OLGX — Application Programmer Interface

1.0 Introduction

This document defines functions for OPEN LOOK Graphics Package (OLGX). OLGX is a library for rendering OPEN LOOK items such as scrollbars, buttons, etc. Most application programmers will probably have no need to use the OLGX library unless they want to write extensions to the XView toolkit and/or want to render an OPEN LOOK item.

2.0 Initialization Functions

OLGX initialization consists of setting initial graphical information used in rendering (the graphics information structure is called Graphics_info, henceforth referred to as ginfo). Ginfos contain the values which define graphical characteristics. These characteristics include pixel values, text fonts, and glyph fonts. As you can see, the ginfo is the mechanism by which graphics information is passed between clients and OLGX. Macros are provided to access the members of the ginfo struct, because the ginfo struct is intended to be opaque. Clients should NOT copy the ginfo struct or try to access its members directly

Each ginfo has 9 or less graphics contexts (GCs), depending on whether 2D or 3D is specified. However, in order to save memory and reduce overhead, ginfos will share graphic contexts with other ginfos.

The following describes the OLGX initialization functions.

Graphics_info *
olgx_main_initialize(display, screen, depth, d_flag, glyphfont_struct, textfont_struct, pixvals, stipple_pixmaps***)***

```
Display         *display;
int             screen;
unsigned int    depth;
int             d_flag;
XFontstruct     glyphfont_struct;
XFontstruct     textfont_struct;
unsigned long   pixvals[ ];
Pixmap          stipple_pixmaps[ ];
```

olgx_main_initialize() creates and initializes a graphics information structure. display is a pointer to the display structure associated with an active server connection. screen is the desired screen. depth is the depth of the window. d_flag may be one of the following flags:

OLGX_2D	Two dimensional rendering.
OLGX_3D_COLOR	Three dimensional rendering on color screen.

glyphfont_struct is the XFontstruct of the desired glyphfont. textfont_struct is the XFontstruct of the desired text font. pixvals is an array of five pixel values. The colors associated with each element of pixvals are OLGX_WHITE (0), OLGX_BLACK(1), OLGX_BG1

(2), OLGX_BG2 (3) and OLGX_BG3 (4). In a monochrome environment, only white and black need to be set. The application must maintain the colormap and allocate the colors properly. In 3D mode, applications are encouraged to use ***olgx_calculate_3Dcolors()*** to get the OLGX_BG2, OLGX_BG3, and OLGX_WHITE pixel values. Please refer to the OPEN LOOK specifications for further details on these colors. stipple_pixmaps[ ] is ignored.

olgx_initialize() still exists for backward compatibility reasons.

void
olgx_destroy(ginfo***)***

```
Graphics_info *ginfo;
```

olgx_destroy() destroys an existing Graphics_info structure info. In the process, ***olgx_destroy()*** will free up any server resources that OLGX has associated with the specified ginfo and not shared by other ginfos.

void
olgx_calculate_3Dcolors(fg, bg1, bg2, bg3, white***)***

```
XColor        *fg, *bg1;
XColor        *bg2, *bg3, *white;   /* Return values */
```

olgx_calculate_3Dcolors() is a utility function which calculates a pleasant combination of bg2, bg3, and white (highlight) color values given the fg (foreground) and bg1 (background) values. The calculation adheres to the OPEN LOOK specification.

fg, bg1, bg2, bg3, and white are pre-allocated XColor structs with fg and bg1 initialized with the appropriate foreground and background pixel values. On the return, bg2, bg3, and white XColor structs are initialized with the OPEN LOOK specified values.

3.0 Manipulation Functions

void
olgx_set_text_font(ginfo, font_struct, flag)

```
Graphics_info    *ginfo;
XFontstruct      font_struct;
int              flag;
```

olgx_set_text_font() sets the current text font. ginfo is a pointer to the Graphics Information Structure. font_struct is the XFontstruct of the desired text font. flag may be one of the following:

OLGX_NORMAL	The changed text font will be reflected only in the specified ginfo. Other ginfos that share graphic contexts with the specified ginfo will not be affected.

OLGX_SPECIAL Changes the ginfo without checking whether it is shared or not. The advantage of this is that it does not have the overhead of checking if GCs are shared or not, or creating new GCs. The disadvantage is that this change may also affect other ginfos that share the GCs. It is up to the client to restore the values so that the changed text font does not affect the other ginfos.

Subsequent text operations using ginfo will use this font.

void
olgx_set_glyph_font(ginfo, glyphfont_struct, flag***)***

```
Graphics_info *ginfo;
XFontstruct glyphfont_struct;
int flag;
```

olgx_set_glyph_font() sets the current glyph font. Subsequent graphics operations using ginfo will use this glyph font. ginfo is a pointer to Graphics_info. glyphfont_struct is the XFontstruct of the desired glyph font. flag may be one of the following:

OLGX_NORMAL The changed glyph font will be reflected only in the specified ginfo. Other ginfos that share graphic contexts with the specified ginfo will not be affected.

OLGX_SPECIAL Changes the ginfo without checking whether it is shared or not. The advantage of this is that it does not have the overhead of checking if GCs are shared or not, or creating new GCs. The disadvantage is that this change may also affect other ginfos that share the GCs. It is up to the client to restore the values so that the changed glyphfont does not affect the other ginfos.

void
olgx_set_single_color(ginfo,index,pixval,flag***)***

```
Graphics_info * ginfo;
unsigned long pixval;
int index;
int flag;
```

olgx_set_single_color() sets the color of the index member of the five element array of colors contained in ginfo to the specified pixel value pixval. index can be any one of the five members of the array namely OLGX_WHITE (0), OLGX_BLACK (1), OLGX_BG1 (2), OLGX_BG2 (3), or OLGX_BG3 (4). flag may be one of the following:

OLGX_NORMAL The changed color will be reflected only in the specified ginfo. Other ginfos that share graphic contexts with the specified ginfo will not be affected.

OLGX_SPECIAL Changes the ginfo without checking whether it is shared or not. The advantage of this is that it does not have the overhead of checking if GCs are shared or not, or creating new GCs. The disadvantage is that this change may also affect other ginfos that share the GCs. It is up to the client to restore the values so that the changed color does not affect the other ginfos. OLGX_SPECIAL might be used where XView renders an array of color choice items representing a palette where each choice item is rendered by changing the foreground color.

unsigned long
olgx_get_single_color(ginfo,index***)***

    Graphics_info * ginfo;
    int index;

olgx_get_single_color() returns the pixel value of the index member of the five element array of pixel values stored in ginfo. index can be any one of the five members of the array namely OLGX_WHITE (0), OLGX_BLACK (1), OLGX_BG1 (2), OLGX_BG2 (3), or OLGX_BG3 (4).

4.0 Rendering Functions

void
olgx_draw_button(ginfo, win, x, y, width,height, label, state***)***

    Graphics_info *ginfo;
    Window win;
    int x, y, width;
    int height;
    void *label;
    int state;

This function renders panel buttons and menu items. win is the window in which the object is to be rendered. x and y are the coordinates of the upper left-hand corner of the object, relative to the upper left-hand corner of win. width is the width of the entire object, including the button endcaps if the object is a panel button. The label of a button can either be a character string or an X pixmap. If the label is a character string, label is a pointer to a character. If the label is an X pixmap, label is a pointer to the struct Pixlabel. The struct Pixlabel is explained below and is to be initialized appropriately. If the given label is too large to fit in the button, it will be clipped appropriately. height is the height of the button and is used only when the button label is a pixmap. If the label is not a pixmap, then the height should be 0, in which case the button height is determined from the button glyphs. state is the bitwise OR of a set of flags representing the current state of the object. These flags are as follows:

OLGX_NORMAL	The item is not invoked. This is the default.
OLGX_INVOKED	Item is invoked. In 3D, it is drawn as a recessed button or menu item. In 2D, the item is in reverse video.
OLGX_MENU_ITEM	This object is a menu item, not a panel button. With this flag, no outline is drawn on non-invoked items.

OLGX_ERASE	The background behind the item is to be cleared before the item is drawn.This flag is only valid for transparent items such as non-invoked menu items.
OLGX_BUSY	The item is busy. In either 2D or 3D, this means that it will be filled with a 15% grey stipple.
OLGX_DEFAULT	This item is a default item. OLGX will render it item with a default ring.
OLGX_INACTIVE	Designates the item as inactive. It will be drawn with a 50% grey stipple.
OLGX_VERT_MENU_MARK	The item contains a vertical menu mark.
OLGX_HORIZ_MENU_MARK	The item contains a horizontal menu mark.
OLGX_LABEL_IS_PIXMAP	By default, item labels are strings. This flag must be presented if the label is a pixmap or window ID.

```
Struct Pixlabel{
        XID     pixmap;
        int     width;          /* width of the pixmap */
        int     height;         /* height of the pixmap */
}
```

Note: If the label is a pixmap, the width of the button is always incremented by OLGX_VARHEIGHT_BTN_MARGIN which accommodates the space for a default ring around the pixmap in case OLGX_DEFAULT is set.

void
olgx_draw_abbrev_button(ginfo, win, x, y, state***)***

```
Graphics_info *ginfo;
Window win;
int x, y, state;
```

This function renders an abbreviated menu button at the position (x, y) with in window win. state may be OLGX_NORMAL, OLGX_INVOKED, OLGX_INACTIVE or OLGX_BUSY. Use OLGX_ ERASE to clear the underlying previous state. The Menu-Mark drawn inside the button points down.

void
olgx_draw_slider(ginfo, win, x, y, width, old_value, new_value, state***)***

```
Graphics_info *ginfo;
Window win;
int x, y, width, old_value, new_value, state;
```

olgx_draw_slider() renders or updates a slider. The slider will be drawn within window win at location (x, y). width represents the width of the slider in pixels, including endcaps.

old_value and new_value give the old and new values of the slider in pixels with the origin at either the top or the left of the slider. Finally, state gives information about the current state of the slider. It consists of the bitwise-OR of any of the following flags:

OLGX_VERTICAL	The slider is a vertical slider.
OLGX_HORIZONTAL	The slider is a horizontal slider.
OLGX_UPDATE	Update the slider instead of rendering the whole slider. If OLGX_ UPDATE is not specified, the parameter old_value will not be used.

This function does not draw any of the text associated with the OPEN LOOK sliders. For example, some OPEN LOOK sliders provide tick marks, end boxes, labels on the tick marks and numeric-entry fields beside the slider. Handling these is left to the application.

void
olgx_draw_resize_corner(*ginfo, win, x, y, type, state****)***

```
Graphics_info *ginfo;
Window win;
int x, y, type;
int state;
```

This function places a resize corner of the requested type in a given location. (x, y) and win specify the location and window of the resize corner, respectively. The shape of the resize corner is determined by the type parameter. type must be one of OLGX_UPPER_LEFT, OLG X_LOWER_LEFT, OLGX_UPPER_RIGHT, or OLGX_LOWER_RIGHT. state must be either OLGX_NORMAL or OLGX_INVOKED.

void
olgx_draw_pushpin(*ginfo, win, x, y, state****)***

```
Graphics_info *ginfo;
Window win;
int x, y, state;
```

This function renders a pushpin at the requested location. (x, y) and win specify the location and window of the resize corner, respectively. The state of the pushpin is determined by the state parameter. state must be a bitwise or combination of the following:

OLGX_PUSHPIN_OUT	The pushpin is out.
OLGX_PUSHPIN_IN	The pushpin is in.
OLGX_ERASE	The area behind the pushpin will be erased before the pushpin is drawn. This flag allows the new pushpin to erase the previous one when it changes states.
OLGX_INACTIVE	Pushpin is inactive; it will be drawn with a 50% grey stipple.

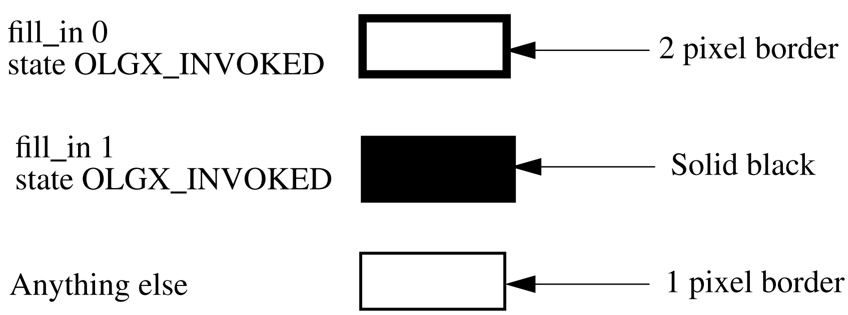
OLGX_DEFAULT	The pushpin is a default pushpin.

void
olgx_draw_box(ginfo, win, x, y, width, height, state, fill_in***)***

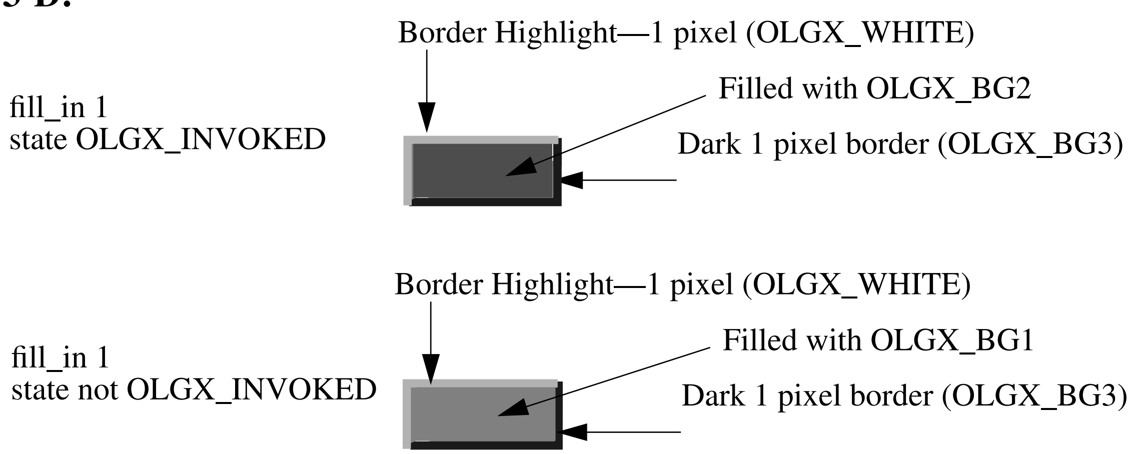
```
Graphics_info *ginfo;
Window win;
int x, y, width, height, state, fill_in;
```

olgx_draw_box() draws a beveled box of the given size at the specified location. state describes the state of the box. It may be either OLGX_NORMAL or OL GX_INVOKED. In 3D mode, an invoked box is drawn as a recessed rectangle and a normal box is drawn as a raised rectangle. If fill_in is nonzero, then the box is filled in. The state flag can take OLGX_ERASE to erase the underlying box before drawing over it.

2-D:

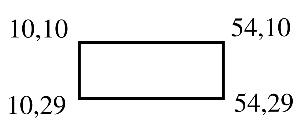


3-D:



Width & Height (2D & 3D):

If width = 45
height = 20
x = 10
y = 10



void

olgx_draw_choice_item(ginfo, win, x, y, width, height, label, state***)***

```
Graphics_info *ginfo;
Window win;
int x, y, width, height;
long label;
int state;
```

olgx_draw_choice_item() will render a choice item with either a pixmap or character string label. win designates what window the item is to be drawn in. x and y give the coordinates of the upper-left corner of the choice item, and width and height denote its outside width and height. The label of a choice item can either be a character string or an X pixmap. If the label is a character string, label is a pointer to a character. If the label is an X pixmap, label is a pointer to the struct Pixlabel. The struct Pixlabel is explained below and is to be initialized appropriately. If the given label is too large to fit in the button, it will be clipped appropriately. Finally, state gives the current state of the item. It consists of a bitwise-OR of any of the following flags:

OLGX_NORMAL	The item is not invoked. In a "3D" environment, it will be drawn as a raised rectangle.
OLGX_INVOKED	The item is invoked. In a "3D" environment, it will be drawn as a recessed rectangle.
OLGX_DEFAULT	This is a default choice item. An inner default rectangle will be drawn to denote it as such.
OLGX_INACTIVE	This item is inactive. It will be drawn with a 50% grey stipple.
OLGX_LABEL_IS_PIXMAP	The label specified is an X Pixmap ID. If this flag is not given, olgx_draw_choice_item will assume that the label is a character string.

```
Struct Pixlabel {
        XID     pixmap;
        int     width;      /* width of the pixmap */
        int     height;     /* height of the pixmap */
}
```

Note: If the label is a pixmap, the width of the choice item is always incremented by OLGX_CHOICE_MARGIN which accommodates the space for a default ring around the pixmap in case OLGX_DEFAULT is set.

void

olgx_draw_check_box(ginfo, win, x, y, state***)***

```
Graphics_info *ginfo;
Window win;
int x, y, state;
```

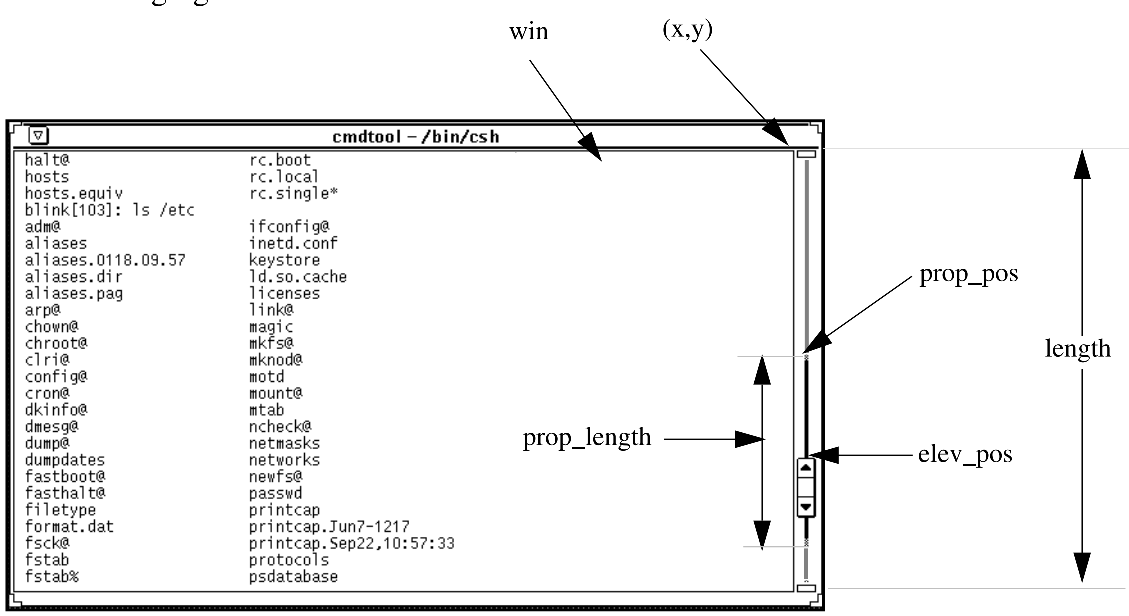
olgx_draw_check_box() renders a check box in a specified state. The upper left-hand corner of the check box will be at location (x, y) within window win. state must be one of the following:

OLGX_NORMAL	The check box is unchecked.
OLGX_INVOKED	The check box is recessed. OPEN LOOK specifies that this is the state of the check box while the mouse button is down over the check box.
OLGX_INACTIVE	The check box is drawn in inactive state.
OLGX_CHECKED	The check box is checked.

void
olgx_draw_scrollbar(ginfo,win,x,y,length,elev_pos,old_elev_pos,prop_pos,prop_length, state***)***

```
Graphics_info * ginfo;
Window win;
int x,y;
int length;
int elev_pos, old_elev_pos;
int prop_pos, prop_length;
int state;
```

olgx_draw_scrollbar() renders or updates a whole scrollbar, including the elevator, cable and proportion indicator. win, x, y, elev_pos, length, prop_pos, prop_length are shown in the following figure.



old_elev_pos is the previous position of the elevator when it is in motion. This is used only when the flag OLGX_UPDATE is set.

Finally state can have any of the following flags bitwise-ORed.

OLGX_VERTICAL	The elevator is vertical.
OLGX_HORIZONTAL	The elevator is horizontal.
OLGX_NORMAL	The elevator is normal.
OLGX_ABBREV	Abbreviated scrollbar elevator
OLGX_SCROLL_ABSOLUTE	The elevator middle box is drawn in invoked fashion.
OLGX_SCROLL_BACKWARD	The elevator backward box is invoked to indicate that the elevator is moving backwards.
OLGX_SCROLL_FORWARD	The elevator forward box is invoked to indicate that the elevator is moving forwards.
OLGX_SCROLL_NO_BACKWARD	The elevator backward box is dimmed, to indicate that the elevator cannot move backward anymore.
OLGX_SCROLL_NO_FORWARD	The elevator forward box is dimmed, to indicate that the elevator cannot move forward anymore.
OLGX_INACTIVE	The whole elevator is drawn dimmed, indicating that the elevator is inactive.
OLGX_UPDATE	Update the scrollbar elevator to its new location. If OLGX_UPDATE is not specified the parameters old_elev_pos will not be used.
OLGX_ERASE	Erases the underlying image before drawing the elevator at location elev_pos.

Use ***olgx_draw_box*** for drawing the end anchors.

void
olgx_draw_text_ledge(ginfo, win, x, y, width***)***

```
Graphics_info *ginfo;
Window win;
int x, y, width;
```

This function draws a 2D or 3D ledge for text items. The ledge is drawn with its left edge at (x,y) within win, and width pixels wide.

void
olgx_draw_text(ginfo,win,string,x,y,max_width,state***)***

```
Graphics_info * ginfo;
Window win;
char * string;
```

int x,y;
int max_width;
int state;

This function renders string in win at x,y. max_width defines, in pixels, the point at which string will be truncated. That is, if the length of string exceeds max_width, it will be truncated. If max_width is 0, the string will not be truncated. By default it is drawn in the normal uninvoked mode. Finally state is a bitwise-OR of any of the following flags.

OLGX_INVOKED	In 2-D, text will appear in reverse video: Like this.
OLGX_NORMAL	In 2-D, text will appear in normal video. 3-D, whether normal or invoked, will always appear in normal video.

olgx_draw_textscroll_button(ginfo,win,x,y,state***)***

Graphics_info * ginfo;
Window win;
int x,y;
int state;

olgx_draw_textscroll_button() renders a text scrolling button. It is drawn within window win at location x,y. By default it is drawn in the normal uninvoked mode. Finally state is a bitwise-OR of any of the following flags.

OLGX_ERASE	Erases the underlying image before drawing the button at location x,y.
OLGX_INVOKED	Will be drawn in Invoked state.
OLGX_SCROLL_FORWARD	Will drawn with the inside menu mark pointing forwards.
OLGX_SCROLL_BACKWARD	Will be drawn with the inside menu mark pointing backwards.
OLGX_INACTIVE	The button is inactive. It will be drawn dimmed.

olgx_draw_numscroll_button(ginfo,win,x,y,state***)***

Graphics_info * ginfo;
Window win;
int x,y;
int state;

olgx_draw_numscroll_button() renders a numeric scrolling button. It is drawn within window win at location x,y. By default it is drawn in the normal uninvoked mode. Finally state is a bitwise-OR of any of the following flags.

OLGX_ERASE	Erases the underlying image before drawing the button at location x,y.
OLGX_SCROLL_FORWARD	Will be drawn with the right box invoked indicating to the user that he is scrolling forward
OLGX_SCROLL_BACKWARD	Will be drawn with the left box invoked indicating to the user that he is scrolling back.
OLGX_SCROLL_NO_FORWARD	Will be drawn with the forward menu mark box dimmed.
OLGX_SCROLL_NO_BACKWARD	Will be drawn with the backward menu mark box dimmed.
OLGX_INACTIVE	The button is inactive. It will be drawn dimmed.

olgx_draw_gauge(ginfo,win,x,y,width,oldval,newval,state***)***

```
Graphics_info * ginfo;
Window win;
int x,y;
int width;
int newval,oldval;
int state;
```

olgx_draw_gauge() renders a gauge. It is drawn within window win at location x,y. width represents the width of the gauge in pixels, including the endcaps. oldval and newval give the old and new values of the gauge in pixels with the origin at either the top (vertical gauges) or the left (horizontal gauges) of the gauge. Finally, state gives information about the current state of the gauge. It consists of the bitwise-OR of any of the following flags:

OLGX_VERTICAL	Vertical gauge
OLGX_HORIZONTAL	Horizontal gauge
OLGX_UPDATE	Update the gauge to newval instead rendering the whole gauge.

If OLGX_UPDATE is not specified, the parameter oldval will not be used.

5.0 OLGX MACROS:

The following macros are provided for convenience and because the ginfo struct is intended to be opaque; clients shouldn't access its members directly. The reason for it being opaque is that OLGX's authors want to retain the option to change the members in the ginfo struct without making existing clients obsolete.

Abbrev_MenuButton_Height(ginfo***)*** returns the height of the abbreviated MenuButton from the specified ginfo struct.

Abbrev_MenuButton_Width(ginfo***)*** returns the width of the abbreviated MenuButton from the specified ginfo struct.

Ascent_of_GlyphFont(*ginfo*) returns the ascent of the glyph font associated with the specified ginfo struct.

Ascent_of_TextFont(*ginfo*) returns the ascent of the text font associated with the specified ginfo struct.

Button_Height(*ginfo*) returns the height of the button from the specified ginfo struct.

ButtonEndcap_Width(*ginfo*) returns the width of the button endcap from the specified ginfo struct.

CheckBox_Height(*ginfo*) returns the height of the checkbox from the specified ginfo struct.

CheckBox_Width(*ginfo*) returns the width of the checkbox from the specified ginfo struct.

Dimension(*ginfo*) returns 1, if 3D is set in the specified ginfo struct, else returns 0.

Descent_of_GlyphFont(*ginfo*) returns the descent of the glyph font associated with the specified ginfo struct.

Descent_of_TextFont(*ginfo*) returns the descent of the text font associated with the specified ginfo struct.

Gauge_EndCapHeight(*ginfo*) returns the height of the gauge endcap from the specified ginfo struct.

Gauge_EndCapOffset(*ginfo*) returns the pixel offset of the value indicator from the left or top endcap of the gauge from the specified ginfo struct.

Gauge_EndCapWidth(*ginfo*) returns the width of the gauge endcap from the specified ginfo struct.

***HorizSliderControl_Width(*ginfo returns the width of the Horizontal slider control section of the ginfo struct. This corresponds to the Vertical Slider Control Height.

HorizSliderControl_Height(*ginfo*) returns the height of the Horizontal slider control section of the specified ginfo struct. This corresponds to the Vertical slider control Width.

MenuMark_Height(*ginfo*) returns the height of the MenuMark used in MenuButtons from the specified ginfo struct.

MenuMark_Width(*ginfo*) returns the width of the MenuMark used in MenuButtons from the specified ginfo struct.

NumScrollButton_Height(*ginfo*) returns the height of the Number scrolling button from the specified ginfo struct.

NumScrollButton_Width(*ginfo*) returns the width of the Number scrolling button from the specified ginfo struct.

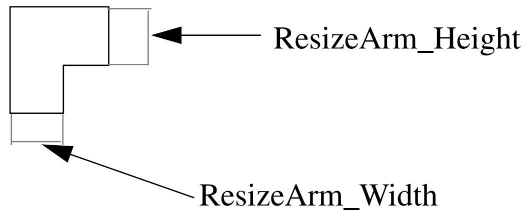
Pointsize_Glyph(*ginfo*) returns the point size of the glyph font associated with the specified ginfo struct.

PushPinOut_Height(*ginfo*) returns the height of the PushPin in its Out state from the specified ginfo struct.

PushPinOut_Width(*ginfo*) returns the width of the PushPin in its Out state from the specified ginfo struct.

ResizeArm_Height(*ginfo*) returns the height of the resize arm from the specified ginfo struct (see figure).

ResizeArm_Width(*ginfo*) returns the width of the resize arm from the specified ginfo struct (see figure).



ScrollbarElevator_Height(*ginfo*) returns the scrollbar elevator height from the specified ginfo struct.

ScrollbarElevator_Width(*ginfo*) returns the scrollbar elevator width from the specified ginfo struct.

SliderEndCap_Width(*ginfo*) returns the width of the slider endcap of the specified ginfo struct. Both vertical and horizontal sliders have the same endcap width.

SliderEndCap_Height(*ginfo*) returns the height of the slider endcap from the specified ginfo struct. Both vertical and horizontal sliders have the same endcap height.

TextFont_Struct(*ginfo*) returns the XFontstruct of the text font from the specified ginfo struct.

TextScrollButton_Height(*ginfo*) returns the height of the Text scrolling button from the specified ginfo struct.

TextScrollButton_Width(*ginfo*) returns the width of the Text scrolling button from the specified ginfo struct.

Vertsb_Endbox_Height(*ginfo*) returns the height of the end anchor box of a vertical scrollbar from the specified ginfo struct.

Vertsb_Endbox_Width(*ginfo*) returns the width of the end anchor box of a vertical scrollbar from the specified ginfo struct.