

libcaca
0.99.beta16

Generated by Doxygen 1.7.6.1

Tue Feb 28 2012 10:54:54

pt3em

Contents

1	libcaca Documentation	1
1.1	Introduction	1
1.2	Developer's documentation	1
1.3	User's documentation	2
1.4	Misc	2
1.5	License	2
2	Libcaca authors	2
3	Libcaca news	3
3.1	Changes between 0.9.beta16 and 0.99.beta15	3
3.2	Changes between 0.9.beta15 and 0.99.beta14	3
3.3	Changes between 0.9.beta14 and 0.99.beta13	3
3.4	Changes between 0.9.beta13 and 0.99.beta12	3
3.5	Changes between 0.9.beta12 and 0.99.beta11	4
3.6	Changes between 0.9.beta11 and 0.99.beta10	4
3.7	Changes between 0.9.beta10 and 0.99.beta9	4
3.8	Changes between 0.9.beta9 and 0.99.beta8	4
3.9	Changes between 0.9.beta8 and 0.99.beta7	5
3.10	Changes between 0.9.beta7 and 0.99.beta6	5
3.11	Changes between 0.9.beta6 and 0.99.beta5	5
3.12	Changes between 0.9.beta5 and 0.99.beta4	5
3.13	Changes between 0.9.beta4 and 0.99.beta3	5
3.14	Changes between 0.9.beta3 and 0.99.beta2	6
3.15	Changes between 0.9.beta2 and 0.99.beta1	6
3.16	Changes between 0.9 and 0.99.beta1	6
3.17	Changes between 0.8 and 0.9	7
3.18	Changes between 0.7 and 0.8	7
3.19	Changes between 0.6 and 0.7	7
3.20	Changes between 0.5 and 0.6	7
3.21	Changes between 0.4 and 0.5	7

pt3em	
3.22 Changes between 0.3 and 0.4	8
3.23 Changes between 0.2 and 0.3	8
3.24 Changes between 0.1 and 0.2	8
3.25 New in 0.1	8
4 Libcaca thanks	9
4.1 Bugfixes and improvements	9
4.2 Reused code	9
4.3 Porters and packagers	9
5 The libcaca canvas format (version 1)	9
6 The libcaca font format (version 1)	10
7 Migrating from libcaca 0.x to the 1.0 API	11
7.1 Overview	12
7.2 Migration strategy	12
7.3 Function equivalence list	13
7.3.1 Basic functions	13
7.3.2 Event handling	13
7.3.3 Character printing	14
7.3.4 Primitives drawing	14
7.3.5 Mathematical functions	15
7.3.6 Sprite handling	15
7.3.7 Bitmap handling	15
7.4 Compilation	16
8 Libcaca coding style	16
8.1 General guidelines	16
8.2 C coding style	16
8.3 C++ coding style	17
9 A libcaca tutorial	17
10 Libcaca environment variables	18

pt3em

11 Libcaca ruby bindings	19
11.1 Libcaca Ruby API	19
12 Module Documentation	22
12.1 libcaca attribute definitions	22
12.1.1 Detailed Description	23
12.1.2 Define Documentation	23
12.2 libcaca basic functions	26
12.2.1 Detailed Description	26
12.2.2 Function Documentation	26
12.3 libcaca canvas drawing	33
12.3.1 Detailed Description	33
12.3.2 Define Documentation	34
12.3.3 Function Documentation	34
12.4 libcaca canvas transformation	41
12.4.1 Detailed Description	41
12.4.2 Function Documentation	41
12.5 libcaca attribute conversions	47
12.5.1 Detailed Description	47
12.5.2 Function Documentation	47
12.6 libcaca character set conversions	55
12.6.1 Detailed Description	55
12.6.2 Function Documentation	55
12.7 libcaca primitives drawing	59
12.7.1 Detailed Description	59
12.7.2 Function Documentation	60
12.8 libcaca canvas frame handling	68
12.8.1 Detailed Description	68
12.8.2 Function Documentation	68
12.9 libcaca bitmap dithering	73
12.9.1 Detailed Description	74
12.9.2 Function Documentation	74
12.10libcaca font handling	85

pt3em	
12.10.1 Detailed Description	85
12.10.2 Function Documentation	85
12.11 libcaca FIGfont handling	90
12.11.1 Detailed Description	90
12.12 libcaca file IO	92
12.12.1 Detailed Description	92
12.13 libcaca importers/exporters from/to various	94
12.13.1 Detailed Description	94
12.14 libcaca display functions	96
12.14.1 Detailed Description	96
12.14.2 Function Documentation	97
12.15 libcaca event handling	105
12.15.1 Detailed Description	105
12.15.2 Function Documentation	105
13 Data Structure Documentation	111
13.1 caca_event Struct Reference	111
13.1.1 Detailed Description	111
14 File Documentation	111
14.1 caca.h File Reference	111
14.1.1 Detailed Description	120
14.1.2 Define Documentation	120
14.1.3 Typedef Documentation	120
14.1.4 Enumeration Type Documentation	121

1 libcaca Documentation

1.1 Introduction

libcaca is a graphics library that outputs text instead of pixels, so that it can work on older video cards or text terminals. It is not unlike the famous AAlib library. *libcaca* can use almost any virtual terminal to work, thus it should work on all Unix systems (including Mac OS X) using either the S-Lang library or the ncurses library, on DOS using the conio library, and on Windows systems using the native Win32 console, the conio library, or using S-Lang or ncurses (through Cygwin emulation). There is also a

pt3em

native X11 driver, and an OpenGL driver (through freeglut) that does not require a text terminal. For machines without a screen, the raw driver can be used to send the output to another machine, using for instance cacaserver.

libcaca is free software, released under the Do What The Fuck You Want To Public - License. This ensures that no one, not even the *libcaca* developers, will ever have anything to say about what you do with the software. It used to be licensed under the GNU Lesser General Public License, but that was not free enough.

1.2 Developer's documentation

The complete *libcaca* programming interface is available from the following header:

- [caca.h](#)

There is language-specific documentation for the various bindings:

- [Libcaca ruby bindings](#)

Some other topics are covered by specific sections:

- [A libcaca tutorial](#)
- [Migrating from libcaca 0.x to the 1.0 API](#)

There is also information specially targeted at *libcaca* developers:

- [The libcaca font format \(version 1\)](#)
- [The libcaca canvas format \(version 1\)](#)
- [Libcaca coding style](#)

1.3 User's documentation

- [Libcaca environment variables](#)

1.4 Misc

- [Libcaca news](#)
- [Libcaca authors](#)
- [Libcaca thanks](#)

pt3em

1.5 License

Permission is granted to copy, distribute and/or modify this document under the terms of the Do What The Fuck You Want To Public License, version 2 as published by Sam Hocevar. For details see <http://sam.zoy.org/wtfpl/>.

2 Libcaca authors

Sam Hocevar <sam@zoy.org>

- main programmer

Jean-Yves Lamoureux <jylam@lnxscene.org>

- cacaball
- OpenGL driver
- Pypycaca Python wrapper
- exporters
- network driver
- C# bindings

John Beppu <beppu@lbox.org>

- Term::Caca Perl wrapper

Ben Wiley Sittler <bsittler@gmail.com>

- numerous bugfixes and enhancements

Pascal Terjan <pterjan@linuxfr.org>

- Ruby bindings

Daniele "Eriol" Tricoli <eriol@mornie.org>

- Python CTypes sample program

3 Libcaca news

3.1 Changes between 0.9.beta16 and 0.99.beta15

- many build fixes, especially for nonstandard platforms

pt3em

3.2 Changes between 0.9.beta15 and 0.99.beta14

- libcucul was merged back into libcaca for more clarity

3.3 Changes between 0.9.beta14 and 0.99.beta13

- internal FIGlet font support
- use C99 types in public headers
- runtime support for output drivers
- BBcode export support

3.4 Changes between 0.9.beta13 and 0.99.beta12

- device-dependent cursor support
- event API rewrite
- minor API improvements and extensions
- img2txt improvements
- Ruby bindings
- Massive C# bindings improvements
- Python sample code
- Visual Studio build solution

3.5 Changes between 0.9.beta12 and 0.99.beta11

- support for 90-degree canvas rotation
- better behaviour when trying to output Unicode on an ASCII terminal
- the built-in font now supports the Geometric Shapes, Halfwidth and Fullwidth - Forms and Miscellaneous Symbols Unicode blocks
- new rotozoom effect in cacademo
- Cocoa output driver for Mac OS X
- preliminary .NET bindings
- many bugfixes and documentation changes

pt3em

3.6 Changes between 0.9.beta11 and 0.99.beta10

- fixed compilation of the C++ bindings
- fixed bugs in `cucul_import_memory()`, `cucul_set_canvas_size()`
- implemented [caca_set_display_title\(\)](#) for ncurses and S-Lang
- minor bugfixes

3.7 Changes between 0.9.beta10 and 0.99.beta9

- new debug mode
- blitting canvases now makes use of the canvas' handle coordinates
- import functions can read streamed data
- attribute to colorspace transformations
- added katakana and hiragana glyphs to the built-in font
- many bugfixes and documentation changes

3.8 Changes between 0.9.beta9 and 0.99.beta8

- support for blink, bold, italics and underline attributes
- allow to import and export zero-sized canvases
- fixed Imlib2 support in cacaview
- fixed buffer overflows in the file importer
- big documentation updates

3.9 Changes between 0.9.beta8 and 0.99.beta7

- allow to build the X11 and GL drivers as separate plugins
- support for fullwidth Unicode characters
- improved `cucul_flip()` and `cucul_rotate()`
- minor bugfixes and documentation updates

pt3em

3.10 Changes between 0.9.beta7 and 0.99.beta6

- transparency support in the UTF-8 importer and exporter
- optimised triangle fill routine
- updated C++ bindings

3.11 Changes between 0.9.beta6 and 0.99.beta5

- ANSI importer now handles transparency and UTF-8
- Unicode support was broken for about 10% of the set
- various memory leaks were fixed

3.12 Changes between 0.9.beta5 and 0.99.beta4

- implemented `cucul_getchar()` and `cucul_get_color()`
- handle transparency in the IRC export
- new cropping and expanding filters
- full Unicode support in the OpenGL driver
- portability fixes for 64-bit architectures, Win32 and MS-DOS
- all demos except `cacafire` were merged into `cacademo`

3.13 Changes between 0.9.beta4 and 0.99.beta3

- added a compatibility layer for pre-1.x `libcaca` applications
- fixed manpage generation
- minor bugfixes and documentation updates

3.14 Changes between 0.9.beta3 and 0.99.beta2

- `libcaca` functions use `errno` for error reporting
- updated C++ bindings
- minor improvements, bugfixes and documentation updates

pt3em

3.15 Changes between 0.9.beta2 and 0.99.beta1

- ANSI importer
- functions use errno for error reporting
- updated C++ bindings
- .NET bindings
- cacadraw, an ANSI viewer that will evolve into an editor
- Unicode input and output support for SLang and ncurses
- built-in fonts work on Win32

3.16 Changes between 0.9 and 0.99.beta1

- license switched to WTFPL
- libcaca was split into libcul, a standalone text manipulation backend, and libcaca, the display and user input frontend
- Unicode support
- TrueColor (more than 16 colours) support
- Floyd-Steinberg dithering
- gamma correction
- export functions for HTML, IRC, ANSI, SVG, PostScript, TGA...
- builtin fonts for device-independent bitmap output
- various text transformation routines (rotation, horizontal flip...)
- OpenGL renderer
- kernel mode to build libcaca programs into a bootable x86 kernel
- cacaserver, a telnet server that can be hooked to libcaca applications
- img2irc, an image to IRC conversion utility

3.17 Changes between 0.8 and 0.9

- fix for a buffer overflow in the line rendering
- fixed resizing in the ncurses and slang drivers
- aspect ratio and finer zoom support in cacaview
- minor compilation fixes

pt3em

3.18 Changes between 0.7 and 0.8

- window resizing support
- native Win32 port
- autorepeat emulation in the ncurses and slang drivers
- support for more keycodes in the ncurses and slang drivers
- cacaplas, a plasma animation example
- cacamoir, a moire circles animation example
- MSVC project file

3.19 Changes between 0.6 and 0.7

- many bugfixes in the event handling
- cacaball, a metaball animation example

3.20 Changes between 0.5 and 0.6

- 30% speed increase in the bitmap rendering routine
- mouse support and various speed optimisations in the X11 driver
- X11 is now the preferred driver
- improved documentation
- minor bugfixes

3.21 Changes between 0.4 and 0.5

- palette optimisation for the S-Lang driver to work around the colour pair shortage bug
- minor compilation fix

pt3em

3.22 Changes between 0.3 and 0.4

- preliminary X11 graphics driver
- support for simultaneously compiled-in drivers
- honour the CACA_DRIVER, CACA_GEOMETRY and CACA_FONT environment variables
- more documentation

3.23 Changes between 0.2 and 0.3

- antialiasing support
- dithering, antialiasing and background mode can now be selected at runtime or in the environment using the CACA_BACKGROUND, CACA_DITHERING and CACA_ANTIALIASING variables
- alpha channel support in cacaview
- BMP loading support in cacaview even if lmlib2 is not present
- cacafire, a libcaca port of aafire

3.24 Changes between 0.1 and 0.2

- rendering now uses 256 colour pairs instead of 16
- mouse support for ncurses
- ncurses is now the preferred backend
- arbitrary color depth and bitmasks in the bitmap renderer
- cacaview, an image viewer based on libcaca

3.25 New in 0.1

- initial release
- slang, ncurses and conio drivers
- basic line, box, ellipse and triangle primitives
- colour bitmap blitting

pt3em

4 Libcaca thanks

4.1 Bugfixes and improvements

- Gildas Bazin <gbazin@netcourrier.com> - win32 driver improvements
- Jari Komppa <jari.komppa at gmail> - win32 speed improvements

4.2 Reused code

- Jan Hubicka <hubicka@freesoftware.cz> - aafire
- Michele Bini <mibini@tin.it> - original SDL plasma
- Free Software Foundation, Inc. - multiboot.S

4.3 Porters and packagers

- Derk-Jan Hartman <thedj@users.sourceforge.net> - Gentoo ebuild file
- Ladislav Hagara <hgr@vabo.cz> - Source Mage spell
- Philip Balinov - Slackware package
- Richard Zidlicky <rz@linux-m68k.org> - rpm specfile
- Thomas Klausner <wiz@NetBSD.org> - NetBSD port maintainer
- Vincent Tarrantini <vinc@FreeBSD-fr.org> - FreeBSD port maintainer

5 The libcaca canvas format (version 1)

All types are big endian.

```
struct
{
    magic:
        uint8_t caca_header[2];    // "\xCA\xCA"
        uint8_t caca_file_type[2]; // "CV"

    canvas_header:
        uint32_t control_size;      // Control size (canvas_data - canvas_header)
        uint32_t data_size;        // Data size (EOF - canvas_data)

        uint16_t version;          // Canvas format version
                                   // bit 0: set to 1 if canvas is compatible
```

```

pt3em
//          with version 1 of the format
//  bits 1-15: unused yet, must be 0

uint32_t frames;           // Frame count

uint16_t flags;           // Feature flags
//  bits 0-15: unused yet, must be 0

frame_info:
struct
{
    uint32_t width;        // Frame width
    uint32_t height;      // Frame height
    uint32_t duration;     // Frame duration in milliseconds, 0 to
// not specify a duration
    uint32_t attr;        // Graphics context attribute
    int32_t cursor_x;     // Cursor X coordinate
    int32_t cursor_y;     // Cursor Y coordinate
    int32_t handle_x;     // Handle X coordinate
    int32_t handle_y;     // Handle Y coordinate
}
frame_list[frames];

control_extension_1:
control_extension_2:
...
control_extension_N:
...                       // reserved for future use

canvas_data:
    uint8_t data[data_size]; // canvas data

data_extension_1:
data_extension_2:
...
data_extension_N:
...                       // reserved for future use
};

```

6 The libcaca font format (version 1)

All types are big endian.

```

struct
{
magic:
    uint8_t caca_header[2]; // "\xCA\xCA"
    uint8_t caca_file_type[2]; // "FT"

font_header:
    uint32_t control_size; // Control size (font_data - font_header)
    uint32_t data_size; // Data size (EOF - font_data)

    uint16_t version; // Font format version
// bit 0: set to 1 if font is compatible
//          with version 1 of the format
//  bits 1-15: unused yet, must be 0

```

```

pt3em
    uint16_t blocks;           // Number of blocks in the font
    uint32_t glyphs;          // Total number of glyphs in the font

    uint16_t bpp;              // Bits per pixel for glyph data (valid
                                // Values are 1, 2, 4 and 8)
    uint16_t width;            // Standard glyph width
    uint16_t height;           // Standard glyph height
    uint16_t maxwidth;         // Maximum glyph width
    uint16_t maxheight;        // Maximum glyph height

    uint16_t flags;            // Feature flags
                                // bit 0: set to 1 if font is fixed width
                                // bits 1-15: unused yet, must be 0

block_info:
    struct
    {
        uint32_t start;        // Unicode index of the first glyph
        uint32_t stop;         // Unicode index of the last glyph + 1
        uint32_t index;        // Glyph info index of the first glyph
    }
    block_list[blocks];

glyph_info:
    struct
    {
        uint16_t width;        // Glyph width in pixels
        uint16_t height;       // Glyph height in pixels
        uint32_t data_offset;   // Offset (starting from data) to the data
                                // for the first character
    }
    glyph_list[glyphs];

control_extension_1:
control_extension_2:
    ...
control_extension_N:
    ...                        // reserved for future use

font_data:
    uint8_t data[data_size];   // glyph data

data_extension_1:
data_extension_2:
    ...
data_extension_N:
    ...                        // reserved for future use
};

```

7 Migrating from libcaca 0.x to the 1.0 API

This section will guide you through the migration of a *libcaca* 0.x application to the latest API version.

pt3em

7.1 Overview

The most important change in the 1.0 API of *libcaca* is the object-oriented design. See these two examples for a rough idea of what changed:

pt3em	pt3em
<pre> #include <caca.h> /* libcaca program - 0.x API */ int main(void) { /* Initialise libcaca */ caca_init(); /* Set window title */ caca_set_window_title("Window"); /* Choose drawing colours */ caca_set_color(CACA_COLOR_BLACK, CACA_COLOR_WHITE); /* Draw a string at (0, 0) */ caca_putstr(0, 0, "Hello world!"); /* Refresh display */ caca_refresh(); /* Wait for a key press event */ caca_wait_event(CACA_EVENT_KEY_PRESS); /* Clean up library */ caca_end(); return 0; } </pre>	<pre> #include <caca.h> /* libcaca program - 1.0 API */ int main(void) { /* Initialise libcaca */ caca_canvas_t *cv; caca_display_t *dp; dp = caca_create_display(cv); cv = caca_get_canvas(dp); /* Set window title */ caca_set_display_title(dp, "Window"); /* Choose drawing colours */ caca_set_color_ansi(cv, CACA_BLACK, CACA_WHITE); /* Draw a string at (0, 0) */ caca_put_str(cv, 0, 0, "Hello world!"); /* Refresh display */ caca_refresh_display(); /* Wait for a key press event */ caca_get_event(dp, CACA_EVENT_KEY_PRESS, NULL, -1); /* Clean up library */ caca_free_display(dp); return 0; } </pre>

Note the following important things:

- Functions now take an object handle as their first argument.
- All input/output functions start with **caca_** and all drawing and text handling functions start with **caca_**.

7.2 Migration strategy

You have two ways to migrate your application to use *libcaca* 1.x:

- Port your code using the function equivalence list. This is the preferred way because new functions are thread safe and offer much more features to both the programmer and the end user.
- Use the legacy compatibility layer.

Using the compatibility layer is as easy as adding the following three lines:

pt3em

pt3em	pt3em
<pre>#include <caca.h> /* libcaca program - 0.x API */ ...</pre>	<pre>#include <caca.h> #ifdef CACA_API_VERSION_1 # include <caca0.h> #endif /* libcaca program - 0.x API */ ...</pre>

7.3 Function equivalence list

7.3.1 Basic functions

- **caca_init()**: use [caca_create_canvas\(\)](#) to create a *libcaca* canvas, followed by [caca_create_display\(\)](#) to attach a *libcaca* display to it.
- **caca_set_delay()**: use [caca_set_display_time\(\)](#).
- **caca_get_feature()**: deprecated.
- **caca_set_feature()**: deprecated, see [caca_set_dither_antialias\(\)](#), [caca_set_dither_color\(\)](#) and [caca_set_dither_mode\(\)](#) instead.
- **caca_get_feature_name()**: deprecated, see [caca_get_dither_mode_list\(\)](#), [caca_get_dither_antialias_list\(\)](#) and [caca_get_dither_color_list\(\)](#) instead.
- **caca_get_rendertime()**: use [caca_get_display_time\(\)](#).
- **caca_get_width()**: use [caca_get_canvas_width\(\)](#).
- **caca_get_height()**: use [caca_get_canvas_height\(\)](#).
- **caca_set_window_title()**: use [caca_set_display_title\(\)](#).
- **caca_get_window_width()**: use [caca_get_display_width\(\)](#).
- **caca_get_window_height()**: use [caca_get_display_height\(\)](#).
- **caca_refresh()**: use [caca_refresh_display\(\)](#).
- **caca_end()**: use [caca_free_display\(\)](#) to detach the *libcaca* display, followed by [caca_free_canvas\(\)](#) to free the underlying *libcaca* canvas.

7.3.2 Event handling

- **caca_get_event()**: unchanged, but the event information retrieval changed a lot.
- **caca_wait_event()**: use [caca_get_event\(\)](#) with a `timeout` argument of `-1`.
- **caca_get_mouse_x()**: unchanged.
- **caca_get_mouse_y()**: unchanged.

pt3em

7.3.3 Character printing

- **caca_set_color()**: use [caca_set_color_ansi\(\)](#) or [caca_set_color_argb\(\)](#).
- **caca_get_fg_color()**: use [caca_get_attr\(\)](#).
- **caca_get_bg_color()**: use [caca_get_attr\(\)](#).
- **caca_get_color_name()**: this function is now deprecated due to major uselessness.
- **caca_putchar()**: use [caca_put_char\(\)](#).
- **caca_putstr()**: use [caca_put_str\(\)](#).
- **caca_printf()**: use [caca_printf\(\)](#).
- **caca_clear()**: use [caca_clear_canvas\(\)](#).

7.3.4 Primitives drawing

These functions are almost unchanged, except for Unicode support and the fact that they now act on a given canvas.

- **caca_draw_line()**: use [caca_draw_line\(\)](#).
- **caca_draw_polyline()**: use [caca_draw_polyline\(\)](#).
- **caca_draw_thin_line()**: use [caca_draw_thin_line\(\)](#).
- **caca_draw_thin_polyline()**: use [caca_draw_thin_polyline\(\)](#).
- **caca_draw_circle()**: use [caca_draw_circle\(\)](#).
- **caca_draw_ellipse()**: use [caca_draw_ellipse\(\)](#).
- **caca_draw_thin_ellipse()**: use [caca_draw_thin_ellipse\(\)](#).
- **caca_fill_ellipse()**: use [caca_fill_ellipse\(\)](#).
- **caca_draw_box()**: use [caca_draw_box\(\)](#).
- **caca_draw_thin_box()**: use [caca_draw_thin_box\(\)](#) or [caca_draw_cp437_box\(\)](#).
- **caca_fill_box()**: use [caca_fill_box\(\)](#).
- **caca_draw_triangle()**: use [caca_draw_triangle\(\)](#).
- **caca_draw_thin_triangle()**: use [caca_draw_thin_triangle\(\)](#).
- **caca_fill_triangle()**: use [caca_fill_triangle\(\)](#).

pt3em

7.3.5 Mathematical functions

- **caca_rand()**: use [caca_rand\(\)](#). The second argument is different, make sure you take that into account.
- **caca_sqrt()**: this function is now deprecated, use your system's **sqrt()** call instead.

7.3.6 Sprite handling

The newly introduced canvases can have several frames. Sprites are hence completely deprecated.

- **caca_load_sprite()**: use [caca_import_file\(\)](#).
- **caca_get_sprite_frames()**: use [caca_get_frame_count\(\)](#).
- **caca_get_sprite_width()**: use [caca_get_canvas_width\(\)](#).
- **caca_get_sprite_height()**: use [caca_get_canvas_height\(\)](#).
- **caca_get_sprite_dx()**: use [caca_get_canvas_handle_x\(\)](#).
- **caca_get_sprite_dy()**: use [caca_get_canvas_handle_y\(\)](#).
- **caca_draw_sprite()**: use [caca_set_frame\(\)](#) and [caca_blit\(\)](#).
- **caca_free_sprite()**: use [caca_free_canvas\(\)](#).

7.3.7 Bitmap handling

Bitmaps have been renamed to dithers, because these objects do not in fact store any pixels, they just have information on how bitmaps will be dithered.

- **caca_create_bitmap()**: use [caca_create_dither\(\)](#).
- **caca_set_bitmap_palette()**: use [caca_set_dither_palette\(\)](#).
- **caca_draw_bitmap()**: use [caca_dither_bitmap\(\)](#).
- **caca_free_bitmap()**: use [caca_free_dither\(\)](#).

pt3em

7.4 Compilation

The `caca-config` utility is deprecated in favour of the standard `pkg-config` interface:

```
gcc -c foobar.c -o foobar.o `pkg-config --cflags caca`  
gcc foobar.o -o foobar `pkg-config --libs caca`
```

`caca-config` is still provided as a convenience tool but may be removed in the future.

8 Libcaca coding style

8.1 General guidelines

A pretty safe rule of thumb is: look at what has already been done and try to do the same.

- Tabulations should be avoided and replaced with *eight* spaces.
- Indentation is generally 4 spaces.
- Lines should wrap at most at 79 characters.
- Do not leave whitespace at the end of lines.
- Do not use multiple spaces for anything else than indentation.
- Code qui fait des warnings == code de porc == deux baffes dans ta gueule

8.2 C coding style

Try to use short names whenever possible (`i` for indices, `w` for width, `cv` for canvas...). Macros are always uppercase, variable and function names are always lowercase. Use the underscore to separate words within names:

```
#define BROKEN 0  
#define MAX(x, y) ((x > y) ? (x) : (y))  
  
unsigned int x, y, w, h;  
char *font_name;  
void frobulate_every_three_seconds(void);
```

`const` is a *suffix*. It's `char const *foo`, not `const char *foo`.

Use spaces after commas and between operators. Do not use spaces after an opening parenthesis or before a closing one:

pt3em

```
a += 2;
b = (a * (c + d));
x = min(x1, x2, x3);
```

Do not put a space between functions and the corresponding opening parenthesis:

```
int function(int);

if(a == b)
    return;
```

Do not put parentheses around return values:

```
return a + (b & x) + d[10];
```

Opening braces should be on a line of their own, aligned with the current block. Braces are optional for one-liners:

```
int function(int a)
{
    if(a & 0x84)
        return a;

    if(a < 0)
    {
        return -a;
    }
    else
    {
        a /= 2;

        switch(a)
        {
            case 0:
            case 1:
                return -1;
                break;
            default:
                return a;
        }
    }
}
```

8.3 C++ coding style

Nothing here yet.

9 A libcaca tutorial

First, a very simple working program, to check for basic libcaca functionalities.

```

pt3em

#include <caca.h>

int main(void)
{
    caca_canvas_t *cv; caca_display_t *dp; caca_event_t ev;

    dp = caca_create_display(NULL);
    if(!dp) return 1;
    cv = caca_get_canvas(dp);

    caca_set_display_title(dp, "Hello!");
    caca_set_color_ansi(cv, CACA_BLACK, CACA_WHITE);
    caca_put_str(cv, 0, 0, "This is a message");
    caca_refresh_display(dp);
    caca_get_event(dp, CACA_EVENT_KEY_PRESS, &ev, -1);
    caca_free_display(dp);

    return 0;
}

```

What does it do?

- Create a display. Physically, the display is either a window or a context in a terminal (ncurses, slang) or even the whole screen (VGA).
- Get the display's associated canvas. A canvas is the surface where everything happens: writing characters, sprites, strings, images... It is unavoidable. Here the size of the canvas is set by the display.
- Set the display's window name (only available in windowed displays, does nothing otherwise).
- Set the current canvas colours to black background and white foreground.
- Write the string "This is a message" using the current colors onto the canvas.
- Refresh the display.
- Wait for an event of type "CACA_EVENT_KEY_PRESS".
- Free the display (release memory). Since it was created together with the display, the canvas will be automatically freed as well.

You can then compile this code on an UNIX-like system using the following command (requiring pkg-config and gcc):

```
gcc `pkg-config --libs --cflags caca` example.c -o example
```

10 Libcaca environment variables

Some environment variables can be used to change the behaviour of *libcaca* without having to modify the program which uses them. These variables are:

pt3em

- **CACA_DRIVER:** set the backend video driver. In order of preference:
 - `conio` uses the DOS `conio.h` interface.
 - `ncurses` uses the `ncurses` library.
 - `slang` uses the S-Lang library.
 - `x11` uses the native X11 driver.
 - `gl` uses `freeglut` and `opengl` libraries.
 - `raw` outputs to the standard output instead of rendering the canvas. This can be used together with `cacaserver`.
- **CACA_GEOMETRY:** set the video display size. The format of this variable must be `XxY`, with `X` and `Y` being integer values. This option currently works with the `raw`, `X11` and `GL` drivers.
- **CACA_FONT:** set the rendered font. The format of this variable is implementation dependent, but since it currently only works with the `X11` driver, an `X11` font name such as `fixed` or `5x7` is expected.

11 Libcaca ruby bindings

There is no real documentation yet for the Ruby binding but `methods` on any object should help you :)

I tried to follow Ruby spirit meaning that :

- most of the methods return `self`
- the methods `set_foo` with only an argument are also available as `foo=` (returning the value instead of `self`)
- the methods originally named `get_foo` are available only as `foo`

For the list of methods and some sample code, read:

[Libcaca Ruby API](#)

11.1 Libcaca Ruby API

The classes available for `libcaca` are :

- **Caca::Canvas** : functions that have a `caca_canvas_t*` as first argument
- **Caca::Dither** : functions that have a `caca_dither_t*` as first argument
- **Caca::Font** : functions that have a `caca_font_t*` as first argument (The constructor can currently only accept the name of a builtin font)

pt3em

- **Caca::Display**
- **Caca::Event**
- **Caca::Event::Key**
- **Caca::Event::Key::Press**
- **Caca::Event::Key::Release**
- **Caca::Event::Mouse**
- **Caca::Event::Mouse::Press**
- **Caca::Event::Mouse::Release**
- **Caca::Event::Mouse::Motion**
- **Caca::Event::Resize**
- **Caca::Event::Quit**

The character set conversion functions are not available yet in the binding.

```
$irb -rcaca
irb(main):001:0>class Object
irb(main):002:1>def Object.my_instance_methods
irb(main):003:2>instance_methods.sort - ancestors[1].instance_methods
irb(main):004:2>end
irb(main):005:1>def Object.my_methods
irb(main):006:2>methods.sort - ancestors[1].methods
irb(main):007:2>end
irb(main):008:1>end

irb(main):009:0>Caca.constants
=>["BROWN", "BOLD", "GREEN", "LIGHTMAGENTA", "LIGHTBLUE", "BLINK",
"MAGENTA","DEFAULT", "TRANSPARENT", "BLUE", "LIGHTRED", "DARKGRAY",
"UNDERLINE","RED", "WHITE", "BLACK", "LIGHTCYAN", "LIGHTGRAY",
"ITALICS","CYAN", "YELLOW", "LIGHTGREEN", "Canvas", "Dither", "Font"]

irb(main):010:0>Caca.my_methods
=>["version"]

irb(main):011:0>Caca::Canvas.my_methods
=>["export_list", "import_list"]

irb(main):012:0>Caca::Canvas.my_instance_methods
=>["attr=", "blit", "clear", "create_frame", "cursor_x", "cursor_y",
"dither_bitmap","draw_box", "draw_circle", "draw_cp437_box", "draw_ellipse",
"draw_line","draw_polyline", "draw_thin_box", "draw_thin_ellipse",
"draw_thin_line","draw_thin_polyline", "draw_thin_triangle",
"draw_triangle","export_memory", "fill_box", "fill_ellipse",
"fill_triangle","flip", "flop", "frame=", "frame_count", "frame_name",
"frame_name=", "free_frame", "get_attr", "get_char", "gotoxy",
"handle_x","handle_y", "height", "height=", "import_file",
"import_memory","invert", "printf", "put_attr", "put_char", "put_str",
```

```

pt3em
"rotate_180", "rotate_left", "rotate_right", "set_attr",
"set_boundaries", "set_color_ansi", "set_color_argb", "set_frame",
"set_frame_name", "set_handle", "set_height", "set_size", "set_width",
"stretch_left", "stretch_right", "width", "width="]

irb(main):013:0>Caca::Font.my_methods
=>["list"]

irb(main):014:0>Caca::Font.my_instance_methods
=>["blocks", "height", "width"]

irb(main):015:0>Caca::Dither.my_instance_methods
=>["algorithm=", "algorithm_list", "antialias=", "antialias_list",
"brightness=", "charset=", "charset_list", "color=", "color_list",
"contrast=", "gamma=", "palette=", "set_algorithm", "set_antialias",
"set_brightness", "set_charset", "set_color", "set_contrast",
"set_gamma", "set_palette"]

irb(main):010:0>Caca::Display.my_instance_methods
=>["canvas", "get_event", "height", "mouse=", "mouse_x", "mouse_y", "refresh",
"set_mouse", "set_time", "set_title", "time", "time=", "title=", "width"]

irb(main):011:0>Caca::Event.constants
=>["Key", "Quit", "TYPE", "Mouse", "Resize"]

irb(main):012:0>Caca::Event.my_instance_methods
=>["quit?"]

irb(main):013:0>Caca::Event::Key.my_instance_methods
=>["ch", "utf32", "utf8"]

irb(main):014:0>Caca::Event::Mouse.my_instance_methods
=>["button", "x", "y"]

irb(main):015:0>Caca::Event::Resize.my_instance_methods
=>["w", "h"]

$ruby -rcaca -e 'c=Caca::Canvas.new(6, 3).fill_box(0,0,2,2,"#[0]);
c2=Caca::Canvas.new(1,1).put_str(0,0,"x"); c.blit(1,1,c2); puts
c.export_memory("irc")'
###
#x#
###

$ruby -e 'puts Caca::Canvas.new(6,3).draw_thin_polyline([[0,0], [0,2],
[5,2],[0,0]]).export_memory("irc")'
-
| \
----\'

$ruby -rcaca -e 'p Caca::Canvas.export_list'
[["caca", "native libcaca format"], ["ansi", "ANSI"], ["utf8", "UTF-8
withANSI escape codes"], ["utf8cr", "UTF-8 with ANSI escape codes and
MS-DOS\\r"], ["html", "HTML"], ["html3", "backwards-compatible HTML"],
["irc", "IRC with mIRC colours"], ["ps", "PostScript document"], ["svg",
"SVGvector image"], ["tga", "TGA image"]]

```

pt3em

```
$ruby -rcaca -e 'p Caca::Font.list'
["Monospace9", "Monospace Bold 12"]

require 'caca'
c= Caca::Canvas.new(20,10)
c.put_str(2,3, "plop!")
c.draw_thin_polyline([[0,0],[0,2], [5,2], [0,0]])
d= Caca::Display.new(c)
d.title= "Test !"
d.refresh

#Redefine Event::Key#quit? so that q, Q, and Esc become exit keys
module Caca
  class Event::Key
    def quit?
      "qQ^[\".split(' ').member?(@ch.chr)
    end
  end
end

while((e= d.get_event(Caca::Event, -1)) && ! e.quit?)
  p e
  d.refresh
end
```

12 Module Documentation

12.1 libcaca attribute definitions

Defines

- #define [CACA_BLACK](#) 0x00
- #define [CACA_BLUE](#) 0x01
- #define [CACA_GREEN](#) 0x02
- #define [CACA_CYAN](#) 0x03
- #define [CACA_RED](#) 0x04
- #define [CACA_MAGENTA](#) 0x05
- #define [CACA_BROWN](#) 0x06
- #define [CACA_LIGHTGRAY](#) 0x07
- #define [CACA_DARKGRAY](#) 0x08
- #define [CACA_LIGHTBLUE](#) 0x09
- #define [CACA_LIGHTGREEN](#) 0x0a
- #define [CACA_LIGHTCYAN](#) 0x0b
- #define [CACA_LIGHTRED](#) 0x0c
- #define [CACA_LIGHTMAGENTA](#) 0x0d
- #define [CACA_YELLOW](#) 0x0e
- #define [CACA_WHITE](#) 0x0f
- #define [CACA_DEFAULT](#) 0x10
- #define [CACA_TRANSPARENT](#) 0x20

pt3em

- #define [CACA_BOLD](#) 0x01
- #define [CACA_ITALICS](#) 0x02
- #define [CACA_UNDERLINE](#) 0x04
- #define [CACA_BLINK](#) 0x08

12.1.1 Detailed Description

Colours and styles that can be used with [caca_set_attr\(\)](#).

12.1.2 Define Documentation

12.1.2.1 #define [CACA_BLACK](#) 0x00

The colour index for black.

Referenced by [caca_attr_to_ansi\(\)](#), [caca_attr_to_argb64\(\)](#), [caca_attr_to_rgb12_bg\(\)](#), and [caca_dither_bitmap\(\)](#).

12.1.2.2 #define [CACA_BLUE](#) 0x01

The colour index for blue.

12.1.2.3 #define [CACA_GREEN](#) 0x02

The colour index for green.

12.1.2.4 #define [CACA_CYAN](#) 0x03

The colour index for cyan.

12.1.2.5 #define [CACA_RED](#) 0x04

The colour index for red.

12.1.2.6 #define [CACA_MAGENTA](#) 0x05

The colour index for magenta.

12.1.2.7 #define [CACA_BROWN](#) 0x06

The colour index for brown.

12.1.2.8 #define [CACA_LIGHTGRAY](#) 0x07

The colour index for light gray.

Referenced by [caca_attr_to_ansi\(\)](#), [caca_attr_to_argb64\(\)](#), and [caca_attr_to_rgb12_fg\(\)](#).

12.1.2.9 **#define CACA_DARKGRAY 0x08**

The colour index for dark gray.

12.1.2.10 **#define CACA_LIGHTBLUE 0x09**

The colour index for blue.

12.1.2.11 **#define CACA_LIGHTGREEN 0x0a**

The colour index for light green.

12.1.2.12 **#define CACA_LIGHTCYAN 0x0b**

The colour index for light cyan.

12.1.2.13 **#define CACA_LIGHTRED 0x0c**

The colour index for light red.

12.1.2.14 **#define CACA_LIGHTMAGENTA 0x0d**

The colour index for light magenta.

12.1.2.15 **#define CACA_YELLOW 0x0e**

The colour index for yellow.

12.1.2.16 **#define CACA_WHITE 0x0f**

The colour index for white.

12.1.2.17 **#define CACA_DEFAULT 0x10**

The output driver's default colour.

Referenced by `caca_attr_to_argb64()`, `caca_attr_to_rgb12_bg()`, `caca_attr_to_rgb12_fg()`, and `caca_create_canvas()`.

12.1.2.18 **#define CACA_TRANSPARENT 0x20**

The transparent colour.

Referenced by `caca_attr_to_argb64()`, `caca_attr_to_rgb12_bg()`, `caca_attr_to_rgb12_fg()`, and `caca_create_canvas()`.

12.1.2.19 **#define CACA_BOLD 0x01**

The style mask for bold.

12.1.2.20 #define CACA_ITALICS 0x02

The style mask for italics.

12.1.2.21 #define CACA_UNDERLINE 0x04

The style mask for underline.

12.1.2.22 #define CACA_BLINK 0x08

The style mask for blink.

pt3em

12.2 libcaca basic functions

Functions

- `__extern caca_canvas_t * caca_create_canvas (int, int)`
Initialise a libcaca canvas.
- `__extern int caca_manage_canvas (caca_canvas_t *, int(*) (void *), void *)`
Manage a canvas.
- `__extern int caca_unmanage_canvas (caca_canvas_t *, int(*) (void *), void *)`
Unmanage a canvas.
- `__extern int caca_set_canvas_size (caca_canvas_t *, int, int)`
Resize a canvas.
- `__extern int caca_get_canvas_width (caca_canvas_t const *)`
Get the canvas width.
- `__extern int caca_get_canvas_height (caca_canvas_t const *)`
Get the canvas height.
- `__extern uint8_t const * caca_get_canvas_chars (caca_canvas_t const *)`
Get the canvas character array.
- `__extern uint8_t const * caca_get_canvas_attrs (caca_canvas_t const *)`
Get the canvas attribute array.
- `__extern int caca_free_canvas (caca_canvas_t *)`
Uninitialise libcaca.
- `__extern int caca_rand (int, int)`
Generate a random integer within a range.
- `__extern char const * caca_get_version (void)`
Return the libcaca version.

12.2.1 Detailed Description

These functions provide the basic *libcaca* routines for library initialisation, system information retrieval and configuration.

12.2.2 Function Documentation

12.2.2.1 `__extern caca_canvas_t * caca_create_canvas ( int width, int height )`

Initialise internal *libcaca* structures and the backend that will be used for subsequent graphical operations. It must be the first *libcaca* function to be called in a function. `caca_free_canvas()` should be called at the end of the program to free all allocated resources.

Both the cursor and the canvas' handle are initialised at the top-left corner.

If an error occurs, NULL is returned and **errno** is set accordingly:

pt3em

- `EINVAL` Specified width or height is invalid.
- `ENOMEM` Not enough memory for the requested canvas size.

pt3emParameters

pt3emwidth	pt3emThe desired canvas width
pt3emheight	pt3emThe desired canvas height

pt3emReturns

A libcaca canvas handle upon success, NULL if an error occurred.

References `CACA_DEFAULT`, `caca_set_color_ansi()`, and `CACA_TRANSPARENT`.

Referenced by `caca_create_display_with_driver()`, and `caca_set_canvas_boundaries()`.

12.2.2.2 `__extern int caca_manage_canvas ( caca_canvas_t * cv, int(*) (void *) callback, void * p )`

Lock a canvas to prevent it from being resized. If non-NULL, the *callback* function pointer will be called upon each *caca_set_canvas_size* call and if the returned value is zero, the canvas resize request will be denied.

This function is only useful for display drivers such as the *libcaca* library.

If an error occurs, -1 is returned and **errno** is set accordingly:

- `EBUSY` The canvas is already being managed.

pt3emParameters

pt3emcv	pt3emA libcaca canvas.
pt3emcallback	pt3emAn optional callback function pointer.
pt3emp	pt3emThe argument to be passed to <i>callback</i> .

pt3em

pt3emReturns

0 in case of success, -1 if an error occurred.

Referenced by `caca_create_display_with_driver()`.

12.2.2.3 `__extern int caca_unmanage_canvas (caca_canvas_t * cv, int (*)(void *)
callback, void * p)`

Unlock a canvas previously locked by `caca_manage_canvas()`. For safety reasons, the callback and callback data arguments must be the same as for the `caca_manage_canvas()` call.

This function is only useful for display drivers such as the *libcaca* library.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** The canvas is not managed, or the callback arguments do not match.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
	pt3em	pt3em	pt3em
	pt3em	pt3em	pt3em
	pt3em	pt3em	pt3em

pt3emReturns

0 in case of success, -1 if an error occurred.

Referenced by `caca_create_display_with_driver()`, and `caca_free_display()`.

12.2.2.4 `__extern int caca_set_canvas_size ( caca_canvas_t * cv, int width, int height )`

Set the canvas' width and height, in character cells.

The contents of the canvas are preserved to the extent of the new canvas size. Newly allocated character cells at the right and/or at the bottom of the canvas are filled with spaces.

If as a result of the resize the cursor coordinates fall outside the new canvas boundaries, they are readjusted. For instance, if the current X cursor coordinate is 11 and the requested width is 10, the new X cursor coordinate will be 10.

It is an error to try to resize the canvas if an output driver has been attached to the canvas using `caca_create_display()`. You need to remove the output driver using `caca_free_display()` before you can change the canvas size again. However, the caca output driver can cause a canvas resize through user interaction. See the `caca_event()` documentation for more about this.

pt3em

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Specified width or height is invalid.
- **EBUSY** The canvas is in use by a display driver and cannot be resized.
- **ENOMEM** Not enough memory for the requested canvas size. If this happens, the canvas handle becomes invalid and should not be used.

pt3emParameters

pt3em	pt3em	pt3em	A libcaca canvas.
	pt3em	width	The desired canvas width.
	pt3em	height	The desired canvas height.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.2.2.5 `__extern int caca_get_canvas_width ( caca_canvas_t const * cv )`

Return the current canvas' width, in character cells.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	A libcaca canvas.
-------	-------	-------	-------------------

pt3emReturns

The canvas width.

Referenced by `caca_get_mouse_x()`.

12.2.2.6 `__extern int caca_get_canvas_height ( caca_canvas_t const * cv )`

Returns the current canvas' height, in character cells.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	A libcaca canvas.
-------	-------	-------	-------------------

pt3em

pt3emReturns

The canvas height.

Referenced by `caca_get_mouse_y()`.

12.2.2.7 `__extern uint8_t const* caca_get_canvas_chars ( caca_canvas_t const * cv )`

Return the current canvas' internal character array. The array elements consist in native endian 32-bit Unicode values as returned by `caca_get_char()`.

This function is only useful for display drivers such as the *libcaca* library.

This function never fails.

pt3emParameters

pt3em	pt3emcv	pt3emA libcaca canvas.
-------	---------	------------------------

pt3emReturns

The canvas character array.

12.2.2.8 `__extern uint8_t const* caca_get_canvas_attr ( caca_canvas_t const * cv )`

Returns the current canvas' internal attribute array. The array elements consist in native endian 32-bit attribute values as returned by `caca_get_attr()`.

This function is only useful for display drivers such as the *libcaca* library.

This function never fails.

pt3emParameters

pt3em	pt3emcv	pt3emA libcaca canvas.
-------	---------	------------------------

pt3emReturns

The canvas attribute array.

12.2.2.9 `__extern int caca_free_canvas ( caca_canvas_t * cv )`

Free all resources allocated by `caca_create_canvas()`. After this function has been called, no other *libcaca* functions may be used unless a new call to `caca_create_canvas()` is done.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EBUSY** The canvas is in use by a display driver and cannot be freed.

pt3emParameters

pt3empt3emcv	pt3emA libcacca canvas.
--------------	-------------------------

0 in case of success, -1 if an error occurred.

12.2.2.10 `__extern int caca_rand ( int min, int max )`

This function never fails.

pt3em $\min$	pt3emThe lower bound of the integer range.
pt3em $\max$	pt3emThe upper bound of the integer range.

A random integer comprised between `min` and `max - 1` (inclusive).

This function never fails.

The *libcaca* version information.

pt3em

pt3em

12.3 libcaca canvas drawing

Defines

- `#define CACA_MAGIC_FULLWIDTH 0x000ffffe`

Functions

- `__extern int caca_gotoxy (caca_canvas_t *, int, int)`
Set cursor position.
- `__extern int caca_get_cursor_x (caca_canvas_t const *)`
Get X cursor position.
- `__extern int caca_get_cursor_y (caca_canvas_t const *)`
Get Y cursor position.
- `__extern int caca_put_char (caca_canvas_t *, int, int, uint32_t)`
Print an ASCII or Unicode character.
- `__extern uint32_t caca_get_char (caca_canvas_t const *, int, int)`
Get the Unicode character at the given coordinates.
- `__extern int caca_put_str (caca_canvas_t *, int, int, char const *)`
Print a string.
- `__extern int caca_printf (caca_canvas_t *, int, int, char const *,...)`
Print a formatted string.
- `__extern int caca_clear_canvas (caca_canvas_t *)`
Clear the canvas.
- `__extern int caca_set_canvas_handle (caca_canvas_t *, int, int)`
Set cursor handle.
- `__extern int caca_get_canvas_handle_x (caca_canvas_t const *)`
Get X handle position.
- `__extern int caca_get_canvas_handle_y (caca_canvas_t const *)`
Get Y handle position.
- `__extern int caca_blit (caca_canvas_t *, int, int, caca_canvas_t const *, caca_canvas_t const *)`
Blit a canvas onto another one.
- `__extern int caca_set_canvas_boundaries (caca_canvas_t *, int, int, int, int)`
Set a canvas' new boundaries.

12.3.1 Detailed Description

These functions provide low-level character printing routines and higher level graphics functions.

pt3em

12.3.2 Define Documentation

12.3.2.1 #define CACA_MAGIC_FULLWIDTH 0x000ffffe

Used to indicate that the previous character was a fullwidth glyph.

Referenced by `caca_blit()`, `caca_flip()`, `caca_put_attr()`, `caca_put_char()`, and `caca_rotate_180()`.

12.3.3 Function Documentation

12.3.3.1 __extern int caca_gotoxy (caca_canvas_t * cv, int x, int y)

Put the cursor at the given coordinates. Functions making use of the cursor will use the new values. Setting the cursor position outside the canvas is legal but the cursor will not be shown.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em	pt3em	A handle to the libcaca canvas.
	pt3em	x		pt3em	X cursor coordinate.
	pt3em	y		pt3em	Y cursor coordinate.

pt3emReturns

This function always returns 0.

12.3.3.2 __extern int caca_get_cursor_x (caca_canvas_t const * cv)

Retrieve the X coordinate of the cursor's position.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em	pt3em	A handle to the libcaca canvas.
-------	-------	-------	-------	-------	---------------------------------

pt3emReturns

The cursor's X coordinate.

12.3.3.3 __extern int caca_get_cursor_y (caca_canvas_t const * cv)

Retrieve the Y coordinate of the cursor's position.

This function never fails.

pt3em	pt3emcv	pt3emA handle to the libcaca canvas.
-------	---------	--------------------------------------

The cursor's Y coordinate.

This function never fails.

<code>pt3emcv</code>	pt3emA handle to the libcaca canvas.
<code>pt3emx</code>	pt3emX coordinate.
<code>pt3emy</code>	pt3emY coordinate.
<code>pt3emch</code>	pt3emThe character to print.

This function always returns 0.

If the coordinates are outside the canvas boundaries, a space (0x20) is returned.

pt3em

A special exception is when CACA_MAGIC_FULLWIDTH is returned. This value is guaranteed not to be a valid Unicode character, and indicates that the character at the left of the requested one is a fullwidth character.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.

pt3emReturns

This function always returns 0.

12.3.3.6 `__extern int caca_put_str ( caca_canvas_t * cv, int x, int y, char const * s )`

Print an UTF-8 string at the given coordinates, using the default foreground and background values. The coordinates may be outside the canvas boundaries (eg. a negative Y coordinate) and the string will be cropped accordingly if it is too long.

See [caca_put_char\(\)](#) for more information on how fullwidth characters are handled when overwriting each other or at the canvas' boundaries.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.
	pt3ems	pt3emThe string to print.

pt3emReturns

This function always returns 0.

References [caca_put_char\(\)](#), [caca_utf32_is_fullwidth\(\)](#), and [caca_utf8_to_utf32\(\)](#).

Referenced by [caca_printf\(\)](#).

12.3.3.7 `__extern int caca_printf ( caca_canvas_t * cv, int x, int y, char const * format, ... )`

Format a string at the given coordinates, using the default foreground and background values. The coordinates may be outside the canvas boundaries (eg. a negative Y coordinate) and the string will be cropped accordingly if it is too long. The syntax of the format string is the same as for the C `printf()` function.

pt3em

This function never fails.

pt3emParameters

pt3em	pt3em	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.
	pt3emformat	pt3emThe format string to print.
	pt3em...	pt3emArguments to the format string.

pt3emReturns

This function always returns 0.

References `caca_put_str()`.

12.3.3.8 `__extern int caca_clear_canvas ( caca_canvas_t * cv )`

Clear the canvas using the current foreground and background colours.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3emThe canvas to clear.
-------	-------	---------------------------

pt3emReturns

This function always returns 0.

12.3.3.9 `__extern int caca_set_canvas_handle ( caca_canvas_t * cv, int x, int y )`

Set the canvas' handle. Blitting functions will use the handle value to put the canvas at the proper coordinates.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX handle coordinate.
	pt3emy	pt3emY handle coordinate.

pt3emReturns

This function always returns 0.

12.3.3.10 `__extern int caca_get_canvas_handle_x( caca_canvas_t const * cv )`

This function never fails.

pt3empt3emcv	pt3emA handle to the libcaca canvas.
--------------	--------------------------------------

The canvas' handle's X coordinate.

This function never fails.

pt3empt3emcv	pt3emA handle to the libcaca canvas.
--------------	--------------------------------------

The canvas' handle's Y coordinate.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** A mask was specified but the mask size and source canvas size do not match.

pt3em <i>dst</i>	pt3emThe destination canvas.
pt3em <i>x</i>	pt3emX coordinate.
pt3em <i>y</i>	pt3emY coordinate.
pt3em <i>src</i>	pt3emThe source canvas.
pt3em <i>mask</i>	pt3emThe mask canvas.

pt3em

pt3emReturns

0 in case of success, -1 if an error occurred.

References CACA_MAGIC_FULLWIDTH.

Referenced by caca_set_canvas_boundaries().

12.3.3.13 `__extern int caca_set_canvas_boundaries ( caca_canvas_t * cv, int x, int y, int w, int h )`

Set new boundaries for a canvas. This function can be used to crop a canvas, to expand it or for combinations of both actions. All frames are affected by this function.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Specified width or height is invalid.
- **EBUSY** The canvas is in use by a display driver and cannot be resized.
- **ENOMEM** Not enough memory for the requested canvas size. If this happens, the canvas handle becomes invalid and should not be used.

pt3emParameters

pt3emcv	pt3emThe canvas to crop.
pt3emx	pt3emX coordinate of the top-left corner.
pt3emy	pt3emY coordinate of the top-left corner.
pt3emw	pt3emThe width of the cropped area.
pt3emh	pt3emThe height of the cropped area.

pt3emReturns

0 in case of success, -1 if an error occurred.

References caca_blit(), caca_create_canvas(), caca_create_frame(), caca_get_frame_count(), and caca_set_frame().

pt3em

pt3em

12.4 libcaca canvas transformation

Functions

- `__extern int caca_invert (caca_canvas_t *)`
Invert a canvas' colours.
- `__extern int caca_flip (caca_canvas_t *)`
Flip a canvas horizontally.
- `__extern int caca_flop (caca_canvas_t *)`
Flip a canvas vertically.
- `__extern int caca_rotate_180 (caca_canvas_t *)`
Rotate a canvas.
- `__extern int caca_rotate_left (caca_canvas_t *)`
Rotate a canvas, 90 degrees counterclockwise.
- `__extern int caca_rotate_right (caca_canvas_t *)`
Rotate a canvas, 90 degrees counterclockwise.
- `__extern int caca_stretch_left (caca_canvas_t *)`
Rotate and stretch a canvas, 90 degrees counterclockwise.
- `__extern int caca_stretch_right (caca_canvas_t *)`
Rotate and stretch a canvas, 90 degrees clockwise.

12.4.1 Detailed Description

These functions perform horizontal and vertical canvas flipping.

12.4.2 Function Documentation

12.4.2.1 `__extern int caca_invert ( caca_canvas_t * cv )`

Invert a canvas' colours (black becomes white, red becomes cyan, etc.) without changing the characters in it.

This function never fails.

Parameters

<code>cv</code>	The canvas to invert.
-----------------	-----------------------

Returns

This function always returns 0.

12.4.2.2 `__extern int caca_flip ( caca_canvas_t * cv )`

Flip a canvas horizontally, choosing characters that look like the mirrored version wherever possible. Some characters will stay unchanged by the process, but the operation is guaranteed to be involutive: performing it again gives back the original canvas.

This function never fails.

Parameters

<code>cv</code>	The canvas to flip.
-----------------	---------------------

Returns

This function always returns 0.

References CACA_MAGIC_FULLWIDTH.

12.4.2.3 `__extern int caca_flop ( caca_canvas_t * cv )`

Flip a canvas vertically, choosing characters that look like the mirrored version wherever possible. Some characters will stay unchanged by the process, but the operation is guaranteed to be involutive: performing it again gives back the original canvas.

This function never fails.

Parameters

<code>cv</code>	The canvas to flop.
-----------------	---------------------

Returns

This function always returns 0.

12.4.2.4 `__extern int caca_rotate_180 ( caca_canvas_t * cv )`

Apply a 180-degree transformation to a canvas, choosing characters that look like the upside-down version wherever possible. Some characters will stay unchanged by the process, but the operation is guaranteed to be involutive: performing it again gives back the original canvas.

This function never fails.

Parameters

<code>cv</code>	The canvas to rotate.
-----------------	-----------------------

pt3emReturns

This function always returns 0.

References CACA MAGIC FULLWIDTH.

Apply a 90-degree transformation to a canvas, choosing characters that look like the rotated version wherever possible. Characters cells are rotated two-by-two. Some characters will stay unchanged by the process, some others will be replaced by close equivalents. Fullwidth characters at odd horizontal coordinates will be lost. The operation is not guaranteed to be reversible at all.

Note that the width of the canvas is divided by two and becomes the new height. Height is multiplied by two and becomes the new width. If the original width is an odd number, the division is rounded up.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EBUSY** The canvas is in use by a display driver and cannot be rotated.
- **ENOMEM** Not enough memory to allocate the new canvas size. If this happens, the previous canvas handle is still valid.

pt3empt3emcv	pt3emThe canvas to rotate left.
--------------	---------------------------------

0 in case of success, -1 if an error occurred.

Apply a 90-degree transformation to a canvas, choosing characters that look like the rotated version wherever possible. Characters cells are rotated two-by-two. Some characters will stay unchanged by the process, some others will be replaced by close equivalents. Fullwidth characters at odd horizontal coordinates will be lost. The operation is not guaranteed to be reversible at all.

Note that the width of the canvas is divided by two and becomes the new height. Height is multiplied by two and becomes the new width. If the original width is an odd number, the division is rounded up.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EBUSY** The canvas is in use by a display driver and cannot be rotated.

pt3em

- **ENOMEM** Not enough memory to allocate the new canvas size. If this happens, the previous canvas handle is still valid.

pt3emParameters

pt3em	pt3emcv	pt3emThe canvas to rotate right.
-------	---------	----------------------------------

pt3emReturns

0 in case of success, -1 if an error occurred.

12.4.2.7 `__extern int caca_stretch_left ( caca_canvas_t * cv )`

Apply a 90-degree transformation to a canvas, choosing characters that look like the rotated version wherever possible. Some characters will stay unchanged by the process, some others will be replaced by close equivalents. Fullwidth characters will be lost. The operation is not guaranteed to be reversible at all.

Note that the width and height of the canvas are swapped, causing its aspect ratio to look stretched.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EBUSY** The canvas is in use by a display driver and cannot be rotated.
- **ENOMEM** Not enough memory to allocate the new canvas size. If this happens, the previous canvas handle is still valid.

pt3emParameters

pt3em	pt3emcv	pt3emThe canvas to rotate left.
-------	---------	---------------------------------

pt3emReturns

0 in case of success, -1 if an error occurred.

12.4.2.8 `__extern int caca_stretch_right ( caca_canvas_t * cv )`

Apply a 270-degree transformation to a canvas, choosing characters that look like the rotated version wherever possible. Some characters will stay unchanged by the process, some others will be replaced by close equivalents. Fullwidth characters will be lost. The operation is not guaranteed to be reversible at all.

Note that the width and height of the canvas are swapped, causing its aspect ratio to look stretched.

If an error occurs, -1 is returned and **errno** is set accordingly:

pt3em

- **EBUSY** The canvas is in use by a display driver and cannot be rotated.
- **ENOMEM** Not enough memory to allocate the new canvas size. If this happens, the previous canvas handle is still valid.

pt3emParameters

pt3em	pt3emcv	pt3emThe canvas to rotate right.
-------	---------	----------------------------------

pt3emReturns

0 in case of success, -1 if an error occurred.

pt3em

pt3em

12.5 libcaca attribute conversions

Functions

- `__extern uint32_t caca_get_attr (caca_canvas_t const *, int, int)`
Get the text attribute at the given coordinates.
- `__extern int caca_set_attr (caca_canvas_t *, uint32_t)`
Set the default character attribute.
- `__extern int caca_put_attr (caca_canvas_t *, int, int, uint32_t)`
Set the character attribute at the given coordinates.
- `__extern int caca_set_color_ansi (caca_canvas_t *, uint8_t, uint8_t)`
Set the default colour pair for text (ANSI version).
- `__extern int caca_set_color_argb (caca_canvas_t *, uint16_t, uint16_t)`
Set the default colour pair for text (truecolor version).
- `__extern uint8_t caca_attr_to_ansi (uint32_t)`
Get DOS ANSI information from attribute.
- `__extern uint8_t caca_attr_to_ansi_fg (uint32_t)`
Get ANSI foreground information from attribute.
- `__extern uint8_t caca_attr_to_ansi_bg (uint32_t)`
Get ANSI background information from attribute.
- `__extern uint16_t caca_attr_to_rgb12_fg (uint32_t)`
Get 12-bit RGB foreground information from attribute.
- `__extern uint16_t caca_attr_to_rgb12_bg (uint32_t)`
Get 12-bit RGB background information from attribute.
- `__extern void caca_attr_to_argb64 (uint32_t, uint8_t[8])`
Get 64-bit ARGB information from attribute.

12.5.1 Detailed Description

These functions perform conversions between attribute values.

12.5.2 Function Documentation

12.5.2.1 `__extern uint32_t caca_get_attr ( caca_canvas_t const * cv, int x, int y )`

Get the internal *libcaca* attribute value of the character at the given coordinates. The attribute value has 32 significant bits, organised as follows from MSB to LSB:

- 3 bits for the background alpha
- 4 bits for the background red component
- 4 bits for the background green component

pt3em

- 3 bits for the background blue component
- 3 bits for the foreground alpha
- 4 bits for the foreground red component
- 4 bits for the foreground green component
- 3 bits for the foreground blue component
- 4 bits for the bold, italics, underline and blink flags

If the coordinates are outside the canvas boundaries, the current attribute is returned.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3emReturns

The requested attribute.

Referenced by `caca_dither_bitmap()`.

12.5.2.2 `__extern int caca_set_attr ( caca_canvas_t * cv, uint32_t attr )`

Set the default character attribute for drawing. Attributes define foreground and background colour, transparency, bold, italics and underline styles, as well as blink. String functions such as `caca_printf()` and graphical primitive functions such as `caca_draw_line()` will use this attribute.

The value of *attr* is either:

- a 32-bit integer as returned by `caca_get_attr()`, in which case it also contains colour information,
- a combination (bitwise OR) of style values (`CACA_UNDERLINE`, `CACA_BLINK`, `CACA_BOLD` and `CACA_ITALICS`), in which case setting the attribute does not modify the current colour information.

To retrieve the current attribute value, use `caca_get_attr(-1,-1)`.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

Returns

This function always returns 0.

Referenced by `caca_dither_bitmap()`.

12.5.2.3 `__extern int caca_put_attr ( caca_canvas_t * cv, int x, int y, uint32_t attr )`

Set the character attribute, without changing the character's value. If the character at the given coordinates is a fullwidth character, both cells' attributes are replaced.

The value of *attr* is either:

- a 32-bit integer as returned by `caca_get_attr()`, in which case it also contains colour information,
- a combination (bitwise OR) of style values (`CACA_UNDERLINE`, `CACA_BLINK`, `CACA_BOLD` and `CACA_ITALICS`), in which case setting the attribute does not modify the current colour information.

This function never fails.

Parameters

<code>cv</code>	A handle to the libcaca canvas.
<code>x</code>	X coordinate.
<code>y</code>	Y coordinate.
<code>attr</code>	The requested attribute value.

Returns

This function always returns 0.

References `CACA_MAGIC_FULLWIDTH`.

12.5.2.4 `__extern int caca_set_color_ansi ( caca_canvas_t * cv, uint8_t fg, uint8_t bg )`

Set the default ANSI colour pair for text drawing. String functions such as `caca_printf()` and graphical primitive functions such as `caca_draw_line()` will use these attributes.

Color values are those defined in `caca.h`, such as `CACA_RED` or `CACA_TRANSPARENT`.

If an error occurs, 0 is returned and `errno` is set accordingly:

- `EINVAL` At least one of the colour values is invalid.

pt3em

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3emReturns

0 in case of success, -1 if an error occurred.

Referenced by `caca_create_canvas()`, and `caca_dither_bitmap()`.

12.5.2.5 `__extern int caca_set_color_argb ( caca_canvas_t * cv, uint16_t fg, uint16_t bg )`

Set the default ARGB colour pair for text drawing. String functions such as `caca_printf()` and graphical primitive functions such as `caca_draw_line()` will use these attributes.

Colors are 16-bit ARGB values, each component being coded on 4 bits. For instance, 0xf088 is solid dark cyan (A=15 R=0 G=8 B=8), and 0x8fff is white with 50% alpha (A=8 R=15 G=15 B=15).

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

12.5.2.6 `__extern uint8_t caca_attr_to_ansi ( uint32_t attr )`

Get the ANSI colour pair for a given attribute. The returned value is an 8-bit value whose higher 4 bits are the background colour and lower 4 bits are the foreground colour.

If the attribute has ARGB colours, the nearest colour is used. Special attributes such as `CACA_DEFAULT` and `CACA_TRANSPARENT` are not handled and are both replaced with `CACA_LIGHTGRAY` for the foreground colour and `CACA_BLACK` for the background colour.

This function never fails. If the attribute value is outside the expected 32-bit range, higher order bits are simply ignored.

Parameters

Parameters

<code>attr</code>	The requested attribute value.
-------------------	--------------------------------

Returns

The corresponding DOS ANSI value.

References `CACA_BLACK`, and `CACA_LIGHTGRAY`.

12.5.2.7 `__extern uint8_t caca_attr_to_ansi_fg ( uint32_t attr )`

Get the ANSI foreground colour value for a given attribute. The returned value is either one of the `CACA_RED`, `CACA_BLACK` etc. predefined colours, or the special value `CACA_DEFAULT` meaning the media's default foreground value, or the special value `CACA_TRANSPARENT`.

If the attribute has ARGB colours, the nearest colour is returned.

This function never fails. If the attribute value is outside the expected 32-bit range, higher order bits are simply ignored.

Parameters

<code>attr</code>	The requested attribute value.
-------------------	--------------------------------

Returns

The corresponding ANSI foreground value.

12.5.2.8 `__extern uint8_t caca_attr_to_ansi_bg ( uint32_t attr )`

Get the ANSI background colour value for a given attribute. The returned value is either one of the `CACA_RED`, `CACA_BLACK` etc. predefined colours, or the special value `CACA_DEFAULT` meaning the media's default background value, or the special value `CACA_TRANSPARENT`.

If the attribute has ARGB colours, the nearest colour is returned.

This function never fails. If the attribute value is outside the expected 32-bit range, higher order bits are simply ignored.

Parameters

<code>attr</code>	The requested attribute value.
-------------------	--------------------------------

pt3em

pt3emReturns

The corresponding ANSI background value.

12.5.2.9 `__extern uint16_t caca_attr_to_rgb12_fg ( uint32_t attr )`

Get the 12-bit foreground colour value for a given attribute. The returned value is a native-endian encoded integer with each red, green and blue values encoded on 8 bits in the following order:

- 8-11 most significant bits: red
- 4-7 most significant bits: green
- least significant bits: blue

This function never fails. If the attribute value is outside the expected 32-bit range, higher order bits are simply ignored.

pt3emParameters

pt3em	pt3emattr	pt3emThe requested attribute value.
-------	-----------	-------------------------------------

pt3emReturns

The corresponding 12-bit RGB foreground value.

References CACA_DEFAULT, CACA_LIGHTGRAY, and CACA_TRANSPARENT.

12.5.2.10 `__extern uint16_t caca_attr_to_rgb12_bg ( uint32_t attr )`

Get the 12-bit background colour value for a given attribute. The returned value is a native-endian encoded integer with each red, green and blue values encoded on 8 bits in the following order:

- 8-11 most significant bits: red
- 4-7 most significant bits: green
- least significant bits: blue

This function never fails. If the attribute value is outside the expected 32-bit range, higher order bits are simply ignored.

pt3emParameters

pt3em	pt3emattr	pt3emThe requested attribute value.
-------	-----------	-------------------------------------

pt3em

pt3emReturns

The corresponding 12-bit RGB background value.

References CACA_BLACK, CACA_DEFAULT, and CACA_TRANSPARENT.

12.5.2.11 `_extern void caca_attr_to_argb64 ( uint32_t attr, uint8_t argb[8] )`

Get the 64-bit colour and alpha values for a given attribute. The values are written as 8-bit integers in the *argb* array in the following order:

- *argb*[0]: background alpha value
- *argb*[1]: background red value
- *argb*[2]: background green value
- *argb*[3]: background blue value
- *argb*[4]: foreground alpha value
- *argb*[5]: foreground red value
- *argb*[6]: foreground green value
- *argb*[7]: foreground blue value

This function never fails. If the attribute value is outside the expected 32-bit range, higher order bits are simply ignored.

pt3emParameters

pt3emattr	pt3emThe requested attribute value.
pt3emargb	pt3emAn array of 8-bit integers.

References CACA_BLACK, CACA_DEFAULT, CACA_LIGHTGRAY, and CACA_TRANSPARENT.

Referenced by `caca_render_canvas()`.

pt3em

pt3em

12.6 libcaca character set conversions

Functions

- `__extern uint32_t caca_utf8_to_utf32 (char const *, size_t *)`
Convert a UTF-8 character to UTF-32.
- `__extern size_t caca_utf32_to_utf8 (char *, uint32_t)`
Convert a UTF-32 character to UTF-8.
- `__extern uint8_t caca_utf32_to_cp437 (uint32_t)`
Convert a UTF-32 character to CP437.
- `__extern uint32_t caca_cp437_to_utf32 (uint8_t)`
Convert a CP437 character to UTF-32.
- `__extern char caca_utf32_to_ascii (uint32_t)`
Convert a UTF-32 character to ASCII.
- `__extern int caca_utf32_is_fullwidth (uint32_t)`
Tell whether a UTF-32 character is fullwidth.

12.6.1 Detailed Description

These functions perform conversions between usual character sets.

12.6.2 Function Documentation

12.6.2.1 `__extern uint32_t caca_utf8_to_utf32 ( char const * s, size_t * bytes )`

Convert a UTF-8 character read from a string and return its value in the UTF-32 character set. If the second argument is not null, the total number of read bytes is written in it.

If a null byte was reached before the expected end of the UTF-8 sequence, this function returns zero and the number of read bytes is set to zero.

This function never fails, but its behaviour with illegal UTF-8 sequences is undefined.

Parameters

<code>s</code>	A string containing the UTF-8 character.
<code>bytes</code>	A pointer to a <code>size_t</code> to store the number of bytes in the character, or NULL.

pt3em

pt3emReturns

The corresponding UTF-32 character, or zero if the character is incomplete.

Referenced by `caca_put_str()`.

12.6.2.2 `__extern size_t caca_utf32_to_utf8 ( char * buf, uint32_t ch )`

Convert a UTF-32 character read from a string and write its value in the UTF-8 character set into the given buffer.

This function never fails, but its behaviour with illegal UTF-32 characters is undefined.

pt3emParameters

pt3emA pointer to a character buffer where the UTF-8 sequence will be written.	pt3em
pt3emThe UTF-32 character.	pt3em

pt3emReturns

The number of bytes written.

12.6.2.3 `__extern uint8_t caca_utf32_to_cp437 ( uint32_t ch )`

Convert a UTF-32 character read from a string and return its value in the CP437 character set, or "?" if the character has no equivalent.

This function never fails.

pt3emParameters

pt3emThe UTF-32 character.	pt3em
----------------------------	-------

pt3emReturns

The corresponding CP437 character, or "?" if not representable.

12.6.2.4 `__extern uint32_t caca_cp437_to_utf32 ( uint8_t ch )`

Convert a CP437 character read from a string and return its value in the UTF-32 character set, or zero if the character is a CP437 control character.

This function never fails.

Parameters

Parameters

<code>ch</code>	The CP437 character.
-----------------	----------------------

Returns

The corresponding UTF-32 character, or zero if not representable.

12.6.2.5 `__extern char caca_utf32_to_ascii ( uint32_t ch )`

Convert a UTF-32 character into an ASCII character. When no equivalent exists, a graphically close equivalent is sought.

This function never fails, but its behaviour with illegal UTF-32 characters is undefined.

Parameters

<code>ch</code>	The UTF-32 character.
-----------------	-----------------------

Returns

The corresponding ASCII character, or a graphically close equivalent if found, or "?" if not representable.

12.6.2.6 `__extern int caca_utf32_is_fullwidth ( uint32_t ch )`

Check whether the given UTF-32 character should be printed at twice the normal width (fullwidth characters). If the character is unknown or if its status cannot be decided, it is treated as a standard-width character.

This function never fails.

Parameters

<code>ch</code>	The UTF-32 character.
-----------------	-----------------------

Returns

1 if the character is fullwidth, 0 otherwise.

Referenced by `caca_put_char()`, and `caca_put_str()`.

pt3em

pt3em

12.7 libcaca primitives drawing

Functions

- `__extern int caca_draw_line (caca_canvas_t *, int, int, int, int, uint32_t)`
Draw a line on the canvas using the given character.
- `__extern int caca_draw_polyline (caca_canvas_t *, int const x[], int const y[], int, uint32_t)`
Draw a polyline.
- `__extern int caca_draw_thin_line (caca_canvas_t *, int, int, int, int)`
Draw a thin line on the canvas, using ASCII art.
- `__extern int caca_draw_thin_polyline (caca_canvas_t *, int const x[], int const y[], int)`
Draw an ASCII art thin polyline.
- `__extern int caca_draw_circle (caca_canvas_t *, int, int, int, uint32_t)`
Draw a circle on the canvas using the given character.
- `__extern int caca_draw_ellipse (caca_canvas_t *, int, int, int, int, uint32_t)`
Draw an ellipse on the canvas using the given character.
- `__extern int caca_draw_thin_ellipse (caca_canvas_t *, int, int, int, int)`
Draw a thin ellipse on the canvas.
- `__extern int caca_fill_ellipse (caca_canvas_t *, int, int, int, int, uint32_t)`
Fill an ellipse on the canvas using the given character.
- `__extern int caca_draw_box (caca_canvas_t *, int, int, int, int, uint32_t)`
Draw a box on the canvas using the given character.
- `__extern int caca_draw_thin_box (caca_canvas_t *, int, int, int, int)`
Draw a thin box on the canvas.
- `__extern int caca_draw_cp437_box (caca_canvas_t *, int, int, int, int)`
Draw a box on the canvas using CP437 characters.
- `__extern int caca_fill_box (caca_canvas_t *, int, int, int, int, uint32_t)`
Fill a box on the canvas using the given character.
- `__extern int caca_draw_triangle (caca_canvas_t *, int, int, int, int, int, int, uint32_t)`
Draw a triangle on the canvas using the given character.
- `__extern int caca_draw_thin_triangle (caca_canvas_t *, int, int, int, int, int, int)`
Draw a thin triangle on the canvas.
- `__extern int caca_fill_triangle (caca_canvas_t *, int, int, int, int, int, int, uint32_t)`
Fill a triangle on the canvas using the given character.

12.7.1 Detailed Description

These functions provide routines for primitive drawing, such as lines, boxes, triangles and ellipses.

pt3em

12.7.2 Function Documentation

12.7.2.1 `__extern int caca_draw_line ( caca_canvas_t * cv, int x1, int y1, int x2, int y2, uint32_t ch )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

Referenced by `caca_draw_box()`, `caca_draw_triangle()`, and `caca_fill_ellipse()`.

12.7.2.2 `__extern int caca_draw_polyline ( caca_canvas_t * cv, int const x[], int const y[], int n, uint32_t ch )`

Draw a polyline on the canvas using the given character and coordinate arrays. The first and last points are not connected, hence in order to draw a polygon you need to specify the starting point at the end of the list as well.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3em

pt3emReturns

This function always returns 0.

12.7.2.3 `__extern int caca_draw_thin_line ( caca_canvas_t * cv, int x1, int y1, int x2, int y2 )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

Referenced by `caca_draw_thin_triangle()`.

12.7.2.4 `__extern int caca_draw_thin_polyline ( caca_canvas_t * cv, int const x[], int const y[], int n )`

Draw a thin polyline on the canvas using the given coordinate arrays and with ASCII art. The first and last points are not connected, so in order to draw a polygon you need to specify the starting point at the end of the list as well.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

12.7.2.5 `__extern int caca_draw_circle ( caca_canvas_t * cv, int x, int y, int r, uint32_t ch )`

This function never fails.

pt3em

pt3emParameters

pt3em	pt3em
pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emx	pt3emCenter X coordinate.
pt3emy	pt3emCenter Y coordinate.
pt3emr	pt3emCircle radius.
pt3emch	pt3emUTF-32 character to be used to draw the circle outline.

pt3emReturns

This function always returns 0.

12.7.2.6 `__extern int caca_draw_ellipse ( caca_canvas_t * cv, int xo, int yo, int a, int b, uint32_t ch )`

This function never fails.

pt3emParameters

pt3em	pt3em
pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emxo	pt3emCenter X coordinate.
pt3emyo	pt3emCenter Y coordinate.
pt3ema	pt3emEllipse X radius.
pt3emb	pt3emEllipse Y radius.
pt3emch	pt3emUTF-32 character to be used to draw the ellipse outline.

pt3emReturns

This function always returns 0.

12.7.2.7 `__extern int caca_draw_thin_ellipse ( caca_canvas_t * cv, int xo, int yo, int a, int b )`

This function never fails.

pt3emParameters

pt3em	pt3em
pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emxo	pt3emCenter X coordinate.
pt3emyo	pt3emCenter Y coordinate.
pt3ema	pt3emEllipse X radius.
pt3emb	pt3emEllipse Y radius.

pt3em

pt3emReturns

This function always returns 0.

12.7.2.8 `__extern int caca_fill_ellipse ( caca_canvas_t * cv, int xo, int yo, int a, int b, uint32_t ch )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The handle to the libcaca canvas.
	pt3em	xo	Center X coordinate.
	pt3em	yo	Center Y coordinate.
	pt3em	a	Ellipse X radius.
	pt3em	b	Ellipse Y radius.
	pt3em	ch	UTF-32 character to be used to fill the ellipse.

pt3emReturns

This function always returns 0.

References `caca_draw_line()`.

12.7.2.9 `__extern int caca_draw_box ( caca_canvas_t * cv, int x, int y, int w, int h, uint32_t ch )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The handle to the libcaca canvas.
	pt3em	x	X coordinate of the upper-left corner of the box.
	pt3em	y	Y coordinate of the upper-left corner of the box.
	pt3em	w	Width of the box.
	pt3em	h	Height of the box.
	pt3em	ch	UTF-32 character to be used to draw the box.

pt3em

pt3emReturns

This function always returns 0.

References caca_draw_line().

12.7.2.10 `__extern int caca_draw_thin_box ( caca_canvas_t * cv, int x, int y, int w, int h )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

References caca_put_char().

12.7.2.11 `__extern int caca_draw_cp437_box ( caca_canvas_t * cv, int x, int y, int w, int h )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em
pt3em	pt3em	pt3em	pt3em

pt3em

pt3emReturns

This function always returns 0.

References `caca_put_char()`.

12.7.2.12 `__extern int caca_fill_box ( caca_canvas_t * cv, int x, int y, int w, int h, uint32_t ch )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The handle to the libcaca canvas.
	pt3emx	pt3em	X coordinate of the upper-left corner of the box.
	pt3emy	pt3em	Y coordinate of the upper-left corner of the box.
	pt3emw	pt3em	Width of the box.
	pt3emh	pt3em	Height of the box.
	pt3emch	pt3em	UTF-32 character to be used to draw the box.

pt3emReturns

This function always returns 0.

References `caca_put_char()`.

12.7.2.13 `__extern int caca_draw_triangle ( caca_canvas_t * cv, int x1, int y1, int x2, int y2, int x3, int y3, uint32_t ch )`

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The handle to the libcaca canvas.
	pt3emx1	pt3em	X coordinate of the first point.
	pt3emy1	pt3em	Y coordinate of the first point.
	pt3emx2	pt3em	X coordinate of the second point.
	pt3emy2	pt3em	Y coordinate of the second point.
	pt3emx3	pt3em	X coordinate of the third point.
	pt3emy3	pt3em	Y coordinate of the third point.
	pt3emch	pt3em	UTF-32 character to be used to draw the triangle outline.

pt3emReturns

This function always returns 0.

References `caca_draw_line()`.

```
12.7.2.14 __extern int caca_draw_thin_triangle ( caca_canvas_t * cv, int x1, int y1, int
x2, int y2, int x3, int y3 )
```

pt3emParameters

pt3em	pt3em	The handle to the libcacanvas.
	pt3emx1	X coordinate of the first point.
	pt3emy1	Y coordinate of the first point.
	pt3emx2	X coordinate of the second point.
	pt3emy2	Y coordinate of the second point.
	pt3emx3	X coordinate of the third point.
	pt3emy3	Y coordinate of the third point.

References `caca_draw_thin_line()`.

```
12.7.2.15 __extern int caca_fill_triangle ( caca_canvas_t * cv, int x1, int y1, int x2, int y2,
int x3, int y3, uint32_t ch )
```

pt3emParameters

pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emx1	pt3emX coordinate of the first point.
pt3emy1	pt3emY coordinate of the first point.
pt3emx2	pt3emX coordinate of the second point.
pt3emy2	pt3emY coordinate of the second point.
pt3emx3	pt3emX coordinate of the third point.
pt3emy3	pt3emY coordinate of the third point.
pt3emch	pt3emUTF-32 character to be used to fill the triangle.

Referenced by `caca_fill_triangle()`.

pt3em

pt3em

12.8 libcaca canvas frame handling

Functions

- `__extern int caca\_get\_frame\_count (caca\_canvas\_t const *)`
Get the number of frames in a canvas.
- `__extern int caca\_set\_frame (caca\_canvas\_t *, int)`
Activate a given canvas frame.
- `__extern char const * caca\_get\_frame\_name (caca\_canvas\_t const *)`
Get the current frame's name.
- `__extern int caca\_set\_frame\_name (caca\_canvas\_t *, char const *)`
Set the current frame's name.
- `__extern int caca\_create\_frame (caca\_canvas\_t *, int)`
Add a frame to a canvas.
- `__extern int caca\_free\_frame (caca\_canvas\_t *, int)`
Remove a frame from a canvas.

12.8.1 Detailed Description

These functions provide high level routines for canvas frame insertion, removal, copying etc.

12.8.2 Function Documentation

12.8.2.1 `__extern int caca\_get\_frame\_count ( caca\_canvas\_t const * cv )`

Return the current canvas' frame count.

This function never fails.

Parameters

<code>caca_canvas_t const * <i>cv</i></code>	A libcaca canvas
--	------------------

Returns

The frame count

Referenced by `caca\_set\_canvas\_boundaries()`.

12.8.2.2 `__extern int caca\_set\_frame ( caca\_canvas\_t * cv, int id )`

Set the active canvas frame. All subsequent drawing operations will be performed on that frame. The current painting context set by `caca\_set\_attr()` is inherited.

pt3em

If the frame index is outside the canvas' frame range, nothing happens.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Requested frame is out of range.

Parameters

pt3em	pt3em	pt3emA libcaca canvas
	pt3em	pt3emThe canvas frame to activate

Returns

0 in case of success, -1 if an error occurred.

Referenced by `caca_set_canvas_boundaries()`.

12.8.2.3 `__extern char const* caca_get_frame_name ( caca_canvas_t const * cv )`

Return the current frame's name. The returned string is valid until the frame is deleted or `caca_set_frame_name()` is called to change the frame name again.

This function never fails.

Parameters

pt3em	pt3em	pt3emA libcaca canvas.
-------	-------	------------------------

Returns

The current frame's name.

12.8.2.4 `__extern int caca_set_frame_name ( caca_canvas_t * cv, char const * name )`

Set the current frame's name. Upon creation, a frame has a default name of "frame#xxxxxxx" where xxxxxxxx is a self-incrementing hexadecimal number.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **ENOMEM** Not enough memory to allocate new frame.

Parameters

pt3em	pt3em	pt3emA libcaca canvas.
	pt3em	pt3emThe name to give to the current frame.

Returns

0 in case of success, -1 if an error occurred.

12.8.2.5 `__extern int caca_create_frame ( caca_canvas_t * cv, int id )`

Create a new frame within the given canvas. Its contents and attributes are copied from the currently active frame.

The frame index indicates where the frame should be inserted. Valid values range from 0 to the current canvas frame count. If the frame index is greater than or equals the current canvas frame count, the new frame is appended at the end of the canvas. If the frame index is less than zero, the new frame is inserted at index 0.

The active frame does not change, but its index may be renumbered due to the insertion.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **ENOMEM** Not enough memory to allocate new frame.

Parameters

<code>cv</code>	A libcaca canvas
<code>id</code>	The index where to insert the new frame

Returns

0 in case of success, -1 if an error occurred.

Referenced by `caca_set_canvas_boundaries()`.

12.8.2.6 `__extern int caca_free_frame ( caca_canvas_t * cv, int id )`

Delete a frame from a given canvas.

The frame index indicates the frame to delete. Valid values range from 0 to the current canvas frame count minus 1. If the frame index is greater than or equals the current canvas frame count, the last frame is deleted.

If the active frame is deleted, frame 0 becomes the new active frame. Otherwise, the active frame does not change, but its index may be renumbered due to the deletion.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Requested frame is out of range, or attempt to delete the last frame of the canvas.

pt3em

pt3emParameters

pt3em	pt3emcv	pt3emA libcaca canvas
	pt3emid	pt3emThe index of the frame to delete

pt3emReturns

0 in case of success, -1 if an error occurred.

pt3em

pt3em

12.9 libcaca bitmap dithering

Functions

- `__extern caca_dither_t * caca_create_dither (int, int, int, int, uint32_t, uint32_t, uint32_t, uint32_t)`
Create an internal dither object.
- `__extern int caca_set_dither_palette (caca_dither_t *, uint32_t r[], uint32_t g[], uint32_t b[], uint32_t a[])`
Set the palette of an 8bpp dither object.
- `__extern int caca_set_dither_brightness (caca_dither_t *, float)`
Set the brightness of a dither object.
- `__extern float caca_get_dither_brightness (caca_dither_t const *)`
Get the brightness of a dither object.
- `__extern int caca_set_dither_gamma (caca_dither_t *, float)`
Set the gamma of a dither object.
- `__extern float caca_get_dither_gamma (caca_dither_t const *)`
Get the gamma of a dither object.
- `__extern int caca_set_dither_contrast (caca_dither_t *, float)`
Set the contrast of a dither object.
- `__extern float caca_get_dither_contrast (caca_dither_t const *)`
Get the contrast of a dither object.
- `__extern int caca_set_dither_antialias (caca_dither_t *, char const *)`
Set dither antialiasing.
- `__extern char const *const * caca_get_dither_antialias_list (caca_dither_t const *)`
Get available antialiasing methods.
- `__extern char const * caca_get_dither_antialias (caca_dither_t const *)`
Get current antialiasing method.
- `__extern int caca_set_dither_color (caca_dither_t *, char const *)`
Choose colours used for dithering.
- `__extern char const *const * caca_get_dither_color_list (caca_dither_t const *)`
Get available colour modes.
- `__extern char const * caca_get_dither_color (caca_dither_t const *)`
Get current colour mode.
- `__extern int caca_set_dither_charset (caca_dither_t *, char const *)`
Choose characters used for dithering.
- `__extern char const *const * caca_get_dither_charset_list (caca_dither_t const *)`
Get available dither character sets.
- `__extern char const * caca_get_dither_charset (caca_dither_t const *)`

pt3em

Get current character set.

- `__extern int caca_set_dither_algorithm (caca_dither_t *, char const *)`

Set dithering algorithm.

- `__extern char const *const * caca_get_dither_algorithm_list (caca_dither_t const *)`

Get dithering algorithms.

- `__extern char const * caca_get_dither_algorithm (caca_dither_t const *)`

Get current dithering algorithm.

- `__extern int caca_dither_bitmap (caca_canvas_t *, int, int, int, int, caca_dither_t const *, void *)`

Dither a bitmap on the canvas.

- `__extern int caca_free_dither (caca_dither_t *)`

Free the memory associated with a dither.

12.9.1 Detailed Description

These functions provide high level routines for dither allocation and rendering.

12.9.2 Function Documentation

12.9.2.1 `__extern caca_dither_t* caca_create_dither ( int bpp, int w, int h, int pitch, uint32_t rmask, uint32_t gmask, uint32_t bmask, uint32_t amask )`

Create a dither structure from its coordinates (depth, width, height and pitch) and pixel mask values. If the depth is 8 bits per pixel, the mask values are ignored and the colour palette should be set using the `caca_set_dither_palette()` function. For depths greater than 8 bits per pixel, a zero alpha mask causes the alpha values to be ignored.

If an error occurs, NULL is returned and **errno** is set accordingly:

- **EINVAL** Requested width, height, pitch or bits per pixel value was invalid.
- **ENOMEM** Not enough memory to allocate dither structure.

pt3emParameters

pt3embpp	pt3emBitmap depth in bits per pixel.
pt3emw	pt3emBitmap width in pixels.
pt3emh	pt3emBitmap height in pixels.
pt3empitch	pt3emBitmap pitch in bytes.
pt3emrmask	pt3emBitmask for red values.
pt3emgmask	pt3emBitmask for green values.
pt3embmask	pt3emBitmask for blue values.
pt3emamask	pt3emBitmask for alpha values.

pt3em

pt3emReturns

Dither object upon success, NULL if an error occurred.

12.9.2.2 `__extern int caca_set_dither_palette ( caca_dither_t * d, uint32_t red[], uint32_t green[], uint32_t blue[], uint32_t alpha[] )`

Set the palette of an 8 bits per pixel bitmap. Values should be between 0 and 4095 (0xffff).

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Dither bits per pixel value is not 8, or one of the pixel values was outside the range 0 - 4095.

pt3emParameters

pt3em pt3emd	pt3emDither object.
pt3emred	pt3emArray of 256 red values.
pt3emgreen	pt3emArray of 256 green values.
pt3emblue	pt3emArray of 256 blue values.
pt3emalpha	pt3emArray of 256 alpha values.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.3 `__extern int caca_set_dither_brightness ( caca_dither_t * d, float brightness )`

Set the brightness of dither.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Brightness value was out of range.

pt3emParameters

pt3em pt3emd	pt3emDither object.
pt3em brightness	pt3embrightness value.

pt3emReturns

0 in case of success, -1 if an error occurred.

pt3em

12.9.2.4 `__extern float caca_get_dither_brightness ( caca_dither_t const * d )`

Get the brightness of the given dither object.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

Brightness value.

12.9.2.5 `__extern int caca_set_dither_gamma ( caca_dither_t * d, float gamma )`

Set the gamma of the given dither object. A negative value causes colour inversion.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Gamma value was out of range.

pt3emParameters

pt3em pt3emd	pt3emDither object.
pt3em gamma	pt3emGamma value.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.6 `__extern float caca_get_dither_gamma ( caca_dither_t const * d )`

Get the gamma of the given dither object.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

Gamma value.

pt3em

12.9.2.7 `__extern int caca_set_dither_contrast ( caca_dither_t * d, float contrast )`

Set the contrast of dither.

If an error occurs, -1 is returned and **errno** is set accordingly:

- `EINVAL` Contrast value was out of range.

pt3emParameters

pt3em pt3em d	pt3emDither object.
pt3em contrast	pt3emcontrast value.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.8 `__extern float caca_get_dither_contrast ( caca_dither_t const * d )`

Get the contrast of the given dither object.

This function never fails.

pt3emParameters

pt3em pt3em d	pt3emDither object.
---------------	---------------------

pt3emReturns

Contrast value.

12.9.2.9 `__extern int caca_set_dither_antialias ( caca_dither_t * d, char const * str )`

Tell the renderer whether to antialias the dither. Antialiasing smoothens the rendered image and avoids the commonly seen staircase effect.

- `"none"`: no antialiasing.
- `"prefilter"` or `"default"`: simple prefilter antialiasing. This is the default value.

If an error occurs, -1 is returned and **errno** is set accordingly:

- `EINVAL` Invalid antialiasing mode.

pt3em

pt3emParameters

pt3em pt3emd	pt3emDither object.
pt3em str	pt3emA string describing the antialiasing method that will be used for the dithering.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.10 `__extern char const* const* caca_get_dither_antialias_list ( caca_dither_t const * d )`

Return a list of available antialiasing methods for a given dither. The list is a NUL-terminated array of strings, interleaving a string containing the internal value for the antialiasing method to be used with [caca_set_dither_antialias\(\)](#), and a string containing the natural language description for that antialiasing method.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

An array of strings.

12.9.2.11 `__extern char const* caca_get_dither_antialias ( caca_dither_t const * d )`

Return the given dither's current antialiasing method.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

A static string.

12.9.2.12 `__extern int caca_set_dither_color ( caca_dither_t * d, char const * str )`

Tell the renderer which colours should be used to render the bitmap. Valid values for `str` are:

pt3em

- "mono": use light gray on a black background.
- "gray": use white and two shades of gray on a black background.
- "8": use the 8 ANSI colours on a black background.
- "16": use the 16 ANSI colours on a black background.
- "fullgray": use black, white and two shades of gray for both the characters and the background.
- "full8": use the 8 ANSI colours for both the characters and the background.
- "full16" or "default": use the 16 ANSI colours for both the characters and the background. This is the default value.

If an error occurs, -1 is returned and **errno** is set accordingly:

- EINVAL Invalid colour set.

pt3emParameters

pt3em pt3emd	pt3emDither object.
pt3emstr	pt3emA string describing the colour set that will be used for the dithering.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.13 `__extern char const* const* caca_get_dither_color_list ( caca_dither_t const * d )`

Return a list of available colour modes for a given dither. The list is a NULL-terminated array of strings, interleaving a string containing the internal value for the colour mode, to be used with [caca_set_dither_color\(\)](#), and a string containing the natural language description for that colour mode.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

An array of strings.

pt3em

12.9.2.14 `__extern char const* caca_get_dither_color ( caca_dither_t const * d )`

Return the given dither's current colour mode.

This function never fails.

pt3emParameters

pt3em	pt3emd	pt3emDither object.
-------	--------	---------------------

pt3emReturns

A static string.

12.9.2.15 `__extern int caca_set_dither_charset ( caca_dither_t * d, char const * str )`

Tell the renderer which characters should be used to render the dither. Valid values for `str` are:

- `"ascii"` or `"default"`: use only ASCII characters. This is the default value.
- `"shades"`: use Unicode characters "U+2591 LIGHT SHADE", "U+2592 MEDIUM SHADE" and "U+2593 DARK SHADE". These characters are also present in the CP437 codepage available on DOS and VGA.
- `"blocks"`: use Unicode quarter-cell block combinations. These characters are only found in the Unicode set.

If an error occurs, -1 is returned and **errno** is set accordingly:

- `EINVAL` Invalid character set.

pt3emParameters

pt3em	pt3emd	pt3emDither object.
	pt3emstr	pt3emA string describing the characters that need to be used for the dithering.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.16 `__extern char const* const* caca_get_dither_charset_list ( caca_dither_t const * d )`

Return a list of available character sets for a given dither. The list is a NULL-terminated array of strings, interleaving a string containing the internal value for the character set,

pt3em

to be used with [caca_set_dither_charset\(\)](#), and a string containing the natural language description for that character set.

This function never fails.

pt3emParameters

pt3em	pt3emd	pt3emDither object.
-------	--------	---------------------

pt3emReturns

An array of strings.

12.9.2.17 `__extern char const* caca_get_dither_charset ( caca_dither_t const * d )`

Return the given dither's current character set.

This function never fails.

pt3emParameters

pt3em	pt3emd	pt3emDither object.
-------	--------	---------------------

pt3emReturns

A static string.

12.9.2.18 `__extern int caca_set_dither_algorithm ( caca_dither_t * d, char const * str )`

Tell the renderer which dithering algorithm should be used. Dithering is necessary because the picture being rendered has usually far more colours than the available palette. Valid values for `str` are:

- "none": no dithering is used, the nearest matching colour is used.
- "ordered2": use a 2x2 Bayer matrix for dithering.
- "ordered4": use a 4x4 Bayer matrix for dithering.
- "ordered8": use a 8x8 Bayer matrix for dithering.
- "random": use random dithering.
- "fstein": use Floyd-Steinberg dithering. This is the default value.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Unknown dithering mode.

pt3em

pt3emParameters

pt3em pt3emd	pt3emDither object.
pt3em str	pt3emA string describing the algorithm that needs to be used for the dithering.

pt3emReturns

0 in case of success, -1 if an error occurred.

12.9.2.19 `__extern char const* const* caca_get_dither_algorithm_list ( caca_dither_t const * d )`

Return a list of available dithering algorithms for a given dither. The list is a NULL-terminated array of strings, interleaving a string containing the internal value for the dithering algorithm, to be used with `caca_set_dither_dithering()`, and a string containing the natural language description for that algorithm.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

An array of strings.

12.9.2.20 `__extern char const* caca_get_dither_algorithm ( caca_dither_t const * d )`

Return the given dither's current dithering algorithm.

This function never fails.

pt3emParameters

pt3em pt3emd	pt3emDither object.
--------------	---------------------

pt3emReturns

```
12.9.2.21 __extern int caca_dither_bitmap ( caca_canvas_t * cv, int x, int y, int w, int h,
caca_dither_t const * d, void * pixels )
```

This function never fails.

pt3em	pt3emA handle to the libcaca canvas.
pt3emx	pt3emX coordinate of the upper-left corner of the drawing area.
pt3emy	pt3emY coordinate of the upper-left corner of the drawing area.
pt3emw	pt3emWidth of the drawing area.
pt3emh	pt3emHeight of the drawing area.
pt3emd	pt3emDither object to be drawn.
pt3empixels	pt3emBitmap's pixels.

References `CACA_BLACK`, `caca_get_attr()`, `caca_put_char()`, `caca_set_attr()`, and `caca_set_color_ansi()`.

This function never fails.

pt3em pt3emd	pt3emDither object.
--------------	---------------------

This function always returns 0.

pt3em

pt3em

12.10 libcaca font handling

Functions

- `__extern caca_font_t * caca_load_font (void const *, size_t)`
Load a font from memory for future use.
- `__extern char const *const * caca_get_font_list (void)`
Get available builtin fonts.
- `__extern int caca_get_font_width (caca_font_t const *)`
Get a font's standard glyph width.
- `__extern int caca_get_font_height (caca_font_t const *)`
Get a font's standard glyph height.
- `__extern uint32_t const * caca_get_font_blocks (caca_font_t const *)`
Get a font's list of supported glyphs.
- `__extern int caca_render_canvas (caca_canvas_t const *, caca_font_t const *, void *, int, int, int)`
Render the canvas onto an image buffer.
- `__extern int caca_free_font (caca_font_t *)`
Free a font structure.

12.10.1 Detailed Description

These functions provide font handling routines and high quality canvas to bitmap rendering.

12.10.2 Function Documentation

12.10.2.1 `__extern caca_font_t* caca_load_font ( void const * data, size_t size )`

This function loads a font and returns a handle to its internal structure. The handle can then be used with `caca_render_canvas()` for bitmap output.

Internal fonts can also be loaded: if `size` is set to 0, `data` must be a string containing the internal font name.

If `size` is non-zero, the `size` bytes of memory at address `data` are loaded as a font. This memory are must not be freed by the calling program until the font handle has been freed with `caca_free_font()`.

If an error occurs, NULL is returned and **errno** is set accordingly:

- **ENOENT** Requested built-in font does not exist.
- **EINVAL** Invalid font data in memory area.
- **ENOMEM** Not enough memory to allocate font structure.

Parameters

<code>data</code>	The memory area containing the font or its name.
<code>size</code>	The size of the memory area, or 0 if the font name is given.

Returns

A font handle or NULL in case of error.

References `caca_load_font()`.

Referenced by `caca_load_font()`.

12.10.2.2 `__extern char const* const* caca_get_font_list ( void )`

Return a list of available builtin fonts. The list is a NULL-terminated array of strings.

This function never fails.

Returns

An array of strings.

12.10.2.3 `__extern int caca_get_font_width ( caca_font_t const * f )`

Return the standard value for the current font's glyphs. Most glyphs in the font will have this width, except fullwidth characters.

This function never fails.

Parameters

<code>f</code>	The font, as returned by caca_load_font()
----------------	---

Returns

The standard glyph width.

12.10.2.4 `__extern int caca_get_font_height ( caca_font_t const * f )`

Returns the standard value for the current font's glyphs. Most glyphs in the font will have this height.

This function never fails.

Parameters

<code>pt3em</code>	<code>pt3emf</code>	The font, as returned by caca_load_font()
--------------------	---------------------	---

Returns

The standard glyph height.

12.10.2.5 `__extern uint32_t const* caca_get_font_blocks ( caca_font_t const * f )`

This function returns the list of Unicode blocks supported by the given font. The list is a zero-terminated list of indices. Here is an example:

```
{
    0x0000, 0x0080,    // Basic latin: A, B, C, a, b, c
    0x0080, 0x0100,    // Latin-1 supplement: "A, 'e, ^u
    0x0530, 0x0590,    // Armenian
    0x0000, 0x0000,    // END
};
```

This function never fails.

Parameters

<code>pt3em</code>	<code>pt3emf</code>	The font, as returned by caca_load_font()
--------------------	---------------------	---

Returns

The list of Unicode blocks supported by the font.

12.10.2.6 `__extern int caca_render_canvas ( caca_canvas_t const * cv, caca_font_t const * f, void * buf, int width, int height, int pitch )`

This function renders the given canvas on an image buffer using a specific font. The pixel format is fixed (32-bit ARGB, 8 bits for each component).

The required image width can be computed using [caca_get_canvas_width\(\)](#) and [caca_get_font_width\(\)](#). The required height can be computed using [caca_get_canvas_height\(\)](#) and [caca_get_font_height\(\)](#).

Glyphs that do not fit in the image buffer are currently not rendered at all. They may be cropped instead in future versions.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Specified width, height or pitch is invalid.

Parameters

<code>pt3em</code>	<code>pt3emcv</code>	The canvas to render
--------------------	----------------------	----------------------

pt3em

pt3em	pt3em	pt3em	The font, as returned by caca_load_font()
pt3em	pt3embuf	pt3em	The image buffer
pt3em	pt3emwidth	pt3em	The width (in pixels) of the image buffer
pt3em	pt3emheight	pt3em	The height (in pixels) of the image buffer
pt3em	pt3empitch	pt3em	The pitch (in bytes) of an image buffer line.

pt3emReturns

0 in case of success, -1 if an error occurred.

References [caca_attr_to_argb64\(\)](#).

12.10.2.7 `__extern int caca_free_font ( caca_font_t * f )`

This function frees all data allocated by [caca_load_font\(\)](#). The font structure is no longer usable by other libcaca functions. Once this function has returned, the memory area that was given to [caca_load_font\(\)](#) can be freed.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The font, as returned by caca_load_font()
-------	-------	-------	---

pt3emReturns

This function always returns 0.

pt3em

pt3em

12.11 libcaca FIGfont handling

Functions

- `__extern int caca_canvas_set_figfont (caca_canvas_t *, char const *)`
- `__extern int caca_put_figchar (caca_canvas_t *, uint32_t)`
- `__extern int caca_flush_figlet (caca_canvas_t *)`

12.11.1 Detailed Description

These functions provide FIGlet and TOLlet font handling routines.

pt3em

pt3em

12.12 libcaca file IO

Functions

- `__extern caca\_file\_t * caca_file_open (char const *, const char *)`
- `__extern int caca_file_close (caca\_file\_t *)`
- `__extern uint64_t caca_file_tell (caca\_file\_t *)`
- `__extern size_t caca_file_read (caca\_file\_t *, void *, size_t)`
- `__extern size_t caca_file_write (caca\_file\_t *, const void *, size_t)`
- `__extern char * caca_file_gets (caca\_file\_t *, char *, int)`
- `__extern int caca_file_eof (caca\_file\_t *)`

12.12.1 Detailed Description

These functions allow to read and write files in a platform-independent way.

pt3em

pt3em

12.13 libcaca importers/exporters from/to various

Functions

- `__extern ssize_t caca_import_memory (caca\_canvas\_t *, void const *, size_t, char const *)`
- `__extern ssize_t caca_import_file (caca\_canvas\_t *, char const *, char const *)`
- `__extern char const *const * caca_get_import_list (void)`
- `__extern void * caca_export_memory (caca\_canvas\_t const *, char const *, size_t *)`
- `__extern char const *const * caca_get_export_list (void)`

12.13.1 Detailed Description

formats

These functions import various file formats into a new canvas, or export the current canvas to various text formats.

pt3em

pt3em

12.14 libcaca display functions

Functions

- `__extern caca\_display\_t * caca\_create\_display (caca\_canvas\_t *)`
Attach a caca graphical context to a caca canvas.
- `__extern caca_display_t * caca_create_display_with_driver (caca_canvas_t *,
char const *)`
Attach a specific caca graphical context to a caca canvas.
- `__extern char const *const * caca\_get\_display\_driver\_list (void)`
Get available display drivers.
- `__extern char const * caca\_get\_display\_driver (caca\_display\_t *)`
Return a caca graphical context's current output driver.
- `__extern int caca\_set\_display\_driver (caca\_display\_t *, char const *)`
Set the output driver.
- `__extern int caca\_free\_display (caca\_display\_t *)`
Detach a caca graphical context from a caca backend context.
- `__extern caca\_canvas\_t * caca\_get\_canvas (caca\_display\_t *)`
Get the canvas attached to a caca graphical context.
- `__extern int caca\_refresh\_display (caca\_display\_t *)`
Flush pending changes and redraw the screen.
- `__extern int caca\_set\_display\_time (caca\_display\_t *, int)`
Set the refresh delay.
- `__extern int caca\_get\_display\_time (caca\_display\_t const *)`
Get the display's average rendering time.
- `__extern int caca\_get\_display\_width (caca\_display\_t const *)`
Get the display width.
- `__extern int caca\_get\_display\_height (caca\_display\_t const *)`
Get the display height.
- `__extern int caca\_set\_display\_title (caca\_display\_t *, char const *)`
Set the display title.
- `__extern int caca\_set\_mouse (caca\_display\_t *, int)`
Show or hide the mouse pointer.
- `__extern int caca\_set\_cursor (caca\_display\_t *, int)`
Show or hide the cursor.

12.14.1 Detailed Description

These functions provide the basic *libcaca* routines for display initialisation, system information retrieval and configuration.

12.14.2 Function Documentation

Create a graphical context using device-dependent features (ncurses for terminals, an X11 window, a DOS command window...) that attaches to a libcacanvas. Everything that gets drawn in the libcacanvas can then be displayed by the libcacanvas driver.

See also [caca create display with driver\(\)](#).

- ENOMEM Not enough memory.
- ENODEV Graphical device could not be initialised.

pt3empt3emcv	pt3emThe caca canvas or NULL to create a canvas automatically.
--------------	--

The caca graphical context or NULL if an error occurred.

Create a graphical context using device-dependent features (ncurses for terminals, an X11 window, a DOS command window...) that attaches to a libcac canvas. Everything that gets drawn in the libcac canvas can then be displayed by the libcac driver.

If no driver name is provided, *libcaca* will try to autodetect the best output driver it can.

If an error occurs, `NULL` is returned and `errno` is set accordingly:

- ENOMEM Not enough memory.
- ENODEV Graphical device could not be initialised.

Parameters

<code>cv</code>	The caca canvas or NULL to create a canvas automatically.
<code>driver</code>	A string describing the desired output driver or NULL to choose the best driver automatically.

Returns

The caca graphical context or NULL if an error occurred.

References `caca_create_canvas()`, `caca_free_canvas()`, `caca_manage_canvas()`, and `caca_unmanage_canvas()`.

Referenced by `caca_create_display()`.

12.14.2.3 `__extern char const* const* caca_get_display_driver_list ( void )`

Return a list of available display drivers. The list is a NULL-terminated array of strings, interleaving a string containing the internal value for the display driver, and a string containing the natural language description for that driver.

This function never fails.

Returns

An array of strings.

12.14.2.4 `__extern char const* caca_get_display_driver ( caca_display_t * dp )`

Return the given display's current output driver.

This function never fails.

Parameters

<code>dp</code>	The caca display.
-----------------	-------------------

Returns

A static string.

12.14.2.5 `__extern int caca_set_display_driver ( caca_display_t * dp, char const * driver )`

Dynamically change the given display's output driver.

FIXME: decide what to do in case of failure

Parameters

<code>dp</code>	The caca display.
<code>driver</code>	A string describing the desired output driver or NULL to choose the best driver automatically.

Returns

0 in case of success, -1 if an error occurred.

12.14.2.6 `extern int caca_free_display ( caca_display_t * dp )`

Detach a graphical context from its caca backend and destroy it. The libcaca canvas continues to exist and other graphical contexts can be attached to it afterwards.

If the caca canvas was automatically created by [caca_create_display\(\)](#), it is automatically destroyed and any handle to it becomes invalid.

This function never fails.

Parameters

<code>dp</code>	The libcaca graphical context.
-----------------	--------------------------------

Returns

This function always returns 0.

References [caca_free_canvas\(\)](#), and [caca_unmanage_canvas\(\)](#).

12.14.2.7 `extern caca_canvas_t* caca_get_canvas ( caca_display_t * dp )`

Return a handle on the `caca_canvas_t` object that was either attached or created by [caca_create_display\(\)](#).

This function never fails.

Parameters

<code>dp</code>	The libcaca graphical context.
-----------------	--------------------------------

pt3em

pt3emReturns

The libcaca canvas.

12.14.2.8 `__extern int caca_refresh_display ( caca_display_t * dp )`

Flush all graphical operations and print them to the display device. Nothing will show on the screen until this function is called.

If [caca_set_display_time\(\)](#) was called with a non-zero value, [caca_refresh_display\(\)](#) will use that value to achieve constant framerate: if two consecutive calls to [caca_refresh_display\(\)](#) are within a time range shorter than the value set with [caca_set_display_time\(\)](#), the second call will be delayed before performing the screen refresh.

This function never fails.

pt3emParameters

pt3em pt3em	<code>dp</code>	pt3emThe libcaca display context.
----------------	-----------------	-----------------------------------

pt3emReturns

This function always returns 0.

12.14.2.9 `__extern int caca_set_display_time ( caca_display_t * dp, int usec )`

Set the refresh delay in microseconds. The refresh delay is used by [caca_refresh_display\(\)](#) to achieve constant framerate. See the [caca_refresh_display\(\)](#) documentation for more details.

If the argument is zero, constant framerate is disabled. This is the default behaviour.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **EINVAL** Refresh delay value is invalid.

pt3emParameters

pt3em pt3em	<code>dp</code>	pt3emThe libcaca display context.
pt3em	<code>usec</code>	pt3emThe refresh delay in microseconds.

pt3emReturns

12.14.2.10 `__extern int caca_get_display_time ( caca_display_t const * dp )`

This function never fails.

pt3em	pt3em	pt3em	The libcac display context.
-------	-------	-------	-----------------------------

The render time in microseconds.

```
12.14.2.11 __extern int caca_get_display_width ( caca_display_t const * dp )
```

This function never fails.

pt3em	pt3em	pt3em	The libcac display context.
-------	-------	-------	-----------------------------

The display width.

```
12.14.2.12 __extern int caca_get_display_height( caca_display_t const * dp )
```

This function never fails.

Parameters

<code>dp</code>	The libcaca display context.
-----------------	------------------------------

Returns

The display height.

12.14.2.13 `__extern int caca_set_display_title ( caca_display_t * dp, char const * title )`

If libcaca runs in a window, try to change its title. This works with the ncurses, S-Lang, OpenGL, X11 and Win32 drivers.

If an error occurs, -1 is returned and **errno** is set accordingly:

- ENOSYS Display driver does not support setting the window title.

Parameters

<code>dp</code>	The libcaca display context.
<code>title</code>	The desired display title.

Returns

0 upon success, -1 if an error occurred.

12.14.2.14 `__extern int caca_set_mouse ( caca_display_t * dp, int flag )`

Show or hide the mouse pointer. This function works with the ncurses, S-Lang and X11 drivers.

If an error occurs, -1 is returned and **errno** is set accordingly:

- ENOSYS Display driver does not support hiding the mouse pointer.

Parameters

<code>dp</code>	The libcaca display context.
<code>flag</code>	0 hides the pointer, 1 shows the system's default pointer (usually an arrow). Other values are reserved for future use.

pt3em

pt3emReturns

0 upon success, -1 if an error occurred.

12.14.2.15 `__extern int caca_set_cursor ( caca_display_t * dp, int flag )`

Show or hide the cursor, for devices that support such a feature.

If an error occurs, -1 is returned and **errno** is set accordingly:

- **ENOSYS** Display driver does not support showing the cursor.

pt3emParameters

pt3empt3em <i>dp</i>	pt3emThe libcaca display context.
pt3em <i>flag</i>	pt3em0 hides the cursor, 1 shows the system's default cursor (usually a white rectangle). Other values are reserved for future use.

pt3emReturns

0 upon success, -1 if an error occurred.

pt3em

pt3em

12.15 libcaca event handling

Functions

- `__extern int caca\_get\_event (caca\_display\_t *, int, caca\_event\_t *, int)`
Get the next mouse or keyboard input event.
- `__extern int caca\_get\_mouse\_x (caca\_display\_t const *)`
Return the X mouse coordinate.
- `__extern int caca\_get\_mouse\_y (caca\_display\_t const *)`
Return the Y mouse coordinate.
- `__extern enum caca\_event\_type caca\_get\_event\_type (caca\_event\_t const *)`
Return an event's type.
- `__extern int caca\_get\_event\_key\_ch (caca\_event\_t const *)`
Return a key press or key release event's value.
- `__extern uint32_t caca\_get\_event\_key\_utf32 (caca\_event\_t const *)`
Return a key press or key release event's Unicode value.
- `__extern int caca\_get\_event\_key\_utf8 (caca\_event\_t const *, char *)`
Return a key press or key release event's UTF-8 value.
- `__extern int caca\_get\_event\_mouse\_button (caca\_event\_t const *)`
Return a mouse press or mouse release event's button.
- `__extern int caca\_get\_event\_mouse\_x (caca\_event\_t const *)`
Return a mouse motion event's X coordinate.
- `__extern int caca\_get\_event\_mouse\_y (caca\_event\_t const *)`
Return a mouse motion event's Y coordinate.
- `__extern int caca\_get\_event\_resize\_width (caca\_event\_t const *)`
Return a resize event's display width value.
- `__extern int caca\_get\_event\_resize\_height (caca\_event\_t const *)`
Return a resize event's display height value.

12.15.1 Detailed Description

These functions handle user events such as keyboard input and mouse clicks.

12.15.2 Function Documentation

12.15.2.1 `__extern int caca\_get\_event ( caca\_display\_t * dp, int event_mask, caca\_event\_t * ev, int timeout )`

Poll the event queue for mouse or keyboard events matching the event mask and return the first matching event. Non-matching events are discarded. If `event_mask` is zero, the function returns immediately.

pt3em

The timeout value tells how long this function needs to wait for an event. A value of zero returns immediately and the function returns zero if no more events are pending in the queue. A negative value causes the function to wait indefinitely until a matching event is received.

If not null, `ev` will be filled with information about the event received. If null, the function will return but no information about the event will be sent.

This function never fails.

pt3emParameters

pt3em	pt3em	dp	pt3emThe libcaca graphical context.
pt3em	event_mask		pt3emBitmask of requested events.
pt3em	timeout		pt3emA timeout value in microseconds, -1 for blocking behaviour
pt3em	ev		pt3emA pointer to a caca_event structure, or NULL.

pt3emReturns

1 if a matching event was received, or 0 if the wait timed out.

References CACA_EVENT_NONE.

12.15.2.2 `__extern int caca_get_mouse_x ( caca_display_t const * dp )`

Return the X coordinate of the mouse position last time it was detected. This function is not reliable if the ncurses or S-Lang drivers are being used, because mouse position is only detected when the mouse is clicked. Other drivers such as X11 work well.

This function never fails.

pt3emParameters

pt3em	pt3em	dp	pt3emThe libcaca graphical context.
-------	-------	----	-------------------------------------

pt3emReturns

The X mouse coordinate.

References `caca_get_canvas_width()`.

12.15.2.3 `__extern int caca_get_mouse_y ( caca_display_t const * dp )`

Return the Y coordinate of the mouse position last time it was detected. This function is not reliable if the ncurses or S-Lang drivers are being used, because mouse position is only detected when the mouse is clicked. Other drivers such as X11 work well.

This function never fails.

pt3em

pt3emParameters

pt3em	pt3em	pt3em	The libcaca graphical context.
-------	-------	-------	--------------------------------

pt3emReturns

The Y mouse coordinate.

References [caca_get_canvas_height\(\)](#).

12.15.2.4 `__extern enum caca_event_type caca_get_event_type ( caca_event_t const * ev )`

Return the type of an event. This function may always be called on an event after [caca_get_event\(\)](#) was called, and its return value indicates which other functions may be called:

- `CACA_EVENT_NONE` : no other function may be called.
- `CACA_EVENT_KEY_PRESS`, `CACA_EVENT_KEY_RELEASE` : [caca_get_event_key_ch\(\)](#), [caca_get_event_key_utf32\(\)](#) and [caca_get_event_key_utf8\(\)](#) may be called.
- `CACA_EVENT_MOUSE_PRESS`, `CACA_EVENT_MOUSE_RELEASE` : [caca_get_event_mouse_button\(\)](#) may be called.
- `CACA_EVENT_MOUSE_MOTION` : [caca_get_event_mouse_x\(\)](#) and [caca_get_event_mouse_y\(\)](#) may be called.
- `CACA_EVENT_RESIZE` : [caca_get_event_resize_width\(\)](#) and [caca_get_event_resize_height\(\)](#) may be called.
- `CACA_EVENT_QUIT` : no other function may be called.

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The libcaca event.
-------	-------	-------	--------------------

pt3emReturns

The event's type.

12.15.2.5 `__extern int caca_get_event_key_ch ( caca_event_t const * ev )`

Return either the ASCII value for an event's key, or if the key is not an ASCII character, an appropriate *enum caca_key* value.

pt3em

This function never fails, but must only be called with a valid event of type `CACA_EVENT_KEY_PRESS` or `CACA_EVENT_KEY_RELEASE`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3em	pt3em	pt3em
-------	-------	-------

pt3emReturns

The key value.

12.15.2.6 `__extern uint32_t caca_get_event_key_utf32 ( caca_event_t const * ev )`

Return the UTF-32/UCS-4 value for an event's key if it resolves to a printable character.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_KEY_PRESS` or `CACA_EVENT_KEY_RELEASE`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3em	pt3em	pt3em
-------	-------	-------

pt3emReturns

The key's Unicode value.

12.15.2.7 `__extern int caca_get_event_key_utf8 ( caca_event_t const * ev, char * utf8 )`

Write the UTF-8 value for an event's key if it resolves to a printable character. Up to 6 UTF-8 bytes and a null termination are written.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_KEY_PRESS` or `CACA_EVENT_KEY_RELEASE`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3em	pt3em	pt3em
-------	-------	-------

pt3emReturns

This function always returns 0.

pt3em

12.15.2.8 `__extern int caca_get_event_mouse_button ( caca_event_t const * ev )`

Return the mouse button index for an event.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_MOUSE_PRESS` or `CACA_EVENT_MOUSE_RELEASE`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3em	pt3em	pt3em	The libcaca event.
-------	-------	-------	--------------------

pt3emReturns

The event's mouse button.

12.15.2.9 `__extern int caca_get_event_mouse_x ( caca_event_t const * ev )`

Return the X coordinate for a mouse motion event.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_MOUSE_MOTION`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3em	pt3em	pt3em	The libcaca event.
-------	-------	-------	--------------------

pt3emReturns

The event's X mouse coordinate.

12.15.2.10 `__extern int caca_get_event_mouse_y ( caca_event_t const * ev )`

Return the Y coordinate for a mouse motion event.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_MOUSE_MOTION`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3em	pt3em	pt3em	The libcaca event.
-------	-------	-------	--------------------

pt3em

pt3emReturns

The event's Y mouse coordinate.

12.15.2.11 `__extern int caca_get_event_resize_width ( caca_event_t const * ev )`

Return the width value for a display resize event.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_RESIZE`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3empt3em	pt3emThe libcaca event.
------------	-------------------------

pt3emReturns

The event's new display width value.

12.15.2.12 `__extern int caca_get_event_resize_height ( caca_event_t const * ev )`

Return the height value for a display resize event.

This function never fails, but must only be called with a valid event of type `CACA_EVENT_RESIZE`, or the results will be undefined. See [caca_get_event_type\(\)](#) for more information.

pt3emParameters

pt3empt3em	pt3emThe libcaca event.
------------	-------------------------

pt3emReturns

The event's new display height value.

pt3em

13 Data Structure Documentation

13.1 caca_event Struct Reference

Handling of user events.

Data Fields

- enum [caca_event_type](#) **type**
 - union {
 - struct {
 - int **x**
 - int **y**
 - int **button**
 - mouse**
 - struct {
 - int **w**
 - int **h**
 - resize**
 - struct {
 - int **ch**
 - uint32_t **utf32**
 - char **utf8** [8]
 - key**
 - data**
- uint8_t **padding** [16]

13.1.1 Detailed Description

This structure is filled by [caca_get_event\(\)](#) when an event is received. It is an opaque structure that should only be accessed through [caca_event_get_type\(\)](#) and similar functions. The struct members may no longer be directly accessible in future versions.

14 File Documentation

14.1 caca.h File Reference

The *libcaca* public header.

pt3em

Data Structures

- struct [caca_event](#)
Handling of user events.

Defines

- #define [CACA_API_VERSION_1](#)
- #define [CACA_BLACK](#) 0x00
- #define [CACA_BLUE](#) 0x01
- #define [CACA_GREEN](#) 0x02
- #define [CACA_CYAN](#) 0x03
- #define [CACA_RED](#) 0x04
- #define [CACA_MAGENTA](#) 0x05
- #define [CACA_BROWN](#) 0x06
- #define [CACA_LIGHTGRAY](#) 0x07
- #define [CACA_DARKGRAY](#) 0x08
- #define [CACA_LIGHTBLUE](#) 0x09
- #define [CACA_LIGHTGREEN](#) 0x0a
- #define [CACA_LIGHTCYAN](#) 0x0b
- #define [CACA_LIGHTRED](#) 0x0c
- #define [CACA_LIGHTMAGENTA](#) 0x0d
- #define [CACA_YELLOW](#) 0x0e
- #define [CACA_WHITE](#) 0x0f
- #define [CACA_DEFAULT](#) 0x10
- #define [CACA_TRANSPARENT](#) 0x20
- #define [CACA_BOLD](#) 0x01
- #define [CACA_ITALICS](#) 0x02
- #define [CACA_UNDERLINE](#) 0x04
- #define [CACA_BLINK](#) 0x08
- #define [CACA_MAGIC_FULLWIDTH](#) 0x000fffe

Typedefs

- typedef struct caca_canvas [caca_canvas_t](#)
- typedef struct caca_dither [caca_dither_t](#)
- typedef struct caca_font [caca_font_t](#)
- typedef struct caca_file [caca_file_t](#)
- typedef struct caca_display [caca_display_t](#)
- typedef struct [caca_event](#) [caca_event_t](#)

pt3em

Enumerations

- enum `caca_event_type` { `CACA_EVENT_NONE` = 0x0000, `CACA_EVENT_KEY_PRESS` = 0x0001, `CACA_EVENT_KEY_RELEASE` = 0x0002, `CACA_EVENT_MOUSE_PRESS` = 0x0004, `CACA_EVENT_MOUSE_RELEASE` = 0x0008, `CACA_EVENT_MOUSE_MOTION` = 0x0010, `CACA_EVENT_RESIZE` = 0x0020, `CACA_EVENT_QUIT` = 0x0040, `CACA_EVENT_ANY` = 0xffff }

User event type enumeration.

- enum `caca_key` { `CACA_KEY_UNKNOWN` = 0x00, `CACA_KEY_CTRL_A` = 0x01, `CACA_KEY_CTRL_B` = 0x02, `CACA_KEY_CTRL_C` = 0x03, `CACA_KEY_CTRL_D` = 0x04, `CACA_KEY_CTRL_E` = 0x05, `CACA_KEY_CTRL_F` = 0x06, `CACA_KEY_CTRL_G` = 0x07, `CACA_KEY_BACKSPACE` = 0x08, `CACA_KEY_CTRL_TAB` = 0x09, `CACA_KEY_CTRL_J` = 0x0a, `CACA_KEY_CTRL_K` = 0x0b, `CACA_KEY_CTRL_L` = 0x0c, `CACA_KEY_RETURN` = 0x0d, `CACA_KEY_CTRL_N` = 0x0e, `CACA_KEY_CTRL_O` = 0x0f, `CACA_KEY_CTRL_P` = 0x10, `CACA_KEY_CTRL_Q` = 0x11, `CACA_KEY_CTRL_R` = 0x12, `CACA_KEY_PAUSE` = 0x13, `CACA_KEY_CTRL_T` = 0x14, `CACA_KEY_CTRL_U` = 0x15, `CACA_KEY_CTRL_V` = 0x16, `CACA_KEY_CTRL_W` = 0x17, `CACA_KEY_CTRL_X` = 0x18, `CACA_KEY_CTRL_Y` = 0x19, `CACA_KEY_CTRL_Z` = 0x1a, `CACA_KEY_ESCAPE` = 0x1b, `CACA_KEY_DELETE` = 0x7f, `CACA_KEY_UP` = 0x111, `CACA_KEY_DOWN` = 0x112, `CACA_KEY_LEFT` = 0x113, `CACA_KEY_RIGHT` = 0x114, `CACA_KEY_INSERT` = 0x115, `CACA_KEY_HOME` = 0x116, `CACA_KEY_END` = 0x117, `CACA_KEY_PAGEUP` = 0x118, `CACA_KEY_PAGEDOWN` = 0x119, `CACA_KEY_F1` = 0x11a, `CACA_KEY_F2` = 0x11b, `CACA_KEY_F3` = 0x11c, `CACA_KEY_F4` = 0x11d, `CACA_KEY_F5` = 0x11e, `CACA_KEY_F6` = 0x11f, `CACA_KEY_F7` = 0x120, `CACA_KEY_F8` = 0x121, `CACA_KEY_F9` = 0x122, `CACA_KEY_F10` = 0x123, `CACA_KEY_F11` = 0x124, `CACA_KEY_F12` = 0x125, `CACA_KEY_F13` = 0x126, `CACA_KEY_F14` = 0x127, `CACA_KEY_F15` = 0x128 }

Special key values.

Functions

- __extern `caca_canvas_t` * `caca_create_canvas` (int, int)
Initialise a libcaca canvas.
- __extern int `caca_manage_canvas` (`caca_canvas_t` *, int(*) (void *), void *)
Manage a canvas.
- __extern int `caca_unmanage_canvas` (`caca_canvas_t` *, int(*) (void *), void *)
Unmanage a canvas.
- __extern int `caca_set_canvas_size` (`caca_canvas_t` *, int, int)
Resize a canvas.
- __extern int `caca_get_canvas_width` (`caca_canvas_t` const *)
Get the canvas width.
- __extern int `caca_get_canvas_height` (`caca_canvas_t` const *)

pt3em

Get the canvas height.

- `__extern uint8_t const * caca_get_canvas_chars (caca_canvas_t const *)`

Get the canvas character array.

- `__extern uint8_t const * caca_get_canvas_attrs (caca_canvas_t const *)`

Get the canvas attribute array.

- `__extern int caca_free_canvas (caca_canvas_t *)`

Uninitialise libcaca.

- `__extern int caca_rand (int, int)`

Generate a random integer within a range.

- `__extern char const * caca_get_version (void)`

Return the libcaca version.

- `__extern int caca_gotoxy (caca_canvas_t *, int, int)`

Set cursor position.

- `__extern int caca_get_cursor_x (caca_canvas_t const *)`

Get X cursor position.

- `__extern int caca_get_cursor_y (caca_canvas_t const *)`

Get Y cursor position.

- `__extern int caca_put_char (caca_canvas_t *, int, int, uint32_t)`

Print an ASCII or Unicode character.

- `__extern uint32_t caca_get_char (caca_canvas_t const *, int, int)`

Get the Unicode character at the given coordinates.

- `__extern int caca_put_str (caca_canvas_t *, int, int, char const *)`

Print a string.

- `__extern int caca_printf (caca_canvas_t *, int, int, char const *,...)`

Print a formatted string.

- `__extern int caca_clear_canvas (caca_canvas_t *)`

Clear the canvas.

- `__extern int caca_set_canvas_handle (caca_canvas_t *, int, int)`

Set cursor handle.

- `__extern int caca_get_canvas_handle_x (caca_canvas_t const *)`

Get X handle position.

- `__extern int caca_get_canvas_handle_y (caca_canvas_t const *)`

Get Y handle position.

- `__extern int caca_blit (caca_canvas_t *, int, int, caca_canvas_t const *, caca_canvas_t const *)`

Blit a canvas onto another one.

- `__extern int caca_set_canvas_boundaries (caca_canvas_t *, int, int, int, int)`

Set a canvas' new boundaries.

- `__extern int caca_invert (caca_canvas_t *)`

Invert a canvas' colours.

- `__extern int caca_flip (caca_canvas_t *)`

Flip a canvas horizontally.

pt3em

- `__extern int caca_flop (caca_canvas_t *)`
Flip a canvas vertically.
- `__extern int caca_rotate_180 (caca_canvas_t *)`
Rotate a canvas.
- `__extern int caca_rotate_left (caca_canvas_t *)`
Rotate a canvas, 90 degrees counterclockwise.
- `__extern int caca_rotate_right (caca_canvas_t *)`
Rotate a canvas, 90 degrees counterclockwise.
- `__extern int caca_stretch_left (caca_canvas_t *)`
Rotate and stretch a canvas, 90 degrees counterclockwise.
- `__extern int caca_stretch_right (caca_canvas_t *)`
Rotate and stretch a canvas, 90 degrees clockwise.
- `__extern uint32_t caca_get_attr (caca_canvas_t const *, int, int)`
Get the text attribute at the given coordinates.
- `__extern int caca_set_attr (caca_canvas_t *, uint32_t)`
Set the default character attribute.
- `__extern int caca_put_attr (caca_canvas_t *, int, int, uint32_t)`
Set the character attribute at the given coordinates.
- `__extern int caca_set_color_ansi (caca_canvas_t *, uint8_t, uint8_t)`
Set the default colour pair for text (ANSI version).
- `__extern int caca_set_color_argb (caca_canvas_t *, uint16_t, uint16_t)`
Set the default colour pair for text (truecolor version).
- `__extern uint8_t caca_attr_to_ansi (uint32_t)`
Get DOS ANSI information from attribute.
- `__extern uint8_t caca_attr_to_ansi_fg (uint32_t)`
Get ANSI foreground information from attribute.
- `__extern uint8_t caca_attr_to_ansi_bg (uint32_t)`
Get ANSI background information from attribute.
- `__extern uint16_t caca_attr_to_rgb12_fg (uint32_t)`
Get 12-bit RGB foreground information from attribute.
- `__extern uint16_t caca_attr_to_rgb12_bg (uint32_t)`
Get 12-bit RGB background information from attribute.
- `__extern void caca_attr_to_argb64 (uint32_t, uint8_t[8])`
Get 64-bit ARGB information from attribute.
- `__extern uint32_t caca_utf8_to_utf32 (char const *, size_t *)`
Convert a UTF-8 character to UTF-32.
- `__extern size_t caca_utf32_to_utf8 (char *, uint32_t)`
Convert a UTF-32 character to UTF-8.
- `__extern uint8_t caca_utf32_to_cp437 (uint32_t)`
Convert a UTF-32 character to CP437.
- `__extern uint32_t caca_cp437_to_utf32 (uint8_t)`
Convert a CP437 character to UTF-32.

pt3em

- `__extern char caca_utf32_to_ascii (uint32_t)`
Convert a UTF-32 character to ASCII.
- `__extern int caca_utf32_is_fullwidth (uint32_t)`
Tell whether a UTF-32 character is fullwidth.
- `__extern int caca_draw_line (caca_canvas_t *, int, int, int, int, uint32_t)`
Draw a line on the canvas using the given character.
- `__extern int caca_draw_polyline (caca_canvas_t *, int const x[], int const y[], int, uint32_t)`
Draw a polyline.
- `__extern int caca_draw_thin_line (caca_canvas_t *, int, int, int, int)`
Draw a thin line on the canvas, using ASCII art.
- `__extern int caca_draw_thin_polyline (caca_canvas_t *, int const x[], int const y[], int)`
Draw an ASCII art thin polyline.
- `__extern int caca_draw_circle (caca_canvas_t *, int, int, int, uint32_t)`
Draw a circle on the canvas using the given character.
- `__extern int caca_draw_ellipse (caca_canvas_t *, int, int, int, int, uint32_t)`
Draw an ellipse on the canvas using the given character.
- `__extern int caca_draw_thin_ellipse (caca_canvas_t *, int, int, int, int)`
Draw a thin ellipse on the canvas.
- `__extern int caca_fill_ellipse (caca_canvas_t *, int, int, int, int, uint32_t)`
Fill an ellipse on the canvas using the given character.
- `__extern int caca_draw_box (caca_canvas_t *, int, int, int, int, uint32_t)`
Draw a box on the canvas using the given character.
- `__extern int caca_draw_thin_box (caca_canvas_t *, int, int, int, int)`
Draw a thin box on the canvas.
- `__extern int caca_draw_cp437_box (caca_canvas_t *, int, int, int, int)`
Draw a box on the canvas using CP437 characters.
- `__extern int caca_fill_box (caca_canvas_t *, int, int, int, int, uint32_t)`
Fill a box on the canvas using the given character.
- `__extern int caca_draw_triangle (caca_canvas_t *, int, int, int, int, int, int, uint32_t)`
Draw a triangle on the canvas using the given character.
- `__extern int caca_draw_thin_triangle (caca_canvas_t *, int, int, int, int, int, int)`
Draw a thin triangle on the canvas.
- `__extern int caca_fill_triangle (caca_canvas_t *, int, int, int, int, int, int, uint32_t)`
Fill a triangle on the canvas using the given character.
- `__extern int caca_get_frame_count (caca_canvas_t const *)`
Get the number of frames in a canvas.
- `__extern int caca_set_frame (caca_canvas_t *, int)`
Activate a given canvas frame.
- `__extern char const * caca_get_frame_name (caca_canvas_t const *)`

pt3em

Get the current frame's name.

- `__extern int caca_set_frame_name (caca_canvas_t *, char const *)`

Set the current frame's name.

- `__extern int caca_create_frame (caca_canvas_t *, int)`

Add a frame to a canvas.

- `__extern int caca_free_frame (caca_canvas_t *, int)`

Remove a frame from a canvas.

- `__extern caca_dither_t * caca_create_dither (int, int, int, int, uint32_t, uint32_t, uint32_t, uint32_t)`

Create an internal dither object.

- `__extern int caca_set_dither_palette (caca_dither_t *, uint32_t r[], uint32_t g[], uint32_t b[], uint32_t a[])`

Set the palette of an 8bpp dither object.

- `__extern int caca_set_dither_brightness (caca_dither_t *, float)`

Set the brightness of a dither object.

- `__extern float caca_get_dither_brightness (caca_dither_t const *)`

Get the brightness of a dither object.

- `__extern int caca_set_dither_gamma (caca_dither_t *, float)`

Set the gamma of a dither object.

- `__extern float caca_get_dither_gamma (caca_dither_t const *)`

Get the gamma of a dither object.

- `__extern int caca_set_dither_contrast (caca_dither_t *, float)`

Set the contrast of a dither object.

- `__extern float caca_get_dither_contrast (caca_dither_t const *)`

Get the contrast of a dither object.

- `__extern int caca_set_dither_antialias (caca_dither_t *, char const *)`

Set dither antialiasing.

- `__extern char const *const * caca_get_dither_antialias_list (caca_dither_t const *)`

Get available antialiasing methods.

- `__extern char const * caca_get_dither_antialias (caca_dither_t const *)`

Get current antialiasing method.

- `__extern int caca_set_dither_color (caca_dither_t *, char const *)`

Choose colours used for dithering.

- `__extern char const *const * caca_get_dither_color_list (caca_dither_t const *)`

Get available colour modes.

- `__extern char const * caca_get_dither_color (caca_dither_t const *)`

Get current colour mode.

- `__extern int caca_set_dither_charset (caca_dither_t *, char const *)`

Choose characters used for dithering.

- `__extern char const *const * caca_get_dither_charset_list (caca_dither_t const *)`

pt3em

Get available dither character sets.

- `__extern char const * caca\_get\_dither\_charset (caca\_dither\_t const *)`

Get current character set.

- `__extern int caca\_set\_dither\_algorithm (caca\_dither\_t *, char const *)`

Set dithering algorithm.

- `__extern char const *const * caca\_get\_dither\_algorithm\_list (caca\_dither\_t const *)`

Get dithering algorithms.

- `__extern char const * caca\_get\_dither\_algorithm (caca\_dither\_t const *)`

Get current dithering algorithm.

- `__extern int caca\_dither\_bitmap (caca\_canvas\_t *, int, int, int, int, caca\_dither\_t const *, void *)`

Dither a bitmap on the canvas.

- `__extern int caca\_free\_dither (caca\_dither\_t *)`

Free the memory associated with a dither.

- `__extern caca\_font\_t * caca\_load\_font (void const *, size_t)`

Load a font from memory for future use.

- `__extern char const *const * caca\_get\_font\_list (void)`

Get available builtin fonts.

- `__extern int caca\_get\_font\_width (caca\_font\_t const *)`

Get a font's standard glyph width.

- `__extern int caca\_get\_font\_height (caca\_font\_t const *)`

Get a font's standard glyph height.

- `__extern uint32_t const * caca\_get\_font\_blocks (caca\_font\_t const *)`

Get a font's list of supported glyphs.

- `__extern int caca\_render\_canvas (caca\_canvas\_t const *, caca\_font\_t const *, void *, int, int, int)`

Render the canvas onto an image buffer.

- `__extern int caca\_free\_font (caca\_font\_t *)`

Free a font structure.

- `__extern int caca\_canvas\_set\_figfont (caca\_canvas\_t *, char const *)`

- `__extern int caca\_put\_figchar (caca\_canvas\_t *, uint32_t)`

- `__extern int caca\_flush\_figlet (caca\_canvas\_t *)`

- `__extern caca\_file\_t * caca\_file\_open (char const *, const char *)`

- `__extern int caca\_file\_close (caca\_file\_t *)`

- `__extern uint64_t caca\_file\_tell (caca\_file\_t *)`

- `__extern size_t caca\_file\_read (caca\_file\_t *, void *, size_t)`

- `__extern size_t caca\_file\_write (caca\_file\_t *, const void *, size_t)`

- `__extern char * caca\_file\_gets (caca\_file\_t *, char *, int)`

- `__extern int caca\_file\_eof (caca\_file\_t *)`

- `__extern ssize_t caca\_import\_memory (caca\_canvas\_t *, void const *, size_t, char const *)`

- `__extern ssize_t caca\_import\_file (caca\_canvas\_t *, char const *, char const *)`

pt3em

- `__extern char const *const * caca_get_import_list (void)`
- `__extern void * caca_export_memory (caca_canvas_t const *, char const *, size_t *)`
- `__extern char const *const * caca_get_export_list (void)`
- `__extern caca_display_t * caca_create_display (caca_canvas_t *)`
Attach a caca graphical context to a caca canvas.
- `__extern caca_display_t * caca_create_display_with_driver (caca_canvas_t *, char const *)`
Attach a specific caca graphical context to a caca canvas.
- `__extern char const *const * caca_get_display_driver_list (void)`
Get available display drivers.
- `__extern char const * caca_get_display_driver (caca_display_t *)`
Return a caca graphical context's current output driver.
- `__extern int caca_set_display_driver (caca_display_t *, char const *)`
Set the output driver.
- `__extern int caca_free_display (caca_display_t *)`
Detach a caca graphical context from a caca backend context.
- `__extern caca_canvas_t * caca_get_canvas (caca_display_t *)`
Get the canvas attached to a caca graphical context.
- `__extern int caca_refresh_display (caca_display_t *)`
Flush pending changes and redraw the screen.
- `__extern int caca_set_display_time (caca_display_t *, int)`
Set the refresh delay.
- `__extern int caca_get_display_time (caca_display_t const *)`
Get the display's average rendering time.
- `__extern int caca_get_display_width (caca_display_t const *)`
Get the display width.
- `__extern int caca_get_display_height (caca_display_t const *)`
Get the display height.
- `__extern int caca_set_display_title (caca_display_t *, char const *)`
Set the display title.
- `__extern int caca_set_mouse (caca_display_t *, int)`
Show or hide the mouse pointer.
- `__extern int caca_set_cursor (caca_display_t *, int)`
Show or hide the cursor.
- `__extern int caca_get_event (caca_display_t *, int, caca_event_t *, int)`
Get the next mouse or keyboard input event.
- `__extern int caca_get_mouse_x (caca_display_t const *)`
Return the X mouse coordinate.
- `__extern int caca_get_mouse_y (caca_display_t const *)`
Return the Y mouse coordinate.
- `__extern enum caca_event_type caca_get_event_type (caca_event_t const *)`
Return an event's type.

pt3em

- `__extern int caca_get_event_key_ch (caca_event_t const *)`
Return a key press or key release event's value.
- `__extern uint32_t caca_get_event_key_utf32 (caca_event_t const *)`
Return a key press or key release event's Unicode value.
- `__extern int caca_get_event_key_utf8 (caca_event_t const *, char *)`
Return a key press or key release event's UTF-8 value.
- `__extern int caca_get_event_mouse_button (caca_event_t const *)`
Return a mouse press or mouse release event's button.
- `__extern int caca_get_event_mouse_x (caca_event_t const *)`
Return a mouse motion event's X coordinate.
- `__extern int caca_get_event_mouse_y (caca_event_t const *)`
Return a mouse motion event's Y coordinate.
- `__extern int caca_get_event_resize_width (caca_event_t const *)`
Return a resize event's display width value.
- `__extern int caca_get_event_resize_height (caca_event_t const *)`
Return a resize event's display height value.

14.1.1 Detailed Description

pt3emVersion

\$Id\$

pt3emAuthor

Sam Hocevar <sam@zoy.org> This header contains the public types and functions that applications using *libcaca* may use.

14.1.2 Define Documentation

14.1.2.1 #define CACA_API_VERSION_1

libcaca API version

14.1.3 Typedef Documentation

14.1.3.1 typedef struct caca_canvas caca_canvas_t

libcaca canvas

14.1.3.2 typedef struct caca_dither caca_dither_t

dither structure

pt3em**14.1.3.3 typedef struct caca_font caca_font_t**

font structure

14.1.3.4 typedef struct caca_file caca_file_t

file handle structure

14.1.3.5 typedef struct caca_display caca_display_t

libcaca display context

14.1.3.6 typedef struct caca_event caca_event_t

libcaca event structure

14.1.4 Enumeration Type Documentation**14.1.4.1 enum caca_event_type**

This enum serves two purposes:

- Build listening masks for [caca_get_event\(\)](#).
- Define the type of a *caca_event_t*.

pt3emEnumerator:

CACA_EVENT_NONE No event.

CACA_EVENT_KEY_PRESS A key was pressed.

CACA_EVENT_KEY_RELEASE A key was released.

CACA_EVENT_MOUSE_PRESS A mouse button was pressed.

CACA_EVENT_MOUSE_RELEASE A mouse button was released.

CACA_EVENT_MOUSE_MOTION The mouse was moved.

CACA_EVENT_RESIZE The window was resized.

CACA_EVENT_QUIT The user requested to quit.

CACA_EVENT_ANY Bitmask for any event.

14.1.4.2 enum caca_key

Special key values returned by [caca_get_event\(\)](#) for which there is no printable ASCII equivalent.

pt3emEnumerator:

CACA_KEY_UNKNOWN Unknown key.

pt3em

CACA_KEY_CTRL_A The Ctrl-A key.
CACA_KEY_CTRL_B The Ctrl-B key.
CACA_KEY_CTRL_C The Ctrl-C key.
CACA_KEY_CTRL_D The Ctrl-D key.
CACA_KEY_CTRL_E The Ctrl-E key.
CACA_KEY_CTRL_F The Ctrl-F key.
CACA_KEY_CTRL_G The Ctrl-G key.
CACA_KEY_BACKSPACE The backspace key.
CACA_KEY_TAB The tabulation key.
CACA_KEY_CTRL_J The Ctrl-J key.
CACA_KEY_CTRL_K The Ctrl-K key.
CACA_KEY_CTRL_L The Ctrl-L key.
CACA_KEY_RETURN The return key.
CACA_KEY_CTRL_N The Ctrl-N key.
CACA_KEY_CTRL_O The Ctrl-O key.
CACA_KEY_CTRL_P The Ctrl-P key.
CACA_KEY_CTRL_Q The Ctrl-Q key.
CACA_KEY_CTRL_R The Ctrl-R key.
CACA_KEY_PAUSE The pause key.
CACA_KEY_CTRL_T The Ctrl-T key.
CACA_KEY_CTRL_U The Ctrl-U key.
CACA_KEY_CTRL_V The Ctrl-V key.
CACA_KEY_CTRL_W The Ctrl-W key.
CACA_KEY_CTRL_X The Ctrl-X key.
CACA_KEY_CTRL_Y The Ctrl-Y key.
CACA_KEY_CTRL_Z The Ctrl-Z key.
CACA_KEY_ESCAPE The escape key.
CACA_KEY_DELETE The delete key.
CACA_KEY_UP The up arrow key.
CACA_KEY_DOWN The down arrow key.
CACA_KEY_LEFT The left arrow key.
CACA_KEY_RIGHT The right arrow key.
CACA_KEY_INSERT The insert key.
CACA_KEY_HOME The home key.
CACA_KEY_END The end key.
CACA_KEY_PAGEUP The page up key.
CACA_KEY_PAGEDOWN The page down key.

pt3em

CACA_KEY_F1 The F1 key.
CACA_KEY_F2 The F2 key.
CACA_KEY_F3 The F3 key.
CACA_KEY_F4 The F4 key.
CACA_KEY_F5 The F5 key.
CACA_KEY_F6 The F6 key.
CACA_KEY_F7 The F7 key.
CACA_KEY_F8 The F8 key.
CACA_KEY_F9 The F9 key.
CACA_KEY_F10 The F10 key.
CACA_KEY_F11 The F11 key.
CACA_KEY_F12 The F12 key.
CACA_KEY_F13 The F13 key.
CACA_KEY_F14 The F14 key.
CACA_KEY_F15 The F15 key.