Nano-X API Reference Manual

Nano-X

Table of Contents

`fncb` :

GrGetScreenInfo ()

```
                                        GR_COORD x,
                                        GR_COORD y,
                                        GR_SIZE width,
                                        GR_SIZE height);
void        GrDestroyWindow            (GR_WINDOW_ID wid);
void        GrMapWindow                (GR_WINDOW_ID wid);
void        GrUnmapWindow              (GR_WINDOW_ID wid);
void        GrRaiseWindow              (GR_WINDOW_ID wid);
void        GrLowerWindow              (GR_WINDOW_ID wid);
void        GrMoveWindow               (GR_WINDOW_ID wid,
                                        GR_COORD x,
                                        GR_COORD y);
void        GrResizeWindow             (GR_WINDOW_ID wid,
                                        GR_SIZE width,
                                        GR_SIZE height);
void        GrReparentWindow           (GR_WINDOW_ID wid,
                                        GR_WINDOW_ID pwid,
                                        GR_COORD x,
                                        GR_COORD y);
void        GrGetWindowInfo            (GR_WINDOW_ID wid,
                                        GR_WINDOW_INFO *infoptr);
void        GrSetWMProperties          (GR_WINDOW_ID wid,
                                        GR_WM_PROPERTIES *props);
void        GrGetWMProperties          (GR_WINDOW_ID wid,
                                        GR_WM_PROPERTIES *props);
void        GrSetFocus                 (GR_WINDOW_ID wid);
GR_WINDOW_ID GrGetFocus                (GR_WINDOW_ID *pwid, GR_WM_PROPERTIES
```

Returns : the ID of the newly created window

GrNewPixmap ()

y : the Y coordinate of the new window relative to the parent window
width : the width of the new window
height : the height of the new window
Returns : the ID of the newly created window

GrDestroyWindow ()

```
void        GrDestroyWindow                 (GR_WINDOW_ID wid);
```

Recursively unmaps and frees the data structures associated with the specified window and all of its children.

wid : the ID of the window to destroy

GrMapWindow ()

```
void        GrMapWindow                     (GR_WINDOW_ID wid);
```

Recursively maps (makes visible) the specified window and all of the child windows which have a sufficient map count. The border and background of the window are painted, and an exposure event is generated for the window and every child which becomes visible.

wid : the ID of the window to map

GrUnmapWindo

GrGetWMProperties ()

```
void
```

Clears the specified window by setting it to its background color. If the exposeflag parameter is non zero, an exposure event is generated for the window after it has been cleared.

wid : the ID of the window to clear
exposeflag : a flag indicating whether to also generate an exposure event

w08

GrCloseWindow ()

```
void        GrCloseWindow                    (GR_WINDOW_ID wid);
```

Sends a CLOSE_REQ event to the specified window if the client has selected to receive CLOSE_REQ events on this window. Used tov4Used it by whether to ve

GR_SIZE

```
                                        GR_COORD y,
                                        GR_SIZE width,
                                        GR_SIZE height,
                                        GR_IMAGE_ID imageid);
void        GrDrawImageBits            (GR_DRAW_ID id,
                                        GR_GC_ID gc,
                                        GR_COORD x,
                                        GR_COORD y,
                                        GR_IMAGE_HDR *pimage);
void        GrText                     (GR_DRAW_ID id,
                                        GR_GC_ID gc,
                                        GR_COORD x,
                                        GR_COORD y,
                                        void *str,
                                        GR_COUNT count,
                                        int flags);
```

Description

Details

GrNewGC ()

```
GR_GC_ID    GrNewGC                    (void);
```

Creates a new graphics context structure and returns the ID used to refer to it. The structure is initialised with a set of default parameters.

Returns : the ID of the newly created graphics context or 0 on error

GrCopyGC ()

```
GR_GC_ID    GrCopyGC                        (GR_GC_ID gc);
```

Creates a new graphics context structure and fills it in with the values from the specified already existing graphics context.

gc : the already existing graphics context to copy the parameters from

GrLine ()

```
void        GrLine                          (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COORD x1,
                                             GR_COORD y1,
                                             GR_COORD x2,
                                             GR_COORD y2);
```

Draws a line using the specified graphics context on the specified drawable from (x1, y1) to (x2, y2), with coordinates given relative to the drawable.

id : the ID of the drawable to draw the line on

gc : the ID of the graphics context to use when drawing the line

x1 : the X coordinate of the start of the line relative to the drawable

y1 : the Y coordinate of the start of the line relative to the drawable

x2 : the X coordinate of the end of the line relative to the drawable

x2 : theX

x2 : theX

n the wable from

x : the X coordinate to draw the point at relative to the drawable

y : the Y coordinate to draw the point at relative to the drawable

GrPoints ()

```
void        GrPoints                        (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COUNT count,
                                             GR_POINT *point-
table);
```

Draws a set of points using the specified graphics context at the positions specified by the point table on the specified drawable.

id : the ID of the drawable to draw a point on

gc : the ID of the graphics context to use when drawing the point

count : the number of points in the point table

pointtable

```
                                                    GR_GC_ID gc,
                                                    GR_COUNT count,
                                                    GR_POINT *point-
table);
```

Draws an unfilled polygon on the specified drawthespecifiedda

Draws a filled ellipse at the specified position using the specified dimensions and graphics context on the specified drawable.

`id` : the ID of the drawable to draw the filled ellipse on
`gc` : the ID of the graphics context to use when drawing the ellipse
`x` : the X coordinate to draw the ellipse at relative to the drawable
`y` : the Y coordinate to draw the ellipse at relative to the drawable
`rx` : the radius of the ellipse on the X axis
`ry` : the radius of the ellipse on the Y

`y` : the Y coordinate to draw the arc at relative to the drawable
`rx` : the radius of the arc on the X axis
`ry` : the radius of the arc on the Y axis
`ax` : the X coordinate of the start of the arc relative to the drawable
`ay` : the Y coordinate of the start of the arc relative to the drawable
`bx` : the X coordinate of the end of the arc relative to the drawable
`by` : the Y coordinate of the end of the arc relative to the drawable
`type` : the fill style to use when drawing the arc

rx : the capitulum 1.

GrSetGCUseBackground ()

```
void        GrSetGCUseBackground            (GR_GC_ID gc,
                                             GR_BOOL flag);
```

Sets the flag which chooses whether or not the background colour is used

gc : the ID of the graphics conte

```
                                                GR_SIZE width,
                                                GR_SIZE height,
                                                GR_PIXELVAL *pix-
els);
```

Reads the pixel data of the specified

an unsigned long containing RGBX data.

id : the ID of the drawable to draw the area onto
gc : the ID of the graphics context to use when drawing the area
x : the X coordinate to draw the area at relative to the drawable
y : the Y coordinate to draw the area at relative to the drawable
width : the width of the area
height : the height of the area
pixels : pointer to an array containing the pixel data
pixtype : the format of the pixel data

GrCopyArea ()

```
void        GrCopyArea                      (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COORD x,
                                             GR_COORD y,
                                             GR_SIZE width,
                                             GR_SIZE height,
                                             GR_DRAW_ID srcid,
                                             GR_COORD srcx,
                                             GR_COORD srcy,
                                             int op);
```

Copies the specified area of the specified size between the specified drawables at the specified positions using the specified graphics context and ROP codes. 0 is a sensible default ROP code in most cases.

id : the ID of the drawable to copy the area to
gc : the ID of the graphics context to use when copying the area
x : the X coordinate to copy the area to within the destination drawable

`y` : the Y coordinate to copy the area to within the destination dra

height : the height of the bitmap

imagebits :

GrFreeImage

variants of PBM, PGM, and PPM. However the image types supported by a particular server depend on which image types were enabled in the server configuration at build time.

path : string containing the filename of the image to load

flags : flags specific to the particular image loader

Returns : ID of the image buffer the image was loaded into

GrDrawImageToFit ()

```
void        GrDrawImageToFit                (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COORD x,
                                             GR_COORD y,
                                             GR_SIZE width,
                                             GR_SIZE height,
                                             GR_IMAGE_ID im-
ageid);
```

Draws the image from the specified image buffer at the specified position on the specified on**f**lags specific

GrDrawImageBits ()

```
void        GrDrawImageBits                 (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COORD x,
                                             GR_COORD y,
                                             GR_IMAGE_HDR *pim-
age);
```

Draws the image contained in the specified image structure onto the specified drawable at the specified coordinates using the specified graphics context.

id : the ID of the drawable to draw the image onto

gc : the ID of the graphics context to use when drawing the image

x : the X coordinate to draw the image at relative to the drawable

y

```
                                                GR_TIMEOUT time-
out);
```

Gets the next event from the event queue and places it in the specified

Description

Details

GrCreateFont ()

```
GR_FONT_ID  GrCreateFont                    (GR_CHAR *name,
                                             GR_COORD height,
                                             GR_LOGFONT *plog-
font);
```

Attempts to locate a

GrDestroyFont ()

```
void        GrDestroyFont                   (GR_FONT_ID fontid);
```

Frees all resources associated with the specified font ID, and if the font is a non built in type and this is the last ID referring to it, unloads the font from memory.

`fontid` : the ID of the font to destroy

GrGetFontInfo ()

```
void        GrGetFontInfo                   (GR_FONT_ID font,
                                             GR_FONT_INFO *fip);
```

Fills in the specified GR_FONT_INFO structure with information

colours (3)

Names

`pal`: pointer to a palette structure to fill

GrSetGCRegion ()

```
void        GrSetGCRegion                   (GR_GC_ID gc,
                                             GR_REGION_ID re-
gion);
```

Sets the clip mask of the specified graphics

GrEqualRegion ()

```
GR_BOOL
GrEqualRegion
```

Fills in the specified

Details

GrSetSelectionOwner ()

```
void        GrSetSelectionOwner             (GR_WINDOW_ID wid,
                                             GR_CHAR *typelist);
```

Sets the current selection (otherwise known as the clipboard) ownership to the specified window. Specifying an owner of 0 disowns the selection. The typelist argument is a list of mime types.

(c) peratd

check the value of it before using it.

`typelist` : pointer used to return the list of available mime types

Returns

machine). Apart from

GrRegisterInput ()

```
void        GrRegisterInput                 (int fd);
```

Register an extra file descriptor to monitor in the main select() call. An event will be returned when the fd has data waiting to be read if that event has been selected for.

fd : the file descriptor to monitor

GrPrepareSelect ()

```
void        GrPrepareSelect                 (int *maxfd,
                                             void *rfdset);
```

Prepare for a GrServiceSelect function by asking what events

Anordataweitarou62 Tf 36.1875d (Prepare)Tj

rfdset : pointer to the file descriptor set to monitor

fncb : pointer to the function to call when an event needs handling

GrBell ()

```
void        GrBell                          (void);
```

Asks the server to ring the console bell on behalf of the client