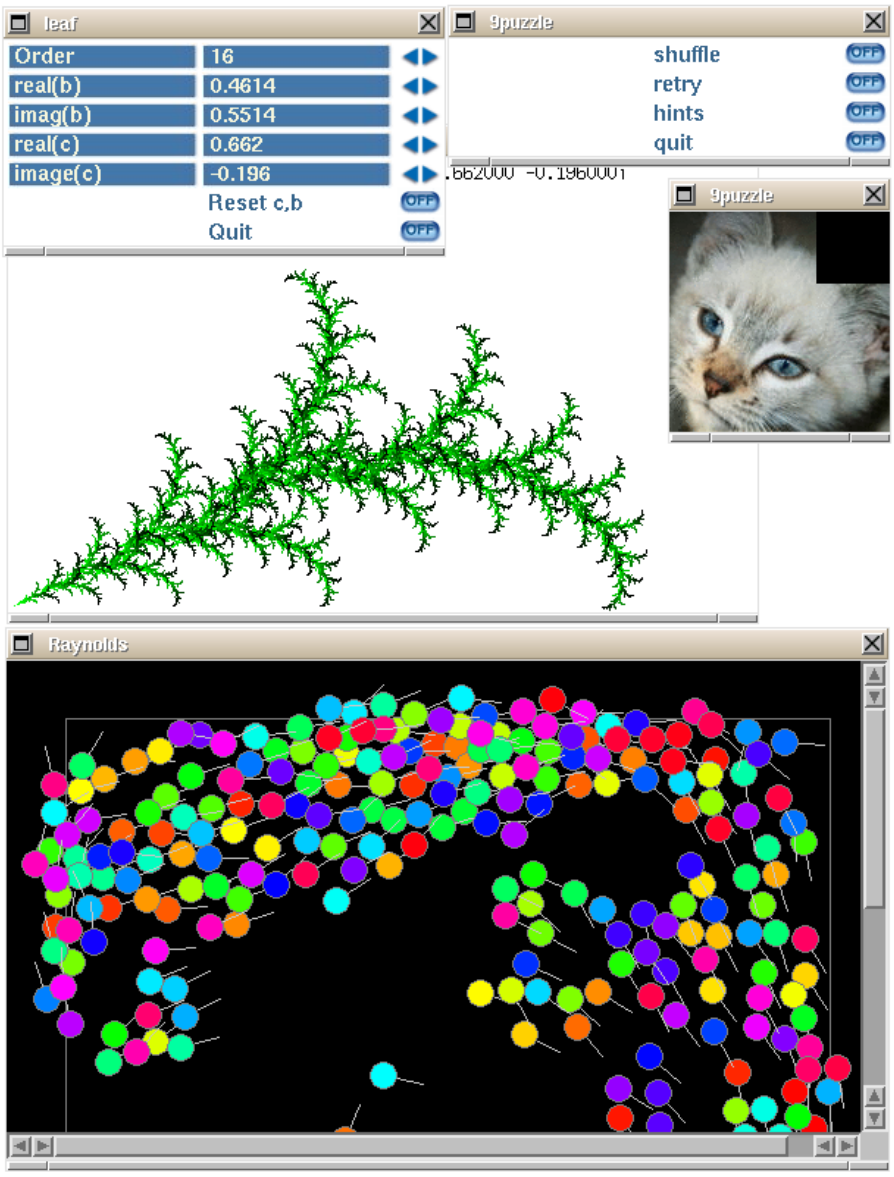
Easy and Gratifying Graphics Library for X11

EGGX/ProCALL version 0.93

User's Guide

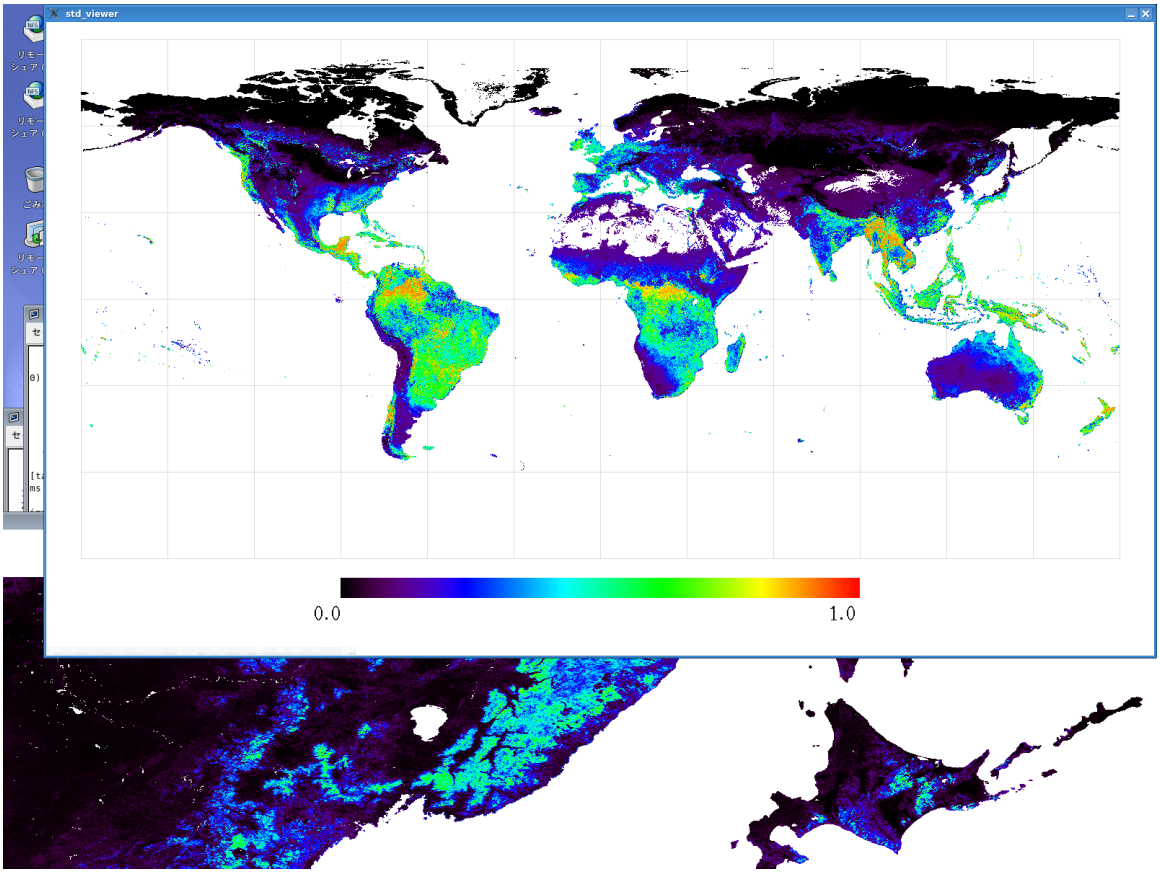
Presented by Chisato Yamauchi, Mar. 5, 2010

Translated by Space Engineering Development Co.,LTD. and Sakura Academia Corporation



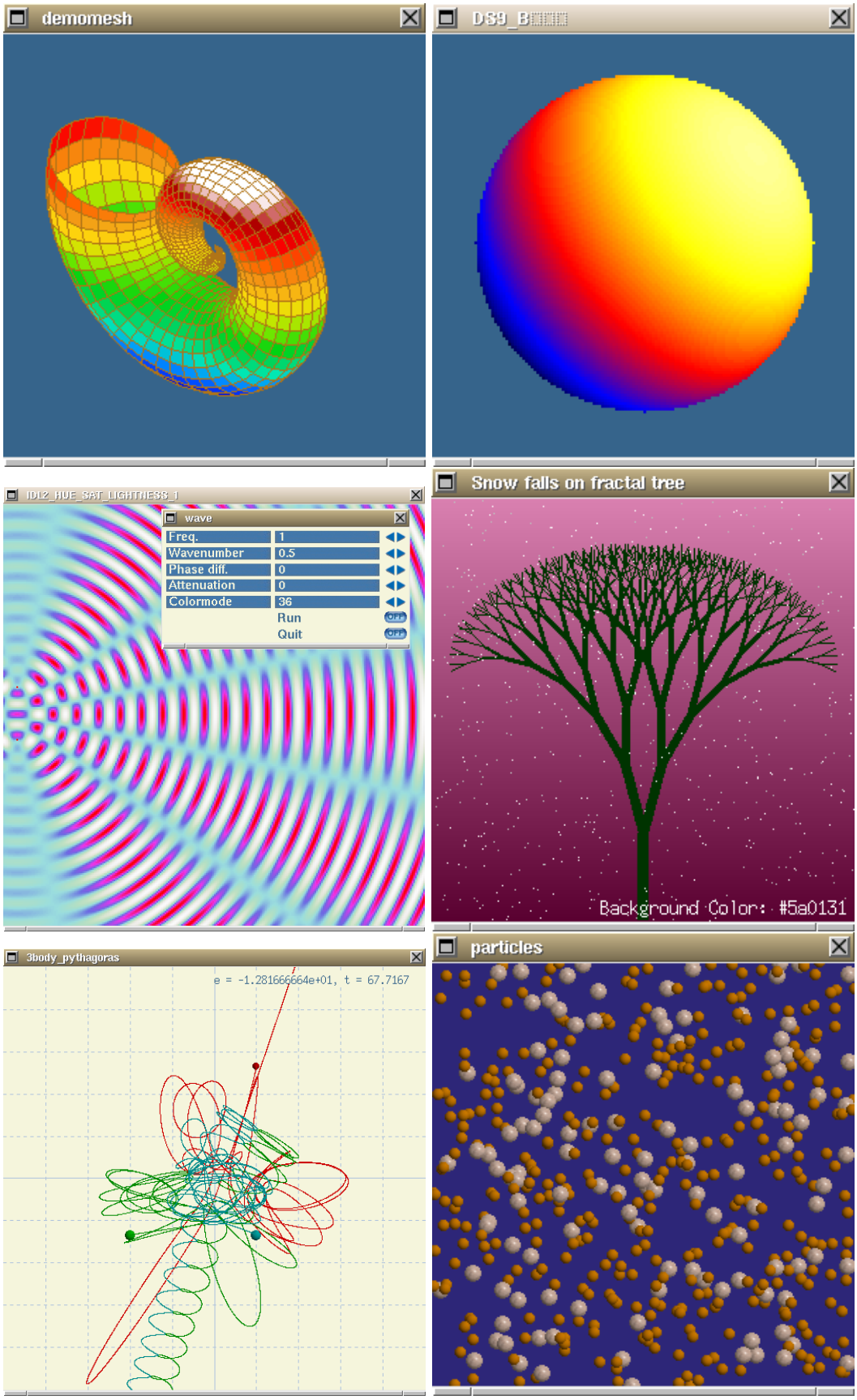
The screenshots on the cover were made with the codes written by Dr. Matsuda (Tokyo Denki University) and Mr.Yasuda (Kyoto Sangyo University).

Gallery



A viewer for remote-sensing by Dr. Sasai (Nagoya University).

An example program “loupe.c”.



Academic example programs by Dr. Matsuda (Tokyo Denki University).

1 INTRODUCTION

1.1 ABOUT EGGX/ProCALL

EGGX/ProCALL is a simple graphics library developed to achieve **extreme simplicity**. It can be called from C or FORTRAN, and has almost the same graphics functions as classical BASIC. We call the group of C functions "EGGX"(§2.2) and the group of FORTRAN subroutines "ProCALL"(§3.1). We have created this library out of our desire to deliver simple usability and the fun that we had when using BASIC in the age of the 8-bit machine to whoever starts to learn programming. This will become one of the optimum libraries especially for elementary programming education.

Since EGGX/ProCALL is a simple library, you do not need to install any dedicated runtime files. Naturally, executable files created by users become real application software. Installation is simple because the library uses only Xlib as X11-related libraries.

1.2 FEARTURES OF EGGX/ProCALL

Simplicity of user functions is by far the best. You can open a window on X Window and use graphics functions immediately by calling "only one function". Users can draw an image by only listing drawing functions without worrying about complicated events, because redrawing is done by the library when a window is hidden. For example, it is easy to insert some drawing functions to a code of numerical calculation and monitor its condition during calculation only when necessary upon debugging.

With this library, it is possible to open multiple graphics windows of any size, to draw lines, points, polygons, and circles whose colors can be specified in 24 bits, to draw font sets (a string containing one- and two-byte characters can be drawn as it is), to draw various arrows needed for visualization of numerical calculation, to generate (about 50 kinds of) color bars, to transfer 24-bit images whose background can be transparent, to scroll a screen (corresponding to the splite and the hardware scroll of video games), to generate smooth animation using the layer function (corresponding to 2 plain of V-RAM of old PCs), to have key input by using `ggetch()` (§2.4.47), and to have mouse and key input by using `ggetevent()` (§2.4.48).

As you may understand, the availability of 24-bit colors and animation means that this library has enough basic elements for today's 2D graphics. You can visualize numerical calculation or games, etc. in any fancy manner, depending on your programming.

Additionally, specific functions are prepared to save images continuously in various formats through various commands of netpbm [1)] , or the convert command of ImageMagick [2)]. Since a lot of frame images can be created at once by using these functions, gif animations and mpeg movies can be created easily.

Furthermore, the scrollbar interface is offered since Version 0.91, so you can easily handle images whose size is too large to fit into a display.

1.3 INFORMATION ABOUT EGGX/ProCALL

EGGX/ProCALL Web page:

`http://www.ir.isas.jaxa.jp/~cyamauch/eggx_procall/`

Sample programs, release informations, derived libraries and applications in informational education are available on this web page. Please use it in accordance with your purpose.

[1)] http://sourceforge.net/projects/netpbm/

[2)] http://www.imagemagick.org

2 C

2.1 TUTORIAL

2.1.1 Basic Usage

In the beginning of the user's program declare as follows[3] :

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

Write a program using EGGX functions just like you would with C standard functions and compile using the egg command.

Example `egg program.c -o program`

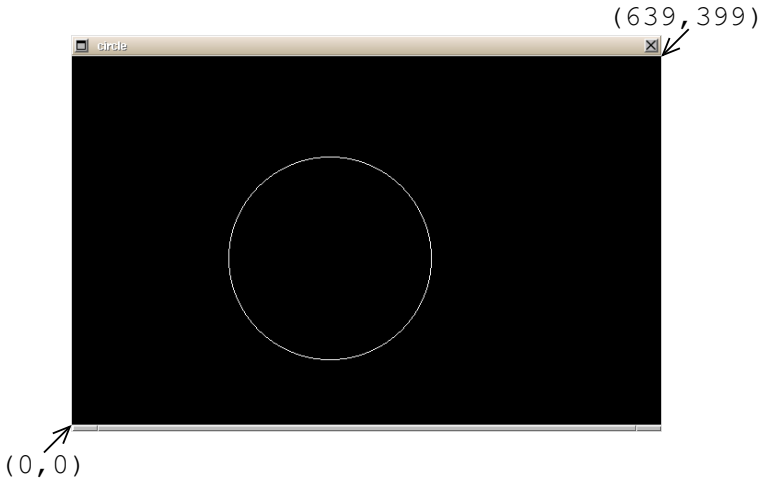
2.1.2 Sample Program

A program to draw a circle using EGGX is shown below:

```
#include <eggx.h>                       /* necessary to use EGGX*/
int main()
{
    int win;                            /* variable to store window index*/
    win = gopen(640,400);               /* open 640x400 dot window for graphics*/
    circle(win, 280, 180, 110, 110);    /* draw the circle with center(280,180) and radius 110 */
    ggetch();                           /* wait until key input*/
    gclose(win);                        /* close the window for graphics*/
    return 0;                           /* end*/
}
```

It opens a window and draws a circle in the lower left slightly down from the center of the window. To prevent the program from finishing in a blink of an eye, `ggetch()` is used before the window is closed.

The following picture is the screenshot of the result.



The origin of the coordinate system is the bottom-left corner of the window by default. Points, lines, polygons, symbols, and strings can be drawn by specifying the window index and the coordinate in the same way as this example.

[3] We recommend the use of `eggxlib.h` when using EGGX in developing full-fledged software. See §2.3.4.

2.2 EGGX FUNCTIONS LIST

2.2.1 Standard Functions

Section	Function	Description
§2.4.1	gopen	Open a graphics screen of any size
§2.4.2	gclose	Close a window for graphics
§2.4.3	gcloseall	Close all windows for graphics and disconnect from the X server
§2.4.4	gresize	Resize the graphics drawing area
§2.4.5	winname	Change the title of a window
§2.4.6	coordinate	Change the application coordinate system (set the reference point and the scale)
§2.4.7	window	Change the application coordinate system (set the the bottom-left and the top-right coordinates)
§2.4.8	layer	Configure a layer
§2.4.9	copylayer	Copy a layer
§2.4.10	gsetbgcolor	Set the background color (in gclr) of a window
§2.4.11	gclr	Clear the drawing layer
§2.4.12	tclr	Clear a terminal screen
§2.4.13	newpen	Change a drawing color (16 colors)
§2.4.14	newcolor	Change a drawing color (set a color contained on the X server directly)
§2.4.15	newrgbcolor	Change a drawing color (set the brightness of Red, Green, Blue)
§2.4.16	newhsvcolor	Change a drawing color (set Hue, Saturation, Value)
§2.4.17	makecolor	Generate a color from a variables (generate a color bar)
§2.4.18	newlinewidth	Change line width
§2.4.19	newlinestyle	Change the line style
§2.4.20	newgcfunction	Change the GC function
§2.4.21	pset	Draw a point
§2.4.22	drawline	Draw a line
§2.4.23	moveto, lineto	Draw a line continuously
§2.4.24	drawpts	Draw points
§2.4.25	drawlines	Draw a polygonal line
§2.4.26	drawpoly	Draw a polygon
§2.4.27	fillpoly	Fill a polygon
§2.4.28	drawrect	Draw a rectangle
§2.4.29	fillrect	Fill a rectangle
§2.4.30	drawcirc, circle	Draw a circle by specifying the center coordinate and the radius
§2.4.31	fillcirc	Fill a circle by specifying the center coordinate and the radius
§2.4.32	drawarc	Draw an arc by specifying the center and radius of a circle and the angle of the starting point and the ending point
§2.4.33	fillarc	Fill an arc by specifying the center and radius of a circle and the angle of the starting point and the ending point
§2.4.34	drawsym	Draw a symbol
§2.4.35	drawsyms	Draw symbols
§2.4.36	drawarrow	Draw various types of arrows
§2.4.37	newfontset	Specify a font set (Japanese font)
§2.4.38	drawstr	Draw a string
§2.4.39	gscroll	Scroll the drawing layer in pixel unit
§2.4.40	gputarea	Copy an image in any window, layer, and area to the drawing layer
§2.4.41	gputimage	Transfer images data (with a mask) in a memory buffer to the drawing layer collectively
§2.4.42	ggetimage	Load image data in any window, layer, and area to a memory
§2.4.43	gsaveimage	Save an image in any window, layer, and area to a file through a converter
§2.4.44	readimage	Load image data in a file to a memory buffer through a converter (netpbm, etc.)
§2.4.45	writeimage	Save image data in a memory buffer to a file through a converter (netpbm, etc.)
§2.4.46	gsetnonblock	Configure the operating mode of `ggetch()`, `ggetevent()` and `ggetxpress()`
§2.4.47	ggetch	Return a character inputted from the keyboard
§2.4.48	ggetevent	Return input information from the mouse or the keyboard
§2.4.49	ggetxpress	Return information of clicking mouse buttons or input from the keyboard
§2.4.50	msleep	Wait an execution in milliseconds

2.2.2 Advanced Functions (for Intermediate/Expert Users)

Section	Function	Description
§2.5.1	ggetdisplayinfo	Get information on the X server (depth and screen size)
§2.5.2	gsetnonflush, ggetnonflush	Configure the flush in drawing functions
§2.5.3	gflush	Flush drawing commands
§2.5.4	gsetinitialattributes	Set the attributes of the windows opened with `gopen()`
§2.5.5	ggetinitialattributes	Get the attributes of the windows opened with `gopen()`
§2.5.6	gsetinitialbgcolor	Set the background color of the windows opened with `gopen()`
§2.5.7	gsetborder	Set the width and color of window borders
§2.5.8	gsetinitialborder	Set the borders of the windows opened with `gopen()`
§2.5.9	gsetinitialgeometry	Set appearance position and other attributes of the windows opened with `gopen()` from a string including x and y values
§2.5.10	gsetinitialwinname	Set a window name, an icon name, a resource name, and a class name of the windows opened with `gopen()`
§2.5.11	gsetscrollbarkeymask	Set a keymask for the key operation of EGGX's scrollbar
§2.5.12	generatecolor	Generate a color from variables (contrast, brightness and gamma can be modified)

2.3 TIPS

2.3.1 How to Speed Up Drawing

- Use layers (double buffering)

  Draw always in an invisible layer when you use drawing functions by using the `layer()`(§2.4.8) and `copylayer()`(§2.4.9). After finishing drawing, copy the invisible layer to the visible layer (with the `copylayer()`). This approach can enhance the drawing speed appreciably.

  The following example shows a typical code written to draw an animation.

```
#include <eggx.h>

int main()
{
    int win;
    win = gopen(640,400);
    layer(win,0,1);             /* Set layer 0 for display and layer 1 for drawing */
    while ( 1 ) {
        gclr(win);              /* Initialize layer 1 */
            :                   /* Write drawing functions here */
        copylayer(win,1,0);     /* Copy the layer 1 to layer 0 ‘‘instantly’’  */
        msleep(10);             /* Wait 10 millisecond (to adjust animation speed) */
    }
```

- Use other functions than `newcolor()` to set a color

  Use the functions that specify a color by numerical values such as `newpen()` and `newrgbcolor()` as far as possible.

- Minimize the frequency of color setting

  If colors are set frequently as follows, drawing performance might become worse: `newpen()` → `pset()` → `newpen()` → `pset()` → `newpen()` → `pset()` → `newpen()` → `pset()` → ...

  It is better to arrange drawing commands by the same color as follows, as far as possible: `newpen()` → `pset()` → `pset()` → `pset()` → `newpen()` → `pset()` → `pset()` → `pset()` → ...

- Use manual flush (for expert users)

  See §2.5.2 and §2.5.3

2.3.2 How to Change the Origin (0,0) of the Window Coordinate to Top Left

As below, disable the `BOTTOM_LEFT_ORIGIN` attribute by using `gsetinitialattributes()` (§2.5.4) and then open a window:

Example
```
gsetinitialattributes(DISABLE, BOTTOM_LEFT_ORIGIN);
```

2.3.3 Usage from C++

In the case of C++, include `eggx.h` as is the case with C, because "`extern "C" {...}`" is declared in the header file that EGGX offers. The compiling method is the same as C. Add the suffix "`.cc`" to the source filename.

Example
```
egg program.cc -o program
```

The compiler is "`g++`" by default. To use another compiler, edit script egg.

2.3.4 When Using EGGX in Fully-Fledged Software Development

All actual names of the functions (names of symbols) in EGGX start with `eggx_`. In consideration of usability in informational education, etc., the macro to shorten the names of functions is defined in the header file `eggx.h` as below:

```
#define gopen eggx_gopen
```

However, such macro definitions might cause trouble when the structure or class of C++ is used.

Therefore, it is recommended to use `eggxlib.h` instead of `eggx.h` when using EGGX in fully-fledged software development as below:

```
#include <eggxlib.h>
```

You need to write the actual names of function in the source code because the macro on the names of functions is not defined in `eggxlib.h`. However, you can avoid such troubles as the error caused by the macro definition mentioned above, which replaces the member name of structure. In this case, add "`eggx_`" to the beginning of the names of the functions written in this manual as below:

Example
```
win = eggx_goepn(800,600) ;
```

2.4 EGGX STANDARD FUNCTIONS REFERENCE

2.4.1 int gopen(int xsize, int ysize)

Description Open a graphics screen of any size

This function opens a window for graphics and returns the window index used in the EGGX. In the case of EGGX, graphics are handled by giving this number to the drawing function.

Specify horizontal and vertical pixels of the drawing area by the arguments `xsize` and `ysize`, respectively. The maximum pixels are 32767.

If the drawing area that is almost the same size as or larger than the root window (background) is specified, a smaller window than the drawing area is opened by default [4)]. In this case, the scrollbar interface is offered, and it is possible to display anywhere in the drawing area by using the mouse or the keyboard.

Example
```
win = gopen(800,600) ;
```

2.4.2 void gclose(int wn)

Description Close a window for the graphics

This function closes a window specified with `wn`.

Example
```
gclose(win) ;
```

2.4.3 void gcloseall(void)

Description Close all windows for graphics and disconnect from the X server

This function closes all windows, disconnects from the X server, and frees the memory area used by the internal processing of the library.

Example
```
gcloseall() ;
```

2.4.4 void gresize(int wn, int xsize, int ysize)

Description Resize the graphics drawing area

This function changes the size of the drawing area in the window specified with `wn`. The number of pixels in the horizontal and vertical directions is specified for the arguments `xsize` and `ysize`, respectively.

The window size is changed together with the drawing area by default. However, when the scrollbar interface is enabled, the window size matches the specification by `gsetinitialgeometry()` (§2.5.9) if it is specified.

The size change in the drawing area is performed with the origin set to the bottom-left corner. On other hand, if the `BOTTOM_LEFT_ORIGIN` attribute is disabled with `gsetinitialattributes()` (§2.5.4), the top-left corner becomes the origin.

Example
```
gresize(win, 1280,960) ;
```

2.4.5 int winname(int wn, const char *argsformat, ...)

Description Change the title of a window

The user's execution filename is set to the title of the windows opened with `gopen()` by default. This title can be changed freely. The first argument `wn` is a window index, and the string specified with `argsformat` (and subsequent arguments) is set as the title. This can be used to displays the value of a variable as shown in the example below.

The return value is the length of the string set to the window title.

Example
```
winname(win,"penguin x=\%f y=\%f",x,y) ;
```

[4)] If Xinerama is available upon the build of EGGX/ProCALL, it is compared with the pixel size of the smallest display.

2.4.6 void coordinate(int wn, int xw, int yw, double xa, double ya, double xscale, double yscale)

Description Change the application coordinate system (set the reference point and the scale)

This function changes the application coordinate system in the window specified with `wn`.

The bottom-left corner is (`0`, `0`) and the top-right corner is (`xsize-1`, `ysize-1`) in the window coordinate system. (The coordinate value is an integer.) The coordinate value of the application coordinate system corresponds to that of the window coordinate system by default. (The coordinate value of the application coordinate system is a real number.)

(`xa`, `ya`) of the application coordinate system can be corresponded to (`xw`, `yw`) of the window coordinate system by using the `coordinate()`, and each scaling factor is specified with `xscale` and `yscale`. This means that the conversion from the application coordinates (x, y) to the window coordinates (`x`, `y`) in the drawing functions, etc. is done by the following expressions:

$\texttt{x} = \texttt{xw} + (x - \texttt{xa}) \cdot \texttt{xscale}$

$\texttt{y} = \texttt{yw} + (y - \texttt{ya}) \cdot \texttt{yscale}$

Once this function is called, each drawing function converts the coordinate system automatically because the application coordinate system is used in the EGGX drawing functions.

In the following example, the bottom-left of the window is set to (-40.0, -20.0) in the application coordinate system, and scaling factors of both x and y are set to 2.0:

Example
```
coordinate(win, 0,0, -40.0,-20.0, 2.0,2.0) ;
```

When the `BOTTOM_LEFT_ORIGIN` attribute in `gsetinitialattributes()` (§2.5.4) is disabled, the origin in the window coordinate system (`0`, `0`) is at the top-left corner.

You can also use `window()` (§2.4.7) to change the coordinate system. Please consider it.

2.4.7 void window(int wn, double xs, double ys, double xe, double ye)

Description Change the application coordinate system (set the the bottom-left and the top-right coordinates)

This function changes the application coordinate system in the window specified with `wn`. (The size of an actual graphics area does not change.)

The bottom-left corner is (`0`, `0`) and the top-right corner is (`xsize-1`, `ysize-1`) in the window coordinate system. (The coordinate value is an integer.) The coordinate value of the application coordinate system corresponds to that of the window coordinate system by default. (The coordinate value of the application coordinate system is a real number.)

The bottom-left corner ((`0`, `0`) in the window coordinate system) and the top-right corner can be changed to (`xs`, `ys`) and (`xe`, `ye`), respectively, by using `window()`.

Once this functions is called, each drawing function converts the coordinate system automatically because the application coordinate system is used in the EGGX drawing functions.

In the following example, the bottom-left and the top-right corner in the application coordinate system are set to (-20.0, -10.0) and (799.0, 599.0), respectively:

Example
```
window(win, -20.0, -10.0, 799.0, 599.0) ;
```

When the `BOTTOM_LEFT_ORIGIN` attribute in `gsetinitialattributes()` (§2.5.4) is disabled, the origin in the window coordinate system (`0`, `0`) is at the top-left corner.

You can also use `coordinate()` (§2.4.6) to change the coordinate system. Please consider it.

2.4.8 void layer(int wn, int lys, int lyw)

Description Configure a layer

In EGGX, each window for the graphics has eight layers. You can specify the layers for displaying and drawing separately. Set the window index for `wn`. Set the layer index for displaying and for drawing in the range of `0` to `7` for `lys` and `lyw`, respectively.

Consecutive execution of drawing functions against the currently displayed layer (in the case of `lys == lyw`) might cause decrease in drawing performance. If high-speed drawing is necessary, draws in the hidden layer and copy an image of the drawing layer to the displayed layer by using `copylayer()` (§2.4.9).

`layer(wn,0,0)` is set by default.

Example
```
layer(win,0,1) ;
```

2.4.9 void copylayer(int wn, int lysrc, int lydest)

Description Copy a layer

This function copies an image in the layer `lysrc` in the window `wn` to the layer `lydest` as it is. Copying is instantaneously done, so it can be used for playing an animation.

Example
```
copylayer(win,1,0) ;
```

2.4.10 void gsetbgcolor(int wn, const char *argsformat, ...)

Description Set the background color of a window

This function changes the background color (initialized with `gclr()` (§2.4.11)) of the windows specified with `wn`. The character string specified with `argsformat` (and the subsequent arguments) is set to the background color. These arguments are variable arguments as is the case with `printf()` in C standard functions. Specify a color included in `rgb.txt`[5] on the X server or hexadecimal RGB values such as `"#c0c0ff"` as this string of the background color.

Example
```
gsetbgcolor(win,"white") ;
```

2.4.11 void gclr(int wn)

Description Clear the drawing layer

This function initializes the drawing layer of the windows specified with `wn` with the colors specified with `gsetinitialbgcolor()` (§2.5.6) or `gsetbgcolor()` (§2.4.10). If the color is not specified with `gsetbgcolor()` and `gsetinitialbgcolor()`, it is initialized with the black.

Example
```
gclr(win) ;
```

2.4.12 void tclr(void)

Description Clear a terminal

This function clears a terminal, and restores the cursor position to the home position.

Example
```
tclr() ;
```

2.4.13 void newpen(int wn, int cn)

Description Change a drawing color

This function changes a drawing color in the window specified with `wn`. The correspondence between `cn` and the colors is as follows:

0: Black	1: White	2: Red	3: Green	4: Blue	5: Cyan	6: Magenta	7: Yellow
8: DimGray	9: Gray	10: red4	11: green4	12: blue4	13: cyan4	14: magenta4	15: yellow4

The colors end with letter "4" such as red4 and green4, which are dark red, dark green.

White is set by default.

Example
```
newpen(win,2) ;
```

[5] `rgb.txt` may be included in /usr/X11R6/lib/X11/ in UNIX OS.

2.4.14 void newcolor(int wn, const char *argsformat, ...)

Description Change a drawing color

This function changes a drawing color in the window specified with `wn`. The character string specified with `argsformat` (and the subsequent arguments) is set to the drawing color. These arguments are variables argument as is the case with `printf()` in C standard functions. Specify a color included in `rgb.txt`[6] on the X server or hexadecimal RGB values such as `"#c0c0ff"` as this string of the drawing color.

Example
```
newcolor(win,"Violet") ;
```

2.4.15 void newrgbcolor(int wn, int r, int g, int b)

Description Change a drawing color

This function changes a drawing color in the window specified with `wn`. Specify the brightness of Red, Green, and Blue in an integer in the range of 0 to 255 as `r,g,b`.

Example
```
newrgbcolor(win,255,127,0) ;
```

2.4.16 void newhsvcolor(int wn, int h, int s, int v)

Description Change a drawing color

This function changes a drawing color in the window specified with `wn`. Specify Hue, Saturation, and Value as `h`, `s`, and `v`, respectively [7]. Specify an integer in the range of 0 to 255 as `s` and `v`. Specify an integer in the range of 0 to 359 (angle) as `h`.

Example
```
newhsvcolor(win,120,250,240) ;
```

2.4.17 int makecolor(int cmode, double dmin, double dmax, double data, int *r, int *g, int *b)

Description Generate a color from a variables (generate a color bar)

This function generates 256 RGB color values to `r`, `g` and `b` from the values of a variable `data`. Specify the minimum and the maximum of the variable with `dmin` and `dmax`. The values of `r`, `n` and `b` acquired through this function can be used to set for the argument of `newrgbcolor` (§2.4.15) as it is.

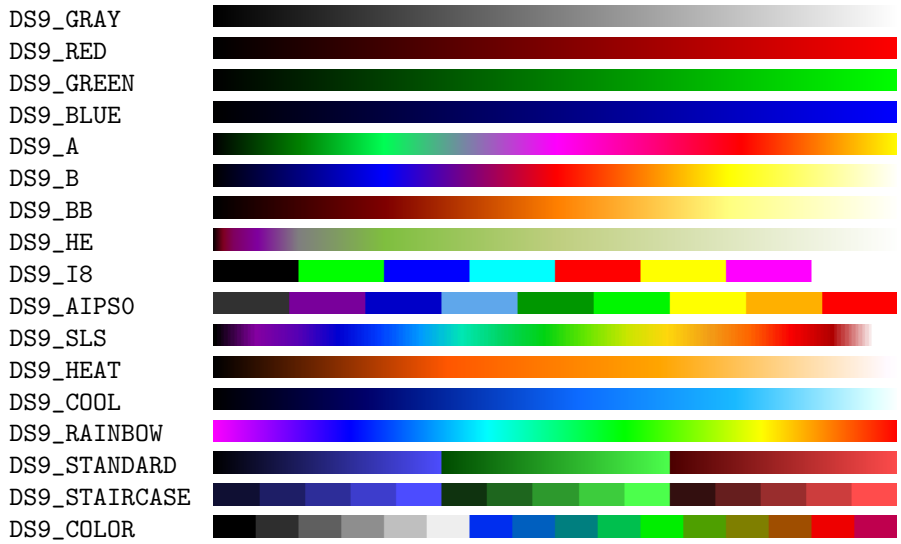
This function returns `0` if `data` is in the range of `dmin` to `dmax`, a negative value if `data` is smaller than `dmin` or a positive value if `data` is larger than `dmax`.

Specify the color pattern index with `cmode`. About 50 kinds of the color patterns shown below are available. The names such as `DS9 GRAY` are the macros, and an actual value is an integer starting from 0. See `eggx_color.h` for more information.

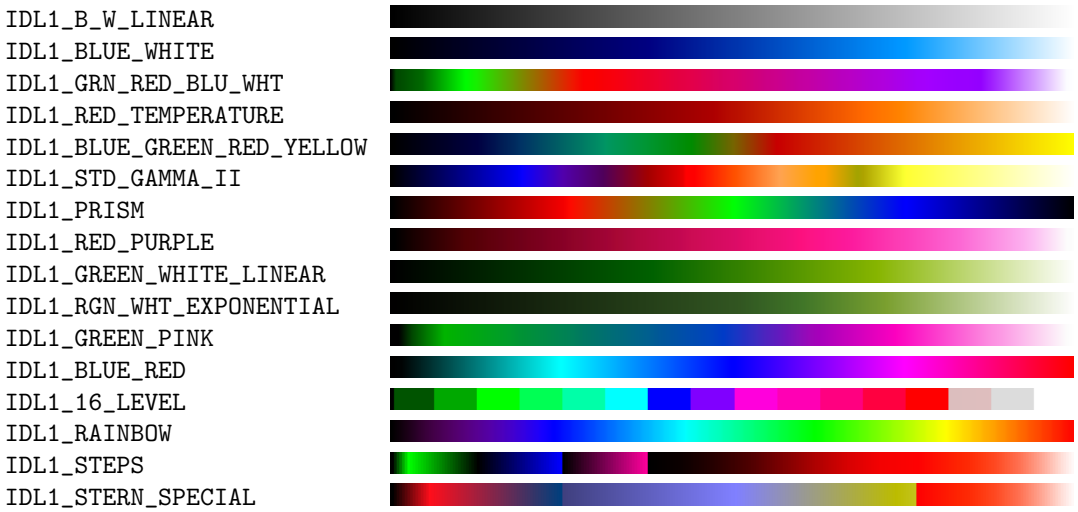
The following are the fits viewer DS9-compatible color patterns.

[6] `rgb.txt` may be included in /usr/X11R6/lib/X11/ in UNIX OS.

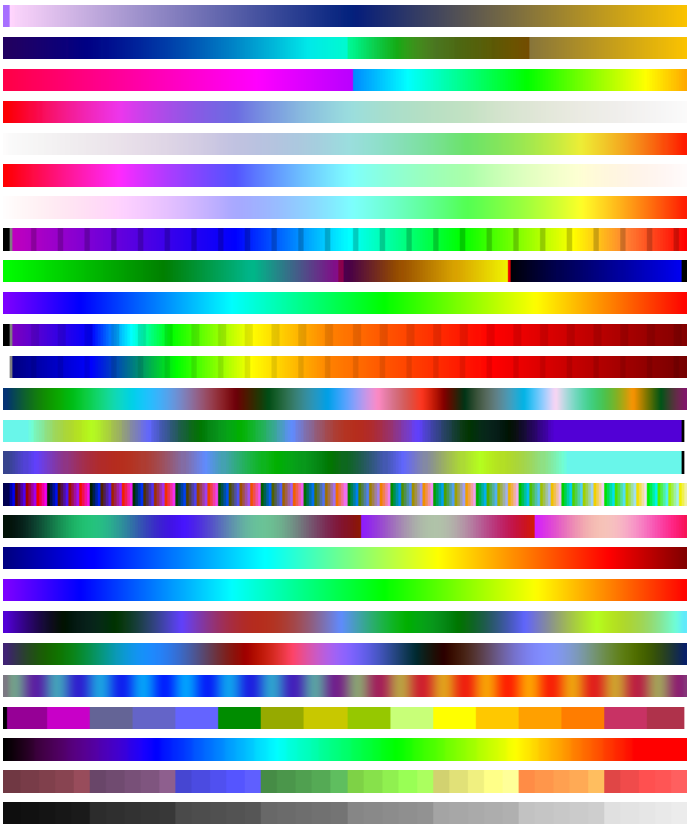
[7] The `newhsvcolor()` function is offered by Mr. Yasuda, Kyoto Sangyo University.



The following are the visualization software IDL-compatible color patterns.



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



Example `makecolor(DS9_SLS,v_min,v_max,v,&r,&g,&b) ;`

2.4.18 void newlinewidth(int wn, int width)

Description Change line width

This function changes the width of a line to be drawn in the window specified with `wn`. The line width 1 is set by default.

Note that once the line width is changed by this routine, graphics drawn by `drawsym()` (§2.4.34) or `drawarrow()` (§2.4.36), etc. are also affected.

Example `newlinewidth(win, 2) ;`

2.4.19 void newlinestyle(int wn, int style)

Description Change the line style

This function changes the style of a line to be drawn in the window specified with `wn`. Set `LineSolid` to draw a solid line or `LineOnOffDash` to draw a dashed line for the argument `style`. The solid line (`LineSolid`) is set by default.

Note that once the style of a line is changed by this routine, graphics drawn by `drawsym()` (§2.4.34) or `drawarrow()` (§2.4.36), etc. are also affected.

Example `newlinestyle(win, LineOnOffDash) ;`

2.4.20 void newgcfunction(int wn, int fnc)

Description Change the GC function

This function changes the GC function when executing of drawing functions (including image transfers) in the window specified with `wn`.

The GC function is a mechanism to determine the RGB values to be drawn by the logical operation of the source RGB value (for example, an RGB value specified in the `newcolor()`) and the RGB value of the drawing destination pixel.

Sixteen kinds of GC functions shown below can be specified. `GXcopy` is set by default. "src" and "dst" indicate the source RGB value and the RGB value of the drawing destination pixel, respectively.

Function Name	Logical Operation
`GXclear`	0
`GXand`	src AND dst
`GXandReverse`	src AND NOT dst
`GXcopy`	src
`GXandInverted`	(NOT src) AND dst
`GXnoop`	dst
`GXxor`	src XOR dst
`GXor`	src OR dst
`GXnor`	(NOT src) AND (NOT dst)
`GXequiv`	(NOT src) XOR dst
`GXinvert`	NOT dst
`GXorReverse`	src OR (NOT dst)
`GXcopyInverted`	NOT src
`GXorInverted`	(NOT src) OR dst
`GXnand`	(NOT src) OR (NOT dst)
`GXset`	1

"`GXxor`" is one of the commonly used GC functions. Since it sets the value of XOR from the source RGB value and the RGB value of the drawing destination pixel, the RGB value of a pixel is restored to the former value after it is drawn twice. This is often used when the cursor drawing and the region are displayed. Additionally, it can be used in such cases as the reversal of an image and the combination of Red, Green, and Blue.

The GC function set by this function is effective in various drawing functions, `gputarea()` (§2.4.40), and `gputimage()` (§2.4.41).

With a GC function that produces different results each time something is drawn in the same area more than once being specified, the result of the graphics drawn by `drawsym()` (§2.4.34), `drawsyms()` (§2.4.35), and `drawarrow()` (§2.4.36) is not defined.

Example
```
newgcfunction(win, GXxor) ;
```

2.4.21 void pset(int wn, double x, double y)

Description Draw a point

This function draws a point in the window specified with `wn`.

In the application coordinate system using by drawing functions, the bottom-left corner is (`0.0`, `0.0`) and the top-right corner is (`xsize-1.0`, `ysize-1.0`) by default. This coordinate system can be changed with `coordinate()` (§2.4.6) or `window()` (§2.4.7).

Example
```
pset(win,gx,gy) ;
```

2.4.22 void drawline(int wn, double x0, double y0, double x1, double y1)

Description Draw a line

This function draws a line from (`x0`, `y0`) to (`x1`, `y1`) in the window specified with `nw`.

Example
```
drawline(win,gx0,gy0,gx1,gy1) ;
```

2.4.23 void moveto(int wn, double x, double y), void lineto(int wn, double x, double y)

Description Draw a line continuously

Lines can be drawn continuously in the window specified with `wn` by using `lineto()` multiple times.

The `moveto()` function sets (`x`, `y`) as the initial position of `lineto()`. The `lineto()` function draws a line from the point specified when `moveto()` or `lineto()` were called previously to (`x`, `y`). It might make sense that `moveto()` is like lifting and moving a pen and `lineto()` is like lowering a pen and drawing something.

Example
```
lineto(win,gx,gy) ;
```

2.4.24 void drawpts(int wn, const double x[], const double y[], int n)

Description Draw points

This function draws `n` points in the window specified with `wn`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put the coordinate of each point to `x[0]` to `x[n-1]` and `y[0]` to `y[n-1]` preliminarily.

When the arguments `x` and `y` are float type arrays, the same function name as that of double type is available.

Example
```
drawpts(win,plx,ply,5) ;
```

2.4.25 void drawlines(int wn, const double x[], const double y[], int n)

Description Draw a polygonal line

This function draws a polygonal line in the window specified with `wn`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put the coordinate of each point of the polygonal line to `x[0]` to `x[n-1]` and `y[0]` to `y[n-1]` preliminarily. When the arguments `x` and `y` are float type arrays, the same function name as that of double type is available.

Example
```
drawlines(win,plx,ply,5) ;
```

2.4.26 void drawpoly(int wn, const double x[], const double y[], int n)

Description Draw a polygon

This function draws a polygon in the window specified with `wn`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put each vertex coordinate of the polygon to `x[0]` to `x[n-1]` and `y[0]` to `y[n-1]` preliminarily.

When the arguments `x` and `y` are float type arrays, the same function name as that of double type is available.

Example
```
drawpoly(win,plx,ply,5) ;
```

2.4.27 void fillpoly(int wn, const double x[], const double y[], int n, int i)

Description Fill a polygon

This function fills a polygon in the window specified with `wn`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put each vertex coordinate of the polygon to `x[0]` to `x[n-1]` and `y[0]` to `y[n-1]` preliminarily. `1` represents the shape of the polygon to be filled. Set `0` normally or `1` in the case of a convex polygon for `i`.

When the arguments `x` and `y` are float type arrays, the same function name as that of double type is available.

Example
```
fillpoly(win,plx,ply,5,0) ;
```

2.4.28 void drawrect(int wn, double x, double y, double w, double h)

Description Draw a rectangle

This function draws a rectangle beginning at (`x`, `y`) with the given width `w` and the height `h` in the window specified with `wn`.

Example
```
drawrect(win,50.0,60.0,30.0,20.0) ;
```

2.4.29 void fillrect(int wn, double x, double y, double w, double h)

Description Fill a rectangle

This function fills a rectangle beginning at (`x`, `y`) with the given width `w` and the height `h` in the window specified with `wn`.

Example
```
fillrect(win,50.0,60.0,30.0,20.0) ;
```

2.4.30 void drawcirc(int wn, double xcen, double ycen, double xrad, double yrad)

Description Draw a circle by specifying the center coordinate and the radius

This function draws a circle with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `wn`.
`circle` can be used as an alias. The following two example codes work identically:

Example
```
drawcirc(win,50.0,60.0,30.0,40.0) ;   circle(win,50.0,60.0,30.0,40.0) ;
```

2.4.31 void fillcirc(int wn, double xcen, double ycen, double xrad, double yrad)

Description Fill a circle by specifying the center coordinate and the radius

This function fills a circle with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `wn`.

Example
```
fillcirc(win,50.0,60.0,30.0,40.0) ;
```

2.4.32 void drawarc(int wn, double xcen, double ycen, double xrad, double yrad, double sang, double eang, int idir)

Description Draw an arc

This function draws an arc with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `wn`. `sang` and `eang` are the angle to start and to end, respectively, expressed in degrees. `idir` is a direction to draw the arc to. When `1` is set for `idir` it draws the arc counterclockwise, and when `-1` is set it draws the arc clockwise.

Example
```
drawarc(win,50.0,60.0,30.0,40.0,-10.0,-170.0,-1) ;
```

2.4.33 void fillarc(int wn, double xcen, double ycen, double xrad, double yrad, double sang, double eang, int idir)

Description Fill an arc

This function fills an arc with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `wn`. `sang` and `eang` are the angle to start and to end, respectively, expressed in degrees. `idir` is a direction to draw the arc to. When `1` is set for `idir` it draws the arc counterclockwise, and when `-1` is set it draws the arc clockwise.

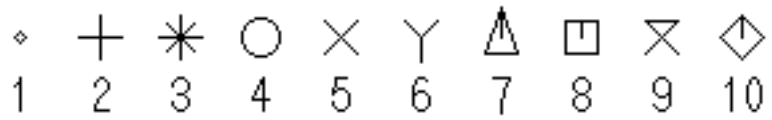
Example
```
fillarc(win,50.0,60.0,30.0,40.0,-10.0,-170.0,-1) ;
```

2.4.34 void drawsym(int wn, int x, int y, int size, int sym_type)

Description Draw a center symbol

This function draws a center symbol at the position (`x`, `y`) in the window specified with `wn`. Specify the symbol size in pixels with `size` and the kind of the symbol with `sym_type`.

The correspondence between `sym_type` and the symbols is as follows:



Example
```
drawsym(win,gx,gy,16,2) ;
```

2.4.35 void drawsyms(int wn, const double x[], const double y[], int n, int size, int sym_type)

Description Draw symbols

This function draws `n` number of symbols in the window specified with `wn`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put a coordinate of each symbol to `x[0]` to `x[n-1]` and `y[0]` to `y[n-1]` preliminarily.

Specify the symbol size in pixels with `size` and the kind of the symbol by `sym_type`.

See §2.4.34 for the correspondence between `sym_type` and the symbols.

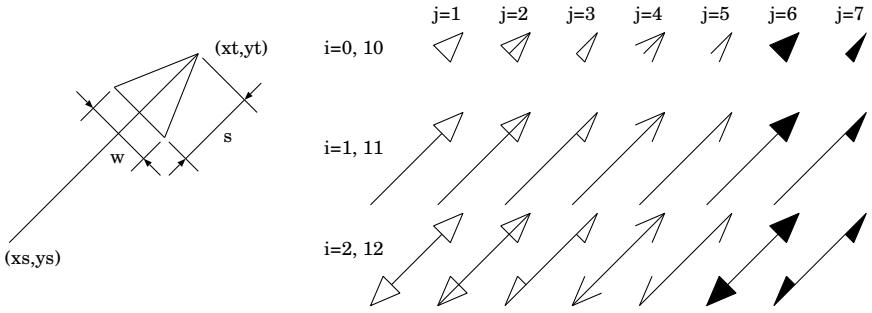
If the arguments `x` and `y` are a float typed array, this function can be used under the same name as that of double type.

Example
```
drawsyms(win,plx,ply,5,16,8) ;
```

2.4.36 void drawarrow(int wn, double xs, double ys, double xt, double yt, double s, double w, int 10*i+j)

Description Draw various types of arrows

Specify `s` and `w` in real numbers to decide the shape of the arrow as follows. When `i` is in the range of `0` to `2`, set pixels for `w` and `s`. When `i` is in the range of `10` to `12`, set the ratios against the arrow length in the range of `0.0` to `1.0`.



Example
```
drawarrow(win,gx0,gy0,gx1,gy1,0.3,0.2,114) ;
```

2.4.37 int newfontset(int wn, const char *argsformat, ...)

Description Specify a font set

This function sets a font set to draw in the window specified with `wn`. To draw a string, use `drawstr()`.

The string specified with `argsformat` (and the subsequent arguments) is set as the font set. Since the arguments after `argsformat` are variable arguments as is the case with `printf()` in C standard functions,

it is very convenient if the arguments are given as the example below, for example when enlarging or reducing a string.

The return value is `0` if the specified font set can be acquired; it is positive if the alternative font set can be acquired; or it is negative if it fails to acquire the font set.

To set the font set, you need to specify a font installed to the X server, so it depends on your OS and distribution. To ensure that a string is displayed, we recommend the following settings:

14 dot font	`"-*-fixed-medium-r-normal--14-*"`
16 dot font	`"-*-fixed-medium-r-normal--16-*"`
24 dot font	`"-*-fixed-medium-r-normal--24-*"`

Example
```
st = newfontset(win,"-kochi-gothic-medium-r-normal--%d-*-*-*-*-*-*-*",fsize) ;
```

Example
```
st = newfontset(win,"-adobe-helvetica-medium-r-*-*-12-*-*-*-*-*-iso8859-1,"
                    "-*-fixed-medium-r-normal--14-*") ;
```

2.4.38 int drawstr(int wn, double x, double y, int size, double theta, const char *argsformat, ...)

Description Draw a string

This function draws a string at the position `(x,y)` in the window specified with `wn`. Specify a character size in pixels for `size`. `theta` is the argument to specify the rotation of a string, however, it is disabled in the current version. The string specified with `argsformat` (and the subsequent arguments) is drawn. Since the arguments after `argsformat` are variable arguments as is the case with `printf()` in C standard functions, values of variables can also be drawn as the example below.

The character size `size` can be specified in the range of 1 to 24. The correspondence between `size` and the actual font size is shown in the table below. In this case, only English one byte characters can be drawn.

To draw a multi byte character (Kanji), set `FONTSET` for `size`. Use `newfontset()` (§2.4.37)) to specify a font in this case. If a font is not specified with `newfontset()`, a string is drawn in the default 14-dot font set.

The return value of this function is the length of the drawn string.

`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						

Example
```
drawstr(win,gx,gy,16,0,"velocity v=\%f",v) ;
```

Example
```
drawstr(win,gx,gy,FONTSET,0,"It is also possible to draw specified font.") ;
```

2.4.39 void gscroll(int wn, int inc_x, int inc_y, int clr)

Description Scroll the drawing layer in pixel unit

This function scrolls the drawing layer of the windows specified with `wn` for the amount equivalent to (`inc_x`, `inc_y`) pixels. The arguments `inc_x` and `inc_y` give the increment in the window coordinate system (the coordinate value is an integer). When `clr` is `0`, it scrolls with the left, right, top, and bottom of the screen all connected. However, when `clr` is `1`, the part where the scroll in was done is initialized with the background color.

Example
```
gscroll(win,0,2,1) ;
```

When the `BOTTOM_LEFT_ORIGIN` attribute is nullified by the gsetinitialattributes() function (§2.5.4), it scrolls down as shown in the above example.

2.4.40 void gputarea(int wn, double x, double y, int src_wn, int src_ly, double src_xs, double src_ys, double src_xe, double src_ye)

Description Copy an image data in any window, layer and area to the drawing layer

This function copies an image data in the range from (`src_xs`, `src_ys`) to (`src_xe`, `src_ye`) in the layer `src_ly` in the window `src_wn` to the position (`x`, `y`) in the window `wn`. The origin of the copied image is set to (`x`, `y`).

While the origin of the image copied from `src_wn` is regarded as the lower left corner by default, it is regarded as the top-left corner if `BOTTOM_LEFT_ORIGIN` is disabled in `gsetinitialattributes()` (§2.5.4).

Behaviors of this function are affected by the settings of `newgcfunction()` (§2.4.20).

Example
```
gputarea(win,gx,gy, wn1,0, 0.0,0.0,99.0,99.0) ;
```

2.4.41 int gputimage(int wn, double x, double y, unsigned char *buf, int width, int height, int msk)

Description Transfer images (with a mask) in a memory buffer to the drawing layer (background can be transparent)

This function transfers image data with the width `width` and the height `height` prepared in `buf` to the position (`x`, `y`) in the window specified with `wn` collectively. While the origin of the image prepared in the buffer is regarded as the bottom-left corner by default, it is regarded as the top-left corner if `BOTTOM_LEFT_ORIGIN` is disabled in `gsetinitialattributes()` (§2.5.4).

Behaviors of this function are affected by the settings of `newgcfunction()` (§2.4.20).

In the buffer `buf`, data are stored in the order of Alpha(mask value), Red, Green and Blue, scanning the image horizontally from top down (that is, buf[0]=0x0ff, buf[1]=red[0], buf[2]=green[0], buf[3]=blue[0],...). Set the level of brightness in the range of `0x000` to `0x0ff` for the values of Red, Green and Blue, and set 0x0ff (opacity) or 0x000 (transparency) for the mask value.

Configure the presence of the mask value using the argument `msk`. To enable the mask value (to enable transparent background) set `1`, or otherwise set `0` (other values than these are to be used when Alpha is supported in the future).

In case of errors, (e.g., the depth of the X server is less than 16) this function returns a negative value; otherwise it returns `0`. The depth of the X server can be checked by `ggetdisplayinfo()` (§2.5.1).

Using `tools/ppmtoh.c` or `tools/xpmtoh.c`[8] included in the EGGX/ProCALL source package, the values and the array to be given to the fourth to sixth arguments of `gputimage()` can be generated from a ppm or xpm file. It is convenient if you create a header file as follows.

```
$ ./ppmtoh my_image1.ppm >> my_images.h
$ ./xpmtoh my_image1.xpm >> my_images.h
```

Example
```
gputimage(win,gx,gy,buffer,640,400,1) ;
```

2.4.42 unsigned char *ggetimage(int wn, int ly, double xs, double ys, double xe, double ye, int *r_width, int *r_height)

Description Load an image data in any window, layer and area to a memory buffer

This function loads image data in the range from (`xs`, `ys`) to (`xe`, `ye`) in the layer `ly` in the window `wn` to a memory buffer and returns an address of the buffer. The width and height of the loaded image data are returned to `*r_width` and `*r_height`.

In the buffer `buf`, data are stored in the order of Alpha(mask value), Red, Green and Blue, scanning the image horizontally from top down (that is, buf[0]=0x0ff, buf[1]=red[0], buf[2]=green[0], buf[3]=blue[0],...). Set the level of brightness in the range of `0x000` to `0x0ff` for the values of Alpha, Red, Green and Blue.

The returned buffer has to be released with `free()` from a user's program.

This function returns `NULL` in case of errors, for example where a value given to an argument is invalid.

Example
```
buffer = ggetimage(win, 0, 0.0,0.0, 639.0,399.0, &width, &height) ;
```

[8] netpbm needs to be installed to use xpmtoh.

2.4.43 int gsaveimage(int wn, int ly, double xs, double ys, double xe, double ye, const char *conv, int nd, const char *argsformat, ...)

Description Save an image in any window, layer, and area to a file through a converter (netpbm, etc.)

This function runs a background process and saves an image in the range from (`xs`, `ys`) to (`xe`, `ye`) in the layer `ly` in the window `wn` to a file through a converter (various commands of netpbm, or convert of ImageMagick, etc.[9]). Set the command line of the converter to convert from ppm to each image format for `conv`. For example, to save in the png format by using netpbm, set `"pnmtopng"`. When using ImageMagick, set `"convert"`. Of course, option switches can be included, so such commands as `"pnmtops -scale 0.125"` are also enabled. `nd` is the subtractive color parameter. Set a gradation level per each channel in R,G,B for it. As for `nd`, while around `16` is enough for simple graphics, set the maximum value `256` in case many colors are used.

Strings specified with `argsformat` (and subsequent arguments) are treated as filenames. Variable values can be included in the filename as shown in the example below, because these arguments are variable arguments as is the case with `printf()` in C standard functions. This is very convenient when making an animation.

Examples of commands in netpbm to convert from ppm to another image format are listed below. You can check the usage of each command by typing "`man command-name`" from your terminal. When using the convert command of ImageMagick, a file format is determined by the suffix of a filename (a string after a dot, for example, .jpg, .png or .eps).

Image Format	Converter Name
AutoCAD DXB	`ppmtoacad`
windows bitmap	`ppmtobmp`
Berkeley YUV	`ppmtoeyuv`
GIF	`ppmtogif`
NCSA ICR graphics	`ppmtoicr`
IFF ILBM	`ppmtoilbm`
Interleaf image	`ppmtoleaf`
HP LaserJet PCL 5 Color	`ppmtolj`
map	`ppmtomap`
Mitsubishi S340-10 printer	`ppmtomitsu`
Atari Neochrome .neo	`ppmtoneo`
PPC Paintbrush	`ppmtopcx`
portable graymap	`ppmtopgm`
Atari Degas .pi1	`ppmtopi1`
Macintosh PICT	`ppmtopict`
HP PaintJet	`ppmtopj`
HP PaintJet XL PCL	`ppmtopjxl`
X11 "puzzle"	`ppmtopuzz`
three portable graymaps	`ppmtorgb3`
DEC sixel	`ppmtosixel`
TrueVision Targa	`ppmtotga`

Image Format	Converer Name
Motif UIL icon	`ppmtouil`
Windows .ico	`ppmtowinicon`
XPM format	`ppmtoxpm`
Abekas YUV	`ppmtoyuv`
YUV triplets	`ppmtoyuvsplit`
DDIF	`pnmtoddif`
FIASCO compressed	`pnmtofiasco`
FITS	`pnmtofits`
JBIG	`pnmtojbig`
JFIF ("JPEG") image	`pnmtojpeg`
Palm pixmap	`pnmtopalm`
plain (ASCII) anymap	`pnmtoplainpnm`
Portable Network Graphics	`pnmtopng`
PostScript	`pnmtops`
Sun raster	`pnmtorast`
RLE image	`pnmtorle`
sgi image	`pnmtosgi`
Solitaire image recorder	`pnmtosir`
TIFF file	`pnmtotiff`
CMYK encoded TIFF	`pnmtotiffcmyk`
X11 window dump	`pnmtoxwd`

Converters other than described here can also be used as long as they can import ppm format data to the standard input and export converted data from the standard output.

To save in the ppm format directly without using a converter, set `""` for `conv`. Please note that the size of the saved file is rather big because the ppm format is uncompressed binary data.

Since this function transfers an image data from the X server and saves it, it takes a long time if the network speed is slow. In consideration of this, EGGX allows a child process to start and image data to be transferred from the X server and written to the disk. Therefore, another operation can be done immediately after `gsaveimage();` in a user's program. However, if a drawing function in EGGX is called immediately after `gsaveimage();` it is suspended until the transfer and the saving of an image are finished, because drawing on the X server cannot be performed until this child process is finished.

If you use `gsaveimage`, please make sure to call `gclose()` (§2.4.2) before finishing the program.

The return value is `-1` if the child process fails to start; or otherwise it is `0`.

[9] netpbm is distributed in http://sourceforge.net/projects/netpbm/, andImageMagick is distributed in http://www.imagemagick.org/.

Example `gsaveimage(win,0, 0.0,0.0, 639.0,399.0, "pnmtopng",256,"img\%d.png",i) ;`

This is an example to save in the png format. To save in the gif format, set `"ppmtogif"` for `conv`.

Example
```
gsaveimage(win,0, 0.0,0.0, 639.0,399.0,
"pnmtops -noturn -dpi 72 -equalpixels -psfilter -flate -ascii85",256,"figure.eps") ;
```

This is an example to save in the PostScript format. pnmtops in netpbm supports RunLength compression and GZIP compression (lossless compression). By adding "`-psfilter -flate -ascii85`" as above, the file size can be reduced without deterioration of image quality.

2.4.44 unsigned char *readimage(const char *conv, const char *filename, int *r_width, int *r_height, int *r_msk)

Description Load image data in a file to a memory buffer through a converter (netpbm, etc.)

This function loads image data in the file `filename` to a memory buffer through a converter (various commands of netpbm, or convert of ImageMagick, etc.) and returns an address of the buffer. The width and height of the loaded image data are returned to `*r_width` and `*r_height`. When Alpha value is loaded, a value not less than `1` is returned to `*r_msk`; or otherwise `0` is returned. If Alpha value is `0x000` and `0x0ff` binary, `1` is returned.

Set a command line of a converter to convert each image format to any one of pbm, pgm, ppm and pam for `conv`. For example, to load a png-format file by using netpbm, set `"pngtopnm"`(`"pngtopam"` in the newer version). When using ImageMagick, set `"convert"`. Also other converters than netpbm and ImageMagick are can be used as long as they import a content of `filename` to the standard input and export converted data from the standard output. Of course, option switches can be included to `conv`.

To load a file directly without using a converter, set `""` for `conv`. Please note that loadable formats are limited to binary pbm, pgm, ppm and pam.

In the memory buffer, an image data are stored in the order of Alpha (mask value), Red, Green and Blue, scanning the image horizontally from top down (that is, buf[0]=0x0ff, buf[1]=red[0], buf[2]=green[0], buf[3]=blue[0],...). Set the level of brightness in the range of `0x000` to `0x0ff` for the values of Alpha, Red, Green and Blue.

The returned buffer has to be released with `free()` from a user's program.

This function returns `NULL` in case of errors, for example where it fails to read the file.

Example `buffer = readimage("pngtopnm", "myimage.png", &width, &height, &msk) ;`

2.4.45 int writeimage(const unsigned char *buf, int width, int height, int msk, const char *conv, int nd, const char *argsformat, ...)

Description Save image data in a memory buffer through a converter (netpbm, etc.)

This function saves image data with the width `width` and the height `height` stored in the memory buffer specified with `buf` through a converter (various commands of netpbm, or convert of ImageMagick, etc). Set a value not less than `1` for the argument `msk` when saving alpha value (mask); or otherwise set `0`. Set a command line of a converter to convert the ppm or pam format to each image format for `conv`. For example, to save in the png format by using netpbm, set `"pnmtopng"`(without alpha) or `"pamrgbatopng"`(with alpha). When using ImageMagick, set `"convert"`. Of course, option switches can be included, so such commands as "pnmtops -scale 0.125" are also enabled. `nd` is the subtractive color parameter. Set a gradation level per channel in R,G,B for it. As for `nd`, while around `16` is enough for simple graphics, set the maximum value `256` in case many colors are used.

Strings specified with `argsformat` (and subsequent arguments) are treated as filenames. Variable values can be included in the filename as shown in the example below, because these arguments are variable arguments as is the case with `printf()` in C standard functions. This is very convenient when making an animation.

In the buffer `buf`, data are stored in the order of Alpha(mask value), Red, Green and Blue, scanning the image horizontally from top down (that is, buf[0]=0x0ff, buf[1]=red[0], buf[2]=green[0], buf[3]=blue[0],...). Set the level of brightness in the range of `0x000` to `0x0ff` for the values of Alpha, Red, Green and Blue.

When using the convert command of ImageMagick, file format is determined by suffix of filename (a string after a dot, for example, .jpg, .png or .eps).

Also other converters than netpbm and ImageMagick can be used as long as they can import ppm or pam-format data to the standard input and export converted data from the standard output.

To save in the ppm or pam format directly without using a converter, set `""` for `conv`. Please note that the size of the saved file is rather big because the ppm and pam format is uncompressed binary data.

Whether ppm or pam is outputted to a converter or a file is determined by the argument `msk` and a suffix of a filename. When the suffix of a filename is `".pam"` or `msk` is not less than `1`, pam is outputted. Otherwise, ppm is outputted.

The return value is a negative value in case of errors, for example where the child process fails to start; or otherwise it is `0`.

See §2.4.43 for more information about the commands of converters.

Example
```
writeimage(buffer,640,400,0,"pnmtopng",256,"img\%d.png",i) ;
```

2.4.46 void gsetnonblock(int flag)

Description Configure the operating mode of `ggetch()`, `ggetevent()`, and `ggetxpress()`

When `ggetch()`, `ggetevent()`, and `ggetxpress()`, which are the functions to get input information from the keyboard or the mouse, are called, they wait inside themselves until an input occurs by default (in the blocking mode).

If `gsetnonblock()` is called with `flag` set to `ENABLE`, it changes to the non-blocking mode, and then `ggetch()`, `ggetevent()`, and `ggetxpress()` will return immediately whether an input occurs or not.

To restore the default blocking mode, set `flag` to `DISABLE`.

`gsetnonblock()` can be called anytime before or after a window is opened.

Example
```
gsetnonblock(ENABLE) ;
```

2.4.47 int ggetch()

Description Return a character inputted from the keyboard

This function returns input information from the keyboard from all windows opened in EGGX. While it waits until a key input occurs in the blocking mode (default), it finishes immediately whether a key input occurs or not in the non-blocking mode. (See `gsetnonblock()` in §2.4.46 for the operation mode.) A negative value is returned if any input does not occur in the non-blocking mode.

While `ggetch()` is similar to the function `fgetc(stdin)` to input a character from a terminal, there are differences: (1) the former does not wait until the break; and (2) it picks up the input of "special key" 0x001∼0x01f, 0x07f and "Ctrl + Alphabet."

The following table indicates hexadecimal character codes. The upper figures of two hexadecimal-digits are shown in Italic.

	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

For example, "a" is 0x061 and "A" is 0x041. There are some special keys in the range of 0x001 - 0x01a. Their codes are shared with those of Ctrl + Alphabet. For example, both codes of BackSpace and Ctrl + H are 0x008. As for blanks in the range of 0x001 - 0x01a, codes are assigned only to Ctrl + Alphabet.

Use `ggetxpress()` (§2.4.49) to check the index of a window where keyboard input occured.

Example
```
key=ggetch() ;
```

2.4.48 int ggetevent(int *type, int *button, double *x, double *y)

Description Return input information from the mouse or the keyboard

This function returns input information from the mouse or the keyboard from all windows opened in EGGX. While it waits until an input occurs in the blocking mode (default), it finishes immediately whether an input occurs or not in the non-blocking mode. (See `gsetnonblock()` in §2.4.46 for the operation mode.) A negative value is returned if any input does not occur in the non-blocking mode.

The return value of this function is the index of a window where an input occurred. By using this value, check whether the input occurred in the intended window in a user's program or not.

The values returned to `*type` are `MotionNotify` for mouse motion, `ButtonPress` where the mouse button is pressed, `ButtonRelease` where the mouse button is released, or `KeyPress` for input from the keyboard.

In the case of input from the mouse, click on/off or the index of the clicked button (`1, 2, 3,...`) is returned to `*button`, and the mouse pointer position (application coordinate system) upon a click is returned to `*x`, `*y`.

For key input, the key code is returned to `*button`. The key code is the same as the return value of `ggetch()` (§2.4.47).

If you do not need to get each value of `type`, `button`, `x` and `y`, you can set `NULL` for them. In C++ codes, `NULL` has to be casted to `double` or `float`.

If the arguments `x` and `y` are a float typed pointer variable, this function can be used with the same name as that of double type.

Example
```
win_ev=ggetevent(&type,&b,&x,&y) ;
```

2.4.49 int ggetxpress(int *type, int *button, double *x, double *y)

Description Return information of clicking mouse buttons or input from the keyboard

This function returns information of clicking mouse buttons or input from the keyboard from all windows opened in EGGX. This function is a simplified version of `ggetevent()` (§2.4.48). While it waits until an input occurs in the blocking mode (default), it finishes immediately whether an input occurs or not in the non-blocking mode. (See `gsetnonblock()` in §2.4.46 for the operation mode.) A negative value is returned if any input does not occur in the non-blocking mode.

The return value of this function is the index of a window where an input occurred. By using this value, check whether the input occurred in the intended window in a user's program or not.

The values returned to `*type` are `ButtonPress` for clicking mouse buttons or `KeyPress` for input from the keyboard.

For clicking mouse buttons, the index of the clicked button (`1, 2, 3,...`) and the mouse pointer position (application coordinate system) upon a click are returned to `*button` and `*x`, `*y`, respectively.

For key input, the key code is returned to `*button`. The key code is the same as the return value of `ggetch()` (§2.4.47).

If you do not need to get each value of `type`, `button`, `x` and `y`, you can set `NULL` for them. In C++ codes, `NULL` has to be casted to `double` or `float`.

If the arguments `x` and `y` are a float typed pointer variable, this function can be used with the same name as that of double type.

Example
```
win_ev=ggetxpress(&type,&b,&x,&y) ;
```

2.4.50 void msleep(unsigned long msec)

Description Wait an execution in milliseconds

This function waits for at least `msec` millisecond doing nothing. This can be used for adjusting animation speed.

Generally, the accuracy of the time is in the order of 10 millisecond.

Example
```
msleep(100) ;
```

2.5 EGGX ADVANCED FUNCTIONS REFERENCE

Functions described below are for more detailed control. Some of them are difficult to use for beginners or people who don't have enough knowledge on C. We don't offer FORTRAN routines corresponding to these functions.

2.5.1 int ggetdisplayinfo(int *depth, int *root_width, int *root_height)

Description Get information on the X server (depth and screen size)

This function connects to the X server and checks its depth and screen size. The values returned to `*depth` are those of the depth used upon opening of a window in EGGX, for example `8`(PseudoColor: 256 colors), `16`(TrueColor: 65536 colors), and `24`(TrueColor: 16 million colors). The screen size in pixels of the X server is returned to `*root_width` and `*root_height`.

If you do not need to get each value of `depth`, `root_width` and `root_height`, you can set `NULL` for them.

When the connection to the X server succeeds, the return value of the function becomes `0`. If it fails, it becomes a negative value.

Example
```
status = ggetdisplayinfo(&depth,NULL,NULL) ;
```

2.5.2 void gsetnonflush(int flag), int ggetnonflush()

Description Configure the flush in drawing functions

By default, EGGX functions on drawing or window decoration call `XFlush()` just before the end of the functions and the X server reflects a command from EGGX immediately (auto flush mode). However, on some X servers, drawing performance might be lowered by using `XFlush()` too often.

(Once `gsetnonflush()` is called with `flag` set to `ENABLE`, `XFlush` will never be called inside the EGGX functions. Then you can flush whenever you want by calling `gflush()` (§2.5.3) from a user's program (manual flush mode).

To change the setting back to the default, set `flag` to `DISABLE`. `gsetnonflush()` can be called anytime before or after a window is opened.

`ggetnonflush()` gets the current value on the settings of auto flush. It returns the value set by `gsetnonflush()`.

Example
```
gsetnonflush(ENABLE) ;
```

2.5.3 void gflush()

Description Flush drawing commands

This function flushes a series of commands against the X server ordered by drawing functions. Use this when the manual flush mode is set by `gsetnonflush()`.

Example
```
gflush() ;
```

2.5.4 void gsetinitialattributes(int values, int att_msk)

Description Set the attributes of the window opened with `gopen()`

This function sets various window attributes opened with `gopen()`. Once an attribute is set by `gsetinitialattributes()`, the attribute is reflected to all windows opened with `gopen()` after that.

The window attribute is expressed with four-bit flags shown below, `values` is specified with `ENABLE`(all is effective) or `DISABLE` (all is invalid) or the arbitrary property value, `att_msk` is specified by a change flag to each attribute (mask).

- `BOTTOM_LEFT_ORIGIN`

  This attribute sets the bottom-left corner of a window as the origin (0,0) of the window coordinate system. This attribute is enabled by default.

  If this attribute is set to disabled, the origin is changed to the top-left of the window.

- `SCROLLBAR_INTERFACE`

  The scrollbar interface by EGGX is offered. This attribute is enabled by default.

  If this attribute is disabled, the scrollbar does not appear even if a window is resized.

- `OVERRIDE_REDIRECT`

  The OverrideRedirect attribute in X is specified. This attribute is disabled by default.

  The window opened with the OverrideRedirect attribute is not interposed by a window manager and has no window frame. This window is displayed on top at all times. This mode is sometimes used for displaying a banner upon launch of an application.

- `DOCK_APPLICATION`

  If this attribute is enabled, the window is configured to become an applet of AfterStep or WindowMaker. This attribute is disabled by default.

  If this attribute is enabled, the scrollbar does not appear even if the window is resized.

The following is an example to enable the `DOCK_APPLICATION` attribute:

Example
```
gsetinitialattributes(ENABLE, DOCK_APPLICATION) ;
```

To change multiple attributes, set as follows. In this example, `OVERRIDE_REDIRECT` is enabled and `BOTTOM_LEFT_ORIGIN` is disabled.

Example
```
gsetinitialattributes(OVERRIDE_REDIRECT, BOTTOM_LEFT_ORIGIN | OVERRIDE_REDIRECT) ;
```

To set all bits of the attribute, set `-1` for `att_msk` (that is, set 1 for all bits) as follows:

Example
```
gsetinitialattributes(SCROLLBAR_INTERFACE | BOTTOM_LEFT_ORIGIN, -1) ;
```

2.5.5 int ggetinitialattributes()

Description Get the attributes of the window opened with `gopen()`

This function reads the current attribute value for `gopen()`. The value set by `gsetinitialattributes()` is returned.

Example
```
att=ggetinitialattributes() ;
```

2.5.6 void gsetinitialbgcolor(const char *argsformat, ...)

Description Set the background color of the window opened with `gopen()`

This function specifies the background color (that is initialized with `gclr()` (§2.4.11)) of the windows opened with `gopen()`. The character string specified with `argsformat` (and the subsequent arguments) is set to the background color. These arguments are variable arguments as is the case with `printf()` in C standard functions. Specify a color included in `rgb.txt`[10] on the X server or hexadecimal RGB values such as `"#c0c0ff"` as this string of the background color.

If `NULL` is set for `argsformat`, it restore to the default setting (black).

Example
```
gsetinitialbgcolor("white") ;
```

2.5.7 void gsetborder(int wn, int width, const char *argsformat, ...)

Description Set the width and color of window borders

This changes the width and color of borders (frames) of a the window specified with `wn`. Set the border width in dots for the argument `width`. The string specified with `argsformat` (and subsequent arguments) is set as the border color. The third and higher arguments are variable arguments as is the case with `printf()` in C standard functions.

[10] `rgb.txt` may be included in /usr/X11R6/lib/X11/ in UNIX OS.

Generally, since the window border is reset by a window manager, the value set by this function is disabled. However, as for a window that has the OverrideRedirect attribute, this setting is reflected. See the description of `gsetinitialattributes()` (§2.5.4) for the OverrideRedirect attribute.

If the `width` is a negative value or `argsformat` is `NULL`, the setting is not changed.

Example
```
gsetborder(win,1,"white") ;
```

2.5.8 void gsetinitialborder(int width, const char *argsformat, ...)

Description Set the borders of the window opened with `gopen()`

This specifies the width and color of borders (frames) of the window opened with `gopen()`. Set the border width in pixels for the argument `width`. The string specified with `argsformat` (and subsequent arguments) is set as the border color. The second and higher arguments are variable arguments as is the case with `printf()` in C standard functions.

If the `width` is a negative value or `argsformat` is `NULL`, the setting is not changed.

The border width and the border color are set to `0` and `Black`, respectively, by default.

Example
```
gsetinitialborder(1,"White") ;
```

2.5.9 void gsetinitialgeometry(const char *argsformat, ...)

Description Set the size and the appearance position of new windows from a string

This specifies the size and the appearance position of new window opened with `gopen()` from strings specified with `argsformat` (and subsequent arguments). The second and higher arguments are variable arguments as is the case with `printf()` in C standard functions. Command options following "`-geometry`" such as string `"800x600+100-200"`, which are standard in X11 clients, can be set for the argument. If `argsformat` is `NULL`, this function restores to the default setting, which means neither size nor the appearance position is specified. Integer values also can be set for the argument as follows:

Example
```
gsetinitialgeometry("800x600\%+d\%+d",-30,40) ;
```

2.5.10 void gsetinitialwinname(const char *storename, const char *iconname, const char *resname, const char *classname)

Description Set a window name, an icon name, a resource name and a class name of new windows

A window name, an icon name, a resource name, and a class name are determined from a command name of a user's program. However, by using this function, those names of new window opened with `gopen()` can be specified with the arguments `storename`, `iconname`, `resname` and`classname`.

If `NULL` is set for each argument, the corresponding name is restored to the default setting.

Example
```
gsetinitialwinname("AyuClock","AyuClock","ayuclock","AyuClock") ;
```

2.5.11 void gsetscrollbarkeymask(int wn, unsigned int keymask)

Description Set a keymask for the key operation of EGGX's scrollbar

EGGX offers a scrollbar interface when `gopen()` is called with the argument greater than the display size or when a user resizes a window.

This scrollbar can be moved by pressing the Alt-arrow key by default. With this function, you can change this specification.

Set the values listed in the following table for argument `keymask`:

Mask Value	Description
ShiftMask	Key operation of the scrollbar is enabled only while the Shift key is pressed
LockMask	Key operation of the scrollbar is enabled only while the CapsLock key is pressed
ControlMask	Key operation of the scrollbar is enabled only while the Ctrl key is pressed
Mod1Mask	Key operation of the scrollbar is enabled only while the Alt key is pressed
Mod2Mask	Key operation of the scrollbar is enabled only while NumLock is on
Mod5Mask	Key operation of the scrollbar is enabled only while ScrollLock is on
0	Key operation of the scrollbar is enabled at all times

Note that if 0 is set for keymask, input event of arrow keys is disabled in ggetch() (§2.4.47), etc.

Example
```
gsetscrollbarkeymask(win, ShiftMask) ;
```

2.5.12 int generatecolor(color_prms *p, double dmin, double dmax, double data, int *r, int *g, int *b)

Description Generate a color from variables (contrast, brightness and gamma can be modified)

This generates 256 RGB color values to r, g, and b from the values of a variable data. Specify the minimum and the maximum of the variable with dmin and dmax. The values of r, g, and b can be used to set for the argument of newrgbcolor() (§2.4.15) as it is.

This function returns 0 if data is between dmin and dmax, -1 if data is smaller than dmin or 1 if data is larger than dmax.

In the current version of EGGX, the struct color_prms consists of the following members, however, this may be extended in the future. So it is not recommended to set the initial values upon declaration of variables.

```
typedef struct _color_prms {
    int colormode ;
    int flags ;
    double contrast ;
    double brightness ;
    double gamma ;
    int seplevel ;
    void *ptr ;
    void (*function)( double,void *,double,double,double,double *,double *,double * ) ;
} color_prms ;
```

colormode is the index of a color pattern. See makecolor() (§2.4.17) for more information.

flags is the flag to enable the contrast contrast, the brightness brightness, the gamma correction gamma, the color separation level seplevel, and/or the user function function.

Flag	Corresponding Member	Description
C_REVERSE	-	Reverse the order of the color pattern for the value of data
CP_CONTRAST	contrast	Enable the control of the contrast.($0 \leq$ contrast ≤ 1)
CP_BRIGHTNESS	brightness	Enable the control of the brightness.($0 \leq$ brightness ≤ 1)
CP_GAMMA	gamma	Enable the gamma correction.($0 \leq$ gamma ≤ 1)
CP_SEPLEVEL	seplevel	Enable the color separation level.($2 \leq$ seplevel)
CP_FUNCTION	function	Enable to call user functions.

Using seplevel, linear color gradation can be digitized. For example, the color changes from black to white linearly in DS9_GRAY8, however, it changes in a phased manner in the range of 10 colors when seplevel is set to 10.

function is a user function executed at the end of processing by generatecolor(). Arguments of function are normalized data by Max=1.0 and Min=0.0, ptr and values of Red, Green, and Blue (in and out), respectively. Use this part only if you understand the source code of EGGX.

Example
```
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

3.1 ProCALL ROUTINES LIST

3.1.1 ProCALL Standard Routines

Chapter	Routine	Description
§3.4.1	ggetdisplayinfo	Get informations (depth, display size) of the X server
§3.4.2	gopen	Open a graphics window of any size
§3.4.3	gclose	Close any windows for graphics
§3.4.4	gcloseall	Close all windows for graphics and disconnect from the X server
§3.4.5	newcoordinate	Change the application coordinate system (set the reference point and the scale)
§3.4.6	newwindow	Change the application coordinate system (set the the bottom-left and the top-right coordinates)
§3.4.7	layer	Configure a layer setting
§3.4.8	copylayer	Copy a layer
§3.4.9	gsetbgcolor	Set the background color (in `gclr`) of a window
§3.4.10	gclr	Clear a window for graphics
§3.4.11	tclr	Clear a terminal display
§3.4.12	newpencolor	Change a drawing color (16 colors)
§3.4.13	newcolor	Change a drawing color (specify the X server's colors directly)
§3.4.14	newrgbcolor	Change a drawing color (specify the brightness of Red, Green and Blue)
§3.4.15	newhsvcolor	Change a drawing color (specify Hue, Saturation and Value)
§3.4.16	makecolor	Generate a color from variables (generate a color bar)
§3.4.17	newlinewidth	Change line width
§3.4.18	newlinestyle	Change line style
§3.4.19	pset	Draw a point
§3.4.20	drawline	Draw a line
§3.4.21	moveto, lineto	Draw a line continuously
§3.4.23	drawpts	Draw points
§3.4.24	drawlines	Draw a polygonal line
§3.4.25	drawpoly	Draw a polygon
§3.4.26	fillpoly	Fill a polygon
§3.4.27	drawrect	Draw a rectangle
§3.4.28	fillrect	Fill a rectangle
§3.4.29	drawcirc	Draw a circle by specifying the center coordinate and the radius
§3.4.30	fillcirc	Fill a circle by specifying the center coordinate and the radius
§3.4.31	drawarc	Draw an arc by specifying the center and radius of a circle and the angle of the starting point and the ending point
§3.4.32	fillarc	Fill an arc by specifying the center and radius of a circle and the angle of the starting point and the ending point
§3.4.33	drawsym	Draw a symbol
§3.4.34	drawsyms	Draw muliple symbols
§3.4.35	drawarrow	Draw various types of arrows
§3.4.36	newfontset	Specify a font set (Japanese font)
§3.4.37	drawstr	Draw a string
§3.4.38	drawnum	Draw a real number
§3.4.39	putimg24	Transfer images prepared in an integer array to a window collectively
§3.4.40	saveimg	Save an image to a file through the converter (netpbm, ImageMagick)
§3.4.41	gsetnonblock	Configure the operating mode of ggetch, ggetevent and ggetxpress routines
§3.4.42	ggetch	Return a character inputted from the keyboard
§3.4.43	ggetevent	Return input information from the mouse or the keyboard
§3.4.44	ggetxpress	Return information of clicking mouse buttons or input from the keyboard
§3.4.45	selwin	Specify a window to be accessed with Calcomp-compatible routines

3.1.2 Calcomp-Compatible Routines

Chapter	Routine	Description
§3.5.1	plots	Open a 640 × 400 dot window for graphics
§3.5.2	window	Change coordinate system
§3.5.3	newpen	Change drawing color
§3.5.4	clsc	Clear a terminal display
§3.5.5	clsx	Clear a graphics display
§3.5.6	plot	Draw a line or a point
§3.5.7	arc	Draw an arc by specifying the center and radius of a circle and the angle of the starting point and the ending point
§3.5.8	circ1	Draw a circle by specifying the center coordinate and the radius
§3.5.9	arohd	Draw various types of arrows
§3.5.10	symbol	Draw a string or a symbol
§3.5.11	number	Draw a real number

3.1.3 Auxiliary Routines

Chapter	Routine name	Function
§3.6.1	msleep	Wait an execution in milliseconds
§3.6.2	isnan	Check whether a real number variable is "Not a Number"
§3.6.3	rtoc	Convert a real number variable to a string

3.2 BASIC USAGE

There is no special notice on user programs. After coding a program, use the `egg` command for compiling.

Example
```
egg program.f -o program
```

3.3 HOW TO SPEED UP DRAWING

Draw always in an invisible layer when you use drawing routines, by using `layer` (§3.4.7) and `copylayer` (§3.4.8). After finishing drawing, copy the invisible layer to the visible layer by using `copylayer`. This approach can enhance the drawing speed appreciably.

3.4 ProCALL STANDARD ROUTINES REFERENCE

3.4.1 ggetdisplayinfo(ndepth,nrwidth,nrheight)

Description Get information (depth and screen size) on the X server

This routine connects to the X server and check its depth and screen size. The values returned to `ndepth` are those of the depth used upon opening of a window in ProCALL, for example `8` (PseudoColor: 256 colors), `16` (TrueColor: 65536 colors), and `24` (TrueColor: 16 million colors) is returned to `ndepth`. The screen size of the X server is returned to `nrwidth` and `nrheight`. If it fails to connect to the X server, a negative value is returned to `ndepth`.

Example
```
call ggetdisplayinfo(ndepth,nrwidth,nrheight)
```

3.4.2 gopen(nxsize,nysize,nw)

Description Open a graphics screen of any size

This routine opens a window that has a drawing area of arbitrary size.

Specify horizontal and vertical pixels of the drawing area by the argument `nxsize` and `nysize`, respectively. The maximum pixels are 32767.

An integer window index used in ProCALL is returned to `nw`. Since the window index generated by ProCALL is set to `nw` when `gopen` is called, **users do not need to set a value to `nw`.** Use this window index to set it to the drawing routines in ProCALL.

If the drawing area that is almost the same size as or larger than the root window (background) is specified, a smaller window than the drawing area is opened by default [11]. In this case, the scrollbar interface is offered, and it is possible to display anywhere in the drawing area by using the mouse or the keyboard.

The window accessed with compatible routines is specified with the window index `nw` in the routine selwin(§3.4.45).

Example
```
call gopen(800,600,nwin)
```

3.4.3 gclose(nw)

Description Close a window for graphics

This routine closes any window specified with `nw`.

Example
```
call gclose(nwin)
```

3.4.4 gcloseall

Description Close all windows for graphics and disconnect from the X server

This routine closes all windows, disconnects from the X server, and frees the memory area used by the internal processing of the library.

Example
```
call gcloseall
```

3.4.5 newcoordinate(nw,nxw,nyw,xa,ya,xscale,yscale)

Description Change the application coordinate system (set the reference point and the scale)

This routine changes the application coordinate system in the window specified with `nw`.

The bottom-left corner is (`0`, `0`) and the top-right corner is (`nxsize-1`, `nysize-1`) in the window coordinate system. (The coordinate value is an integer.) The coordinate value of the application coordinate system corresponds to that of the window coordinate system by default. (The coordinate value of the application coordinate system is a real number.)

(`xa`, `ya`) of the application coordinate system can correspond to (`nxw`, `nyw`) of the window coordinate system by using `newcoordinate`, and each scaling factor is specified with `xscale` and `yscale`. This means that the conversion from the application coordinates (x, y) to the window coordinates (`nx`, `ny`) in the drawing functions etc. is done by the following expressions:

$\texttt{nx} = \texttt{nxw} + (x - \texttt{xa}) \cdot \texttt{xscale}$
$\texttt{ny} = \texttt{nyw} + (y - \texttt{ya}) \cdot \texttt{yscale}$

Once this routine is called, each drawing routine converts the coordinate system automatically because the application coordinate system is used in the ProCALL drawing routines.

In the following example, the bottom-left of the window is set to (-40.0, -20.0) in the application coordinate system, and scaling factors of both x and y are set to 2.0.

Example
```
call newcoordinate(nwin, 0,0, -40.0,-20.0, 2.0,2.0)
```

You can also use `newwindow` (§3.4.6) to change the coordinate system. Please consider it.

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

Description Change the application coordinate system (set the the bottom-left and the top-right coordinates)

This routine changes the application coordinate system in the window specified with `nw`. (The size of an actual graphics area does not change.)

The bottom-left corner is (`0`, `0`) and the top-right corner is (`nxsize-1`, `nysize-1`) in the window coordinate system. (The coordinate value is an integer.) The coordinate value of the application coordinate system correspond to that of the window coordinate system by default. (The coordinate value of the application coordinate system is a real number.)

[11] If Xinerama is available upon the build of EGGX/ProCALL, it is compared with the pixel size of the smallest display.

The bottom-left corner ((0, 0) in the window coordinate system) and the top-right corner can be changed to (xs, ys) and (xe, ye), respectively, by using newwindow.

Once this routine is called, each drawing routine convert the coordinate system automatically because the application coordinate system is used in the ProCALL drawing routines.

In the following example, the bottom-left and the top-right corner in the application coordinate system are set to (-20.0, -10.0) and (799.0, 599.0), respectively.

Example
```
call newwindow(nwin, -20.0, -10.0, 799.0, 599.0)
```
You can also use newcoordinate (§3.4.5) to change the coordinate system. Please consider it.

3.4.7 layer(nw,lys,lyw)

Description Configure a layer

In ProCALL, each window for graphics has eight layers. You can specify the layers for displaying and drawing separately. Set the window index for nw. Set the layer index for displaying and for drawing in the range of 0 to 7 for lys and lyw, respectively.

Consecutive execution of drawing routines against the currently displayed layer (in the case of lys == lyw) might cause decrease in drawing performance. If high-speed drawing is necessary, draw in the hidden layer and copy an image of the drawing layer to the displayed layer by using the copylayer routine (§3.4.8).

layer(wn,0,0) is set by default.

Example
```
call layer(nwin,0,1)
```

3.4.8 copylayer(nw,lysrc,lydest)

Description Copy a layer

This routine copies the image in the layer lysrc in the window nw to the layer lydest as it is. Copying is instantaneously done, so it can be used for playing an animation.

Example
```
call copylayer(nwin,1,0)
```

3.4.9 gsetbgcolor(nw,strc)

Description Set the background color of a window

This routine changes the background color (initialized with gclr) of the window specified with nw.

For strc, specify a color included in rgb.txt[12] in the X server and add "CHAR(0)" in the end of it just like 'Blue'//CHAR(0). You can also specify by hexadecimal RGB values such as '#c0c0ff'//CHAR(0).

Example
```
call gsetbgcolor(nwin,'white'//CHAR(0)) ;
```

3.4.10 gclr(nw)

Description Clear the drawing layer

This routine initializes the drawing layer with the color specified with gsetbgcolor (§3.4.9). If the color is not specified with gsetbgcolor, it is initialized with black.

Example
```
call gclr(nwin)
```

3.4.11 tclr

Description Clear a terminal

This routine clears a terminal, and restores the cursor position to the home position.

Example
```
call tclr
```

[12] rgb.txt may be included in /usr/X11R6/lib/X11/ in UNIX OS.

3.4.12 newpencolor(nw,nc)

Description Change a drawing color

This routine changes a drawing color in the window specified with `nw`. The correspondence between `nc` and the colors is as follows:

0: Black	1: White	2: Red	3: Green	4: Blue	5: Cyan	6: Magenta	7: Yellow
8: DimGray	9: Gray	10: red4	11: green4	12: blue4	13: cyan4	14: magenta4	15: yellow4

The colors end with letter "4" such as red4 and green4, which are dark red and dark green.

White is set by default.

Example
```
call newpencolor(nwin,2)
```

3.4.13 newcolor(nw,strc)

Description Change a drawing color

This routine changes a drawing color in the window specified with `nw`. For `strc`, specify a color included in `rgb.txt`[13] in the X server and add "`CHAR(0)`" to the end of it just like `'Blue'//CHAR(0)`. You can also specify by hexadecimal RGB values such as `'#c0c0ff'//CHAR(0)`.

Example
```
call newcolor(nwin,'Violet'//CHAR(0))
```

3.4.14 newrgbcolor(nw,nr,ng,nb)

Description Change a drawing color

This routine changes a drawing color in the window specified with `nw`. Specify the brightness of Red, Green, and Blue in an integer in the range of 0 to 255 as `nr,ng,nb`.

Example
```
call newrgbcolor(nwin,255,127,0)
```

3.4.15 newhsvcolor(nw,nh,ns,nv)

Description Change a drawing color

This routine changes a drawing color in the window specified with `nw`. Specify Hue, Saturation, and Value as `h`, `s`, and `v`, respectively. Specify an integer in the range of 0 to 255 as `s` and `v`. Specify an integer in the range of 0 to 359 (angle) as `h`.

Example
```
call newhsvcolor(nwin,120,250,240)
```

3.4.16 makecolor(ncolormode,dmin,dmax,data,nr,ng,nb)

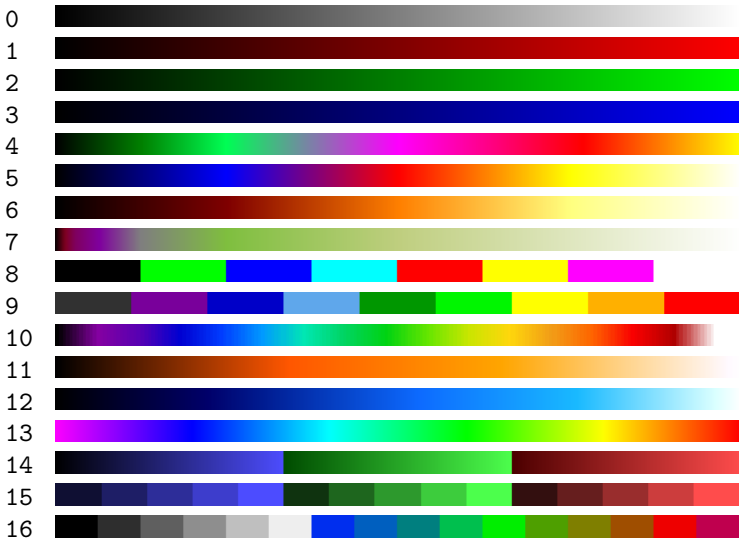
Description Generate a color from variables (generate a color bar)

This routine generates 256 RGB color values to `nr`, `ng` and `nb` from the values of a variable `data`. Specify the minimum and the maximum of the variable with `dmin` and `dmax`. The values of `nr`, `ng` and `nb` acquired through this routine can be used to set for the argument of `newrgbcolor` (§3.4.14) as it is.

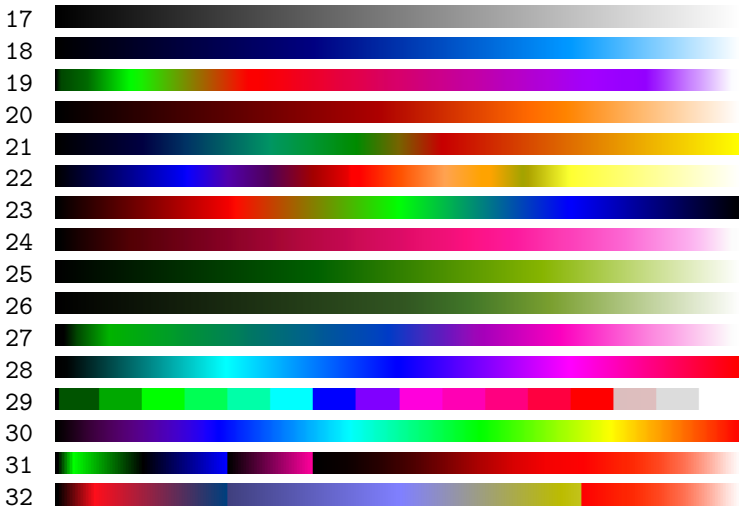
Specify the color pattern index with `ncolormode`. About 50 kinds of the color patterns shown below are available.

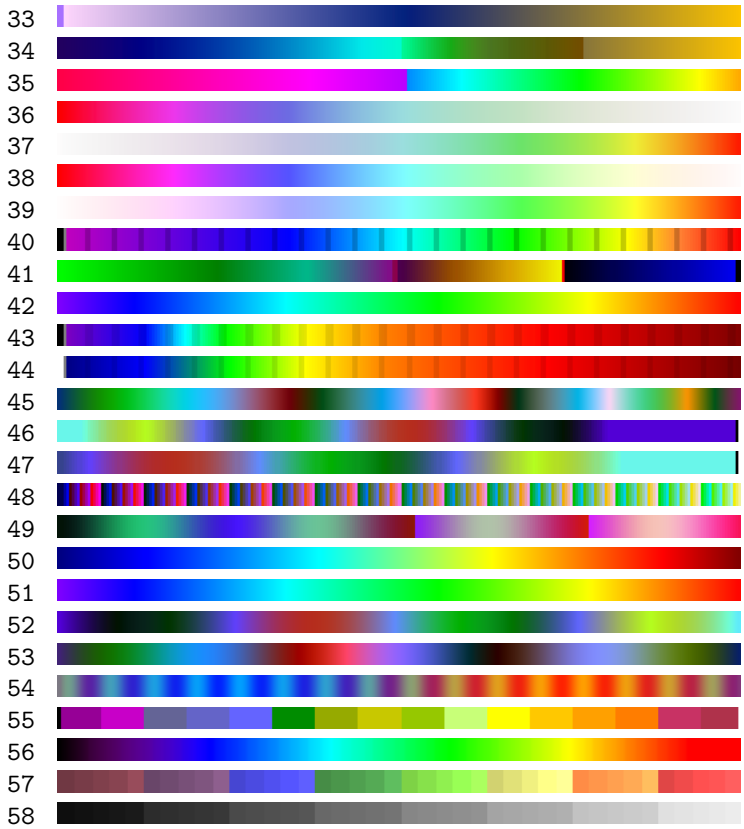
The following are the fits viewer DS9-compatible color patterns:

[13] `rgb.txt` may be included in /usr/X11R6/lib/X11/ in UNIX OS.



The following are the visualization software IDL-compatible color patterns:





Example `call makecolor(10,v_min,v_max,v,nr,ng,nb)`

3.4.17 newlinewidth(nw,nwidth)

Description Change line width

This routine changes the width of a line to be drawn in the window specified with `nw`. The line width 1 is set by default.

Note that once the line width is changed by this routine, graphics drawn by `drawsym` (§3.4.33) or `drawarrow` (§3.4.35), etc. are also affected.

Example `call newlinewidth(nwin, 2)`

3.4.18 newlinestyle(nw,nstyle)

Description Set line style

This routine changes the style of a line to be drawn in the window specified with `nw`. Set `0` to draw a solid line or `1` to draw a dashed line for the argument `style`. The solid line is set by default.

Note that once the style of a line is changed by this routine, graphics drawn by `drawsym` (§3.4.33) or `drawarrow` (§3.4.35), etc. are also affected.

Example `call newlinestyle(nwin, 1)`

3.4.19 pset(nw,xg,yg)

Description Draw a point

This routine draws a point in the window specified with `nw`.

In the application coordinate system used in drawing routines, the bottom-left corner is (`0.0`, `0.0`) and top-right corner is (`nxsize-1.0`, `nysize-1.0`) by default. This coordinate system can be changed with `newcoordinate` (§3.4.5) or `newwindow` (§3.4.6).

Example `call pset(nwin,x,y)`

3.4.20 drawline(nw,xg0,yg0,xg1,yg1)

Description Draw a line

This routine draws a line from (`xg0`, `yg0`) to (`xg1`, `yg1`) in the window specified with `nw`.

Example `call drawline(nwin,x0,y0,x1,y1)`

3.4.21 moveto(nw,xg,yg), lineto(nw,xg,yg)

Description Draw a line continuously

Lines can be drawn continuously in the window specified with `nw` by using `lineto` multiple times.

`moveto` sets (`xg`, `yg`) as the initial position of `lineto`. `lineto` draws a line from the point specified when `moveto` or `lineto` were called previously to (`xg`, `yg`) if `2` is set for `mode`. It might make sense that `moveto` is like lifting and moving a pen and `lineto` lowering a pen and drawing something.

`xg` and `yg` are real number type arguments.

Example `call lineto(nwin,x,y)`

3.4.22 line(nw,xg,yg,mode)

Description Draw a line continuously

This routine works just the same as `moveto` and `lineto` in §3.4.21.

Use `moveto` and `lineto` in a new program.

Lines can be drawn continuously in the window specified with `nw` by using `line` multiple times. This routine draws a line from the point specified when it was called previously to (`xg`, `yg`) if `2` is set for `mode`. If `3` is set for `mode`, (`xg`, `yg`) is set as the initial position of line routine. It might make sense that this routine is like lowering a pen and drawing something where `mode=2` and lifting and moving a pen where `mode=3`.

`xg` and `yg` are real number type arguments.

Example `call line(nwin,x,y,2)`

3.4.23 drawpts(nw,x,y,n)

Description Draw points

This routine draws `n` points in the window specified with `nw`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put the coordinate of each point to `x(1)` to `x(n)` and `y(1)` to `y(n)` preliminarily.

Example `call drawpts(nwin,x,y,5)`

3.4.24 drawlines(nw,x,y,n)

Description Draw a polygonal line

This routine draws a polygonal line in the window specified with `nw`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put the coordinate of each point of the polygonal line to `x(1)` to `x(n)` and `y(1)` to `y(n)` preliminarily.

Example `call drawlines(nwin,x,y,5)`

3.4.25 drawpoly(nw,x,y,n)

Description Draw a polygon

This routine draws a polygon in the window specified with `nw`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put each vertex coordinate of the polygon to `x(1)` to `x(n)` and `y(1)` to `y(n)` preliminarily.

Example `call drawpoly(nwin,x,y,5)`

3.4.26 fillpoly(nw,x,y,n,i)

Description Fill a polygon

This routine fills a polygon in the window specified with `nw`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put each vertex coordinate of the polygon to `x(1)` to `x(n)` and `y(1)` to `y(n)` preliminarily. `1` represents the shape of the polygon to be filled. Set `0` normally or `1` in the case of a convex polygon for `i`.

Example
```
call fillpoly(nwin,x,y,5,0)
```

3.4.27 drawrect(nw,x,y,w,h)

Description Draw a rectangle

This routine draws a rectangle beginning at (`x`, `y`) with the given width `w` and the height `h` in the window specified with `nw`.

Example
```
call drawrect(nwin,50.0,60.0,30.0,20.0)
```

3.4.28 fillrect(nw,x,y,w,h)

Description Fill a rectangle

This routine fills a rectangle beginning at (`x`, `y`) with the given width `w` and the height `h` in the window specified with `nw`.

Example
```
call fillrect(nwin,50.0,60.0,30.0,20.0)
```

3.4.29 drawcirc(nw,xcen,ycen,xrad,yrad)

Description Draw a circle by specifying the center coordinate and the radius

This routine draws a circle with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `nw`.

Example
```
call drawcirc(nwin,50.0,60.0,30.0,40.0)
```

3.4.30 fillcirc(nw,xcen,ycen,xrad,yrad)

Description Fill a circle by specifying the center coordinate and the radius

This routine fills a circle with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `nw`.

Example
```
call fillcirc(nwin,50.0,60.0,30.0,40.0)
```

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

Description Draw an arc

This routine draws an arc with with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `nw`. `sang` and `eang` are the angle to start and to end, respectively, expressed in degrees. `idir` is a direction to draw the arc to. When `1` is set for `idir` it draws the arc counterclockwise, and when `-1` is set it draws the arc clockwise.

Example
```
call drawarc(nwin,50.0,60.0,30.0,40.0,-10.0,-170.0,-1)
```

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

Description Fill an arc

This routine fills an arc with with (`xcen`, `ycen`) as a center, `xrad` as a horizontal radius, and `yrad` as a vertical radius in the window specified with `nw`. `sang` and `eang` are the angle to start and to end, respectively, expressed in degrees. `idir` is a direction to draw the arc to. When `1` is set for `idir` it draws the arc counterclockwise, and when `-1` is set it draws the arc clockwise.

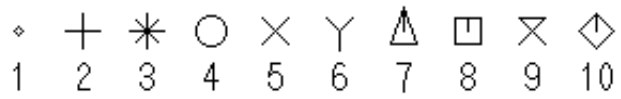
Example
```
call fillarc(nwin,50.0,60.0,30.0,40.0,-10.0,-170.0,-1)
```

3.4.33 drawsym(nw,xg,yg,size,nsym)

Description Draw a center symbol

This routine draws a center symbol at the position (`xg`, `yg`) in the window specified with `nw`. Specify the symbol size in pixels with `size` (real number) and the kind of the symbol by `nsym` (integer).

The correspondence between `nsym` and the symbols is as follows:



`xg` and `yg` are real number type arguments.

Example
```
call drawsym(nwin,x,y,16.0,2)
```

3.4.34 drawsyms(nw,x,y,n,size,nsym)

Description Draw symbols

This routine draws `n` number of symbols in the window specified with `nw`. `x` and `y` are one-dimensional arrays of `n` real numbers. Put a coordinate of each symbol to `x(1)` to `x(n)` and `y(1)` to `y(n)` preliminarily.

Specify the symbol size in pixels with `size` (real number) and the kind of the symbol by `nsym` (integer).

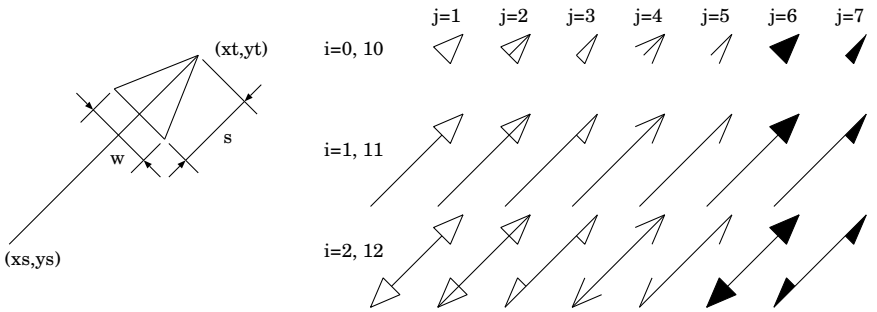
See §3.4.33 for the correspondence between `nsym` and the symbols.

Example
```
call drawsyms(nwin,x,y,5,16.0,8)
```

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

Description Draw various types of arrows

This routine draws an arrow from (`xs`, `ys`) to (`xe`, `ye`) in the window specified with `nw`. Specify `s` and `w` in real numbers to decide the shape of the arrow as follows. When `i` is in the range of `0` to `2`, set pixels for `w` and `s`. When `i` is in the range of `10` to `12`, set the ratios against the arrow length in the range of `0.0` to `1.0`.



Example
```
call drawarrow(nwin,x0,y0,x1,y1,0.3,0.2,114)
```

3.4.36 newfontset(nw,fontset,nstatus)

Description Specify a font set

This routine sets a font set to draw in the window specified with `nw`. To draw a string, use `drawstr` (§3.4.37).

Specify the font set name using `fontset`. The last character of `fontset` has to be "`CHAR(0)`" (termination character).

An integer value to indicate the status of the acquisition of the font set is returned to the last argument `status`. The value returned to `status` is `0` if the font set specified with `fontset` can be acquired; it is a positive value if an alternative font can be acquired; or it is a negative value if the font set cannot be acquired.

To set the font set, you need to specify a font installed to the X server, so it depends on your OS and distribution. To ensure taht a string is displayed, we recommend the following settings:

14 dot font	`'-*-fixed-medium-r-normal--14-*'//CHAR(0)`
16 dot font	`'-*-fixed-medium-r-normal--16-*'//CHAR(0)`
24 dot font	`'-*-fixed-medium-r-normal--24-*'//CHAR(0)`

Example
```
call newfontset(nwin,'-*-fixed-medium-r-normal--16-*'//CHAR(0),nstat)
```

3.4.37 drawstr(nw,xg,yg,size,str,theta,len)

Description Draw a string

This routine draws a string at the position (`xg`, `yg`) in the window specified with `nw`. Specify a character size in real number pixels for `size`. Set a string for `str` and the length of the string (integer) to `len`. You can set `-1` instead of the length of the string for `len` if the last character of the string `str` is the termination character "`CHAR(0)`". `theta` is the argument to specify rotation of a string, however, it is disabled in the current version.

The character size `size` can be specified in the range of `1` to `24`. The correspondence between `size` and the actual font size is shown in the table below. In this case, only English one byte characters can be drawn.

To draw a multi byte character (Kanji), set `0.0` for `size`. Use `newfontset` (§3.4.36) to specify a font in this case. If a font is not specified with `newfontset`, a string is drawn in the default 14-dot font set.

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

Example
```
call drawstr(nwin,x,y,16.0,'foo',0.0,4)
```

Example
```
call drawstr(nwin,x,y,0.0,'fooooo!'//CHAR(0),0.0,-1)
```

3.4.38 drawnum(nw,xg,yg,size,v,theta,n)

Description Draw a value of variables

This routine draws the value of the real number type variable `v` at the position (`xg`, `yg`) in the window specified with `nw`. Set a real number in pixels for `size` to specify the size of a string. Set an integer value for n to specify the number of decimal places of the value to be displayed. The real number type argument `theta` to specify the rotation of a string is disabled in the current version.

Example
```
call drawnum(nwin,x,y,16.0,prm,0.0,3)
```

3.4.39 putimg24(nw,x,y,nw,nh,nbuf)

Description Transfer images prepared in an integer array to a window collectively

This routine transfers images with the width `nw` and the height `nh` prepared in `nbuf` to the position (`x`, `y`) in the window specified with `wn` collectively.

In the integer array `nbuf`, data are stored in the order of Red, Green and Blue, scanning the image horizontally from top down (that is, `nbuf(1)=nRed(1)`, `nbuf(2)=nGreen(1)`, `nbuf(3)=nBlue(1)`, ...). Set the level of brightness in the range of `0` to `255` for the array.

If the depth of the X server is 8 or lower, this routine cannot be used because it does not work normally. The depth of the X server can be checked using ggetdisplayinfo (§3.4.1).

Example
```
call putimg24(nwin,x,y,640,400,nbuffer)
```

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

Description Save an image in a layer through a converter (netpbm, etc.)

This routine runs a background process and saves an image in the range from (`xs`, `ys`) to (`xe`, `ye`) in the layer `ly` in the window `wn` to a file through the converter (various commands of netpbm, or convert of ImageMagick, etc.[14]).

Set a filename for `fname` after adding a termination character `CHAR(0)` to the end of it just like `'image.png'//CHAR(0)`. If a positive number is set for `n`, the number is added to the filename. For example, if `12` is set for `n`, the filename becomes `image12.png`. If `n` is negative, the filename does not change. Set the command line of the converter to convert the format from ppm to each image format for `conv`. For example, to save in the png format by using netpbm, set `'pnmtopng'//CHAR(0)`. When using ImageMagick, set `'convert'//CHAR(0)`. Of course, option switches can be included, so such commands as `'pnmtops -scale 0.125'//CHAR(0)` are also enabled. The last argument `nd` is the subtractive color parameter. Set a gradation level per channel in R,G,B for it. As for `nd`, while around `16` is enough for simple graphics, set the maximum value `256` when many colors are used.

Examples of commands in netpbm to convert the format from ppm to another image format are listed below. You can check the usage of each command by typing "`man command-name`" from your terminal. When using the convert command of ImageMagick, a file format is determined by the suffix of a filename (a string after a dot, for example, `.jpg`, `.png` or `.eps2`).

Image Format	Converter Name
AutoCAD DXB	`ppmtoacad`
windows bitmap	`ppmtobmp`
Berkeley YUV	`ppmtoeyuv`
GIF	`ppmtogif`
NCSA ICR graphics	`ppmtoicr`
IFF ILBM	`ppmtoilbm`
Interleaf image	`ppmtoleaf`
HP LaserJet PCL 5 Color	`ppmtolj`
map	`ppmtomap`
Mitsubishi S340-10 printer	`ppmtomitsu`
Atari Neochrome .neo	`ppmtoneo`
PPC Paintbrush	`ppmtopcx`
portable graymap	`ppmtopgm`
Atari Degas .pi1	`ppmtopi1`
Macintosh PICT	`ppmtopict`
HP PaintJet	`ppmtopj`
HP PaintJet XL PCL	`ppmtopjxl`
X11 "puzzle"	`ppmtopuzz`
three portable graymaps	`ppmtorgb3`
DEC sixel	`ppmtosixel`
TrueVision Targa	`ppmtotga`

Image format	Converter name
Motif UIL icon	`ppmtouil`
Windows .ico	`ppmtowinicon`
XPM format	`ppmtoxpm`
Abekas YUV	`ppmtoyuv`
YUV triplets	`ppmtoyuvsplit`
DDIF	`pnmtoddif`
FIASCO compressed	`pnmtofiasco`
FITS	`pnmtofits`
JBIG	`pnmtojbig`
JFIF ("JPEG") image	`pnmtojpeg`
Palm pixmap	`pnmtopalm`
plain (ASCII) anymap	`pnmtoplainpnm`
Portable Network Graphics	`pnmtopng`
PostScript	`pnmtops`
Sun raster	`pnmtorast`
RLE image	`pnmtorle`
sgi image	`pnmtosgi`
Solitaire image recorder	`pnmtosir`
TIFF file	`pnmtotiff`
CMYK encoded TIFF	`pnmtotiffcmyk`
X11 window dump	`pnmtoxwd`

[14] netpbm is distributed in http://sourceforge.net/projects/netpbm/, and ImageMagick is distributed in http://www.imagemagick.org/.

Converters other than described here can also be used as long as they import ppm-format data to the standard input and export converted data from the standard output.

To save in the ppm format directly without using a converter, see "`CHAR(0)`" for `conv`. Please note that the size of the saved file is rather big because the ppm format is uncompressed binary data.

Since this routine transfers an image data from the X server and saves it, it takes a long time if the network speed is slow. In consideration of this, this routine allows a child process to start and image data to be transfered from the X server and written to the disk. Therefore, another operation can be done immediately after `call saveimg(...)` in a user's program. However, if a drawing routine in ProCALL is called immediately after `call saveimg(...)` it is suspended until the transfer and the saving of an image are finished, because drawing on the X server cannot be performed until this child process is finished.

If you use `saveimg`, please make sure to call gclose (§3.4.3) before finishing the program.

Example
```
call saveimg(nwin,0, 0.0,0.0, 639.0,399.0, 'img.png'//CHAR(0),i, 'pnmtopng'//CHAR(0),256)
```
This is an example to save in the png format. To save in the gif format, set "`'ppmtogif'//CHAR(0)`" for `conv`.

Example
```
call saveimg(nwin,0, 0.0,0.0, 639.0,399.0, 'fig.eps'//CHAR(0),-1,
     'pnmtops -noturn -dpi 72 -equalpixels -psfilter -flate -ascii85'//CHAR(0),256)
```
This is an example to save in the PostScript format. `pnmtops` in netpbm supports RunLength compression and GZIP compression (lossless compression). By adding "`-psfilter -flate -ascii85`" as above, the file size can be reduced without deterioration of image quality.

3.4.41 gsetnonblock(iflag)

Description Configure the operating mode of `ggetch`, `ggetevent` and `ggetxpress`

When `ggetch`, `ggetevent`, and `ggetxpress` which are the routines to get input information from the keyboard or the mouse, are called, they wait inside themselves until an input occurs by default (in the blocking mode).

If gsetnonblock is called with `iflag` set to `1`, it changes to the non-blocking mode, and then `ggetch`, `ggetevent`, and `ggetxpress` will return immediately whether an input occurs or not.

To restore the default blocking mode, set `iflag` to `0`.

gsetnonblock can be called anytime before or after a window is opened.

Example
```
call gsetnonblock(1)
```

3.4.42 ggetch(key)

Description Return a character inputted from the keyboard

This routine returns input information from the keyboard from all windows opened in ProCALL. The character code inputted from the keyboard is set to `key` (in integers). While it waits until a key input occurs in the blocking mode (default), it finishes immediately whether a key input occurs or not in the non-blocking mode. (See `gsetnonblock` in §3.4.41 for the operation mode.) A negative value is returned if any input does not occur in the non-blocking mode.

The following table indicates hexadecimal character codes. The upper figures of two hexadecimal-digits are shown in Italic. For example, "a" is z'61' and "A" is z'41'.

	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

There are some special keys in the range of z'01' to z'1A'. Their codes are shared with those of Ctrl + Alphabet. For example, both codes of BackSpace and Ctrl + H are z'08'. As for blanks in the range of z'01' to z'1A', codes are assigned only to Ctrl + Alphabet.

Example `call ggetch(nwin,key)`

3.4.43 ggetevent(nw,ntype,nbutton,xg,yg)

Description Return input information from the mouse or the keyboard

This routine returns input information from the mouse or the keyboard from all windows opened in ProCALL. While it waits until an input occurs in the blocking mode (default), it finishes immediately whether an input occurs or not in the non-blocking mode. (See gsetnonblock in §3.4.41 for the operation mode.) A negative value is returned if any input does not occur in the non-blocking mode.

The values returned to `nw` is the index of a window where an input occurred. By using this value, check whether the input occurred in the intended window in a user's program or not.

The value returned to `ntype` are `6` for mouse motion, `4` where the mouse button is clicked, `5` where the mouse button is released, or `2` in the case of input from the keyboard.

For input from the mouse, the values assigned to `nbutton` are click on/off or the clicked button (`1, 2, 3,`...). The mouse pointer position (in the application coordinate system) upon clicking is returned to `xg`, `yg`.

For key input, the key code is returned to `nbutton`. The key code is the same as the `key` of the ggetch (§3.4.42).

Example `call ggetevent(nwin,ntype,nb,x,y)`

3.4.44 ggetxpress(nw,ntype,nbutton,xg,yg)

Description Return information of clicking mouse buttons or input from the keyboard

This routine returns information of clicking mouse buttons or input from the keyboard from all windows opened with ProCALL. While it waits until an input occurs in the blocking mode (default), it finishes immediately whether an input occurs or not in the non-blocking mode. (See gsetnonblock in §3.4.41 for the operation mode.) A negative value is returned if any input does not occur in the non-blocking mode.

The index of a window where an input occurred is returned to `nw`. By using this value, check whether the input occurred in the intended window in a user's program or not.

The values assigned to `ntype` are `4` for clicking mouse buttons or `2` for input from the keyboard.

For clicking mouse buttons, the index of the clicked button (`1, 2, 3,`...) and the mouse pointer position (in the application coordinate system) upon clicking are returned to `nbutton` and `xg`, `yg`, respectively.

For key input, the key code is returned to `nbutton`. The key code is the same as the `key` of the ggetch (§3.4.42).

Example `call ggetxpress(nwin,ntype,nb,x,y)`

3.4.45 selwin(nw)

Description Specify a window to be drawn

By using this routine, specify which window you access for CALCOMP-compatible routines. Set the window index acquired through `gopen` (§3.4.2) for `nw`. The window index opened with the CALCOMP-compatible routine `plots` (§3.5.1) is set to `0`. The default value is `0`.

Example `call selwin(nwin)`

3.5 CALCOMP COMPATIBLE ROUTINES REFERENCE

Routines described in this section are Pro-FORTRAN-compatible routines. They might not be necessary except in the case of transitions from Pro-FORTRAN, other GKS or CalComp-compatible FORTRAN.

3.5.1 plots

Description Open a window for graphics

A window for graphics is opened by calling this routine. The graphics area (window size) is 640 × 400 pixels. Use `call gopen` to specify the graphics area or to open multiple windows.

A window index number of a window called by `plots` in ProCALL is 0. (Specify `0` for the window index number `nw` in ProCALL standard routines.)

Example
```
call plots
```

3.5.2 window(xs,ys,xe,ye)

Description Change a coordinate system

In the window coordinate system (whose value is an integer), the bottom-left corner is (0, 0) and the top-right corner is (639, 399). A coordinate value of the application coordinate system (whose value is a real number) equals to that of the window coordinate system.

The bottom-left corner ((`0`, `0`) in the window coordinate system) and the top-right corner in the application coordinate system can be changed to (`xs`, `ys`) and (`xe`, `ye`), respectively, by using the `window` routine.

In the following example, the bottom-left corner and the top-right corner in the application coordinate system are changed to (-2.0, -1.0) and (5.0, 4.0), respectively.

Example
```
call window(-2.0, -1.0, 5.0, 4.0)
```

3.5.3 newpen(nc)

Description Change a drawing color

This routine changes a drawing color in plot, etc. The correspondence between `nc` and the colors is as follows:

`0`: Black `1`: White `2`: Red `3`: Green `4`: Blue `5`: Cyan `6`: Magenta `7`: Yellow
`8`: DimGray `9`: Gray `10`: red4 `11`: green4 `12`: blue4 `13`: cyan4 `14`: magenta4 `15`: yellow4

The colors end with letter "4" such as red4 and green4 are dark red and dark green.

White is set by default.

Example
```
call newpen(2)
```

Compatibility The colors 8 to 15 cannot be used in the original Pro-FORTRAN.

3.5.4 clsc

Description Clear a terminal

This routine clears a terminal and restores a cursor position to the home position.

Example
```
call clsc
```

3.5.5 clsx

Description Clear a graphics display

Example
```
call clsx
```

3.5.6 plot(xg,yg,mode)

Description Draw a line or a point

This routine draws a line from the point specified when it was called previously to (xg, yg) if 2 is set for mode. (xg, yg) is set as the initial position of plot if 3 is set for mode. It might make sense that this routine lowering a pen and drawing somethig where mode=2 and lifting and moving a pen where mode=3.

Additionally, in the case of mode=1, it draws a point at (xg, yg) and update the pen position.

xg and yg are real number type arguments.

Example
```
call plot(x,y,2)
```

Compatibility mode=1 is not available in the original Pro-FORTRAN.

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

Description Draw an arc

This routine draws an arc with (xcen, ycen) as a center and rad as a radius. sang and eang are the angle to start and to end, respectively, expressed in degrees. idir is a direction to draw the arc to. When 1 is set for idir it draws the arc counterclockwise, and when -1 is set it draws the arc clockwise.

Example
```
call arc(50.0,60.0,30.0,-10.0,-170.0,-1)
```

3.5.8 circ1(xc,yc,r)

Description Draw a circle by specifying the center coordinate and the radius

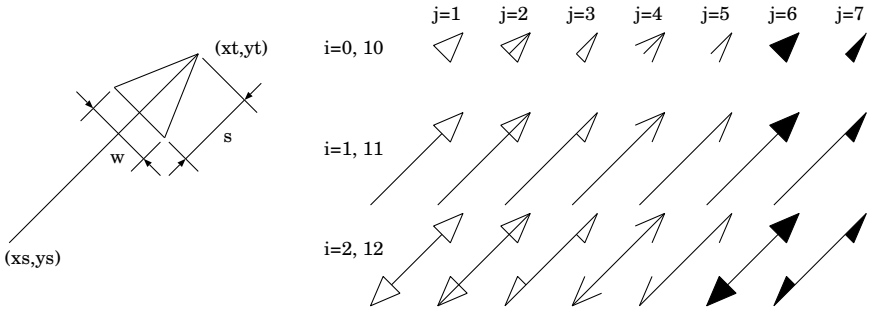
This routine draws a circle with (xc, yc) as a center and r as a radius.

Example
```
call circ1(50.0,60.0,30.0)
```

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

Description Draw various types of arrows

This routine draws an arrow from (xs, ys) to (xe, ye). Specify s and w in real numbers to decide the shape of the arrow as follows. When i is in the range of 0 to 2, set dot numbers for w and s. When i is in the range of 10 to 12, set the ratios against the arrow length in the range of 0.0 to 1.0.



Example
```
call arohd(x0,y0,x1,y1,0.3,0.2,114)
```

Compatibility i=10 to 12 are not available in the original Pro-FORTRAN. In the original Pro-FORTRAN, the size of the symbol actually drawn is changed by specifying window, however, it is not changed in Pro-CALL.

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

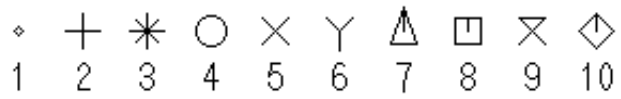
Description Draw a string or a center symbol

This routine draws a string or a center symbol at the position (`xg`, `yg`). Set a real number in dots for `size` to specify the size of a string or a symbol. To draw a string, set the string length (integer) for `len` and the string to `nstr`. To draw a symbol, set `-1` for `len` and the index of the symbol (integer from `1` to `10`) for `nstr`.

Character size `size` can be specified in the range of `1` to `24`. The correspondence between `size` and the actual font is as follows. As for characters, all of English one byte characters can be drawn. Furthermore, they are clearer than those in the original Pro-FORTRAN.

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

The correspondence between `nstr` and the symbols is as follows:



The fourth argument of this routine can be both string type and integer type. "Warning" will appear when compiling a source coded to draw both a string and a symbol by using this `symbol`. Since this "Warning" is quite annoying, we recommend the use of ProCALL standard routines `drawstr` (draw a string) and `drawsym` (draw a symbol) unless you compile in the original Pro-FORTRAN.

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

Compatibility In the original Pro-FORTRAN the size of the symbol actually drawn is changed by specifying `window`, however, it is not changed in ProCALL.

Any character size greater than 24 is fixed to a 12 × 24 dot font. Lower-case alphabets and some symbols cannot be displayed in the original Pro-FORTRAN. The real number type argument `theta` to specify the rotation of a string is disabled in the current version.

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

Description Draw a value of a variables

This routine draws the value of the real number type variable `v` at the position (`xg`, `yg`). Set a real number in dots for `size` to specify the size of a string. Set an integer value for n to specify the number of decimal places of the value to be displayed.

Example
```
call number(x,y,16.0,prm,0.0,3)
```

Compatibility Any character size greater than 24 is fixed to a 12 × 24 dot font. The real number type argument `theta` to specify the rotation of a string is disabled in the current version.

3.5.12 vport,setal

Description Dummy routine

These routines do not work.

Compatibility Only the formats of these routines are being kept so that a Pro-FORTRAN source code can be compiled as it is.

3.6 AUXILIARY ROUTINES REFERENCE

Routines described here are offered to resolve inconvenience caused by FORTRAN as much as possible.

3.6.1 msleep(ms)

Description | Wait an execution in milliseconds

This routine waits for `ms` millisecond doing nothing. Set an integer value up to 999 for `ms`. This can be used for adjusting animation speed.

Example
```
call msleep(100)
```

3.6.2 isnan(v,iflg)

Description | Check whether a real number variable is Not a Number

This routine checks whether a real number variable `v` is Not a Number (NaN), and returns an integer value other than 0 to `iflg` if `v` is NaN.

Example
```
call isnan(x,nf)
```

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

Description | Convert a real number variable to a string

This routine converts a real number `v` into a string with `n` decimal places, and stores it to `str` after adding the delimiter `'\0'` to the end of it. Set a number of characters of `str` for `ns`. An excess number of characters (integer) in storing the string to `str` is returned to `m`. If this value is negative, it is indicated that the number of characters of `str` is not enough to store the whole string.

Example
```
call rtoc(x,4,10,st,m)
```