Elbonia Client Reference Manual

0.1

Generated by Doxygen 1.3

Wed May 14 10:53:14 2003

Contents

Chapter 1

Elbonia Client Compound Index

1.1 Elbonia Client Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Elbonia Client File Index

2.1 Elbonia Client File List

Here is a list of all files with brief descriptions:

Chapter 3

Elbonia Client Class Documentation

3.1 console_line Struct Reference

Public Attributes

- char **str** [40]

3.1.1 Detailed Description

For holding one line of console output.

3.1.2 Member Data Documentation

3.1.2.1 char console_line::str[40]

The documentation for this struct was generated from the following file:

- **console.c**

3.2 link Struct Reference

`#include <navigation.h>`

Public Attributes

- unsigned int **x1**
- unsigned int **y1**
- unsigned int **x2**
- unsigned int **y2**
- char * **url**

3.2.1 Member Data Documentation

3.2.1.1 char* link::url

3.2.1.2 unsigned int link::x1

3.2.1.3 unsigned int link::x2

3.2.1.4 unsigned int link::y1

3.2.1.5 unsigned int link::y2

The documentation for this struct was generated from the following file:

- **navigation.h**

Chapter 4

Elbonia Client File Documentation

4.1 capabilities.c File Reference

```
#include <vmgp.h>
#include "capabilities.h"
```

Functions

- int **get_capabilities** ()

Variables

- int **screen_width**
- int **screen_height**
- int **video_modes**

4.1.1 Function Documentation

4.1.1.1 int get_capabilities (void)

Get capabilites and set variables defined in **capabilities.h**.

Returns:
1 if enough for running Elbonia, else 0.

4.1.2 Variable Documentation

4.1.2.1 int screen_height

height of screen in pixels.

4.1.2.2 int screen_width

width of screen in pixels.

4.1.2.3 int video_modes

supported video modes.

4.2 capabilities.h File Reference

Functions

- int **get_capabilities** (void)

4.2.1 Function Documentation

4.2.1.1 int get_capabilities (void)

Get capabilites and set variables defined in **capabilities.h**.

Returns:
1 if enough for running Elbonia, else 0.

4.3 client.c File Reference

```
#include <vstream.h>
#include <vmgp.h>
#include <stdlib.h>
#include <string.h>
#include "client.h"
#include "net.h"
#include "navigation.h"
#include "graphics.h"
#include "console.h"
#include "capabilities.h"
#include "history.h"
```

Functions

- int **get_page_information** (void)
- **__attribute__** ((destructor)) void savestate(void)
- int **main** (void)

Variables

- uint32_t **stream**
- char **data** [DATA_SIZE]
- char * **image**
- int **data_size**
- int **page_height**
- int **page_width**
- int **scroll_position**

4.3.1 Function Documentation

4.3.1.1 __attribute__ ((destructor))

Called when Elbonia is interrupted (by, for example, an incoming call). We hope this will also be called when the program is forced to quit by the user pressing No for a while. The only place this is documented is in the "Guidelines" provided on the Mophun home page.

4.3.1.2 int get_page_information (void)

Get page width and height.

Returns:
non-zero if client is available to display the video mode of the image, 0 otherwise.

4.3.1.3 int main (void)

4.3.2 Variable Documentation

4.3.2.1 char data[DATA_SIZE]

received data

4.3.2.2 int data_size

the size of the image and the imagemap

4.3.2.3 char* image

pointer to the image

4.3.2.4 int page_height

the page height

4.3.2.5 int page_width

the page width

4.3.2.6 int scroll_position ()

y position of upper edge of screen

4.3.2.7 uint32_t stream

socket

4.4 client.h File Reference

Defines

- #define **SERVER_ADDRESS** "127.0.0.1"
- #define **SERVER_PORT** 4711
- #define **DATA_SIZE** 15000

Functions

- int **get_page_information** (void)

Variables

- int **video_modes**

4.4.1 Define Documentation

4.4.1.1 #define DATA_SIZE 15000

4.4.1.2 #define SERVER_ADDRESS "127.0.0.1"

4.4.1.3 #define SERVER_PORT 4711

4.4.2 Function Documentation

4.4.2.1 int get_page_information (void)

Get page width and height.

Returns:
non-zero if client is available to display the video mode of the image, 0 otherwise.

4.4.3 Variable Documentation

4.4.3.1 int video_modes ()

supported video modes.

4.5 console.c File Reference

```
#include <vmgp.h>
#include <stdarg.h>
#include "console.h"
#include "meta.h"
```

Compounds

- struct **console_line**

Functions

- void **console_init** (void)
- void **console_render** (void)
- void **print** (char *str)
- void **println** (char *str)
- void **print_int** (int n)

Variables

- VMGPFONT **font**

4.5.1 Function Documentation

4.5.1.1 void console_init (void)

Initialize console usage. Must be called before any call to console_render or any of the print functions.

4.5.1.2 void console_render (void)

Render console to screen. This is automatically called after each print call.

4.5.1.3 void print (char * *str*)

Print a message to console. Used mainly for debug purposes.

Parameters:
str Message to print.

4.5.1.4 void print_int (int *n*)

Print an integer to console, followed by a newline. Used mainly for debug purposes.

Parameters:
n Integer to print.

4.5.1.5 void println (char * *str*)

Print a message to console followed by a newline. Used mainly for debug purposes.

Parameters:
str Message to print.

4.5.2 Variable Documentation

4.5.2.1 VMGPFONT font

Initial value:

```
{
    CONSOLE_FONT + CONSOLE_FONT_CHARTBLSIZE,
    CONSOLE_FONT,
    CONSOLE_FONT_BPP,
    CONSOLE_FONT_WIDTH,
    CONSOLE_FONT_HEIGHT,
    CONSOLE_FONT_PALINDEX
}
```

The console font to use.

4.6 console.h File Reference

Functions

- void **console_init** (void)
- void **console_render** (void)
- void **print** (char *str)
- void **println** (char *str)
- void **print_int** (int n)

4.6.1 Function Documentation

4.6.1.1 void console_init (void)

Initialize console usage. Must be called before any call to console_render or any of the print functions.

4.6.1.2 void console_render (void)

Render console to screen. This is automatically called after each print call.

4.6.1.3 void print (char * *str*)

Print a message to console. Used mainly for debug purposes.

Parameters:
str Message to print.

4.6.1.4 void print_int (int *n*)

Print an integer to console, followed by a newline. Used mainly for debug purposes.

Parameters:
n Integer to print.

4.6.1.5 void println (char * *str*)

Print a message to console followed by a newline. Used mainly for debug purposes.

Parameters:
str Message to print.

4.7 graphics.c File Reference

```
#include <stdlib.h>
#include <vmgp.h>
#include "navigation.h"
#include "graphics.h"
```

Functions

- int **render** (const char **image*)
- void **get_and_set_palette** (const char **image*, char ***rest*)

Variables

- **link imagemap** [NUM_LINKS]
- int **scroll_position**
- int **link_active**
- int **page_width**
- int **page_height**
- int **screen_width**
- int **screen_height**

4.7.1 Function Documentation

4.7.1.1 void get_and_set_palette (const char * *image*, char ** *rest*)

Used to extract palette information when indexed palette images are sent to the client.

Parameters:

image The image data

rest Pointer into image indicating start of image data

4.7.1.2 int render (const char * *image*)

Render screen and everything on it.

Parameters:

image The image to draw. Should point to a SPRITE struct, as defined by the Mophun API reference.

Returns:

1 on success, 0 otherwise.

4.7.2 Variable Documentation

4.7.2.1 link imagemap[NUM_LINKS] ()

the links

4.7.2.2 int link_active ()

active link

4.7.2.3 int page_height ()

the page height

4.7.2.4 int page_width ()

the page width

4.7.2.5 int screen_height ()

height of screen in pixels.

4.7.2.6 int screen_width ()

width of screen in pixels.

4.7.2.7 int scroll_position ()

y position of upper edge of screen

4.8 graphics.h File Reference

Functions

- int **render** (const char *)
- void **get_and_set_palette** (const char *, char **)

Variables

- **link imagemap** [NUM_LINKS]

4.8.1 Function Documentation

4.8.1.1 void get_and_set_palette (const char * *image*, char ** *rest*)

Used to extract palette information when indexed palette images are sent to the client.

Parameters:

image The image data

rest Pointer into image indicating start of image data

4.8.1.2 int render (const char * *image*)

Render screen and everything on it.

Parameters:

image The image to draw. Should point to a SPRITE struct, as defined by the Mophun API reference.

Returns:

1 on success, 0 otherwise.

4.8.2 Variable Documentation

4.8.2.1 link imagemap[NUM_LINKS] ()

the links

4.9 history.c File Reference

```
#include <string.h>
#include "history.h"
```

Functions

- void **history_add** (const char *page)
- char * **history_get** (void)
- int **history_is_empty** (void)

Variables

- char **history_item** [NUM_HISTORY_ITEMS][HISTORY_ITEM_LENGTH]
- int **first** = -1
- int **next** = 0

4.9.1 Function Documentation

4.9.1.1 void history_add (const char * *page*)

Add page to history.

Parameters:
page The page to add to history.

4.9.1.2 char* history_get (void)

Get most recently visited page. The behaviour if history is empty is undefined.

Returns:
The address of the most recently visited page.

4.9.1.3 int history_is_empty (void)

Returns:
1 if history is empty, 0 if not.

4.9.2 Variable Documentation

4.9.2.1 int first = -1

first item in list. -1 means empty.

4.9.2.2 char history_item[NUM_HISTORY_ITEMS][HISTORY_ITEM_LENGTH]

all the history items currently stored.

4.9.2.3 int next = 0

next free slot for items.

4.10 history.h File Reference

Defines

- #define **NUM_HISTORY_ITEMS** 5
- #define **HISTORY_ITEM_LENGTH** 256

Functions

- void **history_add** (const char *)
- char * **history_get** (void)
- int **history_is_empty** (void)

4.10.1 Define Documentation

4.10.1.1 #define HISTORY_ITEM_LENGTH 256

4.10.1.2 #define NUM_HISTORY_ITEMS 5

4.10.2 Function Documentation

4.10.2.1 void history_add (const char * *page*)

Add page to history.

Parameters:

page The page to add to history.

4.10.2.2 char* history_get (void)

Get most recently visited page. The behaviour if history is empty is undefined.

Returns:

The address of the most recently visited page.

4.10.2.3 int history_is_empty (void)

Returns:

1 if history is empty, 0 if not.

4.11 navigation.c File Reference

```
#include <string.h>
#include "navigation.h"
#include "client.h"
#include "net.h"
#include "history.h"
```

Functions

- int **link_follow** (int **stream**)
- int **link_is_visible** (int link_index)
- int **next_link** (void)
- int **previous_link** (void)
- int **scroll_up** (void)
- int **scroll_down** (void)
- int **get_imagemap** (const char *__data__, char **__rest__)

Variables

- int **num_links**
- **link imagemap** [NUM_LINKS]
- int **link_active**
- int **scroll_position**
- int **scroll_speed** = 4
- int **screen_height**
- int **page_height**
- char **data** [DATA_SIZE]
- int **data_size**

4.11.1 Function Documentation

4.11.1.1 int get_imagemap (const char * *data*, char ** *rest*)

Extract imagemap from data. Imagemap looks like (everything in network byte order)

- Number of links (4 bytes)
- Each link:
  - x1 (4 bytes)
  - y1 (4 bytes)
  - x2 (4 bytes)
  - y2 (4 bytes)
  - path (null-terminated)

If new page contains any links, the first one is set as active link. Otherwise no link becomes active.

Parameters:
data Data to extract imagemap from.
rest Set to point to byte immediately following imagemap.

Returns:
Number of hotspots in imagemap

4.11.1.2 int link_follow (int *stream*)

Follow active link if it's visible. If a link is followed, the page is set to be viewed from top and the path to the current page is saved.

Parameters:
stream Socket to which request should be sent.

Returns:
1 on success, else 0.

4.11.1.3 int link_is_visible (int *link_index*)

Determine whether a link is entirely visible.

Parameters:
link_index Index of link.

Returns:
1 if entire link is visible, -1 if link doesn't exist and 0 otherwise.

4.11.1.4 int next_link (void)

Move to next visible link (if one is present).

Returns:
New active link.

4.11.1.5 int previous_link (void)

Move to previous visible link (if one is present).

Returns:
New active link.

4.11.1.6 int scroll_down (void)

Scroll down. If active link becomes invisible and next link is visible it becomes the active link. If no link is visible active link is unchanged.

Returns:
New active link.

4.11.1.7 int scroll_up (void)

Scroll up. If active link becomes invisible and previous link is visible it becomes the active link. If no link is visible active link is unchanged.

Returns:
New active link.

4.11.2 Variable Documentation

4.11.2.1 char data[DATA_SIZE] ()

received data

4.11.2.2 int data_size ()

the size of the image and the imagemap

4.11.2.3 link imagemap[NUM_LINKS]

the links

4.11.2.4 int link_active

active link

4.11.2.5 int num_links

number of links on the page

4.11.2.6 int page_height ()

the page height

4.11.2.7 int screen_height ()

height of screen in pixels.

4.11.2.8 int scroll_position

y position of upper edge of screen

4.11.2.9 int scroll_speed = 4

speed of scrolling

4.12 navigation.h File Reference

Compounds

- struct **link**

Defines

- #define **NUM_LINKS** 20

Functions

- int **link_follow** (int)
- int **link_is_visible** (int)
- int **next_link** (void)
- int **previous_link** (void)
- int **scroll_up** (void)
- int **scroll_down** (void)
- int **get_imagemap** (const char *, char **)

4.12.1 Define Documentation

4.12.1.1 #define NUM_LINKS 20

4.12.2 Function Documentation

4.12.2.1 int get_imagemap (const char * *data*, char ** *rest*)

Extract imagemap from data. Imagemap looks like (everything in network byte order)

- Number of links (4 bytes)
- Each link:
  - x1 (4 bytes)
  - y1 (4 bytes)
  - x2 (4 bytes)
  - y2 (4 bytes)
  - path (null-terminated)

If new page contains any links, the first one is set as active link. Otherwise no link becomes active.

Parameters:

data Data to extract imagemap from.

rest Set to point to byte immediately following imagemap.

Returns:

Number of hotspots in imagemap

4.12.2.2 int link_follow (int *stream*)

Follow active link if it's visible. If a link is followed, the page is set to be viewed from top and the path to the current page is saved.

Parameters:
stream Socket to which request should be sent.

Returns:
1 on success, else 0.

4.12.2.3 int link_is_visible (int *link_index*)

Determine whether a link is entirely visible.

Parameters:
link_index Index of link.

Returns:
1 if entire link is visible, -1 if link doesn't exist and 0 otherwise.

4.12.2.4 int next_link (void)

Move to next visible link (if one is present).

Returns:
New active link.

4.12.2.5 int previous_link (void)

Move to previous visible link (if one is present).

Returns:
New active link.

4.12.2.6 int scroll_down (void)

Scroll down. If active link becomes invisible and next link is visible it becomes the active link. If no link is visible active link is unchanged.

Returns:
New active link.

4.12.2.7 int scroll_up (void)

Scroll up. If active link becomes invisible and previous link is visible it becomes the active link. If no link is visible active link is unchanged.

Returns:
New active link.

4.13 net.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include <vstream.h>
#include "net.h"
#include "console.h"
```

Functions

- int **connect** (const char *address, int port)
- int **request** (uint32_t **stream**, const char *location, void *buffer, uint32_t size)
- int **write** (uint32_t **stream**, void *buffer, uint32_t size)
- int **read** (uint32_t **stream**, void *buffer, uint32_t size)
- uint16_t **net_to_hosts** (uint16_t input)
- uint32_t **net_to_hosti** (uint32_t input)

Variables

- int **screen_height**
- int **screen_width**

4.13.1 Function Documentation

4.13.1.1 int connect (const char * *address*, int *port*)

Connect to elbonia server at address:port.

Parameters:

address Remote address.

port Remote port.

Returns:

The connection socket or -1 if failed.

4.13.1.2 uint32_t net_to_hosti (uint32_t *input*)

Convert network (big endian) to host order.

Parameters:

input The int to be converted to host order

Returns:

input converted to host order

4.13.1.3 uint16_t net_to_hosts (uint16_t *input*)

Convert network (big endian) to host order.

Parameters:
input The short to be converted to host order

Returns:
input converted to host order

4.13.1.4 int read (uint32_t *stream*, void * *buffer*, uint32_t *size*)

Read size number of bytes from stream into buffer.

Parameters:
stream Socket to read from.

buffer Buffer to fill with read data.

size Size of buffer.

Returns:
number of bytes read. -1 on error, 0 on end of file.

4.13.1.5 int request (uint32_t *stream*, const char * *location*, void * *buffer*, uint32_t *size*)

Request a page from stream. buffer is filled with response. size indicates size of buffer. TODO: this is not at all done.

Parameters:
stream Socket to write to.

location Identifier of resource to request.

buffer Buffer to fill with response.

size Size of buffer.

Returns:
1 on success, 0 on error.

4.13.1.6 int write (uint32_t *stream*, void * *buffer*, uint32_t *size*)

Write size number of bytes from buffer to stream.

Parameters:
stream Socket to write to.

buffer Buffer to read from.

size Size of buffer.

Returns:
Number of bytes written, -1 on error.

4.13.2 Variable Documentation

4.13.2.1 int screen_height ()

height of screen in pixels.

4.13.2.2 int screen_width ()

width of screen in pixels.

4.14 net.h File Reference

`#include <vmgp.h>`

Functions

- int **connect** (const char *, int)
- int **request** (uint32_t, const char *, void *, uint32_t)
- int **write** (uint32_t, void *, uint32_t)
- int **read** (uint32_t, void *, uint32_t)
- uint16_t **net_to_hosts** (uint16_t)
- uint32_t **net_to_hosti** (uint32_t)

4.14.1 Function Documentation

4.14.1.1 int connect (const char * *address*, int *port*)

Connect to elbonia server at address:port.

Parameters:

address Remote address.

port Remote port.

Returns:

The connection socket or -1 if failed.

4.14.1.2 uint32_t net_to_hosti (uint32_t *input*)

Convert network (big endian) to host order.

Parameters:

input The int to be converted to host order

Returns:

input converted to host order

4.14.1.3 uint16_t net_to_hosts (uint16_t *input*)

Convert network (big endian) to host order.

Parameters:

input The short to be converted to host order

Returns:

input converted to host order

4.14.1.4 int read (uint32_t *stream*, void * *buffer*, uint32_t *size*)

Read size number of bytes from stream into buffer.

Parameters:

stream Socket to read from.

buffer Buffer to fill with read data.

size Size of buffer.

Returns:

number of bytes read. -1 on error, 0 on end of file.

4.14.1.5 int request (uint32_t *stream*, const char * *location*, void * *buffer*, uint32_t *size*)

Request a page from stream. buffer is filled with response. size indicates size of buffer. TODO: this is not at all done.

Parameters:

stream Socket to write to.

location Identifier of resource to request.

buffer Buffer to fill with response.

size Size of buffer.

Returns:

1 on success, 0 on error.

4.14.1.6 int write (uint32_t *stream*, void * *buffer*, uint32_t *size*)

Write size number of bytes from buffer to stream.

Parameters:

stream Socket to write to.

buffer Buffer to read from.

size Size of buffer.

Returns:

Number of bytes written, -1 on error.

Index