GIA

The GIA (Graphics Interface Adapter) module implements an abstraction layer for pixel oriented graphic devices like VGA cards or LCD displays.

GIA implements generic support for mouse and keyboard devices as well as support for fonts.

The GIA module is divided in two components: A gia 'server' and a gia library. Applications should use the gia_lib functions in order to talk to the gia process, not send messages directlry to gia. The reason for this is to decouple the application process from gia for synchronization purposes. This mechanism also makes it possible to use a gia server over the network.

Requirements

Graphics driver process	GIA opens a graphics driver once it receives a *graphics* command.

Process Information

Prototype Name	GIA
Process Priority	driver level
Process Name	gia

Module Options

NONE_YET	none yet

Process Operation

The module implements a queue handler and a main process. All but one messages are handled in the main process. The only messages processed in the queue handler are *EVENT* messages. These messages are simply queued into the event queue of the gia process.

All other messages are handled in the main process:

OPEN	messages are handled by creating a gia openfile context structure and returning it in the dst_context field of the *OPEN* message.
CLOSE	message cause gia to free the gia openfile structure.
COMMAND	messages are used to configure GIA parameters such as mouse or keyboard devices and the graphics device. Valid commands are:

- mouse <mouse_device>
- keyboard <keyboard_device>
- graphics <graphics_device>

These commands are usually sent by the system manager process.

GIA_GRAPH_CMD messages are used to decouple gia_lib from the gia main process. These messages contain the actual operation commands for gia to draw lines boxes etc.

EVENT messages from input devices such as mice and keyboards are handled in the main process. GIA translates these events into GIA events and relays them to the processes that requested gia events.

Shared Library Macros and Routines

The following GIA interface functions are available as a shared library API.

gia_init

void *gia_init*(
void)

The *gia_init* function should be called before any other gia functions. This function initializes gia internal structures and also initializes the *ROME window manager* and the *font support module(s)* if they are present in the system.

gia_lib_init

FILE **gia_lib_init*(
void)

The *gia_lib_init* function opens the gia process. Currently it opens *"gia:"* on the local machine and returns the file handle to the caller. In future versions this function can be extended to open the *gia* process on other machines on the network.

gia_plot

void *gia_plot*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
int *x*,
int y,
uint *col*)

The *gia_plot* function draws a single point (pixel) on the graphics screen. It has to provide a *drawable* and a *drawable graphics context*, as well as the *x* and *y* position and the *color* of the pixel to draw. The *graphic context* of the drawable can be optained using the *gia_default_gc* function.

gia_draw_line

int *gia_draw_line*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
int *x1*,
int *y1*,
int *x2*,
int *y2*,
uint *col*)

The *gia_draw_line* routine draws a line between the points *x1/y2* and *x2/y2* using the color *col*.

gia_draw_box

int *gia_draw_box*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
int *x1*,
int *y1*,
int *x2*,
int *y2*,
uint *col*,
int *fill*)

The *gia_draw_box* routine draws a box with the corner points *x1/y1* and *x2/y2* using the color *col*. If the *fill* flag is set to TRUE, the box will be filled.

gia_draw_frame

int *gia_draw_frame*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
int *width*,
uint *col*)

The *gia_draw_frame* routine draws a frame around the drawable *drawable*. Calling this function is equivalent of calling *gia_draw_box* with the drawable size as parameters.

gia_clear_drawable

int *gia_clear_drawable*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
uint *col*)

The *gia_clear_drawable* function will clear the drawable *drawable* with the color *col*.

gia_new_drawable

GIA_DRAWABLE *gia_new_drawable*(
GIA_DRAWABLE_ATTR **attr*)

The *gia_new_drawable* routine returns the handle of a newly created drawable. The attributes for the drawable have to be provided in a *GIA_DRAWABLE_ATTR* structure.

gia_map_drawable

int *gia_map_drawable*(
GIA_DRAWABLE *drawable*)

The *gia_map_drawable* routine maps the drawable *drawable* on the screen. The drawable will not be visible before this function is called. All previous *plot*, *draw* or other paint operations will happen off screen.

gia_delete_drawable

int *gia_delete_drawable*(
GIA_DRAWABLE *drawable*)

The *gia_delete_drawable* routine will delete the drawable *drawable* and free all associated resources. If the drawable is currently visible on the screen, it will be removed.

gia_default_gc

GIA_DRAWABLE_GC *gia_default_gc*(
GIA_DRAWABLE *drawable*)

The *gia_default_gc* routine will return the default graphic context of the drawable *drawable*. If the drawable is also managed by a window manager, the graphic context will represent the client area of the window, otherwise it will represent the whole drawable area.

gia_frame_gc

GIA_DRAWABLE_GC *gia_frame_gc*(
GIA_DRAWABLE *drawable*)

The *gia_frame_gc* routine will return the graphic context representing the whole drawable. This routine is mainly used by window managers.

gia_repaint_drawable

int *gia_repaint_drawable*(
GIA_DRAWABLE *drawable*)

The *gia_repaint_drawable* routine repaints a drawable on the screen. If the drawable is not mapped, calling this routine will have no effect.

gia_repaint_drawable_area

int *gia_repaint_drawable_area*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
GIA_RECT **rect*)

The *gia_repaint_drawable_area* repaints a certain area of a drawable on the screen. The area to be repainted is defined in the *GIA_RECT* structure. If the drawable is not mapped, calling this routine will have no effect.

gia_get_font_handle

GIA_FONT *gia_get_font_handle*(
const char **font*)

The *gia_get_font_handle* returns the handle to a GIA font which represents the font with the name *font.*

gia_draw_text

void *gia_draw_text*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,
GIA_FONT *hfont*,
const char **text*,
register int *dx*,
register int *dy*,
uint *col*)

The *gia_draw_text* routine draws the text *text* at the offest *dx/dy to* a drawable *drawable* using the font *font* and the color *col*.

gia_draw_text_size_hint

void *gia_draw_text_size_hint*(
GIA_FONT *hfont*,
const char **text*,
GIA_RECT **rect*)

The *gia_draw_text_size_hint* routine returns the size of then given text *text* with the font *font.* The size that the text will need will be returned in the *rect* structure.

gia_select_events

void *gia_select_events*(
GIA_DRAWABLE *drawable*,
FILE **fp*,
int *mask*)

The *gia_select_events* selects certain events to be received from the file *fp*. The list of events is defined using the *mask* parameter.

gia_put_image

void *gia_put_image*(
GIA_DRAWABLE *drawable*,
GIA_DRAWABLE_GC *gc*,

const char **image*,
int *x*,
int *y*,
int *dx*,
int dy,
int *mask*)

The *gia_put_image* routine transfers an image into the drawable *drawable*. The image is given by the buffer location *image* and the size *dx/dy*. The position of the image within the drawable is given by *x/y*. The *mask* value defines whether the image contains transparent values (*mask*=TRUE) or not (*mask*=FALSE).