Nano-X API Reference Manual

Table of Contents

Chapter 1. libnano-X

general (3)

Name

general —

Synopsis

```
 void        GrFlush                          (void);
 int         GrOpen                           (void);
 void        GrClose                          (void);
 void        GrMain-
 Loop                     (GR_FNCALLBACKEVENTfncb);
void        GrGetScreen-
Info                 (GR_SCREEN_INFO *sip);
GR_FNCALLBACKEVENT GrSetErrorHan-
dler        (GR_FNCALLBACKEVENTfncb);
void       GrDefaultErrorHandler           (GR_EVENT *ep);
```

Description

fncb :

GrGetScreenInfo ()

```
void        GrGetScreen-
Info                (GR_SCREEN_INFO *sip);
```

Fills in the specified GR_SCREEN_INFO structure.

sip : pointer to a GR_SCREEN_INFO structure

GrSetErrorHandler ()

```
GR_FNCALLBACKEVENT GrSetErrorHan-
dler        (GR_FNCALLBACKEVENT fncb);
```

Sets an error handling routine that will be called on any errors from the server (assuming the client has asked to receive them). If zero is used as the argument, errors will be returned as regular events instead.

fncb : the function to call to handle error events

Returns : the address of the previous error handlanthe o60andler ()

Generates a human readable error message on stderr describing what error occurred and what function it occured in, then exits.

ep : the error event structure

window (3)

Name

window —

Synopsis

```
GR_WINDOW_ID GrNewWin-
dow                    (GR_WINDOW_ID parent,GR_WINDOW_ID GrNewWin-
                                                (GR_WINDOW_ID
```

Recursively unmaps

Moves the specified window to the specified position relative to its parent window.

`wid` : the ID of the

pwid : the ID of the new parent window

x : the X coordinate to place the window at relative parent

GrGetWMProperties(), GrSetWMProperties()

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

GrCloseWindow ()44 14.38 10.7597 Tf -114.526 -25.92 Td (viid)Tj 77.5092 0

graphics (3)

Name

graphics —

Synopsis

```
GR_GC_ID    GrNewGC                 (void);
GR_GC_ID    GrCopyGC                (GR_GC_ID gc);
void        GrGetGCInfo             (GR_GC_ID gc,
                                     GR_GC_INFO *gcip);
void        GrDestroyGC             (GR_GC_ID gc);
void        GrLine                  (GR_DRAW_ID id,
                                     GR_GC_ID gc,
                                     GR_COORD x1,
                                     GR_COORD y1,
                                     GR_COORD x2,
                                     GR_COORD y2);
void        GrPoint                 (GR_DRAW_ID id,
                                     GR_GC_ID gc,
                                     GR_COORD x,
                                     GR_COORD y);
void        GrPoints                (GR_DRAW_ID id,
                                     GR_GC_ID gc,
                                     GR_COUNT count,
```

```
                                        GR_SIZE height);
void        GrFillRect                 (GR_DRAW_ID id,
                                        GR_GC_ID gc,
                                        GR_COORD x,
                                        GR_COORD y,
                                        GR_SIZE width,
                                        GR_SIZE height);
void        GrPoly                     (GR_DRAW_ID id,
                                        GR_GC_ID gc,
                                        GR_COUNT count,
                                        GR_POINT *pointtable);
void        GrF(GrPoly)Tj 206.493 0 Td ((GR_DRAW_ID)Tj 77.4904 0 Td (id,)Tj
```

```
                                        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

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

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

GrGetGCInfo ()

```
void        GrGetGCInfo                     (GR_GC_ID gc,
                                             GR_GC_INFO *gcip);
```

Fills in the specified GR_GC_INFO structure with information regarding the specified graphics context.

gc : a graphics context

gcip : pointer to a GR_GC_INFO structure

GrDestroyGC ()

```
void        GrDestroyGC                     (GR_GC_ID gc);
```

Destroys the graphics context structure with the specified ID.

gc : the ID of the graphics context structure to destroy

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

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

x : the X coordinate of the rectangle relative to the X

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

Draws an unfilled polygon on the specified drawable using the specified graphics context. The polygon is specified by an array of point structures. The polygon is not automatically closed- if

Chapter 1.

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

GrArcAngle ()

```
void        GrArcAngle                      (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COORD x,
                                             GR_COORD y,
                                             GR_SIZE rx,
                                             GR_SIZE ry,
                                             GR_COORD angle1,
                                             GR_COORD angle2,
                                             int type);
```

Draws an arc with the specified dimensions at the specified position on the specified drawable using the specified graphics context. The type specifies the fill type. Possible values include GR_ARC and GR_PIE. This function requires floating point support, and is slightly slower than the GrArc() function which does not require floating point.

specified graphics e ablse when draable.bx the X
bx :
id : the ID of the drawable to draw the arc)

rx : the radius of the arc on the X axis

ry : the radius of the arc on the Y axis

angle1 : the angle of the start of the arc

angle2 : the angle of the end of the arc

type : the fill style to use when drawing the arc

GrSetGCForeground ()

```
void        GrSetGCForeground               (GR_GC_ID gc,
                                             GR_COLOR
```

GrSetGCUseBackground ()

gc : the ID of the graphics context to set the font of

font : the ID of the font

GrGetGCTextSize ()

```
void        GrGetGCTextSize                 (GR_GC_ID gc,
                                             void *str,
                                             int
```

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

Reads the pixel data of the specified size from the specified position on the specified drawable into the specified pixel array. If the drawable is a window, the data returned will be the pixel values from the relevant position on the screen regardless of whether the window is obscured by other windows. If the window is unmapped, or partially or fully outside a window boundary, black pixel values will be returned.

id : the ID of the drawable to read an area from
x : the X coordinate to read the area from relative to the drawable
y : the Y coordinate to read the area from relative to the drawable
width : the width of the area to read
height : the height of the area to read
pixels : pointer to an area of memory to place the pixel data in

GrArea ()

```
void        GrArea                          (GR_DRAW_ID id,
                                             GR_GC_ID gc,
                                             GR_COORD x,
                                             GR_COORD y,
                                             GR_SIZE width,
```

an unsigned long containing RGBX data.

`id` : the ID of the drawable

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

`width` : the width of the area to copy

`height` : the height of the area to copy

`srcid` : the ID of the drawable to copy the area from

`srcx` : the X coordinate to copy the area from within the source drawable

srcy : the Y

height : the height of the bitmap

imagebits :

GrFreeImage ()

```
void        GrFreeImage                     (GR_IMAGE_ID id);
```

Destroys the specified image buffer and reclaims the memory used by it.

id : ID of the image buffer to free

GrGetImageInfo ()

```
void        GrGetImageInfo                  (GR_IMAGE_ID id,
                                             GR_IMAGE_INFO *iip);
```

Fills in the specified image information structure with the details of the specified image buffer.

id : ID of an image buffer

```
                                        GR_SIZE  height,
                                        char    *path,
                                        int     flags);
```

Loads the specified image file and draws it at the specified position on the specified drawable

GrDrawImageBits ()

void GrDrawImageBits (GR_DRAW_ID

id : the ID of the drawable to draw the text string onto
gc : the ID of the graphics context to use when drawing the text string
x : the X coordinate to draw the string at relative to the drawable
y : the Y coordinate to draw the string at relative to the drawable
str : the text string to draw
count : the number of characters (not bytes) in the string
flags : flags specifying text encoding, alignment, etc.

events (3)

Name

events —

Synopsis

```
void        GrSelectEvents              (GR_WINDOW_ID wid,
                                         GR_EVENT_MASK eventmask);
void        GrGetNextEvent              (GR_EVENT *ep);
void        GrGetNextEventTimeout       (GR_EVENT *ep,
                                         GR_TIMEOUT timeout);
void        GrCheckNextEvent            (GR_EVENT *ep);
int         GrPeekEvent                 (GR_EVENT *ep);
```

Description

GR_TIMEOUT time-

Returns : 1 if an event was returned, or 0 if the queue was empty

fonts (3)

Name

`fonts` —

Synopsis

```
GR_FONT_ID  GrCreateFont                    (GR_CHAR *name,
                                             GR_COORD height,
                                             GR_LOGFONT *plogfont);
void        GrSetFontSize                   (GR_FONT_ID
                                            GR_COORD
                                            void        GrSetRotationSize
```

fontid : the ID number of the font to change the size of
size :

GrSetFontRotation ()

```
void        GrSetFontRotation               (GR_FONT_IDfontid,
                                             int tenthsdegrees);
```

Changes the 16.312 -2 .s fontid, tenthsdegrees);597 Tf -132.249 -26.04 Td (voidAtt (ID)0 Td
size

fontid : the ID number of the font

Synopsis

```
void        GrSetCursor                 (GR_WINDOW_ID wid,
                                         GR_SIZE width,
                                         GR_SIZE height,
                                         GR_COORD hotx,
                                         GR_COORD hoty,
                                         GR_COLOR foreground,
                                         GR_COLOR background,
                                         GR_BITMAP *fbbitmap,
                                         GR_BITMAP *bgbitmap,
                                         int flags);
void        GrMoveCursor                (GR_COORD x,
                                         GR_COORD y);
void        GrInjectPointerEvent        (MWCOORD x,
                                         MWCOORD y,
                                         int button,
                                         int visible);
```

Description

Details

GrSetCursor ()

```
void        GrSetCursor                 (GR_WINDOW_ID wid,
                                         GR_SIZE width,
                                         GR_SIZE height,
                                         GR_COORD hotx,
                                         GR_COORD hoty,
```

Moves the cursor (mouse pointer) to the specified coordinates. The coordinates are relative to the root window, where (0,0) is the upper left hand corner of the screen. The reference point used for the pointer is that of the "hot spot". After moving the pointer, the graphic used for the pointer will change to the graphic defined for use in the window which it is over.

x : the X coordinate to move the pointer to

y : the Y coordinate to move the pointer to

GrInjectPointerEvent()

```
void
```

colours (3)

Name

colours —

Synopsis

```
void        GrGetSystemPalette              (GR_PALETTE *pal);
void        GrSetSystemPalette              (GR_COUNT first,
                                             GR_PALETTE *pal);
void        GrFindColor                     (GR_COLOR c,
                                              GR_PIXELVAL *retpixel);
  GR_COLOR    GrGetSysColor                  (int index);
```

Description

Details

GrGetSystemPalette ()

```
void        GrGetSystemPalette              (GR_PALETTE *pal);
```

Retrieves the system palette and places it in the specified palette structure.

`pal`

index

```
                                                GR_REGION_ID src_rgn1,
                                                GR_REGION_ID src_rgn2);
void        GrIntersectRe-
gion               (GR_REGION_ID dst_rgn,
                                                GR_REGION_ID src_rgn1,
                                                GR_REGION_ID src_rgn2);
void        GrSetGCRegion                      (GR_GC_ID gc,
                                                GR_REGION_ID region);
GR_BOOL     GrPointInRe-
gion                (GR_REGION_ID region,
                                                GR_COORD x,
                                                GR_COORD y);
int         GrRectInRe-
gion                 (GR_REGION_ID region,
                                                GR_COORD x,
                                                GR_COORD y,
                                                GR_COORD w,
                                                GR_COORD h);
GR_BOOL     GrEmptyRe-
gion                 (GR_REGION_ID region);voidGrSetTd (gion)Int22e62R4-0
                                                gion                 (GR_RE
```

GrXorRegion ()

```
void        GrXorRe-
gion                    (GR_REGION_ID dst_rgn,
                                        GR_REGION_ID src_rgn1,
                                        GR_REGION_ID src_rgn2);
```

Performs a logical exclusive OR operation on the specified source regions and places the result in the destination region. The destination region will contain only the parts of the source regions which do not overlap.

dst_rgn : the ID of the destination region

src_rgn1 : the ID of the first source region *: the ID*

GrEqualRegion ()

```
GR_BOOL     GrEqualRegion                   (GR_REGION_ID rgn1,
                                             GR_REGION_ID rgn2);
```

Fills in the specified rectangle structure with a bounding box that would completely enclose the specified region, and also returns the type of the specified region.

`region` : the ID of the region to get the bounding box of

`rect` : pointer to a rectangle structure

Returns

selections (3)

Name

`selections` —

Synopsis

```
void        GrSetSelectionOwner             (GR_WINDOW_ID wid,
                                             GR_CHAR *typelist);e.1 0a777E.(
```

Details

GrSetSelectionOwner ()

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

Sets the current selection (otherwise known as the clipboard) ownership to the specified window0 Td .91162 0 Td (.)Tj 6.46875 0 Td (Specifying)Tj 54.7778 0 Td (an)Tj 14.2617 0 Td (o)Tj 5.

check the value of it before using it.

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

Returns : the ID of the window which currently owns the selection, or 0

GrRequestClientData ()

```
void        GrRequestClientData             (GR_WINDOW_ID wid,
                                             GR_WINDOW_ID rid,
                                             GR_SERIALNO serial,
                                             GR_MIMETYPE mime-
type);
```

Sends a CLIENT_DAT

GrSendClientData ()

```
void        GrSendClientData                (GR_WINDOW_ID wid,
                                             GR_WINDOW_ID did,
                                             GR_SERIALNO
```

machine). Apart from the initial allocation of the area using this call, the use of shared memory is completely transparent. Additionally, if the allocation fails we si(shared)Tj

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 the server to send the next event but not waiting around for