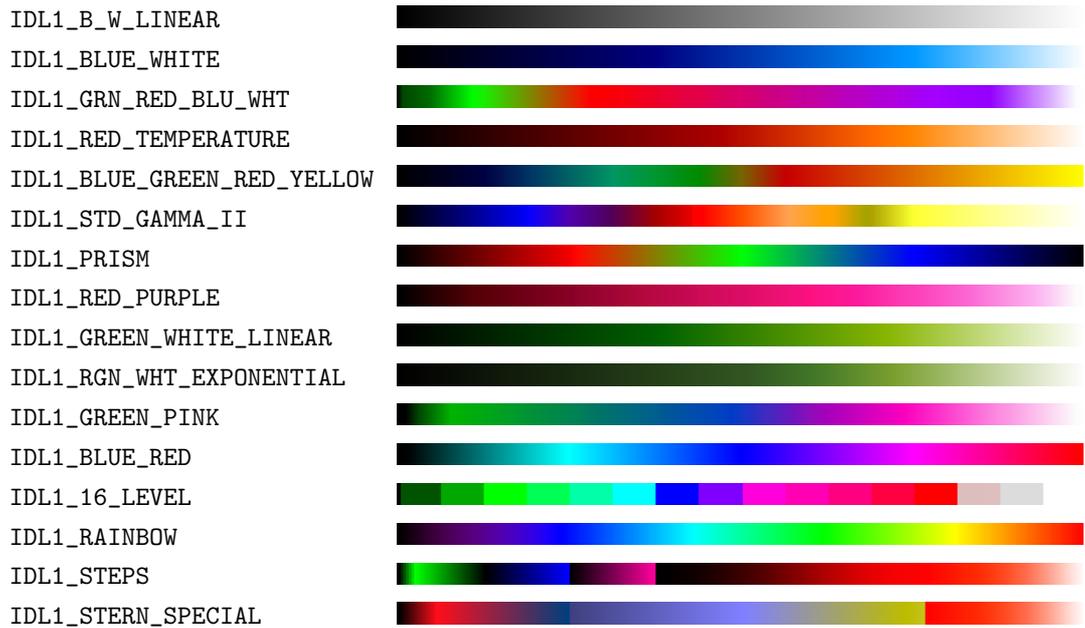
cmode はカラーパターンの番号で、以下に示す約 50 種類のカラーパターンが利用できます。DS9\_GRAY 等はマクロで、実際の値は 0 から始まる整数です。詳細は、eggx\_color.h をご覧ください。

以下は fits ビューア DS9 コンパチのカラーパターンです。

<sup>4)</sup> rgb.txt は UNIX 系の OS なら /usr/X11R6/lib/X11/ などにあります。



以下は可視化ソフトウェア IDL コンパチのカラーパターンです .



IDL2_HAZE	
IDL2_BLUE_PASTEL_RED	
IDL2_PASTELS	
IDL2_HUE_SAT_LIGHTNESS_1	
IDL2_HUE_SAT_LIGHTNESS_2	
IDL2_HUE_SAT_VALUE_1	
IDL2_HUE_SAT_VALUE_2	
IDL2_PURPLE_RED_WITH_STRIPEs	
IDL2_BEACH	
IDL2_MAC_STYLE	
IDL2_EOS_A	
IDL2_EOS_B	
IDL2_HARDCANDY	
IDL2_NATURE	
IDL2_OCEAN	
IDL2_PEPPERMINT	
IDL2_PLASMA	
IDL2_BLUE_RED	
IDL2_RAINBOW	
IDL2_BLUE_WAVES	
IDL2_VALCANO	
IDL2_WAVES	
IDL2_RAINBOW18	
IDL2__RAINBOW	
IDL2_ORBIT_VIEWER_COLOR	
IDL2_ORBIT_VIEWER_GRAY	

**使用例** `makecolor(DS9_SLS,v_min,v_max,v,&r,&g,&b) ;`

### 2.5.15 void tclr( void )

**機能** 端末のクリア

端末をクリアし、カーソルの位置をホームポジションに戻します。

**使用例** `tclr() ;`

### 2.5.16 void gclr( int wn )

**機能** wn で指定したグラフィックス用ウィンドウをクリア

wn で指定したグラフィックス用ウィンドウを gsetinitialbgcolor() 関数 (§2.6.2) あるいは gsetbgcolor() 関数 (§2.5.5) で指定した色で、ウィンドウをクリアします。gsetbgcolor() 関数、gsetinitialbgcolor() 関数での指定がない場合の色は黒となっています。

**使用例** `gclr(win) ;`

### 2.5.17 void pset( int wn,float x,float y )

**機能** 点の描画

wn で指定したウィンドウに点を描画します。

グラフィックス画面の座標は、デフォルトでは左下が (0.0, 0.0) で右上が (xsize-1.0, ysize-1.0) になっています。この座標は window() 関数 (§2.5.8) で変更が可能です。

**使用例** pset(win,gx,gy) ;

### 2.5.18 void line( int wn,float x,float y,int mode )

**機能** 直線の描画

wn で指定したウィンドウに直線を描画します。mode に PENDOWN を指定すると以前 line() 関数が呼ばれた点から、(x,y) へ直線を引きます。mode に PENUP を指定すると (x,y) を line() 関数の初期位置に設定します。mode=PENDOWN でペンを下ろして描画、mode=PENUP でペンを上げて移動と考えるとわかりやすいでしょう。

また、mode=PSET の場合は (x,y) に点を描き、ペンの位置が更新されます。

**使用例** line(win,gx,gy,PENDOWN) ;

### 2.5.19 void circle( int wn, float xcen, float ycen, float xrad, float yrad )

**機能** 中心座標、半径を与えて円弧を描く

wn で指定したウィンドウに、(xcen,ycen) を中心に横方向の半径 xrad、縦方向の半径 yrad の円を描きます。

**使用例** circle(win,50.0,60.0,30.0,40.0) ;

### 2.5.20 void drawarc( int wn, float xcen, float ycen, float xrad, float yrad, float sang, float eang, int idir )

**機能** 円の中心、半径、始点、終点の角度を与えて円弧を描く

wn で指定したウィンドウに、(xcen,ycen) を中心に横方向の半径 xrad、縦方向の半径 yrad の円弧を描きます。sang は開始角、eang は終了角で、度で与えます。idir は円弧を描く方向で 1 で左廻り、-1 で右廻りとなります。

**使用例** drawarc(win,50.0,60.0,30.0,40.0,-10.0,-170.0,-1) ;

### 2.5.21 void fillarc( int wn, float xcen, float ycen, float xrad, float yrad, float sang, float eang, int idir )

**機能** 円の中心、半径、始点、終点の角度を与えて円弧を塗り潰す

wn で指定したウィンドウで、(xcen,ycen) を中心に横方向の半径 xrad、縦方向の半径 yrad の円弧を塗り潰します。sang は開始角、eang は終了角で、度で与えます。idir は円弧を描く方向で 1 で左廻り、-1 で右廻りとなります。

**使用例** fillarc(win,50.0,60.0,30.0,40.0,-10.0,-170.0,-1) ;

### 2.5.22 void drawrect( int wn, float x, float y, float w, float h )

**機能** 長方形の描画

wn で指定したウィンドウに、頂点 (x,y) から正の方向に幅 w , 高さ h の長方形を描きます。

**使用例** drawrect(win,50.0,60.0,30.0,20.0) ;

### 2.5.23 void fillrect( int wn, float x, float y, float w, float h )

**機能** 長方形の領域を塗り潰す

wn で指定したウィンドウで、頂点 (x,y) から正の方向に幅 w , 高さ h の長方形の領域を塗り潰します。

**使用例** fillrect(win,50.0,60.0,30.0,20.0) ;

### 2.5.24 void fillpoly( int wn, float x[], float y[], int n, int i )

**機能** 多角形を塗り潰す

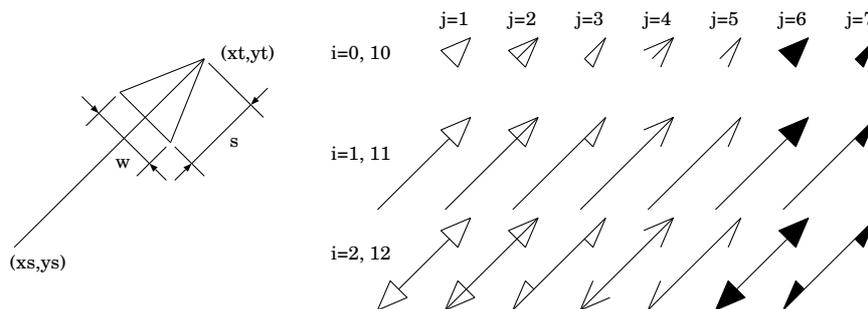
wn で指定したウィンドウで、多角形の領域を塗り潰します。x , y は n 個の実数の一次元配列で、x[0] ~ x[n-1] , y[0] ~ y[n-1] に多角形の各点の座標を入れておきます。i は塗り潰す時の形状で通常は 0 を、凸多角形の場合は 1 を指定します。

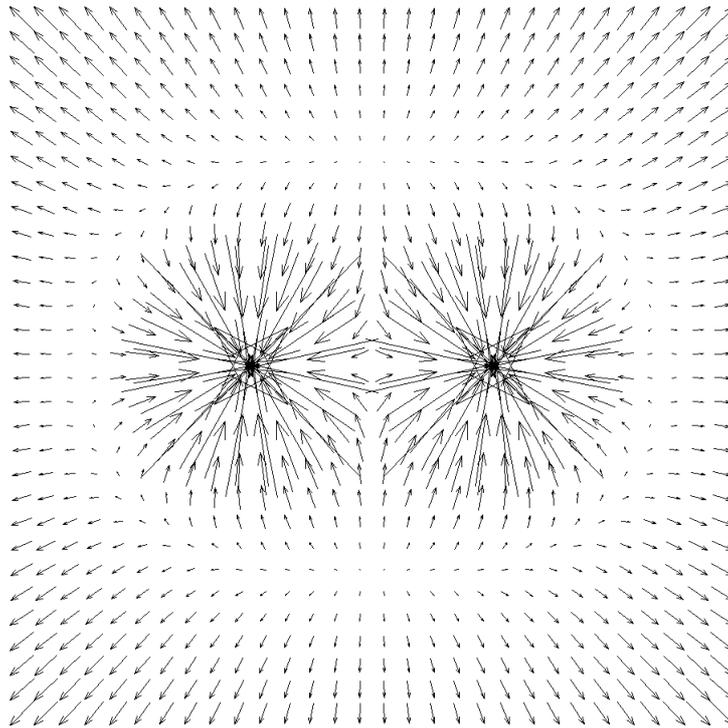
**使用例** fillpoly(win,plx,ply,5,0) ;

### 2.5.25 void drawarrow( int wn, float xs, float ys, float xt, float yt, float s, float w, int 10\*i+j )

**機能** 種々の型の矢印を描く

wn で指定したウィンドウに (xs,ys) から (xt,yt) に向かって矢印を描きます。矢印の形状は以下の図の通りで、s と w は実数で指定します。i が 0~2 の場合には w,s はドット数を、i が 10~12 の場合には w,s は矢印の長さに対する割合で 0.0~1.0 の値を指定します。





i=11 での矢印の描画例

**使用例** `drawarrow(win,gx0,gy0,gx1,gy1,0.3,0.2,114) ;`

## 2.5.26 int drawstr( int wn, float x, float y, int size, float theta, const char \*argsformat, ... )

**機能** 文字列の描画

wnで指定したウィンドウに、文字列を座標(x,y)から描きます。sizeは文字の大きさを、ドット単位で指定します。thetaは文字列の回転を指定する引数ですが、現バージョンでは機能しません。文字列はargsformatに与えますが、この引数以降はprintf()関数の引数と同様のフォーマットになっていますので、使用例のように変数の値などを描く事もできます。

文字のサイズsizeは1~24の範囲で指定できます。sizeと実際のフォントとの関係は以下の表のようになっています。この場合、文字は半角英数字のみ描画できます。

2バイト文字(漢字)を描画する場合は、sizeにはFONTSETを指定します。この場合のフォントの指定は、gsetfontset()関数(§2.5.27)を利用します。gsetfontset()でのフォント指定がない場合は、デフォルトの14ドットのフォントセットで描画されます。

この関数の返り値は描いた文字列の長さです。

1~7 : 5×7	8 : 5×8	9 : 6×9	10~11 : 6×10	12 : 6×12
13 : 7×13	14~15 : 7×14	16~19 : 8×16	20~23 : 10×20	24 : 12×24

**使用例** `drawstr(win,gx,gy,16,0,"velocity v=%f",v) ;`

**使用例** `drawstr(win,gx,gy,FONTSET,0,"日本語の描画も OK!") ;`

### 2.5.27 int gsetfontset( int wn, const char \*argsformat, ... )

**機能** フォントセット (日本語フォント) の指定

wn で指定したウィンドウで描くフォントセットを指定します。文字の描画は drawstr() 関数を利用します。

フォントセット名は argsformat 以降で指定します。argsformat 以降は printf() 関数の引数と同一のフォーマットになっているため、文字を拡大・縮小する時など、使用例のように引数を与えると大変便利です。

返り値は、argsformat 以降で指定したフォントセットが取得できた場合は 0、代替フォントで取得できた場合は 1、フォントセットの取得に失敗した場合は -1 となります。

**使用例** f=gsetfontset(win, "-kochi-gothic-medium-r-normal--%d-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*", fsize) ;

**使用例** gsetfontset(win, "-adobe-helvetica-medium-r-\*-\*-\*-\*12-\*-\*-\*-\*-\*-\*-\*-\*-\*-\*iso8859-1,"  
"-\*-\*fixed-medium-r-normal--14-\*") ;

### 2.5.28 void drawsym( int wn, int x, int y, int size, int nsym )

**機能** センターシンボルの描画

wn で指定したウィンドウにセンターシンボルを座標 (x,y) に描きます。size はシンボルの大きさをドット単位、nsym でシンボルの種類を指定します。

nsym とシンボルの関係は、だいたい次のようなものとなっています。

1:・ 2:+ 3:\* 4: 5:x 6:Y 7: 8: 9: 10:

**使用例** drawsym(win,gx,gy,16,2) ;

### 2.5.29 int putimg24( int wn, float x, float y, int width ,int height, unsigned char \*buf )

**機能** バッファに用意した画像をウィンドウに一括転送する

wn で指定したウィンドウの座標 (x,y) に buf に用意した、幅 width、高さ height の画像を一括転送します。

バッファ buf には、0,Blue,Green,Red の順に 4 つずつ、水平方向に走査しながら画像の上から下へ用意します (buf[0]=0x000, buf[1]=blue[0], buf[2]=green[0], buf[3]=red[0], といった具合です)。バッファには 0x000 ~ 0x0ff の範囲の輝度を与えておきます。

なお X サーバの depth が 16 以上でない場合は何もせず、返り値は -1 となります。それ以外の場合は 0 を返します。X サーバの depth は ggetdisplayinfo() 関数 (§2.5.1) で調べる事ができます。

**使用例** putimg24(win,gx,gy,640,400,buffer) ;

### 2.5.30 int saveimg( int wn, int ly, float xs, float ys, float xe, float ye, char \*conv, int nd, const char \*argsformat, ... )

**機能** 画像をコンバータ (netpbm,ImageMagick) を通してファイルに保存する

ウィンドウ番号 wn、レイヤ番号 ly の (xs,ys) から (xe,ye) の範囲をコンバータ (netpbm,ImageMagick) を通してファイルに保存します。conv には ppm 形式から各画像フォーマットに変換するコンバータのコマンドラインを指定します。例えば、netpbm を利用して png 形式に保存する場合は、"pnmtopng" とします。ImageMagick を利用する場合は、"convert" とします。もちろん、オプションスイッチも含める事もでき、

"pnmtops -scale 0.125",あるいは"convert -compress ZIP"といった指定も可能です。次のndは減色パラメータで、R,G,B 1チャンネルあたりの色の段階数を指定します。ndは簡単な図形では16くらいで十分ですが、多くの色を使っている場合は最大値の256を指定してください。

ファイル名はargsformat以降で指定します。argsformat以降はprintf()関数の引数と同一のフォーマットになっているため、使用例のようにファイル名に変数の値を含む、といった事も可能です。このようにすると動画作成時に大変便利です。

ppmから他の画像フォーマットに変換するnetpbmのコマンドの例を挙げておきます。各コンバータの使用方法に関しては端末からman コンバータ名とタイプする事で調べることができます。ImageMagickのconvertコマンドを利用する場合は、ファイル名のサフィックス(ドット以降の文字列、例えば.jpg,.png,.eps2など)から保存フォーマットが決定されます。

画像フォーマット	コンバータ名	画像フォーマット	コンバータ名
AutoCAD DXB	ppmtoacad	Motif UIL icon	ppmtouil
windows bitmap	ppmtobmp	Windows .ico	ppmtowinicon
Berkeley YUV	ppmtoeyuv	XPM format	ppmtoxpm
GIF	ppmtogif	Abekas YUV	ppmtoyuv
NCSA ICR graphics	ppmtoicr	YUV triplets	ppmtoyuvsplit
IFF ILBM	ppmtoilbm	DDIF	pnmtoddif
Interleaf image	ppmtoleaf	FIASCO compressed	pnmtofiasco
HP LaserJet PCL 5 Color map	ppmtolj	FITS	pnmtofits
Mitsubishi S340-10 printer	ppmtomaps	JBIG	pnmtobjbig
Atari Neochrome .neo	ppmtomitsu	JFIF ("JPEG") image	pnmtjpeg
PPC Paintbrush	ppmtoneo	Palm pixmap	pnmtopalm
portable graymap	ppmtopcx	plain (ASCII) anymap	pnmtoplainpnm
Atari Degas .pil	ppmtopgm	Portable Network Graphics	pnmtopng
Macintosh PICT	ppmtopi1	PostScript	pnmtops
HP PaintJet	ppmtopict	Sun raster	pnmtorast
HP PaintJet XL PCL	ppmtopj	RLE image	pnmtorle
X11 "puzzle"	ppmtopjxl	sgi image	pnmtosgi
three portable graymaps	ppmtopuzz	Solitaire image recorder	pnmtosir
DEC sixel	ppmtorgb3	TIFF file	pnmtotiff
TrueVision Targa	ppmtosixel	CMYK encoded TIFF	pnmtotiffcmk
	ppmtotga	X11 window dump	pnmtowxd

もちろん、コンバータはすでにインストールされている必要があり、インストールされていない場合は指定できません<sup>5)</sup>。ここに挙げた以外のコンバータも、標準入力からppm形式で入力し、標準出力から変換データを出力するものであれば使用する事ができます。

コンバータを通さずに直接ppm形式で保存する場合はconvには""と指定してください。ただし、ppm形式は非圧縮のテキストデータなので、巨大なファイルとなります。

この関数ではXサーバから画像データを転送し、保存するわけですが、ネットワークが遅かったりすると非常に時間がかかります。その事を考慮してEGGXでは子プロセスを起動して、バックグラウンドでXサーバからの画像の転送・ディスクへの書き込みを行うようにしています。したがってユーザプログラムはsaveimg();の後すぐに他の作業ができるようになります。ただし、この子プロセスが終わるまではXサーバでの描画等ができないので、もしsaveimg();のすぐ後にEGGXの描画関数などを呼んだ場合は画像の転送・保存が完了するまで待たされる事になります。

なお、saveimgを使っている場合はプログラムは必ずgcloseall()関数 (§2.5.4) を呼んでから終了するようにしてください。

返り値は、子プロセスの起動に失敗した場合は-1、それ以外の場合は0となります。

**使用例** saveimg(win,0,0.0,0.0,639.0,399.0,"pnmtopng",256,"img%d.png",i);  
png形式に保存する例です。gif形式の場合はconvは"ppmtogif"にします。

<sup>5)</sup> netpbmは<http://sourceforge.net/projects/netpbm/>、ImageMagickは<http://www.imagemagick.org/>などで配布されています。

**使用例** `saveimg(win,0,0.0,0.0,639.0,399.0,"pnmtops -noturn -rle",256,"figure.ps") ;`

**使用例** `saveimg(win,0,0.0,0.0,639.0,399.0,"convert -compress ZIP",256,"figure.eps2") ;`

PostScript 形式に保存する例です。netpbm の pnmtops では RunLength 圧縮 (可逆圧縮) をサポートしています。上記のように「-rle」をつける事で画質を損うことなくファイルサイズを小さくできます。

ImageMagick では RunLength の他、LZW、ZIP(使用例) などのアルゴリズムでより効率良く圧縮したビットマップイメージを Postscript Level 2 ファイルに埋め込む事ができます。ただし、ImageMagick の Postscript Level 2 の出力は 5.3.3 以降で改悪され、一般的な Postscript インタープリタ (Adobe Postscript 3 など) では読めません。ImageMagick-5.3.2 の利用をお勧めします。

### 2.5.31 int ggetch( int wn )

**機能** キーボードから 1 文字入力があるまで待ち、入力された文字を返す

wn で指定したウィンドウにキー入力があるまで待ち、入力されたらその文字コードを返します。端末からの 1 文字入力 fgetc(stdin) と似ていますが、1 つのキー入力があるとすぐに関数から戻る (改行まで待たない) という点と、0x001 ~ 0x01f, 0x07f の「特殊キー」と「Ctrl+アルファベットキー」の入力を拾う点が異なります。

次の表で 16 進の文字コードを知る事ができます。イタリック体の数字が 16 進 2 桁の上位を表します。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0		Home	PageUp	Pause		End	PageDown		BackSpace	Tab				Enter		
1												Esc				
2	Space	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	Delete

例えば、「a」は 0x061、「A」は 0x041 となります。

0x001 ~ 0x01a の中にはいくつか特殊キーが含まれていますが、この区間は「Ctrl+アルファベットキー」のコードと共有となります。例えば、「BackSpace」と「Ctrl+H」のコードはどちらも 0x008 となります。0x001 ~ 0x01a で空白になっている個所は、「Ctrl+アルファベットキー」にのみコードが割り当てられています。

**使用例** `key=ggetch(win) ;`

### 2.5.32 int ggetxpress( int wn, int \*button, float \*x, float \*y )

**機能** マウスからのボタンクリック、キーボードからの入力があるまで待ち、その情報を返す

wn で指定したウィンドウでマウスのボタンクリック、キータイプがあるまで待ち、入力があれば、その情報を返します。

この関数の戻り値はマウスのボタンクリックの場合は ButtonPress、キーボードからの入力の場合は KeyPress となります。

マウスのボタンクリックの場合、\*button にはクリックされたボタンの番号 (1,2,3,...)、クリックされた時のウィンドウ wn 上でのマウスポインタの座標が\*x, \*y に代入されます。

キー入力の場合は、キーコードが\*button に返ります。キーコードは ggetch() 関数 (§2.5.31) の戻り値と同一です。

**使用例** `type=ggetxpress(win,&b,&x,&y) ;`

### 2.5.33 void msleep( unsigned long msec )

**機能** ミリ秒単位で実行を延期する

msec ミリ秒の間、プログラムの実行をなにもせずに待ちます。アニメーション速度の調整に利用できます。

**使用例** msleep(100) ;

## 2.6 EGGX の拡張関数リファレンス

以下でとりあげる関数は、より細かい制御を行なったりするためのもので、中級者以上向けのやや C 言語に習熟していないと利用が難しい関数もあります。なお、これらの関数はそれぞれに対応する FORTRAN のサブルーチンは用意していません。

### 2.6.1 void gsetinitialattributes( int flags, long att )

**機能** gopen での各ウィンドウ属性を指定する

これから gopen() 関数で開くウィンドウの各種属性を指定します。gsetinitialattributes() 関数で一旦ある属性を指定すると、以降 gopen() 関数で開くウィンドウすべてにその指定が反映されます。flags には ENABLE(有効) か DISABLE(無効) を指定し、att には以下に示す各種属性を指定します。

- OVERRIDE

X における OverrideRedirect 属性を指定します。OverrideRedirect 属性で開いたウィンドウは、ウィンドウマネージャの介入をうけず、ウィンドウに枠が付きません。また、ウィンドウを開いた時は常に最前面に表示されます。このモードはアプリケーションのバナー表示などに用いられることがあります。

- AUTOREDRAW

ウィンドウが他のウィンドウによって隠され、露出した時の再描画を子プロセスが自動でおこないません。デフォルトではこの属性は ENABLE になっています。時計アプリケーションのように、ある周期で必ず再描画するプログラムではこの属性を DISABLE にするとメモリ使用量が抑えることができます。

- DOCKAPP

AfterStep, WindowMaker のアプレットとなるための各種設定をおこないます。

**使用例** gsetinitialattributes(ENABLE,DOCKAPP) ;

### 2.6.2 void gsetinitialbgcolor( const char \*argsformat, ... )

**機能** gopen での背景カラーを指定する

これから gopen() 関数で開くウィンドウの背景色 (gclr() 関数 (§2.5.16) で初期化される色) を指定します。argsformat には X サーバの rgb.txt<sup>6)</sup> に設定されている色が、"#c0c0ff"のように、16 進数の Red,Green,Blue で指定します。

**使用例** gsetinitialbgcolor("white") ;

---

<sup>6)</sup> rgb.txt は UNIX 系の OS なら /usr/X11R6/lib/X11/ などにあります。

### 2.6.3 void gsetinitialborder( int width, const char \*argsformat, ... )

**機能** gopen でのウィンドウのボーダーを指定する

これから gopen() 関数で開くウィンドウのボーダー (枠) の幅をカラーを指定します。引数 width にはボーダーの幅をドット単位で指定し、負の値を指定した場合には設定を変更しません。argsformat にはボーダーの色を指定し、NULL の場合には設定変更をおこないません。EGGX/ProCALL のデフォルトではボーダー幅は 0、ボーダーカラーは Black を設定しています。

一般的に、ウィンドウのボーダーはウィンドウマネージャによって再設定されるので、ここで設定した値は無効になります。ただし、OverrideRedirect 属性なウィンドウの場合は、この設定が反映されます。OverrideRedirect 属性については、gsetinitialattributes() 関数 (§2.6.1) の解説を参照してください。

**使用例** gsetinitialborder(1,"White") ;

### 2.6.4 void gsetinitialparsegeometry( const char \*argsformat, ... )

**機能** x,y の値を含む文字列から gopen でのウィンドウの出現座標を設定する

これから gopen() 関数で開くウィンドウの出現位置を文字列 argsformat から決定します。X11 のクライアントでは標準的な -geometry に続くコマンドオプション、例えば "+100-200" ような文字列を argsformat に与えることができます。

使用例のようにすると、整数値を与えることもできます。

**使用例** gsetinitialparsegeometry("%+d%+d",-30,40) ;

### 2.6.5 void gsetinitialwinname( char \*storename, char \*iconname, char \*resname, char \*classname )

**機能** gopen でのウィンドウ名、アイコン名、リソース名、クラス名を指定する

これから gopen() 関数で開くウィンドウのウィンドウ名、アイコン名、リソース名、クラス名を引数 storename, iconname, resname, classname で指定します。

デフォルトでは、ウィンドウ名はアイコン名はユーザプログラムのコマンド名から決定されます。それぞれの引数に NULL を与えると、それぞれをデフォルトの設定に戻します。

**使用例** gsetinitialwinname("AyuClock","AyuClock","ayuclock","AyuClock") ;

### 2.6.6 int generatecolor( struct color\_prms \*p, float dmin, float dmax, float data, int \*r, int \*g, int \*b )

**機能** 変数からカラーを生成する (コントラスト, ブライトネス, 補正などが可能)

変数 data の値を使って、r,g,b それぞれに 256 段階の Red,Green,Blue のカラーを生成します。変数の最小, 最大は dmin,dmax で指定します。この関数で得た、r,g,b の値は newrgbcolor() 関数 (§2.5.13) にそのまま与えて使う事ができます。

戻り値は data が dmin,dmax の範囲内にあった場合は 0 を返し、dmin 未満であれば -1 を、dmax を越えていれば、1 を返します。

現在の EGGX では、構造体 struct color\_prms は次のようなメンバ構成になっていますが、将来拡張する可能性がありますので、変数宣言時に初期値を代入するコードは書かない事をお勧めします。

```

struct color_prms {
    int colormode ;
    int flags ;
    float contrast ;
    float brightness ;
    float gamma ;
    int seplevel ;
    void *ptr ;
    void (*function)( float,void *,float,float,float,float *,float *,float * ) ;
};

```

colormode はカラーパターンの番号で、詳細は、makecolor() 関数 (§2.5.14) をご覧ください。

flags は、コントラスト contrast、ブライツネス brightness、ガンマ補正 gamma、カラーセパレーションレベル seplevel、ユーザ関数 function、のどれを有効にするかを示すためのフラグです。

フラグ	対応するメンバ	解説
C_REVERSE	-	カラーパターンをデータ data の大小と逆にする
CP_CONTRAST	contrast	コントラスト制御を有効化。(0 ≤ contrast ≤ 1)
CP_BRIGHTNESS	brightness	ブライツネス制御を有効化。(0 ≤ brightness ≤ 1)
CP_GAMMA	gamma	ガンマ補正を有効化。(0 ≤ gamma ≤ 1)
CP_SEPLEVEL	seplevel	カラーセパレーションレベルを有効化。(2 ≤ seplevel)
CP_FUNCTION	function	ユーザ関数呼出しを有効化。

seplevel を使うと、リニアに変化するカラーをデジタイズできます。例えば、DS9\_GRAY8 ではリニアに黒から白へ変化しますが、seplevel=10 に設定しておくで、10 色を使って段階的に変化するようになります。

function は generatecolor() が処理の最後に実行するユーザ関数です。function の引数はそれぞれ、data を Max=1.0,Min=0.0 で標準化した値、ptr、Red,Green,Blue の値 (in と out) となっています。この部分は EGGX のソースを見て理解できる方のみご利用ください。

**使用例**

```

struct color_prms cl ;
cl.colormode = DS9_RAINBOW ;
cl.flags = CP_CONTRAST | CP_BRIGHTNESS | CP_GAMMA ;
cl.contrast = 1.0 ;
cl.brightness = 0.0 ;
cl.gamma = 1.0 ;
:
generatecolor(&cl,zmin,zmax,zvalue,&cl_r,&cl_g,&cl_b) ;

```

## 3 FORTRAN 編

### 3.1 ProCALL のサブルーチン一覧

#### 3.1.1 ProCALL 標準サブルーチン

ggetdisplayinfo	X サーバの情報 (depth, 画面サイズ) を取得する	§3.4.1
gopen	任意のサイズのグラフィックス用ウィンドウを開く	§3.4.2
gclose	任意のグラフィックス用ウィンドウを閉じる	§3.4.3
gcloseall	すべてのグラフィックス用ウィンドウを閉じ, X サーバと断線する	§3.4.4
gsetbgcolor	ウィンドウのバックグラウンドカラー (gclr での色) を指定する	§3.4.5
newwindow	座標系の変更	§3.4.6
layer	レイヤの設定を行う	§3.4.7
copylayer	レイヤのコピーを行う	§3.4.8
newpencolor	描画色 (16 色) の変更	§3.4.9
newcolor	描画色の変更 (X サーバの持つ色を直接指定)	§3.4.10
newrgbcolor	描画色の変更 (Red, Green, Blue の輝度を指定)	§3.4.11
makecolor	変数からカラーを生成する (カラーバー生成)	§3.4.12
tclr	端末画面の消去	§3.4.13
gclr	グラフィックス用ウィンドウの消去	§3.4.14
pset	点の描画	§3.4.15
line	直線・点の描画	§3.4.16
drawcirc	中心座標と半径を与えて円を描く	§3.4.17
drawarc	円の中心, 半径, 始点, 終点の角度を与えて円弧を描く	§3.4.18
fillarc	円の中心, 半径, 始点, 終点の角度を与えて円弧を塗り潰す	§3.4.19
drawrect	長方形の描画	§3.4.20
fillrect	長方形の領域を塗り潰す	§3.4.21
fillpoly	多角形を塗り潰す	§3.4.22
drawarrow	種々の矢印の描画	§3.4.23
drawstr	文字列の描画	§3.4.24
gsetfontset	フォントセット (日本語フォント) の指定	§3.4.25
drawsym	1 個のシンボルの描画	§3.4.26
drawnum	実数の値を描く	§3.4.27
putimg24	整数配列に用意した画像をウィンドウに一括転送する	§3.4.28
saveimg	画像をコンバータ (netpbm, ImageMagick) を通してファイルに保存する	§3.4.29
ggetch	キーボードから 1 文字入力があるまで待ち, 入力された文字を返す	§3.4.30
ggetxpress	マウスのボタンクリック, キーボード入力があるまで待ち, その情報を返す	§3.4.31
selwin	互換サブルーチンで, どのウィンドウにアクセスするか指定する	§3.4.32

### 3.1.2 カルコンプ互換サブルーチン

plots	640 × 400 ドットのグラフィックス用ウィンドウを開く	§3.5.1
window	座標系の変更	§3.5.2
newpen	描画色の変更	§3.5.3
clsc	端末画面の消去	§3.5.4
clsx	グラフィックス画面の消去	§3.5.5
plot	直線・点の描画	§3.5.6
arc	円の中心、半径、始点、終点の角度を与えて円弧を描く	§3.5.7
circ1	中心座標と半径を与えて円を描く	§3.5.8
arohd	種々の矢印の描画	§3.5.9
symbol	文字列または 1 個のシンボルの描画	§3.5.10
number	実数の値を描く	§3.5.11

### 3.1.3 補助サブルーチン

msleep	ミリ秒単位で実行を延期する	§3.6.1
isnan	実数型変数が非数 (Not a Number) かどうか調べる	§3.6.2
rtoc	実数型変数を文字列に変換する	§3.6.3

## 3.2 基本的な使用方法

ソースについての特別な注意はありません。ソースを書いたら、コンパイルは `pro` コマンドを用います。

**例** `pro program.f -o program`

## 3.3 描画速度をアップする方法

`layer` サブルーチン (§3.4.7) と `copylayer` サブルーチン (§3.4.8) とを使って、描画サブルーチンでは常に裏画面に描きましょう。描き終わったら、裏画面を表画面にコピーする (`copylayer` サブルーチン) か、裏画面と表画面とを切り替えます (`layer` サブルーチン)。このようにする事で、かなり描画速度が改善されます。

## 3.4 ProCALL 標準サブルーチンのリファレンス

### 3.4.1 `ggetdisplayinfo(ndepth,nrwidth,nrheight)`

**機能** X サーバの情報 (`depth`, 画面サイズ) を取得する

X サーバに接続し、`depth`, 画面サイズを調べます。`ndepth` には ProCALL でウィンドウをオープンした時に使われる `depth` 値 (ピクセルに何ビット使っているか)、すなわち 8(PseudoColor : 256 色カラー), 16(TrueColor : 65536 色カラー), 24(TrueColor : 1677 万色カラー) といった値が返ってきます。`nrwidth`, `nrheight` には、それぞれ X サーバの画面のピクセルサイズが返ります。なお、X サーバへの接続が失敗した場合は `ndepth` には -1 が返ってきます。

`putimg24`(§3.4.28) を call する時は、必ず `depth` 値を調べるようにします。

**使用例** `call ggetdisplayinfo(ndepth,nrwidth,nrheight)`

### 3.4.2 `gopen(nxsize,nysize,nw)`

**機能** 任意のサイズのグラフィックス画面を開く

任意のサイズのウィンドウを開きます。nxsize, nysize にはそれぞれ横方向、縦方向のドット数 (整数) を指定します。nw には ProCALL 内部で使用する、ウィンドウ番号 (整数) が返ってきます。gopen を呼ぶと、このウィンドウ番号には、ProCALL 側で生成されたウィンドウ番号が設定されるため、ユーザが nw に値を代入しておく必要はありません。この取得したウィンドウ番号は ProCALL のグラフィックス描画サブルーチンに渡します。

互換サブルーチンでどのウィンドウにアクセスするかは、このウィンドウ番号 nw をサブルーチン selwin (§3.4.32) で指定します。

**使用例**    call gopen(800,600,nwin)

### 3.4.3 gclose(nw)

**機能**    グラフィックス用ウィンドウを閉じる  
nw で指定されたウィンドウを閉じます。

**使用例**    call gclose(nwin)

### 3.4.4 gcloseall

**機能**    すべてのグラフィックス用ウィンドウを閉じ、X サーバと断つ  
すべてのウィンドウを閉じ、X サーバと断線します。

**使用例**    call gcloseall

### 3.4.5 gsetbgcolor(nw, strc)

**機能**    ウィンドウの背景色を変更する

nw で指定されたウィンドウの背景色 (gclr サブルーチンで初期化される色) を変更します。

strc には X サーバの rgb.txt<sup>7)</sup> に設定されている色を指定し、'Blue\0' のように必ず最後に\0 を入れます。また、'#c0c0ff\0' のように、16 進数の Red, Green, Blue での指定も可能です。

**使用例**    call gsetbgcolor(nwin, 'white\0') ;

### 3.4.6 newwindow(nw, xs, ys, xe, ye)

**機能**    座標系の変更

nw で指定したウィンドウの座標系を変更します (実際のグラフィックスエリアの大きさが変わるわけではありません)。グラフィックス画面の座標は、デフォルトでは左下が (0.0, 0.0) で右上が (xsize-1.0, ysize-1.0) になっています。newwindow サブルーチンでこの座標系を左下を (xs, ys), 右上を (xe, ye) に変更できます。

下の使用例では、左上を (0.0, 0.0), 右下を (799.0, 599.0) に変更します。

**使用例**    call newwindow(nwin, 0.0, 599.0, 799.0, 0.0)

---

<sup>7)</sup> rgb.txt は UNIX 系の OS なら /usr/X11R6/lib/X11/ などにあります。

### 3.4.7 layer(nw,lys,lyw)

**機能** レイヤの設定をする

ProCALL ではグラフィックス用ウィンドウ毎に 2 枚のレイヤを持ち、表示するレイヤと書き込むレイヤを独立に指定できます。nw にはウィンドウ番号を指定し、lys には表示するレイヤ番号、lyw には書き込むレイヤ番号を 0 か 1 の整数で指定します。

なお、現在表示しているレイヤに対して (lys = lyw の場合に) 連続して描画サブルーチンを実行すると、良い描画パフォーマンスが得られません。高速な描画が必要な場合には、現在表示していないレイヤに対して描画し、最後に表示レイヤを layer サブルーチンで切り替えるか、copylayer サブルーチン (§3.4.8) で描画レイヤの画像を表示レイヤにコピーするようにします。

デフォルトでは layer(nw,0,0) の状態となっています。

**使用例** call layer(nwin,0,1)

### 3.4.8 copylayer(nw,lysrc,lydest)

**機能** レイヤのコピーをする

wn のウィンドウ番号の、レイヤ lysrc の画像をレイヤ lydest にそのままコピーします。このコピーは瞬時に行われるため、アニメーションの再生に使うことができます。

**使用例** call copylayer(nwin,1,0)

### 3.4.9 newpencolor(nw,nc)

**機能** 描画色の変更

nw で指定したウィンドウでの描画色を変更します。nc と色との関係は以下の通りです。

0:黒 1:白 2:赤 3:緑 4:青 5:シアン 6:マゼンタ 7:黄

8:DimGray 9:Gray 10:red4 11:green4 12:blue4 13:cyan4 14:magenta4 15:yellow4  
red4, green4... の “4” のつく色は、暗い赤、暗い緑...となっています。

デフォルトでは、白が指定されています。

**使用例** call newpencolor(nwin,2)

### 3.4.10 newcolor(nw,strc)

**機能** 描画色の変更

nw で指定したウィンドウでの描画色を変更します。strc には X サーバの rgb.txt<sup>8)</sup> に設定されている色を指定し、'Blue\0' のように必ず最後に \0 を入れます。また、'#c0c0ff\0' のように、16 進数の Red,Green,Blue での指定も可能です。

**使用例** call newcolor(nwin,'Violet\0')

### 3.4.11 newrgbcolor(nw,nr,ng,rb)

**機能** 描画色の変更

<sup>8)</sup> rgb.txt は UNIX 系の OS なら /usr/X11R6/lib/X11/ などにあります。

nw で指定したウィンドウでの描画色を変更します . nr,ng,nb にはそれぞれ Red,Green,Blue の輝度を 256 段階の整数 (0 ~ 255) で指定します .

**使用例** call newrgbcolor(nwin,255,127,0)

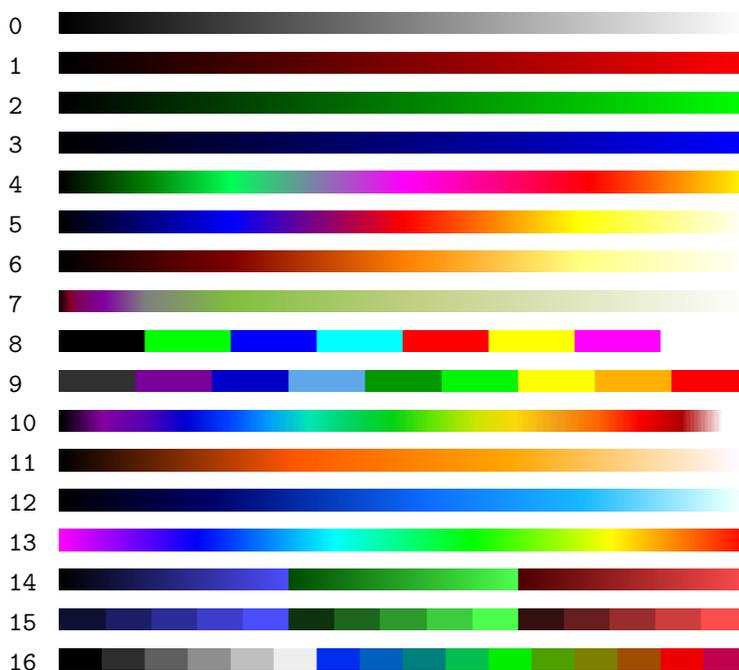
### 3.4.12 makecolor(ncolormode,dmin,dmax,data,nr,ng,nb)

**機能** 変数からカラーを生成する (カラーバー生成)

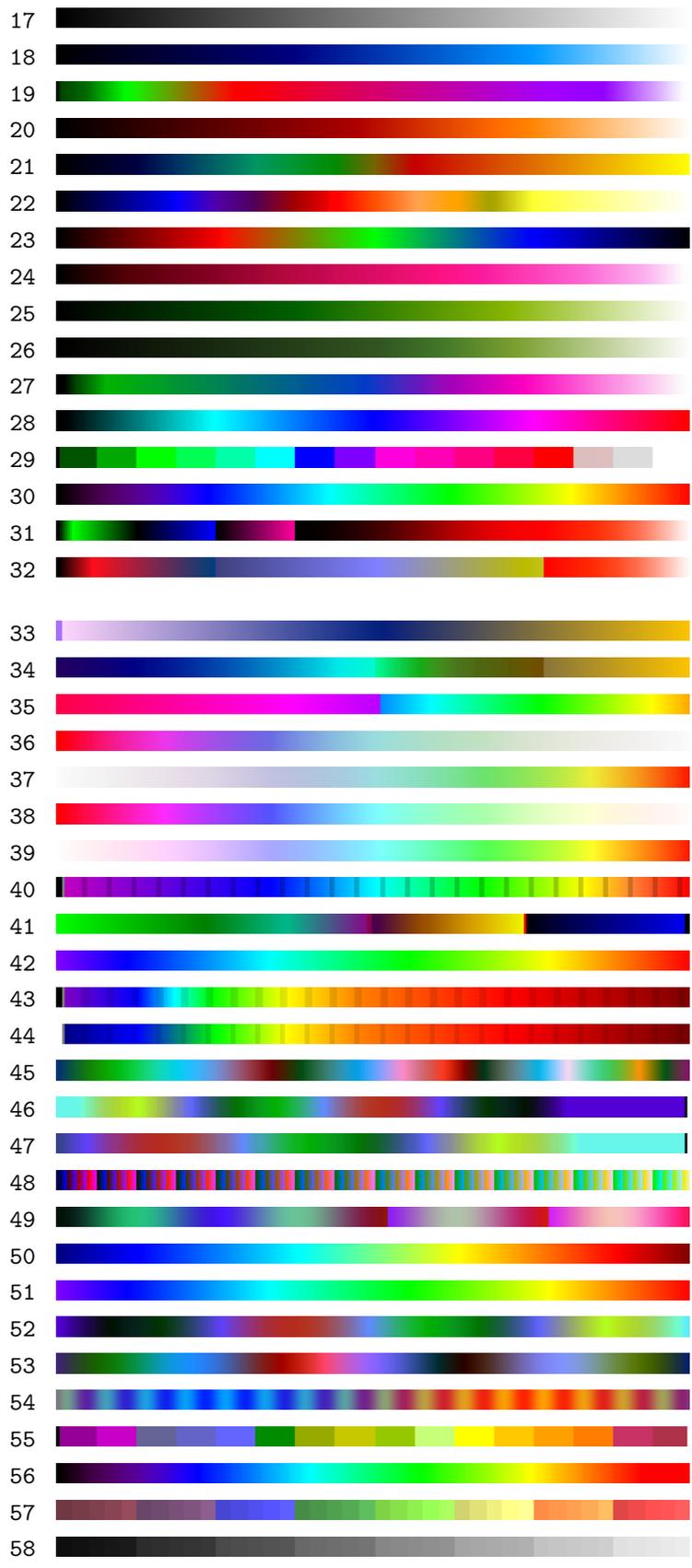
変数 data の値を使って , nr,ng,nb それぞれに 256 段階の Red,Greed,Blue のカラーを生成します . 変数の最小 , 最大は dmin,dmax で指定します . このサブルーチンで得た , nr,ng,nb の値は newrgbcolor サブルーチン (§3.4.11) にそのまま与えて使う事ができます .

ncolormode はカラーパターンの番号で , 以下に示す約 50 種類のカラーパターンが利用できます .

以下は fits ビューア DS9 コンパチのカラーパターンです .



以下は可視化ソフトウェア IDL コンパチのカラーパターンです .



使用例 call makecolor(10,v\_min,v\_max,v,nr,ng,nb)

### 3.4.13 tclr

**機能** 端末のクリア

端末をクリアし、カーソルの位置をホームポジションに戻します。

**使用例** `call tclr`

### 3.4.14 gclr(nw)

**機能** `nw` で指定したグラフィックス用ウィンドウをクリア

`gsetbgcolor` サブルーチン (§3.4.5) で指定した色で、ウィンドウをクリアします。`gsetbgcolor` での指定がない場合の色は黒となっています。

**使用例** `call gclr(nwin)`

### 3.4.15 pset(nw,xg,yg)

**機能** 点の描画

`nw` で指定したウィンドウに点を描画します。

グラフィックス画面の座標は、デフォルトでは左下が (0.0, 0.0) で右上が (nxsize-1.0, nysize-1.0) となっています。この座標は `newwindow` サブルーチン (§3.4.6) で変更が可能です。

**使用例** `call pset(nwin,x,y)`

### 3.4.16 line(nw,xg,yg,mode)

**機能** 直線、点の描画

`nw` で指定したウィンドウに直線を描画します。`mode` に 2 を指定すると以前 `line` サブルーチンが呼ばれた点から、(xg,yg) へ直線を引きます。`mode` に 3 を指定すると (xg,yg) を `line` サブルーチンの初期位置に設定します。`mode=2` でペンを下ろして描画、`mode=3` でペンを上げて移動と考えるとわかりやすいでしょう。また、`mode=1` の場合は (xg,yg) に点を描き、ペンの位置が更新されます。`xg, yg` は実数型の引数です。

**使用例** `call line(nwin,x,y,2)`

### 3.4.17 drawcirc(nw,xcen,ycen,xrad,yrad)

**機能** 中心座標と半径を与えて円を描く

`nw` で指定したウィンドウに、(xcen,ycen) を中心に横方向の半径 `xrad`、縦方向の半径 `yrad` の円を描きます。

**使用例** `call drawcirc(nwin,50.0,60.0,30.0,40.0)`

### 3.4.18 drawarc(nw,xcen,ycen,xrad,yrad,sang,eang,idir)

**機能** 円の中心、半径、始点、終点の角度を与えて円弧を描く

nw で指定したウィンドウに、(xcen,ycen) を中心に横方向の半径 xrad、縦方向の半径 yrad の円弧を描きます。sang は開始角、eang は終了角で、度で与えます。idir は円弧を描く方向で 1 で左廻り、-1 で右廻りとなります。

**使用例**    call drawarc(nwin,50.0,60.0,30.0,40.0,-10.0,-170.0,-1)

### 3.4.19 fillarc(nw,xcen,ycen,xrad,yrad,sang,eang,idir)

**機能**    円の中心、半径、始点、終点の角度を与えて円弧を塗り潰す

nw で指定したウィンドウで、(xcen,ycen) を中心に横方向の半径 xrad、縦方向の半径 yrad の円弧を塗り潰します。sang は開始角、eang は終了角で、度で与えます。idir は円弧を描く方向で 1 で左廻り、-1 で右廻りとなります。

**使用例**    call fillarc(nwin,50.0,60.0,30.0,40.0,-10.0,-170.0,-1)

### 3.4.20 drawrect(nw,x,y,w,h)

**機能**    長方形の描画

nw で指定したウィンドウに、頂点 (x,y) から正の方向に幅 w、高さ h の長方形を描きます。

**使用例**    call drawrect(nwin,50.0,60.0,30.0,20.0)

### 3.4.21 fillrect(nw,x,y,w,h)

**機能**    長方形の領域を塗り潰す

nw で指定したウィンドウで、頂点 (x,y) から正の方向に幅 w、高さ h の長方形の領域を塗り潰します。

**使用例**    call fillrect(nwin,50.0,60.0,30.0,20.0)

### 3.4.22 fillpoly(nw,x,y,n,i)

**機能**    多角形を塗り潰す

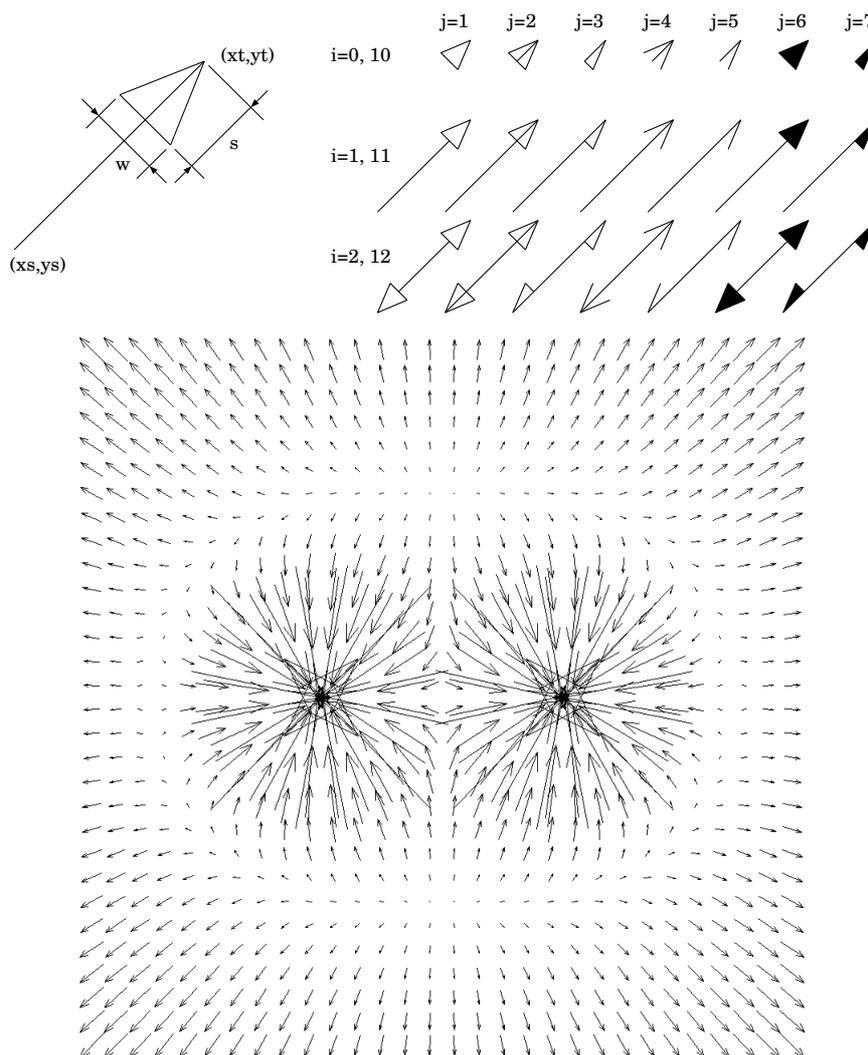
nw で指定したウィンドウで、多角形の領域を塗り潰します。x,y は n 個の実数の一次元配列で、x(1)~x(n), y(1)~y(n) に多角形の各点の座標を入れておきます。i は塗り潰す時の形状で通常は 0 を、凸多角形の時は 1 を指定します。

**使用例**    call fillpoly(nwin,x,y,5,0)

### 3.4.23 drawarrow(nw,xs,ys,xt,yt,s,w,10\*i+j)

**機能**    種々の型の矢印を描く

nw で指定したウィンドウに (xs,ys) から (xe,ye) に向かって矢印を描きます。矢印の形状は以下の図の通りで、s と w は実数で指定します。i が 0~2 の場合には w,s はドット数を、i が 10~12 の場合には w,s は矢印の長さに対する割合で 0.0~1.0 の値を指定します。



i=11 での矢印の描画例

**使用例** `call drawarrow(nwin,x0,y0,x1,y1,0.3,0.2,114)`

### 3.4.24 drawstr(nw,xg,yg,size,str,theta,len)

**機能** 文字列の描画

nw で指定したウィンドウに、文字列を座標 (xg,yg) から描きます。size は文字の大きさで、ドット単位の実数で指定します。str に文字列を、len には文字列の長さ (整数) を与えます。文字列 str の最終文字が終端文字 \0 であれば、len に実際の文字列長のかわりに -1 を与えてもかまいません。theta は文字列の回転を指定する実数の引数ですが、現バージョンでは機能しません。

文字のサイズ size は 1~24 の範囲で指定できます。size と実際のフォントとの関係は以下の表のようになっています。この場合、文字は半角英数字のみ描画できます。

2 バイト文字 (漢字) を描画する場合は、size には 0.0 を指定します。この場合のフォントの指定は、gsetfontset サブルーチン (§3.4.25) を利用します。gsetfontset でのフォント指定がなされていない場合は、デフォルトの 14 ドットのフォントセットで描画されます。

1~7	: 5 × 7	8	: 5 × 8	9	: 6 × 9	10~11	: 6 × 10	12	: 6 × 12
13	: 7 × 13	14~15	: 7 × 14	16~19	: 8 × 16	20~23	: 10 × 20	24	: 12 × 24

**使用例** `call drawstr(nwin,x,y,16.0,'Hoge',0.0,4)`

**使用例** `call drawstr(nwin,x,y,0.0,'日本語の描画も OK!\0',0.0,-1)`

### 3.4.25 gsetfontset(nw,fontset,nstatus)

**機能** フォントセット (日本語フォント) の指定

nw で指定したウィンドウで描くフォントセットを指定します。文字の描画は drawstr サブルーチン (§3.4.24) を利用します。

フォントセット名は fontset で指定します。fontset の最終文字は \0 (終端文字) でなければなりません。最後の引数 status には、フォントセットの取得状況を示す整数値が代入されます。fontset で指定したフォントセットが取得できた場合は 0、代替フォントで取得できた場合は 1、フォントセットの取得に失敗した場合は -1 が status に返ってきます。

**使用例** `call gsetfontset(nwin,'*-fixed-medium-r-normal--16-***-***-***\0',nstat)`

### 3.4.26 drawsym(nw,xg,yg,size,nsym)

**機能** センターシンボルの描画

nw で指定したウィンドウにセンターシンボルを座標 (xg,yg) に描きます。size はシンボルの大きさでドット単位の実数、nsym でシンボルの種類 (整数) を指定します。

nsym とシンボルの関係は、だいたい次のようなものとなっています。

1: · 2: + 3: \* 4: 4: 5: x 6: Y 7: 8: 9: 10:

なお、xg, yg は実数型の引数です。

**使用例** `call drawsym(nwin,x,y,16.0,2)`

### 3.4.27 drawnum(nw,xg,yg,size,v,theta,n)

**機能** 変数の値を描く

nw で指定したウィンドウに実数型変数 v の値を座標 (xg,yg) から描きます。size は文字列の大きさで、ドット単位の実数で指定します。n は表示する小数点の桁数で、整数値を与えます。theta は文字列の回転を指定する実数の引数ですが、現バージョンでは機能しません。

**使用例** `call drawnum(nwin,x,y,16.0,prm,0.0,3)`

### 3.4.28 putimg24(nw,x,y,nw,nh,nbuf)

**機能** 整数配列に用意した画像をウィンドウに一括転送する

nw で指定したウィンドウの座標 (x,y) に nbuf に用意した、幅 nw、高さ nh の画像を一括転送します。

整数配列 nbuf には、Blue, Green, Red の順に 3 つずつ、水平方向に走査しながら画像の上から下へ用意します (nbuf(1)=nBlue(1), nbuf(2)=nGreen(1), nbuf(3)=nRed(1), といった具合です)。配列には 0~255 の範囲の輝度を与えておきます。

なお X サーバの depth が 8 以下の場合はこのサブルーチンは使用できません (正常に動作しません)。X サーバの depth はサブルーチン ggetdisplayinfo (§3.4.1) で調べる事ができます。

**使用例** `call putimg24(nwin,x,y,640,400,nbuffer)`

### 3.4.29 saveimg(nw,ly,xs,ys,xs,ys,xe,ye,fname,n,conv,nd)

**機能** 画像をコンバータ (netpbm,ImageMagick) を通してファイルに保存する

ウィンドウ番号 `nw` , レイヤ番号 `ly` の `(xs,ys)` から `(xe,ye)` の範囲をコンバータ (netpbm,ImageMagick) を通してファイルに保存します . ファイル名は `fname` で指定し , 'image.png\0' のように必ず最後に \0 (終端文字) を入れます . `n` に正の整数 , 例えば `n=12` を指定すると , 実際のファイル名は `image12.png` のように数字がつき , `n` が負の場合は数字はつかずにそのままのファイル名で保存します . `conv` には ppm 形式から各画像フォーマットに変換するコンバータのコマンドラインを指定します . 例えば , netpbm を利用して png 形式に保存する場合は , 'pnmtopng\0' とします . ImageMagick を利用する場合は , 'convert\0' とします . もちろん , オプションスイッチも含める事もでき , 'pnmtops -scale 0.125\0' , あるいは 'convert -compress ZIP\0' といった指定も可能です . 最後の `nd` は減色パラメータで , R,G,B 1 チャンネルあたりの色の段階数を整数で指定します . `nd` は簡単な図形では 16 くらいで十分ですが , 多くの色を使っている場合は最大値の 256 を指定してください .

ppm から他の画像フォーマットに変換する netpbm のコマンドの例を挙げておきます . 各コンバータの使用方法に関しては端末から `man` コンバータ名 とタイプする事で調べることができます . ImageMagick の `convert` コマンドを利用する場合は , ファイル名のサフィックス (ドット以降の文字列 , 例えば .jpg , .png , .eps2 など) から保存フォーマットが決定されます .

画像フォーマット	コンバータ名	画像フォーマット	コンバータ名
AutoCAD DXB	ppmtoacad	Motif UIL icon	ppmtouil
windows bitmap	ppmtobmp	Windows .ico	ppmtowinicon
Berkeley YUV	ppmtoeyuv	XPM format	ppmtoxpm
GIF	ppmtogif	Abekas YUV	ppmtoyuv
NCSA ICR graphics	ppmtoicr	YUV triplets	ppmtoyvsplit
IFF ILBM	ppmtoilbm	DDIF	pnmtoddiff
Interleaf image	ppmtoleaf	FIASCO compressed	pnmtofiasco
HP LaserJet PCL 5 Color	ppmtolj	FITS	pnmtofits
map	ppmtomap	JBIG	pnmtobjbig
Mitsubishi S340-10 printer	ppmtomitsu	JFIF ("JPEG") image	pnmtjpeg
Atari Neochrome .neo	ppmtonao	Palm pixmap	pnmtopalm
PPC Paintbrush	ppmtopcx	plain (ASCII) anymap	pnmtoplainpnm
portable graymap	ppmtopgm	Portable Network Graphics	pnmtopng
Atari Degas .pil	ppmtopi1	PostScript	pnmtops
Macintosh PICT	ppmtopict	Sun raster	pnmtorast
HP PaintJet	ppmtopj	RLE image	pnmtorle
HP PaintJet XL PCL	ppmtopjxl	sgi image	pnmtosgi
X11 "puzzle"	ppmtopuzz	Solitaire image recorder	pnmtosir
three portable graymaps	ppmtorgb3	TIFF file	pnmtotiff
DEC sixel	ppmtosixel	CMYK encoded TIFF	pnmtotiffcmyk
TrueVision Targa	ppmtotga	X11 window dump	pnmtowd

もちろん , コンバータはすでにインストールされている必要があり , インストールされていない場合は指定できません<sup>9)</sup> . ここに挙げた以外のコンバータも , 標準入力から ppm 形式で入力し , 標準出力から変換データを出力するものであれば使用する事ができます .

コンバータを通さずに直接 ppm 形式で保存する場合は `conv` には '\0' と指定してください . ただし , ppm 形式は非圧縮のテキストデータなので , 巨大なファイルとなります .

このサブルーチンでは X サーバから画像データを転送し , 保存するわけですが , ネットワークが遅かったりすると非常に時間がかかります . その事を考慮して ProCALL では子プロセスを起動して , バックグラウンドで X サーバからの画像の転送・ディスクへの書き込みを行うようにしています . したがってプログラムは `call saveimg(...)` の後すぐに他の作業ができるようになります . ただし , この子プロセスが終わ

<sup>9)</sup> netpbm は <http://sourceforge.net/projects/netpbm/> , ImageMagick は <http://www.imagemagick.org/> などで配布されています .

るまでは X サーバでの描画等ができないので、もし `call saveimg(...)` のすぐ後に ProCALL の描画サブルーチンなどを呼んだ場合は画像の転送・保存が完了するまで待たされる事になります。

なお、`saveimg` を使っている場合は プログラムは必ず `gcloseall`(§3.4.4) を `call` してから終了するよう にしてください。

**使用例** `call saveimg(nwin,0,0.0,0.0,639.0,399.0,'img.png\0',i,'pnmtopng\0',256)`  
`png` 形式に保存する例です。gif 形式の場合は `conv` は `'ppmtogif\0'` にします。

**使用例** `call saveimg(nwin,0,0.0,0.0,639.0,399.0,'fig.ps\0',-1,'pnmtops -noturn -rle\0',256)`

**使用例** `call saveimg(nwin,0,0.0,0.0,639.0,399.0,'fig.eps2\0',-1,'convert -compress ZIP\0',256)`  
`PostScript` 形式に保存する例です。`netpbm` の `pnmtops` では `RunLength` 圧縮 (可逆圧縮) をサポートしています。上記のように「`-rle`」をつける事で画質を損うことなく、ファイルサイズを小さくすることができます。

`ImageMagick` では `RunLength` の他、`ZIP`、`LZW` などのアルゴリズムでより効率良く圧縮したビットマップイメージを `Postscript Level 2` ファイルに埋め込む事ができます。最後の使用例は `ZIP` 圧縮を用いた `Postscript Level 2` の `EPS` ファイルを生成する例です。ただし、`ImageMagick` の `Postscript Level 2` の出力は 5.3.3 以降で改悪され、一般的な `Postscript` インタープリタ (`Adobe Postscript 3` など) では読めません。`ImageMagick-5.3.2` の利用をお勧めします。

### 3.4.30 `ggetch(nw,key)`

**機能** キーボードから 1 文字入力があるまで待ち、入力された文字を返す

`nw` で指定したウィンドウにキー入力があるまで待ち、入力されたらその文字コード `key` (整数) を返します。

以下が 16 進の文字コード表です。イタリック体の数字が 16 進 2 桁の上位を表します。例えば、「a」は `z'61'`、「A」は `z'41'` となります。

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0		Home	PageUp	Pause		End	PageDown		BackSpace	Tab				Enter		
1												Esc				
2	Space	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	Delete

`z'01' ~ z'1A'` の中にはいくつか特殊キーが含まれていますが、この区間は「`Ctrl+アルファベットキー`」のコードと共有となります。例えば、「`BackSpace`」と「`Ctrl+H`」のコードはどちらも `z'08'` となります。`z'01' ~ z'1A'` で空白になっている個所は、「`Ctrl+アルファベットキー`」にのみコードが割り当てられています。

**使用例** `call ggetch(nwin,key)`

### 3.4.31 `ggetxpress(nw,ntype,nbutton,xg,yg)`

**機能** マウスからのボタンクリック、キーボードからの入力があるまで待ち、その情報を返す

`nw` で指定したウィンドウでマウスのボタンクリック、キータイプがあるまで待ち、入力があればその情報、`ntype`、`nbutton`、`xg`、`yg` を返します。

マウスのボタンクリックがあった場合、`ntype` には 4 が代入され、キーボードからの入力の場合は 2 が代入されます。

マウスのボタンクリックの場合、`nbutton` にはクリックされたボタンの番号 (1,2,3,...), クリックされた時のウィンドウ `nw` 上でのマウスポインタの座標が `xg`, `yg` に代入されます。

キー入力の場合は、キーコードが `nbutton` に返ります。このキーコードは `ggetch` サブルーチン (§3.4.30) の `key` と同一です。

**使用例** `call ggetxpress(nwin,ntype,nb,x,y)`

### 3.4.32 selwin(nw)

**機能** 描画するウィンドウを指定する

互換サブルーチンでどのウィンドウにアクセスするかを指定します。`nw` には `gopen`(§3.4.2) で得た、ウィンドウ番号を指定します。なお、カルコンプ互換ルーチンの `plots`(§3.5.1) で開いたウィンドウの番号は 0 となります。デフォルトは 0 です。

**使用例** `call selwin(nwin)`

## 3.5 カルコンプ互換サブルーチンのリファレンス

Pro-FORTRAN とコンパチブルなサブルーチン群です。Pro-FORTRAN や他の GKS, カルコンプ互換の FORTRAN から移行する場合以外は、必要としないと思われる。

### 3.5.1 plots

**機能** グラフィックス用ウィンドウを開く

このサブルーチンを呼ぶことで、グラフィックス用ウィンドウを開きます。グラフィックスエリア (ウィンドウサイズ) は 640 × 400 ドットです。グラフィックスエリアを任意にとりたい場合や複数のウィンドウを開きたい場合は `call gopen` を使用してください。

なお、サブルーチン `plots` で呼び出した ProCALL でのウィンドウ番号は 0 となります (ProCALL 標準サブルーチンのウィンドウ番号 `nw` には 0 を指定します)。

**使用例** `call plots`

### 3.5.2 window(xs,ys,xe,ye)

**機能** 座標系の変更

グラフィックス画面の座標は、デフォルトでは左下が (0.0, 0.0), 右上が (639.0, 399.0) になっています。`window` サブルーチンでこの座標系を左下を (`xs`, `ys`), 右上を (`xe`, `ye`) に変更できます。

下の使用例では、左上を (0.0, 0.0), 右下を (639.0, 399.0) に変更します。

**使用例** `call window(0.0, 399.0, 639.0, 0.0)`

### 3.5.3 newpen(nc)

**機能** 描画色の変更

plot などでの描画色を変更します。nc と色との関係は以下の通りです。  
0:黒 1:白 2:赤 3:緑 4:青 5:シアン 6:マゼンタ 7:黄  
8:DimGray 9:Gray 10:red4 11:green4 12:blue4 13:cyan4 14:magenta4 15:yellow4  
red4, green4...の“4”のつく色は、暗い赤、暗い緑...となっています。  
デフォルトでは、白が指定されています。

**使用例** call newpen(2)

**互換性** 8~15の色は本家 Pro-FORTRAN では使用できません。

### 3.5.4 clsc

**機能** 端末のクリア  
端末をクリアし、カーソルの位置をホームポジションに戻します。

**使用例** call clsc

### 3.5.5 clsx

**機能** グラフィックス画面をクリア

**使用例** call clsx

### 3.5.6 plot(xg,yg,mode)

**機能** 直線、点の描画

mode に 2 を指定すると以前 plot が呼ばれた点から、(xg,yg) へ直線を引きます。mode に 3 を指定すると (xg,yg) を plot サブルーチンの初期位置に設定します。mode=2 でペンを下ろして描画、mode=3 でペンを上げて移動と考えるとわかりやすいでしょう。

また、mode=1 の場合は (xg,yg) に点を描き、ペンの位置を更新します。

xg, yg は実数型の引数です。

**使用例** call plot(x,y,2)

**互換性** mode=1 は本家 Pro-FORTRAN では使用できません。

### 3.5.7 arc(xcen,ycen,rad,sang,eang,idir)

**機能** 円の中心、半径、始点、終点の角度を与えて円弧を描く

(xcen,ycen) を中心に半径 rad の円弧を描きます。sang は開始角、eang は終了角で、度で与えます。idir は円弧を描く方向で 1 で左廻り、-1 で右廻りとなります。

**使用例** call arc(50.0,60.0,30.0,-10.0,-170.0,-1)

### 3.5.8 circ1(xc,yc,r)

**機能** 中心座標と半径を与えて円を描く

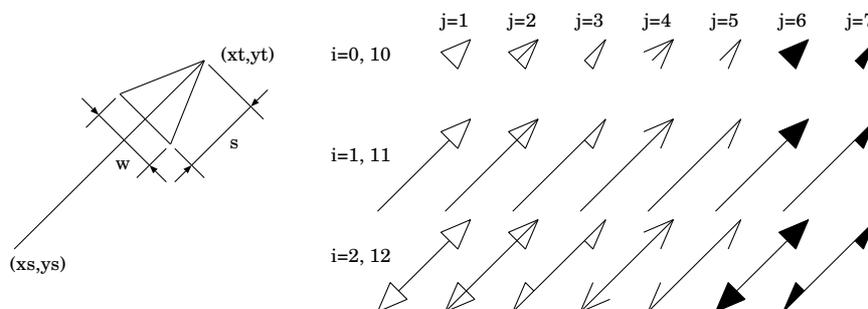
(xc,yc) を中心に半径 r の円を描きます。

**使用例** call circ1(50.0,60.0,30.0)

### 3.5.9 arohd(xs,ys,xt,yt,s,w,10\*i+j)

**機能** 種々の型の矢印を描く

(xs,ys) から (xe,ye) に向かって矢印を描きます。矢印の形状は以下の図の通りで、s と w は実数で指定します。i が 0~2 の場合には w,s はドット数を、i が 10~12 の場合には w,s は矢印の長さに対する割合で 0.0~1.0 の値を指定します。



**使用例** `call arohd(x0,y0,x1,y1,0.3,0.2,114)`

**互換性** i=10~12 は本家 Pro-FORTRAN では使用できません。本家 Pro-FORTRAN では window の指定によって実際に描画される矢印の先端の大きさが変化しますが、ProCALL では変化しません。

### 3.5.10 symbol(xg,yg,size,nstr,theta,len)

**機能** 文字列，センターシンボルの描画

文字列またはセンターシンボルを座標 (xg,yg) に描きます。size は文字列・シンボルの大きさを、ドット単位の実数で指定します。文字列を描く場合は len に文字列の長さ (整数) を、nstr に文字列を与えます。センターシンボルを描く場合は len に -1 を指定し、nstr にシンボルの番号 1~10 の整数を与えます。

文字のサイズ size は 1~24 の範囲で指定できます。size と実際のフォントとの関係は以下のようになっています。文字は半角英数字ならすべて描画できる上、本家 Pro-FORTRAN より綺麗です。

1~7 : 5 × 7	8 : 5 × 8	9 : 6 × 9	10~11 : 6 × 10	12 : 6 × 12
13 : 7 × 13	14~15 : 7 × 14	16~19 : 8 × 16	20~23 : 10 × 20	24 : 12 × 24

nstr とシンボルの関係は、だいたい次のようなものとなっています。

1:・ 2:+ 3:\* 4: 5:× 6:Y 7: 8: 9: 10:

なお、このサブルーチンは 4 番目の引数に、文字列型と整数型の両方をとります。この symbol を使って文字列、シンボルの両方を描くように作成されたソースでは、コンパイルすると Warning が出てきます。この Warning はかなり目障りなので、本家 Pro-FORTRAN でコンパイルしない場合は、ProCALL 標準サブルーチン drawstr(文字列を描く)、drawsym(シンボルを描く) の使用を推奨します。

**使用例** `call symbol(x,y,16.0,'Hoge',0.0,4)`  
`call symbol(x,y,16.0,2,0.0,-1)`

**互換性** 本家 Pro-FORTRAN では window の指定によって実際に描画されるシンボルの大きさが変化しますが、ProCALL では変化しません。

24 より大きい文字サイズは 12 × 24 ドットフォント固定となります。アルファベットの小文字と一部の記号は本家 Pro-FORTRAN では出力できません。theta は文字列の回転を指定する実数の引数ですが、現バージョンでは機能しません。

### 3.5.11 number(xg,yg,size,v,theta,n)

**機能** 変数の値を描く

実数型変数  $v$  の値を座標  $(xg,yg)$  から描きます。size は文字列の大きさを、ドット単位の実数で指定します。n は表示する小数点の桁数で、整数値を与えます。

**使用例** call number(x,y,16.0,prm,0.0,3)

**互換性** 24 より大きい文字サイズは 12 × 24 ドットフォント固定となります。theta は文字列の回転を指定する実数の引数ですが、現バージョンでは機能しません。

### 3.5.12 vport,setal

**機能** ダミーのサブルーチン

**互換性** これらのサブルーチンは機能しません。Pro-FORTRAN のソースをそのままコンパイルできるように、これらのサブルーチンは形だけ残してあります。

## 3.6 補助サブルーチンのリファレンス

FORTRAN の不便さを少しでも解消するように用意したサブルーチンです。

### 3.6.1 msleep(ms)

**機能** ミリ秒単位で実行を延期する

ms ミリ秒の間、プログラムの実行をなにもせずに待ちます。ms には 999 までの整数を指定します。アニメーション速度の調整に利用できます。

**使用例** call msleep(100)

### 3.6.2 isnan(v,iflg)

**機能** 実数型変数を非数 (Not a Number) かどうか調べる

実数型変数  $v$  を非数かどうかを調べ、非数なら iflg に 0 以外の整数値を返します。

**使用例** call isnan(x,nf)

### 3.6.3 rtoc(v,n,ns,str,m)

**機能** 実数型変数を文字列に変換する

実数型変数  $v$  を小数点  $n$  桁まで文字列に変換し、区切り文字 '\0' を最後に付け足して文字型変数  $str$  に格納します。ns は  $str$  で確保されている文字数を指定します。m には  $str$  に文字列を格納した時に余った文字数 (整数値) を返し、この値が負の場合は  $str$  で確保されている文字数が足りなかった事を示します。

**使用例** call rtoc(x,4,10,st,m)