

libcaca
0.99.beta19

Generated by Doxygen 1.7.6.1

Tue May 15 2018 11:02:09

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 0.99.beta18 and 0.99.beta17	3
3.2	Changes between 0.99.beta17 and 0.99.beta16	3
3.3	Changes between 0.99.beta16 and 0.99.beta15	4
3.4	Changes between 0.99.beta15 and 0.99.beta14	4
3.5	Changes between 0.99.beta14 and 0.99.beta13	4
3.6	Changes between 0.99.beta13 and 0.99.beta12	4
3.7	Changes between 0.99.beta12 and 0.99.beta11	5
3.8	Changes between 0.99.beta11 and 0.99.beta10	5
3.9	Changes between 0.99.beta10 and 0.99.beta9	5
3.10	Changes between 0.99.beta9 and 0.99.beta8	5
3.11	Changes between 0.99.beta8 and 0.99.beta7	6
3.12	Changes between 0.99.beta7 and 0.99.beta6	6
3.13	Changes between 0.99.beta6 and 0.99.beta5	6
3.14	Changes between 0.99.beta5 and 0.99.beta4	6
3.15	Changes between 0.99.beta4 and 0.99.beta3	6
3.16	Changes between 0.99.beta3 and 0.99.beta2	7
3.17	Changes between 0.99.beta2 and 0.99.beta1	7
3.18	Changes between 0.9 and 0.99.beta1	7
3.19	Changes between 0.8 and 0.9	8
3.20	Changes between 0.7 and 0.8	8
3.21	Changes between 0.6 and 0.7	8

pt3em	
3.22 Changes between 0.5 and 0.6	8
3.23 Changes between 0.4 and 0.5	8
3.24 Changes between 0.3 and 0.4	9
3.25 Changes between 0.2 and 0.3	9
3.26 Changes between 0.1 and 0.2	9
3.27 New in 0.1	9
4 Libcaca thanks	10
4.1 Bugfixes and improvements	10
4.2 Reused code	10
4.3 Porters and packagers	10
5 The libcaca canvas format (version 1)	10
6 The libcaca font format (version 1)	11
7 Migrating from libcaca 0.x to the 1.0 API	12
7.1 Overview	13
7.2 Migration strategy	13
7.3 Function equivalence list	14
7.3.1 Basic functions	14
7.3.2 Event handling	15
7.3.3 Character printing	15
7.3.4 Primitives drawing	15
7.3.5 Mathematical functions	16
7.3.6 Sprite handling	16
7.3.7 Bitmap handling	17
7.4 Compilation	17
8 Libcaca coding style	17
8.1 General guidelines	17
8.2 C coding style	18
8.3 C++ coding style	19
9 A libcaca tutorial	19

pt3em

10 Libcaca environment variables	20
11 Libcaca ruby bindings	21
11.1 Libcaca Ruby API	21
11.1.1 Classes	21
11.1.2 Samples	23
12 Module Documentation	24
12.1 libcaca attribute definitions	24
12.1.1 Detailed Description	24
12.1.2 Enumeration Type Documentation	24
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 dirty rectangle manipulation	42
12.4.1 Detailed Description	42
12.4.2 Function Documentation	42
12.5 libcaca canvas transformation	47
12.5.1 Detailed Description	47
12.5.2 Function Documentation	47
12.6 libcaca attribute conversions	53
12.6.1 Detailed Description	53
12.6.2 Function Documentation	53
12.7 libcaca character set conversions	62
12.7.1 Detailed Description	62
12.7.2 Function Documentation	62
12.8 libcaca primitives drawing	66
12.8.1 Detailed Description	67
12.8.2 Function Documentation	67

pt3em

12.9 libcacaca canvas frame handling	75
12.9.1 Detailed Description	75
12.9.2 Function Documentation	75
12.10 libcacaca bitmap dithering	80
12.10.1 Detailed Description	81
12.10.2 Function Documentation	81
12.11 libcacaca character font handling	92
12.11.1 Detailed Description	92
12.12 libcacaca bitmap font handling	94
12.12.1 Detailed Description	94
12.12.2 Function Documentation	94
12.13 libcacaca FIGfont handling	99
12.13.1 Detailed Description	99
12.14 libcacaca file IO	101
12.14.1 Detailed Description	101
12.14.2 Function Documentation	101
12.15 libcacaca importers/exporters from/to various	106
12.15.1 Detailed Description	106
12.15.2 Function Documentation	106
12.16 libcacaca display functions	114
12.16.1 Detailed Description	114
12.16.2 Function Documentation	115
12.17 libcacaca event handling	123
12.17.1 Detailed Description	123
12.17.2 Function Documentation	123
12.18 libcacaca process management	130
12.18.1 Detailed Description	130
12.19 libcacaca DOS conio.h compatibility layer	132
12.19.1 Detailed Description	133
13 Data Structure Documentation	134
13.1 caca_conio_text_info Struct Reference	134
13.1.1 Detailed Description	134

pt3em

13.1.2	Field Documentation	134
13.2	caca_event Struct Reference	135
13.2.1	Detailed Description	136
13.2.2	Field Documentation	136
13.3	caca_option Struct Reference	136
13.3.1	Detailed Description	136
14	File Documentation	136
14.1	caca.h File Reference	136
14.1.1	Detailed Description	148
14.1.2	Define Documentation	148
14.1.3	Typedef Documentation	149
14.1.4	Enumeration Type Documentation	149

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 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:

pt3em

- [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](#)

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://www.wtfpl.net/>.

2 Libcaca authors

Sam Hocevar <sam@hocevar.net>

- main programmer

Jean-Yves Lamoureux <jylam@lrxscene.org>

- cacaball

pt3em

- OpenGL driver
- 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

Nicolas Vion <nico@picapo.net>

- PHP bindings

Adrien Grand <jpountz@dinauz.org>

- Java bindings

Alex Foulon <alxf@lavabit.com>

- Python bindings

3 Libcaca news

3.1 Changes between 0.99.beta18 and 0.99.beta17

- new "cacaclock" utility
- improved FIGlet support
- improved Python and Ruby bindings
- numerous improvements and bugfixes

pt3em

3.2 Changes between 0.99.beta17 and 0.99.beta16

- new dirty rectangle framework for accelerated rendering
- new <conio.h> compatible header for old DOS software porting
- XCode and Visual Studio build improvements
- Unicode support in the X11 driver
- triangle texture mapping
- improved HTML output
- new troff output
- improved Ruby and .NET bindings
- new comprehensive PHP bindings
- new Java bindings
- Cocoa driver enhancements
- support for ANSI colour in FIGlet and TOLlet fonts
- documentation updates
- numerous improvements and bugfixes

3.3 Changes between 0.99.beta16 and 0.99.beta15

- many build fixes, especially for nonstandard platforms

3.4 Changes between 0.99.beta15 and 0.99.beta14

- libcucul was merged back into libcaca for more clarity

3.5 Changes between 0.99.beta14 and 0.99.beta13

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

pt3em

3.6 Changes between 0.99.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

3.7 Changes between 0.99.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

3.8 Changes between 0.99.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

pt3em

3.9 Changes between 0.99.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

3.10 Changes between 0.99.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

3.11 Changes between 0.99.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

3.12 Changes between 0.99.beta7 and 0.99.beta6

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

pt3em

3.13 Changes between 0.99.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

3.14 Changes between 0.99.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

3.15 Changes between 0.99.beta4 and 0.99.beta3

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

3.16 Changes between 0.99.beta3 and 0.99.beta2

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

3.17 Changes between 0.99.beta2 and 0.99.beta1

- ANSI importer
- functions use errno for error reporting
- updated C++ bindings
- .NET bindings

pt3em

- 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

3.18 Changes between 0.9 and 0.99.beta1

- license switched to WTFPL
- libcaca was split into libcucul, 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

3.19 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

3.20 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

3.21 Changes between 0.6 and 0.7

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

3.22 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

3.23 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

3.24 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

3.25 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

3.26 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

3.27 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@gmail.com> - win32 speed improvements
- Bastian Märkisch <bmaerkisch@web.de> - bugfixes and win32 driver 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
- Simon Huggins <webmaster::simonhuggins.com> - conio-snake

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
                        //          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)

```

```

pt3em
    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

    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(NULL); 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:

- Most functions now take an object handle as their first argument.

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>

The modified code is guaranteed to build both with *libcaca* 0.x and *libcaca* 1.0.

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. Alternatively, [caca_create_display\(\)](#) with a NULL argument will create a canvas automatically.
- **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. Alternatively, if the canvas was created by [caca_create_display\(\)](#), it will be automatically destroyed by [caca_free_display\(\)](#).

pt3em

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.

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()**: unchanged.
- **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()**: unchanged.
- **caca_draw_polyline()**: unchanged.
- **caca_draw_thin_line()**: unchanged.
- **caca_draw_thin_polyline()**: unchanged.
- **caca_draw_circle()**: unchanged.
- **caca_draw_ellipse()**: unchanged.
- **caca_draw_thin_ellipse()**: unchanged.
- **caca_fill_ellipse()**: unchanged.

pt3em

- **caca_draw_box()**: unchanged, but the argument meaning changed (width and height instead of corner coordinates).
- **caca_draw_thin_box()**: use [caca_draw_thin_box\(\)](#) or [caca_draw_cp437_box\(\)](#), also the argument meaning changed (width and height instead of corner coordinates).
- **caca_fill_box()**: unchanged, but the argument meaning changed (width and height instead of corner coordinates).
- **caca_draw_triangle()**: unchanged.
- **caca_draw_thin_triangle()**: unchanged.
- **caca_fill_triangle()**: unchanged.

7.3.5 Mathematical functions

- **caca_rand()**: unchanged, but 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\(\)](#).

pt3em

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()`.

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

pt3em

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:

```
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);
```

A space can be inserted after keywords such as `for`, `while` or `if`, but consistency with the rest of the page is encouraged:

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

if (p == NULL)
```

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;
    }
}
```



```
pt3em
    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.

```
#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.

pt3em

- 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" onto the canvas, using the current colour pair.
- Refresh the display, causing the text to be effectively displayed.
- 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 commands (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 it. These variables are:

- **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 is 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.

pt3em

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

11.1.1 Classes

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)
- **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**

pt3em

- **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",
"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_to_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_from_memory", "invert", "printf", "put_attr", "put_char", "put_str",
"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", "wherex", "wherey", "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"]
```

pt3em

```
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"]
```

11.1.2 Samples

```
$ 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_to_memory("irc")'
###
#x#
###
```

```
$ ruby -e 'puts Caca::Canvas.new(6,3).draw_thin_polyline([[0,0], [0,2],
[5,2],[0,0]]).export_to_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"]]
```

```
$ 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
```

```
pt3em
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

Enumerations

- enum `caca_color` { `CACA_BLACK` = 0x00, `CACA_BLUE` = 0x01, `CACA_GREEN` = 0x02, `CACA_CYAN` = 0x03, `CACA_RED` = 0x04, `CACA_MAGENTA` = 0x05, `CACA_BROWN` = 0x06, `CACA_LIGHTGRAY` = 0x07, `CACA_DARKGRAY` = 0x08, `CACA_LIGHTBLUE` = 0x09, `CACA_LIGHTGREEN` = 0x0a, `CACA_LIGHTCYAN` = 0x0b, `CACA_LIGHTRED` = 0x0c, `CACA_LIGHTMAGENTA` = 0x0d, `CACA_YELLOW` = 0x0e, `CACA_WHITE` = 0x0f, `CACA_DEFAULT` = 0x10, `CACA_TRANSPARENT` = 0x20 }
- enum `caca_style` { `CACA_BOLD` = 0x01, `CACA_ITALICS` = 0x02, `CACA_UNDERLINE` = 0x04, `CACA_BLINK` = 0x08 }

12.1.1 Detailed Description

Colours and styles that can be used with `caca_set_attr()`.

12.1.2 Enumeration Type Documentation

12.1.2.1 enum `caca_color`

libcaca colour keyword

pt3emEnumerator:

CACA_BLACK The colour index for black.

CACA_BLUE The colour index for blue.

CACA_GREEN The colour index for green.

CACA_CYAN The colour index for cyan.

CACA_RED The colour index for red.

CACA_MAGENTA The colour index for magenta.

pt3em

CACA_BROWN The colour index for brown.
CACA_LIGHTGRAY The colour index for light gray.
CACA_DARKGRAY The colour index for dark gray.
CACA_LIGHTBLUE The colour index for blue.
CACA_LIGHTGREEN The colour index for light green.
CACA_LIGHTCYAN The colour index for light cyan.
CACA_LIGHTRED The colour index for light red.
CACA_LIGHTMAGENTA The colour index for light magenta.
CACA_YELLOW The colour index for yellow.
CACA_WHITE The colour index for white.
CACA_DEFAULT The output driver's default colour.
CACA_TRANSPARENT The transparent colour.

12.1.2.2 enum `caca_style`

libcaca style keyword

pt3emEnumerator:

CACA_BOLD The style mask for bold.
CACA_ITALICS The style mask for italics.
CACA_UNDERLINE The style mask for underline.
CACA_BLINK The style mask for blink.

pt3em

12.2 libcaca basic functions

Functions

- `caca_canvas_t * caca_create_canvas (int, int)`
Initialise a libcaca canvas.
- `int caca_manage_canvas (caca_canvas_t *, int(*) (void *), void *)`
Manage a canvas.
- `int caca_unmanage_canvas (caca_canvas_t *, int(*) (void *), void *)`
unmanage a canvas.
- `int caca_set_canvas_size (caca_canvas_t *, int, int)`
Resize a canvas.
- `int caca_get_canvas_width (caca_canvas_t const *)`
Get the canvas width.
- `int caca_get_canvas_height (caca_canvas_t const *)`
Get the canvas height.
- `uint32_t const * caca_get_canvas_chars (caca_canvas_t const *)`
Get the canvas character array.
- `uint32_t const * caca_get_canvas_attrs (caca_canvas_t const *)`
Get the canvas attribute array.
- `int caca_free_canvas (caca_canvas_t *)`
Free a libcaca canvas.
- `int caca_rand (int, int)`
- `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 `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_conio_movetext()`, `caca_create_display_with_driver()`, `caca_export_area_to_memory()`, `caca_import_area_from_file()`, `caca_import_area_from_memory()`, and `caca_set_canvas_boundaries()`.

12.2.2.2 `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

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

pt3empt3emcv	pt3emA libcaca canvas.
pt3emwidth	pt3emThe desired canvas width.
pt3emheight	pt3emThe desired canvas height.

pt3emReturns

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

Referenced by `caca_canvas_set_figfont()`, `caca_flush_figlet()`, and `caca_put_figchar()`.

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

Return the current canvas' width, in character cells.

This function never fails.

pt3emParameters

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

pt3emReturns

The canvas width.

Referenced by `caca_conio_cleol()`, `caca_conio_movetext()`, and `caca_get_mouse_x()`.

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

Returns the current canvas' height, in character cells.

This function never fails.

pt3emParameters

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

pt3em

pt3emReturns

The canvas height.

Referenced by [caca_flush_figlet\(\)](#), and [caca_get_mouse_y\(\)](#).

12.2.2.7 `uint32_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 probably only useful for *libcaca* 's internal display drivers.

This function never fails.

pt3emParameters

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

pt3emReturns

The canvas character array.

12.2.2.8 `uint32_t const* caca_get_canvas_attrs (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 probably only useful for *libcaca* 's internal display drivers.

This function never fails.

pt3emParameters

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

pt3emReturns

The canvas attribute array.

12.2.2.9 `int caca_free_canvas (caca_canvas_t * cv)`

Free all resources allocated by [caca_create_canvas\(\)](#). The canvas pointer becomes invalid and must no longer be used unless a new call to [caca_create_canvas\(\)](#) is made.

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.

pt3em

pt3emParameters

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

pt3emReturns

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

References caca_canvas_set_figfont().

Referenced by caca_canvas_set_figfont(), caca_create_display_with_driver(), caca_export_area_to_memory(), caca_free_display(), caca_import_area_from_file(), and caca_import_area_from_memory().

12.2.2.10 char const* **caca_get_version**(void)

Return a read-only string with the *libcaca* version information.

This function never fails.

pt3emReturns

The *libcaca* version information.

pt3em

pt3em

12.3 libcaca canvas drawing

Defines

- #define `CACA_MAGIC_FULLWIDTH` 0x000ffffe

Functions

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

pt3empt3emcv	pt3emA handle to the libcaca canvas.
pt3emx	pt3emX cursor coordinate.
pt3emy	pt3emY cursor coordinate.

pt3emReturns

This function always returns 0.

Referenced by `caca_conio_cgets()`, `caca_conio_clrscr()`, `caca_conio_cprintf()`, `caca_conio_cputs()`, `caca_conio_gotoxy()`, `caca_conio_printf()`, and `caca_conio_putch()`.

12.3.3.2 int caca_wherex (caca_canvas_t const * cv)

Retrieve the X coordinate of the cursor's position.

This function never fails.

pt3emParameters

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

pt3em

pt3emReturns

The cursor's X coordinate.

Referenced by `caca_conio_cgets()`, `caca_conio_cleol()`, `caca_conio_cprintf()`, `caca_conio_cputs()`, `caca_conio_printf()`, `caca_conio_putch()`, and `caca_conio_wherex()`.

12.3.3.3 int caca_wherey (caca_canvas_t const * cv)

Retrieve the Y coordinate of the cursor's position.

This function never fails.

pt3emParameters

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

pt3emReturns

The cursor's Y coordinate.

Referenced by `caca_conio_cgets()`, `caca_conio_cleol()`, `caca_conio_cprintf()`, `caca_conio_cputs()`, `caca_conio_printf()`, `caca_conio_putch()`, and `caca_conio_wherey()`.

12.3.3.4 int caca_put_char (caca_canvas_t * cv, int x, int y, uint32_t ch)

Print an ASCII or Unicode character at the given coordinates, using the default foreground and background colour values.

If the coordinates are outside the canvas boundaries, nothing is printed. If a fullwidth Unicode character gets overwritten, its remaining visible parts are replaced with spaces. If the canvas' boundaries would split the fullwidth character in two, a space is printed instead.

The behaviour when printing non-printable characters or invalid UTF-32 characters is undefined. To print a sequence of bytes forming an UTF-8 character instead of an UTF-32 character, use the `caca_put_str()` function.

This function returns the width of the printed character. If it is a fullwidth character, 2 is returned. Otherwise, 1 is returned.

This function never fails.

pt3emParameters

pt3em	pt3emcv	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.
	pt3emch	pt3emThe character to print.

pt3em

pt3emReturns

The width of the printed character: 2 for a fullwidth character, 1 otherwise.

References `caca_add_dirty_rect()`, `CACA_MAGIC_FULLWIDTH`, and `caca_utf32_is_fullwidth()`.

Referenced by `caca_conio_cgets()`, `caca_conio_cputs()`, `caca_conio_putch()`, `caca_dither_bitmap()`, `caca_fill_box()`, `caca_fill_triangle()`, `caca_flush_figlet()`, `caca_put_figchar()`, and `caca_put_str()`.

12.3.3.5 `uint32_t caca_get_char (caca_canvas_t const * cv, int x, int y)`

Get the ASCII or Unicode value of the character at the given coordinates. If the value is less or equal to 127 (0x7f), the character can be printed as ASCII. Otherwise, it must be handled as a UTF-32 value.

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

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	pt3emcv	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.

pt3emReturns

The Unicode character at the given coordinates.

Referenced by `caca_flush_figlet()`, and `caca_put_figchar()`.

12.3.3.6 `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 returns the number of cells printed by the string. It is not the number of characters printed, because fullwidth characters account for two cells.

This function never fails.

pt3em

pt3emParameters

pt3em	pt3em
pt3emcv	pt3emA handle to the libcaca canvas.
pt3emx	pt3emX coordinate.
pt3emy	pt3emY coordinate.
pt3ems	pt3emThe string to print.

pt3emReturns

The number of cells printed.

References `caca_put_char()`, `caca_utf32_is_fullwidth()`, and `caca_utf8_to_utf32()`.

Referenced by `caca_vprintf()`.

12.3.3.7 `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.

This function returns the number of cells printed by the string. It is not the number of characters printed, because fullwidth characters account for two cells.

This function never fails.

pt3emParameters

pt3em	pt3em
pt3emcv	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

The number of cells printed.

References `caca_vprintf()`.

12.3.3.8 `int caca_vprintf (caca_canvas_t * cv, int x, int y, char const * format, va_list args)`

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 X 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 `vprintf()` function.

This function never fails.

pt3em	pt3emA handle to the libcaca canvas.
pt3emx	pt3emX coordinate.
pt3emy	pt3emY coordinate.
pt3emformat	pt3emThe format string to print.
pt3emargs	pt3emA va_list containing the arguments to the format string.

This function never fails.

pt3em	pt3em	pt3em	The canvas to clear.
-------	-------	-------	----------------------

This function never fails.

pt3emcv	pt3emA handle to the libcaca canvas.
pt3emx	pt3emX handle coordinate.
pt3emy	pt3emY handle coordinate.

pt3em

pt3emReturns

This function always returns 0.

Referenced by caca_put_figchar().

12.3.3.11 int caca_get_canvas_handle_x (caca_canvas_t const * cv)

Retrieve the X coordinate of the canvas' handle.

This function never fails.

pt3emParameters

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

pt3emReturns

The canvas' handle's X coordinate.

12.3.3.12 int caca_get_canvas_handle_y (caca_canvas_t const * cv)

Retrieve the Y coordinate of the canvas' handle.

This function never fails.

pt3emParameters

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

pt3emReturns

The canvas' handle's Y coordinate.

12.3.3.13 int caca_blit (caca_canvas_t * dst, int x, int y, caca_canvas_t const * src, caca_canvas_t const * mask)

Blit a canvas onto another one at the given coordinates. An optional mask canvas can be used.

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

pt3emParameters

pt3em	pt3em	pt3emThe destination canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.
	pt3emsrc	pt3emThe source canvas.
	pt3emmask	pt3emThe mask canvas.

pt3emReturns

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

References `caca_add_dirty_rect()`, and `CACA_MAGIC_FULLWIDTH`.

Referenced by `caca_conio_movetext()`, `caca_export_area_to_memory()`, `caca_import_area_from_file()`, `caca_import_area_from_memory()`, `caca_put_figchar()`, and `caca_set_canvas_boundaries()`.

12.3.3.14 `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

pt3em	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_add_dirty_rect()`, `caca_blit()`, `caca_create_canvas()`, `caca_create_frame()`, `caca_get_frame_count()`, and `caca_set_frame()`.

pt3em

pt3em

12.4 libcaca dirty rectangle manipulation

Functions

- int [caca_disable_dirty_rect](#) ([caca_canvas_t](#) *)
Disable dirty rectangles.
- int [caca_enable_dirty_rect](#) ([caca_canvas_t](#) *)
Enable dirty rectangles.
- int [caca_get_dirty_rect_count](#) ([caca_canvas_t](#) *)
Get the number of dirty rectangles in the canvas.
- int [caca_get_dirty_rect](#) ([caca_canvas_t](#) *, int, int *, int *, int *, int *)
Get a canvas's dirty rectangle.
- int [caca_add_dirty_rect](#) ([caca_canvas_t](#) *, int, int, int, int)
Add an area to the canvas's dirty rectangle list.
- int [caca_remove_dirty_rect](#) ([caca_canvas_t](#) *, int, int, int, int)
Remove an area from the dirty rectangle list.
- int [caca_clear_dirty_rect_list](#) ([caca_canvas_t](#) *)
Clear a canvas's dirty rectangle list.

12.4.1 Detailed Description

These functions manipulate dirty rectangles for optimised blitting.

12.4.2 Function Documentation

12.4.2.1 int [caca_disable_dirty_rect](#) ([caca_canvas_t](#) * cv)

Disable dirty rectangle handling for all *libcaca* graphic calls. This is handy when the calling application needs to do slow operations within a known area. Just call [caca_add_dirty_rect\(\)](#) afterwards.

This function is recursive. Dirty rectangles are only reenabled when [caca_enable_dirty_rect\(\)](#) is called as many times.

This function never fails.

Parameters

pt3em	pt3emcv	pt3emA <i>libcaca</i> canvas.
-------	---------	-------------------------------

Returns

This function always returns 0.

12.4.2.2 int **caca_enable_dirty_rect**(**caca_canvas_t** * *cv*)

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

- `EINVAL` Dirty rectangles were not disabled.

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

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

Get the number of dirty rectangles in a canvas. Dirty rectangles are areas that contain cells that have changed since the last reset.

The dirty rectangles are used internally by display drivers to optimise rendering by avoiding to redraw the whole screen. Once the display driver has rendered the canvas, it resets the dirty rectangle list.

Dirty rectangles are guaranteed not to overlap.

This function never fails.

pt3empt3emcv	pt3emA libcaca canvas.
--------------	------------------------

The number of dirty rectangles in the given canvas.

Get the canvas's given dirty rectangle coordinates. The index must be within the dirty rectangle count. See [caca_get_dirty_rect_count\(\)](#) for how to compute this count.

If an error occurs, no coordinates are written in the pointer arguments, -1 is returned and **errno** is set accordingly:

- `EINVAL` Specified rectangle index is out of bounds.

pt3em

pt3emParameters

pt3em	pt3em	pt3emA libcaca canvas.
	pt3emr	pt3emThe requested rectangle index.
	pt3emx	pt3emA pointer to an integer where the leftmost edge of the dirty rectangle will be stored.
	pt3emy	pt3emA pointer to an integer where the topmost edge of the dirty rectangle will be stored.
	pt3emwidth	pt3emA pointer to an integer where the width of the dirty rectangle will be stored.
	pt3emheight	pt3emA pointer to an integer where the height of the dirty rectangle will be stored.

pt3emReturns

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

12.4.2.5 `int caca_add_dirty_rect (caca_canvas_t * cv, int x, int y, int width, int height)`

Add an invalidating zone to the canvas's dirty rectangle list. For more information about the dirty rectangles, see [caca_get_dirty_rect\(\)](#).

This function may be useful to force refresh of a given zone of the canvas even if the dirty rectangle tracking indicates that it is unchanged. This may happen if the canvas contents were somewhat directly modified.

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

- `EINVAL` Specified rectangle coordinates are out of bounds.

pt3emParameters

pt3em	pt3em	pt3emA libcaca canvas.
	pt3emx	pt3emThe leftmost edge of the additional dirty rectangle.
	pt3emy	pt3emThe topmost edge of the additional dirty rectangle.
	pt3emwidth	pt3emThe width of the additional dirty rectangle.
	pt3emheight	pt3emThe height of the additional dirty rectangle.

pt3emReturns

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

Referenced by `caca_blit()`, `caca_clear_canvas()`, `caca_fill_box()`, `caca_flip()`, `caca_flop()`, `caca_free_frame()`, `caca_invert()`, `caca_put_attr()`, `caca_put_char()`, `caca_rotate_180()`, `caca_rotate_left()`, `caca_rotate_right()`, `caca_set_canvas_boundaries()`, `caca_set_frame()`, `caca_stretch_left()`, and `caca_stretch_right()`.

12.4.2.6 `int cacaca_remove_dirty_rect (cacaca_canvas_t * cv, int x, int y, int width, int height)`

Mark a cell area in the canvas as not dirty. For more information about the dirty rectangles, see [cacaca_get_dirty_rect\(\)](#).

Values such that **xmin** > **xmax** or **ymin** > **ymax** indicate that the dirty rectangle is empty. They will be silently ignored.

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

- **EINVAL** Specified rectangle coordinates are out of bounds.

Parameters

<code>cv</code>	A libcacaca canvas.
<code>x</code>	The leftmost edge of the clean rectangle.
<code>y</code>	The topmost edge of the clean rectangle.
<code>width</code>	The width of the clean rectangle.
<code>height</code>	The height of the clean rectangle.

Returns

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

12.4.2.7 `int cacaca_clear_dirty_rect_list (cacaca_canvas_t * cv)`

Empty the canvas's dirty rectangle list.

This function never fails.

Parameters

<code>cv</code>	A libcacaca canvas.
-----------------	---------------------

Returns

This function always returns 0.

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

pt3em

pt3em

12.5 libcaca canvas transformation

Functions

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

12.5.1 Detailed Description

These functions perform horizontal and vertical canvas flipping.

12.5.2 Function Documentation

12.5.2.1 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.

References `caca_add_dirty_rect()`.

12.5.2.2 int caca_flip (caca_canvas t * cv)

This function never fails.

pt3empt3emcv	pt3emThe canvas to flip.
--------------	--------------------------

This function always returns 0.

References `caca` add `dirty` `rect()`, and `CACA` `MAGIC` `FULLWIDTH`.

12.5.2.3 int **caca_flop** (**caca_canvas** t * **cv**)

This function never fails.

pt3empt3emcv	pt3emThe canvas to flop.
--------------	--------------------------

This function always returns 0.

References `caca` `add` `dirty` `rect()`.

```
12.5.2.4 int caca rotate 180 ( caca canvas t * cv )
```

This function never fails.

pt3em	pt3em	cv	pt3em	The canvas to rotate.
-------	-------	----	-------	-----------------------

pt3em

pt3emReturns

This function always returns 0.

References `caca_add_dirty_rect()`, and `CACA_MAGIC_FULLWIDTH`.

12.5.2.5 `int caca_rotate_left (caca_canvas_t * cv)`

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.

pt3emParameters

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

pt3emReturns

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

References `caca_add_dirty_rect()`.

12.5.2.6 `int caca_rotate_right (caca_canvas_t * cv)`

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:

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.

References `caca_add_dirty_rect()`.

12.5.2.7 `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.

References `caca_add_dirty_rect()`.

12.5.2.8 `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,

pt3em

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	pt3em	The canvas to rotate right.
-------	---------	-------	-----------------------------

pt3emReturns

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

References `caca_add_dirty_rect()`.

pt3em

12.6

12.6 libcaca attribute conversions

Functions

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

12.6.1 Detailed Description

These functions perform conversions between attribute values.

12.6.2 Function Documentation

12.6.2.1 `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:

pt3em

- 3 bits for the background alpha
- 4 bits for the background red component
- 4 bits for the background green component
- 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	pt3emcv	pt3emA handle to the libcaca canvas.
	pt3emx	pt3emX coordinate.
	pt3emy	pt3emY coordinate.

pt3emReturns

The requested attribute.

Referenced by `caca_conio_textbackground()`, `caca_conio_textcolor()`, `caca_dither_bitmap()`, `caca_flush_figlet()`, and `caca_put_figchar()`.

12.6.2.2 `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.

pt3em

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

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em
pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

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

12.6.2.3 `int caca_unset_attr (caca_canvas_t * cv, uint32_t attr)`

Unset flags in 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 a combination (bitwise OR) of style values (`CACA_UNDERLINE`, `CACA_BLINK`, `CACA_BOLD` and `CACA_ITALICS`). Unsetting these attributes 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

pt3emReturns

This function always returns 0.

12.6.2.4 `int caca_toggle_attr (caca_canvas_t * cv, uint32_t attr)`

Toggle flags in 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 a combination (bitwise OR) of style values (`CACA_UNDERLINE`, `CACA_BLINK`, `CACA_BOLD` and `CACA_ITALICS`). Toggling these attributes does not modify the current colour information.

pt3em

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

This function never fails.

Parameters

pt3em	pt3em	pt3em
<code>pt3emcv</code>		A handle to the libcaca canvas.
<code>pt3emattr</code>		The requested attribute values to toggle.

Returns

This function always returns 0.

12.6.2.5 `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

pt3em	pt3em	pt3em
<code>pt3emcv</code>		A handle to the libcaca canvas.
<code>pt3emx</code>		X coordinate.
<code>pt3emy</code>		Y coordinate.
<code>pt3emattr</code>		The requested attribute value.

Returns

This function always returns 0.

References `caca_add_dirty_rect()`, and `CACA_MAGIC_FULLWIDTH`.

Referenced by `caca_flush_figlet()`, and `caca_put_figchar()`.

12.6.2.6 `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, -1 is returned and `errno` is set accordingly:

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

Parameters

<code>cv</code>	A handle to the libcaca canvas.
<code>fg</code>	The requested ANSI foreground colour.
<code>bg</code>	The requested ANSI background colour.

Returns

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

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

12.6.2.7 `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.

Parameters

<code>cv</code>	A handle to the libcaca canvas.
<code>fg</code>	The requested ARGB foreground colour.
<code>bg</code>	The requested ARGB background colour.

Returns

This function always returns 0.

12.6.2.8 `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 re-

placed 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.

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 DOS ANSI value.

References *CACA_BLACK*, and *CACA_LIGHTGRAY*.

12.6.2.9 `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.

Referenced by `caca_conio_textbackground()`.

12.6.2.10 `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

Parameters

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

Returns

The corresponding ANSI background value.

Referenced by `caca_conio_textcolor()`.

12.6.2.11 `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.

Parameters

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

Returns

The corresponding 12-bit RGB foreground value.

References `CACA_DEFAULT`, `CACA_LIGHTGRAY`, and `CACA_TRANSPARENT`.

12.6.2.12 `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.

Parameters

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

Returns

The corresponding 12-bit RGB background value.

References CACA_BLACK, CACA_DEFAULT, and CACA_TRANSPARENT.

12.6.2.13 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.

Parameters

<code>attr</code>	The requested attribute value.
<code>argb</code>	An array of 8-bit integers.

References CACA_BLACK, CACA_DEFAULT, CACA_LIGHTGRAY, and CACA_TRANSPARENT.

Referenced by `caca_render_canvas()`.

pt3em

12.7 libcaca character set conversions

Functions

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

12.7.1 Detailed Description

These functions perform conversions between usual character sets.

12.7.2 Function Documentation

12.7.2.1 `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.

Returns

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

Referenced by `caca_put_str()`.

12.7.2.2 `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.

Parameters

<code>buf</code>	A pointer to a character buffer where the UTF-8 sequence will be written.
<code>ch</code>	The UTF-32 character.

Returns

The number of bytes written.

12.7.2.3 `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.

Parameters

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

Returns

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

12.7.2.4 `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.

12.7.2.5 `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.

12.7.2.6 `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

12.8 libcaca primitives drawing

Functions

- int `caca_draw_line` (`caca_canvas_t *`, int, int, int, int, uint32_t)
Draw a line on the canvas using the given character.
- int `caca_draw_polyline` (`caca_canvas_t *`, int const x[], int const y[], int, uint32_t)
Draw a polyline.
- int `caca_draw_thin_line` (`caca_canvas_t *`, int, int, int, int)
Draw a thin line on the canvas, using ASCII art.
- int `caca_draw_thin_polyline` (`caca_canvas_t *`, int const x[], int const y[], int)
Draw an ASCII art thin polyline.
- int `caca_draw_circle` (`caca_canvas_t *`, int, int, int, uint32_t)
Draw a circle on the canvas using the given character.
- int `caca_draw_ellipse` (`caca_canvas_t *`, int, int, int, int, uint32_t)
Draw an ellipse on the canvas using the given character.
- int `caca_draw_thin_ellipse` (`caca_canvas_t *`, int, int, int, int)
Draw a thin ellipse on the canvas.
- int `caca_fill_ellipse` (`caca_canvas_t *`, int, int, int, int, uint32_t)
Fill an ellipse on the canvas using the given character.
- int `caca_draw_box` (`caca_canvas_t *`, int, int, int, int, uint32_t)
Draw a box on the canvas using the given character.
- int `caca_draw_thin_box` (`caca_canvas_t *`, int, int, int, int)
Draw a thin box on the canvas.
- int `caca_draw_cp437_box` (`caca_canvas_t *`, int, int, int, int)
Draw a box on the canvas using CP437 characters.
- int `caca_fill_box` (`caca_canvas_t *`, int, int, int, int, uint32_t)
Fill a box on the canvas using the given character.
- 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.
- int `caca_draw_thin_triangle` (`caca_canvas_t *`, int, int, int, int, int, int)
Draw a thin triangle on the canvas.
- 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.
- int `caca_fill_triangle_textured` (`caca_canvas_t *`cv, int coords[6], `caca_canvas_t *`tex, float uv[6])
Fill a triangle on the canvas using an arbitrary-sized texture.

12.8.1 Detailed Description

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

12.8.2 Function Documentation

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

This function never fails.

Parameters

<code>cv</code>	The handle to the libcaca canvas.
<code>x1</code>	X coordinate of the first point.
<code>y1</code>	Y coordinate of the first point.
<code>x2</code>	X coordinate of the second point.
<code>y2</code>	Y coordinate of the second point.
<code>ch</code>	UTF-32 character to be used to draw the line.

Returns

This function always returns 0.

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

12.8.2.2 `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.

Parameters

<code>cv</code>	The handle to the libcaca canvas.
<code>x</code>	Array of X coordinates. Must have <code>n + 1</code> elements.
<code>y</code>	Array of Y coordinates. Must have <code>n + 1</code> elements.
<code>n</code>	Number of lines to draw.
<code>ch</code>	UTF-32 character to be used to draw the lines.

pt3em

pt3emReturns

This function always returns 0.

12.8.2.3 `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	The handle to the libcaca canvas.
	pt3em	x1	X coordinate of the first point.
	pt3em	y1	Y coordinate of the first point.
	pt3em	x2	X coordinate of the second point.
	pt3em	y2	Y coordinate of the second point.

pt3emReturns

This function always returns 0.

Referenced by `caca_draw_thin_triangle()`.

12.8.2.4 `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	The handle to the libcaca canvas.
	pt3em	x	Array of X coordinates. Must have <code>n + 1</code> elements.
	pt3em	y	Array of Y coordinates. Must have <code>n + 1</code> elements.
	pt3em	n	Number of lines to draw.

pt3emReturns

This function always returns 0.

12.8.2.5 `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	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.

12.8.2.6 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	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.

12.8.2.7 int **caca_draw_thin_ellipse** (**caca_canvas_t** * *cv*, int *xo*, int *yo*, int *a*, int *b*)

This function never fails.

pt3emParameters

pt3em	pt3em	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.

12.8.2.8 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	pt3em	Center X coordinate.
	pt3em	pt3em	Center Y coordinate.
	pt3em	pt3em	Ellipse X radius.
	pt3em	pt3em	Ellipse Y radius.
	pt3em	pt3em	UTF-32 character to be used to fill the ellipse.

pt3emReturns

This function always returns 0.

References `caca_draw_line()`.

12.8.2.9 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	pt3em	X coordinate of the upper-left corner of the box.
	pt3em	pt3em	Y coordinate of the upper-left corner of the box.
	pt3em	pt3em	Width of the box.
	pt3em	pt3em	Height of the box.
	pt3em	pt3em	UTF-32 character to be used to draw the box.

pt3emReturns

This function always returns 0.

References `caca_draw_line()`.

12.8.2.10 int **caca_draw_thin_box** (**caca_canvas_t** * *cv*, int *x*, int *y*, int *w*, int *h*)

This function never fails.

pt3em

pt3emParameters

pt3em	pt3em
pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emx	pt3emX coordinate of the upper-left corner of the box.
pt3emy	pt3emY coordinate of the upper-left corner of the box.
pt3emw	pt3emWidth of the box.
pt3emh	pt3emHeight of the box.

pt3emReturns

This function always returns 0.

12.8.2.11 int **caca_draw_cp437_box** (**caca_canvas_t** * *cv*, int *x*, int *y*, int *w*, int *h*)

This function never fails.

pt3emParameters

pt3em	pt3em
pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emx	pt3emX coordinate of the upper-left corner of the box.
pt3emy	pt3emY coordinate of the upper-left corner of the box.
pt3emw	pt3emWidth of the box.
pt3emh	pt3emHeight of the box.

pt3emReturns

This function always returns 0.

12.8.2.12 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
pt3emcv	pt3emThe handle to the libcaca canvas.
pt3emx	pt3emX coordinate of the upper-left corner of the box.
pt3emy	pt3emY coordinate of the upper-left corner of the box.
pt3emw	pt3emWidth of the box.
pt3emh	pt3emHeight of the box.
pt3emch	pt3emUTF-32 character to be used to draw the box.

pt3emReturns

This function always returns 0.

pt3em

References `caca_add_dirty_rect()`, and `caca_put_char()`.

Referenced by `caca_conio_cleol()`.

12.8.2.13 `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.
pt3em	pt3em	x1	X coordinate of the first point.
pt3em	pt3em	y1	Y coordinate of the first point.
pt3em	pt3em	x2	X coordinate of the second point.
pt3em	pt3em	y2	Y coordinate of the second point.
pt3em	pt3em	x3	X coordinate of the third point.
pt3em	pt3em	y3	Y coordinate of the third point.
pt3em	pt3em	ch	UTF-32 character to be used to draw the triangle outline.

pt3emReturns

This function always returns 0.

References `caca_draw_line()`.

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

This function never fails.

pt3emParameters

pt3em	pt3em	pt3em	The handle to the libcaca canvas.
pt3em	pt3em	x1	X coordinate of the first point.
pt3em	pt3em	y1	Y coordinate of the first point.
pt3em	pt3em	x2	X coordinate of the second point.
pt3em	pt3em	y2	Y coordinate of the second point.
pt3em	pt3em	x3	X coordinate of the third point.
pt3em	pt3em	y3	Y coordinate of the third point.

pt3emReturns

This function always returns 0.

References `caca_draw_thin_line()`.

pt3em

```
12.8.2.15 int caca_fill_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	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
pt3em	pt3em	pt3em	pt3em

pt3emReturns

This function always returns 0.

References caca_fill_triangle(), and caca_put_char().

Referenced by caca_fill_triangle().

```
12.8.2.16 int caca_fill_triangle_textured ( caca_canvas_t * cv, int coords[6],
caca_canvas_t * tex, float uv[6] )
```

This function fails if one or both the canvas are missing

pt3emParameters

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

pt3emReturns

This function return 0 if ok, -1 if canvas or texture are missing.

pt3em

- int `caca_get_frame_count` (`caca_canvas_t` const *)
Get the number of frames in a canvas.
- int `caca_set_frame` (`caca_canvas_t` *, int)
Activate a given canvas frame.
- char const * `caca_get_frame_name` (`caca_canvas_t` const *)
Get the current frame's name.
- int `caca_set_frame_name` (`caca_canvas_t` *, char const *)
Set the current frame's name.
- int `caca_create_frame` (`caca_canvas_t` *, int)
Add a frame to a canvas.
- int `caca_free_frame` (`caca_canvas_t` *, int)
Remove a frame from a canvas.

Returns

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

12.9.2.5 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

<i>cv</i>	A libcacaca canvas
<i>id</i>	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().

12.9.2.6 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

pt3empt3emcv	pt3emA libcaca canvas
pt3em <i>d</i>	pt3emThe index of the frame to delete

pt3emReturns

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

References `caca_add_dirty_rect()`.

pt3em

pt3em

12.10 libcaca bitmap dithering

Functions

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

pt3em

Set dithering algorithm.

- char const *const * [caca_get_dither_algorithm_list](#) ([caca_dither_t](#) const *)

Get dithering algorithms.

- char const * [caca_get_dither_algorithm](#) ([caca_dither_t](#) const *)

Get current dithering algorithm.

- int [caca_dither_bitmap](#) ([caca_canvas_t](#) *, int, int, int, int, [caca_dither_t](#) const *, void const *)

Dither a bitmap on the canvas.

- int [caca_free_dither](#) ([caca_dither_t](#) *)

Free the memory associated with a dither.

12.10.1 Detailed Description

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

12.10.2 Function Documentation

12.10.2.1 [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

pt3em pt3embpp	pt3emBitmap depth in bits per pixel.
pt3em pt3emw	pt3emBitmap width in pixels.
pt3em pt3emh	pt3emBitmap height in pixels.
pt3em pt3empitch	pt3emBitmap pitch in bytes.
pt3em pt3emrmask	pt3emBitmask for red values.
pt3em pt3emgmask	pt3emBitmask for green values.
pt3em pt3embmask	pt3emBitmask for blue values.
pt3em pt3emamask	pt3emBitmask for alpha values.

pt3em

pt3emReturns

Dither object upon success, NULL if an error occurred.

12.10.2.2 `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.

12.10.2.3 `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

12.10.2.4 float **caca_get_dither_brightness** (**caca_dither_t** const * *d*)

Get the brightness of the given dither object.

This function never fails.

pt3emParameters

pt3em	pt3em	<i>d</i>	pt3emDither object.
-------	-------	----------	---------------------

pt3emReturns

Brightness value.

12.10.2.5 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	pt3em	<i>d</i>	pt3emDither object.
	pt3em	<i>gamma</i>	pt3emGamma value.

pt3emReturns

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

12.10.2.6 float **caca_get_dither_gamma** (**caca_dither_t** const * *d*)

Get the gamma of the given dither object.

This function never fails.

pt3emParameters

pt3em	pt3em	<i>d</i>	pt3emDither object.
-------	-------	----------	---------------------

pt3emReturns

Gamma value.

pt3em

12.10.2.7 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 <i>pt3emd</i>	pt3emDither object.
pt3em <i>contrast</i>	pt3emcontrast value.

pt3emReturns

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

12.10.2.8 float **caca_get_dither_contrast** (*caca_dither_t* const * *d*)

Get the contrast of the given dither object.

This function never fails.

pt3emParameters

pt3em <i>pt3emd</i>	pt3emDither object.
---------------------	---------------------

pt3emReturns

Contrast value.

12.10.2.9 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.

12.10.2.10 `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.

12.10.2.11 `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.

12.10.2.12 `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.

12.10.2.13 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

12.10.2.14 `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.

12.10.2.15 `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.

12.10.2.16 `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.

12.10.2.17 `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.

12.10.2.18 `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.

12.10.2.19 **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.

12.10.2.20 **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

```
12.10.2.21 int caca_dither_bitmap ( caca_canvas_t * cv, int x, int y, int w, int h,
caca_dither_t const * d, void const * 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

12.11 libcaca character font handling

Functions

- `caca_charfont_t * caca_load_charfont` (void const *, size_t)
- `int caca_free_charfont` (caca_charfont_t *)

12.11.1 Detailed Description

These functions provide character font handling routines.

pt3em

pt3em

12.12 libcaca bitmap font handling

Functions

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

12.12.1 Detailed Description

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

12.12.2 Function Documentation

12.12.2.1 `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

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()`.

12.12.2.2 `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.

12.12.2.3 `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.

12.12.2.4 `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

pt3em pt3em	pt3emf	pt3emThe font, as returned by caca_load_font()
----------------	--------	--

Returns

The standard glyph height.

12.12.2.5 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

pt3em	pt3emf	pt3emThe font, as returned by caca_load_font()
-------	--------	--

Returns

The list of Unicode blocks supported by the font.

12.12.2.6 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

pt3em pt3emcv	pt3emThe canvas to render
------------------	---------------------------

Parameters

<i>font</i>	The font, as returned by caca_load_font()
<i>buf</i>	The image buffer
<i>width</i>	The width (in pixels) of the image buffer
<i>height</i>	The height (in pixels) of the image buffer
<i>pitch</i>	The pitch (in bytes) of an image buffer line.

Returns

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

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

12.12.2.7 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.

Parameters

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

Returns

This function always returns 0.

pt3em

pt3em

12.13 libcaca FIGfont handling

Functions

- int [caca_canvas_set_figfont](#) ([caca_canvas_t](#) *, char const *)
load a figfont and attach it to a canvas
- int [caca_set_figfont_smush](#) ([caca_canvas_t](#) *, char const *)
set the smushing mode of the figfont rendering
- int [caca_set_figfont_width](#) ([caca_canvas_t](#) *, int)
set the width of the figfont rendering
- int [caca_put_figchar](#) ([caca_canvas_t](#) *, uint32_t)
paste a character using the current figfont
- int [caca_flush_figlet](#) ([caca_canvas_t](#) *)
flush the figlet context

12.13.1 Detailed Description

These functions provide FIGlet and TOLlet font handling routines.

pt3em

pt3em

12.14 libcaca file IO

Functions

- `caca_file_t * caca_file_open` (char const *, const char *)
Open a file for reading or writing.
- int `caca_file_close` (caca_file_t *)
Close a file handle.
- uint64_t `caca_file_tell` (caca_file_t *)
Return the position in a file handle.
- size_t `caca_file_read` (caca_file_t *, void *, size_t)
Read data from a file handle.
- size_t `caca_file_write` (caca_file_t *, const void *, size_t)
Write data to a file handle.
- char * `caca_file_gets` (caca_file_t *, char *, int)
Read a line from a file handle.
- int `caca_file_eof` (caca_file_t *)
Tell whether a file handle reached end of file.

12.14.1 Detailed Description

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

12.14.2 Function Documentation

12.14.2.1 `caca_file_t * caca_file_open` (char const * *path*, const char * *mode*)

Create a caca file handle for a file. If the file is zipped, it is decompressed on the fly.

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

- `ENOTS` Function not implemented.
- `EINVAL` File not found or permission denied.

Parameters

<code>path</code>	The file path
<code>mode</code>	The file open mode

pt3em

pt3emReturns

A file handle to *path*.

Referenced by `caca_import_canvas_from_file()`.

12.14.2.2 `int caca_file_close (caca_file_t * fp)`

Close and destroy the resources associated with a caca file handle.

This function is a wrapper for `fclose()` or, if available, `gzclose()`.

pt3emParameters

pt3empt3em <i>fp</i>	pt3emThe file handle
----------------------	----------------------

pt3emReturns

The return value of `fclose()` or `gzclose()`.

Referenced by `caca_import_canvas_from_file()`.

12.14.2.3 `uint64_t caca_file_tell (caca_file_t * fp)`

Return the file handle position, in bytes.

pt3emParameters

pt3empt3em <i>fp</i>	pt3emThe file handle
----------------------	----------------------

pt3emReturns

The current offset in the file handle.

12.14.2.4 `size_t caca_file_read (caca_file_t * fp, void * ptr, size_t size)`

Read data from a file handle and copy them into the given buffer.

pt3emParameters

pt3empt3em <i>fp</i>	pt3emThe file handle
pt3empt3em <i>ptr</i>	pt3emThe destination buffer
pt3em <i>size</i>	pt3emThe number of bytes to read

pt3em

pt3emReturns

The number of bytes read

Referenced by `caca_import_canvas_from_file()`.

12.14.2.5 `size_t caca_file_write (caca_file_t * fp, const void * ptr, size_t size)`

Write the contents of the given buffer to the file handle.

pt3emParameters

pt3empt3em <i>fp</i>	pt3emThe file handle
pt3em <i>ptr</i>	pt3emThe source buffer
pt3em <i>size</i>	pt3emThe number of bytes to write

pt3emReturns

The number of bytes written

12.14.2.6 `char* caca_file_gets (caca_file_t * fp, char * s, int size)`

Read one line of data from a file handle, up to one less than the given number of bytes. A trailing zero is appended to the data.

pt3emParameters

pt3empt3em <i>fp</i>	pt3emThe file handle
pt3em <i>s</i>	pt3emThe destination buffer
pt3em <i>size</i>	pt3emThe maximum number of bytes to read

pt3emReturns

The number of bytes read, including the trailing zero

12.14.2.7 `int caca_file_eof (caca_file_t * fp)`

Return the end-of-file status of the file handle.

This function is a wrapper for `feof()` or, if available, `gzeof()`.

pt3emParameters

pt3empt3em <i>fp</i>	pt3emThe file handle
----------------------	----------------------

pt3em

pt3emReturns

1 if EOF was reached, 0 otherwise

Referenced by caca_import_canvas_from_file().

pt3em

pt3em

12.15 libcaca importers/exporters from/to various

Functions

- `ssize_t caca_import_canvas_from_memory (caca_canvas_t *, void const *, size_t, char const *)`
Import a memory buffer into a canvas.
- `ssize_t caca_import_canvas_from_file (caca_canvas_t *, char const *, char const *)`
Import a file into a canvas.
- `ssize_t caca_import_area_from_memory (caca_canvas_t *, int, int, void const *, size_t, char const *)`
Import a memory buffer into a canvas area.
- `ssize_t caca_import_area_from_file (caca_canvas_t *, int, int, char const *, char const *)`
Import a file into a canvas area.
- `char const *const * caca_get_import_list (void)`
Get available import formats.
- `void * caca_export_canvas_to_memory (caca_canvas_t const *, char const *, size_t *)`
Export a canvas into a foreign format.
- `void * caca_export_area_to_memory (caca_canvas_t const *, int, int, int, int, char const *, size_t *)`
Export a canvas portion into a foreign format.
- `char const *const * caca_get_export_list (void)`
Get available export formats.

12.15.1 Detailed Description

formats

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

12.15.2 Function Documentation

12.15.2.1 `ssize_t caca_import_canvas_from_memory (caca_canvas_t * cv, void const * data, size_t len, char const * format)`

Import a memory buffer into the given libcaca canvas's current frame. The current frame is resized accordingly and its contents are replaced with the imported data.

Valid values for `format` are:

pt3em

- "": attempt to autodetect the file format.
- "caca": import native libcaca files.
- "text": import ASCII text files.
- "ansi": import ANSI files.
- "utf8": import UTF-8 files with ANSI colour codes.
- "bin": import BIN files.

The number of bytes read is returned. If the file format is valid, but not enough data was available, 0 is returned.

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

- ENOMEM Not enough memory to allocate canvas.
- EINVAL Invalid format requested.

pt3emParameters

pt3em	pt3emA libcaca canvas in which to import the file.
pt3emdata	pt3emA memory area containing the data to be loaded into the canvas.
pt3em/en	pt3emThe size in bytes of the memory area.
pt3emformat	pt3emA string describing the input format.

pt3emReturns

The number of bytes read, or 0 if there was not enough data, or -1 if an error occurred.

Referenced by `caca_import_area_from_memory()`, and `caca_import_canvas_from_file()`.

12.15.2.2 `ssize_t caca_import_canvas_from_file (caca_canvas_t * cv, char const * filename, char const * format)`

Import a file into the given libcaca canvas's current frame. The current frame is resized accordingly and its contents are replaced with the imported data.

Valid values for `format` are:

- "": attempt to autodetect the file format.
- "caca": import native libcaca files.
- "text": import ASCII text files.

Parameters

- `"ansi"`: import ANSI files.
- `"utf8"`: import UTF-8 files with ANSI colour codes.
- `"bin"`: import BIN files.

The number of bytes read is returned. If the file format is valid, but not enough data was available, 0 is returned.

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

- **ENOSYS** File access is not implemented on this system.
- **ENOMEM** Not enough memory to allocate canvas.
- **EINVAL** Invalid format requested. `caca_import_file()` may also fail and set **errno** for any of the errors specified for the routine `fopen()`.

Parameters

<code>cv</code>	A libcaca canvas in which to import the file.
<code>filename</code>	The name of the file to load.
<code>format</code>	A string describing the input format.

Returns

The number of bytes read, or 0 if there was not enough data, or -1 if an error occurred.

References `caca_file_close()`, `caca_file_eof()`, `caca_file_open()`, `caca_file_read()`, and `caca_import_canvas_from_memory()`.

Referenced by `caca_import_area_from_file()`.

12.15.2.3 `ssize_t caca_import_area_from_memory (caca_canvas_t * cv, int x, int y, void const * data, size_t len, char const * format)`

Import a memory buffer into the given libcaca canvas's current frame, at the specified position. For more information, see [caca_import_canvas_from_memory\(\)](#).

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

- **EINVAL** Unsupported format requested or invalid coordinates.
- **ENOMEM** Not enough memory to allocate canvas.

pt3em

pt3emParameters

pt3em	pt3em	pt3emA libcaca canvas in which to import the file.
pt3em	pt3em	pt3emThe leftmost coordinate of the area to import to.
pt3em	pt3em	pt3emThe topmost coordinate of the area to import to.
pt3em	pt3em	pt3emA memory area containing the data to be loaded into the canvas.
pt3em	pt3em	pt3emThe size in bytes of the memory area.
pt3em	pt3em	pt3emA string describing the input format.

pt3emReturns

The number of bytes read, or 0 if there was not enough data, or -1 if an error occurred.

References `caca_blit()`, `caca_create_canvas()`, `caca_free_canvas()`, and `caca_import_canvas_from_memory()`.

12.15.2.4 `ssize_t caca_import_area_from_file (caca_canvas_t * cv, int x, int y, char const * filename, char const * format)`

Import a file into the given libcaca canvas's current frame, at the specified position. For more information, see [caca_import_canvas_from_file\(\)](#).

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

- **ENOSYS** File access is not implemented on this system.
- **ENOMEM** Not enough memory to allocate canvas.
- **EINVAL** Unsupported format requested or invalid coordinates. `caca_import_file()` may also fail and set **errno** for any of the errors specified for the routine `fopen()`.

pt3emParameters

pt3em	pt3em	pt3emA libcaca canvas in which to import the file.
pt3em	pt3em	pt3emThe leftmost coordinate of the area to import to.
pt3em	pt3em	pt3emThe topmost coordinate of the area to import to.
pt3em	pt3em	pt3emThe name of the file to load.
pt3em	pt3em	pt3emA string describing the input format.

pt3emReturns

The number of bytes read, or 0 if there was not enough data, or -1 if an error occurred.

pt3em

References `caca_blit()`, `caca_create_canvas()`, `caca_free_canvas()`, and `caca_import_canvas_from_file()`.

12.15.2.5 `char const* const* caca_get_import_list (void)`

Return a list of available import formats. The list is a NULL-terminated array of strings, interleaving a string containing the internal value for the import format, to be used with `caca_import_canvas()`, and a string containing the natural language description for that import format.

This function never fails.

pt3emReturns

An array of strings.

12.15.2.6 `void* caca_export_canvas_to_memory (caca_canvas_t const * cv, char const * format, size_t * bytes)`

This function exports a libcaca canvas into various foreign formats such as ANSI art, HTML, IRC colours, etc. The returned pointer should be passed to `free()` to release the allocated storage when it is no longer needed.

Valid values for `format` are:

- `"caca"`: export native libcaca files.
- `"ansi"`: export ANSI art (CP437 charset with ANSI colour codes).
- `"html"`: export an HTML page with CSS information.
- `"html3"`: export an HTML table that should be compatible with most navigators, including textmode ones.
- `"irc"`: export UTF-8 text with mIRC colour codes.
- `"ps"`: export a PostScript document.
- `"svg"`: export an SVG vector image.
- `"tga"`: export a TGA image.
- `"troff"`: export a troff source.

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

- `EINVAL` Unsupported format requested.
- `ENOMEM` Not enough memory to allocate output buffer.

pt3emParameters

pt3em

pt3em

pt3emcv	pt3emA libcacca canvas
pt3emformat	pt3emA string describing the requested output format.
pt3embytes	pt3emA pointer to a size_t where the number of allocated bytes will be written.

pt3emReturns

A pointer to the exported memory area, or NULL in case of error.

Referenced by `caca_export_area_to_memory()`.

12.15.2.7 `void* caca_export_area_to_memory (caca_canvas_t const * cv, int x, int y, int w, int h, char const * format, size_t * bytes)`

This function exports a portion of a *libcacca* canvas into various formats. For more information, see [caca_export_canvas_to_memory\(\)](#).

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

- **EINVAL** Unsupported format requested or invalid coordinates.
- **ENOMEM** Not enough memory to allocate output buffer.

pt3emParameters

pt3emcv	pt3emA libcacca canvas
pt3emx	pt3emThe leftmost coordinate of the area to export.
pt3emy	pt3emThe topmost coordinate of the area to export.
pt3emw	pt3emThe width of the area to export.
pt3emh	pt3emThe height of the area to export.
pt3emformat	pt3emA string describing the requested output format.
pt3embytes	pt3emA pointer to a size_t where the number of allocated bytes will be written.

pt3emReturns

A pointer to the exported memory area, or NULL in case of error.

References `caca_blit()`, `caca_create_canvas()`, `caca_export_canvas_to_memory()`, and `caca_free_canvas()`.

12.15.2.8 `char const* const* caca_get_export_list (void)`

Return a list of available export formats. The list is a NULL-terminated array of strings, interleaving a string containing the internal value for the export format, to be used with `caca_export_memory()`, and a string containing the natural language description for that export format.

pt3em

This function never fails.

pt3emReturns

An array of strings.

pt3em

pt3em

12.16 libcaca display functions

Functions

- `caca_display_t * caca_create_display (caca_canvas_t *)`
Attach a caca graphical context to a caca canvas.
- `caca_display_t * caca_create_display_with_driver (caca_canvas_t *, char const *)`
Attach a specific caca graphical context to a caca canvas.
- `char const *const * caca_get_display_driver_list (void)`
Get available display drivers.
- `char const * caca_get_display_driver (caca_display_t *)`
Return a caca graphical context's current output driver.
- `int caca_set_display_driver (caca_display_t *, char const *)`
Set the output driver.
- `int caca_free_display (caca_display_t *)`
Detach a caca graphical context from a caca backend context.
- `caca_canvas_t * caca_get_canvas (caca_display_t *)`
Get the canvas attached to a caca graphical context.
- `int caca_refresh_display (caca_display_t *)`
Flush pending changes and redraw the screen.
- `int caca_set_display_time (caca_display_t *, int)`
Set the refresh delay.
- `int caca_get_display_time (caca_display_t const *)`
Get the display's average rendering time.
- `int caca_get_display_width (caca_display_t const *)`
Get the display width.
- `int caca_get_display_height (caca_display_t const *)`
Get the display height.
- `int caca_set_display_title (caca_display_t *, char const *)`
Set the display title.
- `int caca_set_mouse (caca_display_t *, int)`
Show or hide the mouse pointer.
- `int caca_set_cursor (caca_display_t *, int)`
Show or hide the cursor.

12.16.1 Detailed Description

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

pt3em

12.16.2 Function Documentation

12.16.2.1 `caca_display_t*` `caca_create_display` (`caca_canvas_t*` *cv*)

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 `caca_canvas` is provided, a new one is created. Its handle can be retrieved using `caca_get_canvas()` and it is automatically destroyed when `caca_free_display()` is called.

See also [caca_create_display_with_driver\(\)](#).

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

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

pt3emParameters

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

pt3emReturns

The caca graphical context or NULL if an error occurred.

References `caca_create_display_with_driver()`.

```
12.16.2.2  caca_display_t* caca_create_display_with_driver ( caca_canvas_t * cv,
char const * driver )
```

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 `caca_canvas` is provided, a new one is created. Its handle can be retrieved using `caca_get_canvas()` and it is automatically destroyed when `caca_free_display()` is called.

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

See also `caca create display()`.

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

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

Parameters

<i>cv</i>	The caca canvas or NULL to create a canvas automatically.
<i>driver</i>	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()`.

12.16.2.3 `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.

12.16.2.4 `char const* caca_get_display_driver (caca_display_t * dp)`

Return the given display's current output driver.

This function never fails.

Parameters

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

Returns

A static string.

12.16.2.5 `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

<i>dp</i>	The caca display.
<i>driver</i>	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.

12.16.2.6 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

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

Returns

This function always returns 0.

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

12.16.2.7 `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

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

pt3emReturns

12.16.2.8 int caca_refresh_display (caca_display t * dp)

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.

pt3emParameters

<code>pt3emp3emdp</code>	<code>pt3em</code>	The libcac display context.
--------------------------	--------------------	-----------------------------

pt3emReturns

References `caca` `clear` `dirty` `rect` `list()`.

12.16.2.9 int caca_set_display_time (caca_display t * dp, int usec)

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

<code>pt3emdp</code>	The libcacca display context.
<code>pt3emusec</code>	The refresh delay in microseconds.

pt3emReturns

0 upon success, -1 if an error occurred.

pt3em

12.16.2.10 int **caca_get_display_time** (caca_display_t const * *dp*)

Get the average rendering time, which is the average measured time between two [caca_refresh_display\(\)](#) calls, in microseconds. If constant framerate was activated by calling [caca_set_display_time\(\)](#), the average rendering time will be close to the requested delay even if the real rendering time was shorter.

This function never fails.

pt3emParameters

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

pt3emReturns

The render time in microseconds.

12.16.2.11 int **caca_get_display_width** (caca_display_t const * *dp*)

If libcaca runs in a window, get the usable window width. This value can be used for aspect ratio calculation. If libcaca does not run in a window or if there is no way to know the font size, most drivers will assume a 6x10 font is being used. Note that the units are not necessarily pixels.

This function never fails.

pt3emParameters

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

pt3emReturns

The display width.

12.16.2.12 int **caca_get_display_height** (caca_display_t const * *dp*)

If libcaca runs in a window, get the usable window height. This value can be used for aspect ratio calculation. If libcaca does not run in a window or if there is no way to know the font size, assume a 6x10 font is being used. Note that the units are not necessarily pixels.

This function never fails.

pt3emParameters

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

Returns

The display height.

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

Returns

0 upon success, -1 if an error occurred.

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

Returns

0 upon success, -1 if an error occurred.

12.16.2.15 `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.

<code>pt3emdp</code>	The libcaca display context.
<code>pt3emflag</code>	0 hides the cursor, 1 shows the system's default cursor (usually a white rectangle). Other values are reserved for future use.

0 upon success, -1 if an error occurred.

Referenced by `caca_conio__setcursortype()`.

pt3em

pt3em

12.17 libcaca event handling

Functions

- int `caca_get_event` (`caca_display_t *`, int, `caca_event_t *`, int)
Get the next mouse or keyboard input event.
- int `caca_get_mouse_x` (`caca_display_t` const *)
Return the X mouse coordinate.
- int `caca_get_mouse_y` (`caca_display_t` const *)
Return the Y mouse coordinate.
- enum `caca_event_type` `caca_get_event_type` (`caca_event_t` const *)
Return an event's type.
- int `caca_get_event_key_ch` (`caca_event_t` const *)
Return a key press or key release event's value.
- uint32_t `caca_get_event_key_utf32` (`caca_event_t` const *)
Return a key press or key release event's Unicode value.
- int `caca_get_event_key_utf8` (`caca_event_t` const *, char *)
Return a key press or key release event's UTF-8 value.
- int `caca_get_event_mouse_button` (`caca_event_t` const *)
Return a mouse press or mouse release event's button.
- int `caca_get_event_mouse_x` (`caca_event_t` const *)
Return a mouse motion event's X coordinate.
- int `caca_get_event_mouse_y` (`caca_event_t` const *)
Return a mouse motion event's Y coordinate.
- int `caca_get_event_resize_width` (`caca_event_t` const *)
Return a resize event's display width value.
- int `caca_get_event_resize_height` (`caca_event_t` const *)
Return a resize event's display height value.

12.17.1 Detailed Description

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

12.17.2 Function Documentation

12.17.2.1 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	pt3em	The libcaca graphical context.
	pt3em	pt3em	Bitmask of requested events.
	pt3em	pt3em	A timeout value in microseconds, -1 for blocking behaviour
	pt3em	pt3em	A 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.

Referenced by `caca_conio_getch()`, and `caca_conio_kbhit()`.

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

pt3emReturns

The X mouse coordinate.

References `caca_get_canvas_width()`.

12.17.2.3 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.

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

pt3emReturns

The key value.

Referenced by `caca_conio_getch()`, and `caca_conio_kbhit()`.

12.17.2.6 `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	The libcaca event.
-------	-------	-------	--------------------

pt3emReturns

The key's Unicode value.

12.17.2.7 `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	The libcaca event.
	pt3em	pt3em	A string buffer with enough bytes to hold the pressed key value in UTF-8. Though fewer bytes may be written to it, 7 bytes is the minimum safe size.

pt3em

pt3emReturns

This function always returns 0.

12.17.2.8 `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.

This function returns 1 for the left mouse button, 2 for the right mouse button, and 3 for the middle mouse button.

pt3emParameters

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

pt3emReturns

The event's mouse button.

12.17.2.9 `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

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

pt3emReturns

The event's X mouse coordinate.

12.17.2.10 `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.

pt3em

pt3emParameters

pt3em	pt3em	pt3em	The libcacaca event.
-------	-------	-------	----------------------

pt3emReturns

The event's Y mouse coordinate.

12.17.2.11 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

pt3em	pt3em	pt3em	The libcacaca event.
-------	-------	-------	----------------------

pt3emReturns

The event's new display width value.

12.17.2.12 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

pt3em	pt3em	pt3em	The libcacaca event.
-------	-------	-------	----------------------

pt3emReturns

The event's new display height value.

pt3em

pt3em

12.18 libcaca process management

Functions

- int **caca_getopt** (int, char *const [], char const *, struct [caca_option](#) const *, int *)

Variables

- int **caca_optind**
- char * **caca_optarg**

12.18.1 Detailed Description

These functions help with various process handling tasks such as option parsing, DLL injection.

pt3em

pt3em

12.19 libcaca DOS conio.h compatibility layer

Functions

- char * [caca_conio_cgets](#) (char *str)
DOS conio.h cgets() equivalent.
- void [caca_conio_cleol](#) (void)
DOS conio.h cleol() equivalent.
- void [caca_conio_clrscr](#) (void)
DOS conio.h clrscr() equivalent.
- int [caca_conio_cprintf](#) (const char *format,...)
DOS conio.h cprintf() equivalent.
- int [caca_conio_cputs](#) (const char *str)
DOS conio.h cputs() equivalent.
- int [caca_conio_cscanf](#) (char *format,...)
DOS stdio.h cscanf() equivalent.
- void [caca_conio_delay](#) (unsigned int)
DOS dos.h delay() equivalent.
- void [caca_conio_delline](#) (void)
DOS conio.h delline() equivalent.
- int [caca_conio_getch](#) (void)
DOS conio.h getch() equivalent.
- int [caca_conio_getche](#) (void)
DOS conio.h getche() equivalent.
- char * [caca_conio_getpass](#) (const char *prompt)
DOS conio.h getpass() equivalent.
- int [caca_conio_gettext](#) (int left, int top, int right, int bottom, void *destin)
DOS conio.h gettext() equivalent.
- void [caca_conio_gettextinfo](#) (struct [caca_conio_text_info](#) *r)
DOS conio.h gettextinfo() equivalent.
- void [caca_conio_gotoxy](#) (int x, int y)
DOS conio.h gotoxy() equivalent.
- void [caca_conio_highvideo](#) (void)
DOS conio.h highvideo() equivalent.
- void [caca_conio_inslne](#) (void)
DOS conio.h inslne() equivalent.
- int [caca_conio_kbhit](#) (void)
DOS conio.h kbhit() equivalent.
- void [caca_conio_lowvideo](#) (void)
DOS conio.h lowvideo() equivalent.

pt3em

- int [caca_conio_movetext](#) (int left, int top, int right, int bottom, int destleft, int desttop)
DOS conio.h movetext() equivalent.
- void [caca_conio_normvideo](#) (void)
DOS conio.h normvideo() equivalent.
- void [caca_conio_nosound](#) (void)
DOS dos.h nosound() equivalent.
- int [caca_conio_printf](#) (const char *format,...)
DOS stdio.h printf() equivalent.
- int [caca_conio_putch](#) (int ch)
DOS conio.h putch() equivalent.
- int [caca_conio_puttext](#) (int left, int top, int right, int bottom, void *destin)
DOS conio.h puttext() equivalent.
- void [caca_conio__setcursortype](#) (int cur_t)
DOS conio.h __setcursortype() equivalent.
- void [caca_conio_sleep](#) (unsigned int)
DOS dos.h sleep() equivalent.
- void [caca_conio_sound](#) (unsigned int)
DOS dos.h sound() equivalent.
- void [caca_conio_textattr](#) (int newattr)
DOS conio.h textattr() equivalent.
- void [caca_conio_textbackground](#) (int newcolor)
DOS conio.h textbackground() equivalent.
- void [caca_conio_textcolor](#) (int newcolor)
DOS conio.h textcolor() equivalent.
- void [caca_conio_textmode](#) (int newmode)
DOS conio.h textmode() equivalent.
- int [caca_conio_ungetch](#) (int ch)
DOS conio.h ungetch() equivalent.
- int [caca_conio_wherex](#) (void)
DOS conio.h wherex() equivalent.
- int [caca_conio_wherey](#) (void)
DOS conio.h wherey() equivalent.
- void [caca_conio_window](#) (int left, int top, int right, int bottom)
DOS conio.h window() equivalent.

12.19.1 Detailed Description

These functions implement DOS-like functions for high-level text operations.

pt3em

13 Data Structure Documentation

13.1 caca_conio_text_info Struct Reference

DOS text area information.

Data Fields

- unsigned char [winleft](#)
- unsigned char [wintop](#)
- unsigned char [winright](#)
- unsigned char [winbottom](#)
- unsigned char [attribute](#)
- unsigned char [normattr](#)
- unsigned char [currmode](#)
- unsigned char [screenheight](#)
- unsigned char [screenwidth](#)
- unsigned char [curx](#)
- unsigned char [cury](#)

13.1.1 Detailed Description

This structure stores text area information for the DOS conio.h compatibility layer.

13.1.2 Field Documentation

13.1.2.1 unsigned char caca_conio_text_info::winleft

left window coordinate

13.1.2.2 unsigned char caca_conio_text_info::wintop

top window coordinate

13.1.2.3 unsigned char caca_conio_text_info::winright

right window coordinate

13.1.2.4 unsigned char caca_conio_text_info::winbottom

bottom window coordinate

pt3em

13.1.2.5 unsigned char caca_conio_text_info::attribute

text attribute

13.1.2.6 unsigned char caca_conio_text_info::normattr

normal attribute

13.1.2.7 unsigned char caca_conio_text_info::currmode

current video mode: BW40, BW80, C40, C80, or C4350

13.1.2.8 unsigned char caca_conio_text_info::screenheight

text screen's height

13.1.2.9 unsigned char caca_conio_text_info::screenwidth

text screen's width

13.1.2.10 unsigned char caca_conio_text_info::curx

x-coordinate in current window

13.1.2.11 unsigned char caca_conio_text_info::cury

y-coordinate in current window

13.2 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
```

13.2.1 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.

13.2.2 Field Documentation

13.2.2.1 enum caca_event_type caca_event::type

The event type.

13.2.2.2 union { ... } caca_event::data

The event information data

13.3 caca_option Struct Reference

Option parsing.

Data Fields

- char const * **name**
- int **has_arg**
- int * **flag**
- int **val**

13.3.1 Detailed Description

This structure contains commandline parsing information for systems where [getopt_long\(\)](#) is unavailable.

14 File Documentation

14.1 caca.h File Reference

The *libcaca* public header.

pt3em

Data Structures

- struct [caca_event](#)
Handling of user events.
- struct [caca_option](#)
Option parsing.
- struct [caca_conio_text_info](#)
DOS text area information.

Defines

- #define [CACAPI_VERSION_1](#)
- #define [CACAPI_MAGIC_FULLWIDTH](#) 0x000ffffe

Typedefs

- typedef struct caca_canvas [caca_canvas_t](#)
- typedef struct caca_dither [caca_dither_t](#)
- typedef struct caca_charfont [caca_charfont_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](#)

Enumerations

- enum [caca_color](#) { [CACAPI_BLACK](#) = 0x00, [CACAPI_BLUE](#) = 0x01, [CACAPI_GREEN](#) = 0x02, [CACAPI_CYAN](#) = 0x03, [CACAPI_RED](#) = 0x04, [CACAPI_MAGENTA](#) = 0x05, [CACAPI_BROWN](#) = 0x06, [CACAPI_LIGHTGRAY](#) = 0x07, [CACAPI_DARKGRAY](#) = 0x08, [CACAPI_LIGHTBLUE](#) = 0x09, [CACAPI_LIGHTGREEN](#) = 0x0a, [CACAPI_LIGHTCYAN](#) = 0x0b, [CACAPI_LIGHTRED](#) = 0x0c, [CACAPI_LIGHTMAGENTA](#) = 0x0d, [CACAPI_YELLOW](#) = 0x0e, [CACAPI_WHITE](#) = 0x0f, [CACAPI_DEFAULT](#) = 0x10, [CACAPI_TRANSPARENT](#) = 0x20 }
 - enum [caca_style](#) { [CACAPI_BOLD](#) = 0x01, [CACAPI_ITALICS](#) = 0x02, [CACAPI_UNDERLINE](#) = 0x04, [CACAPI_BLINK](#) = 0x08 }
 - enum [caca_event_type](#) { [CACAPI_EVENT_NONE](#) = 0x0000, [CACAPI_EVENT_KEY_PRESS](#) = 0x0001, [CACAPI_EVENT_KEY_RELEASE](#) = 0x0002, [CACAPI_EVENT_MOUSE_PRESS](#) = 0x0004, [CACAPI_EVENT_MOUSE_RELEASE](#) = 0x0008, [CACAPI_EVENT_MOUSE_MOTION](#) = 0x0010, [CACAPI_EVENT_RESIZE](#) = 0x0020, [CACAPI_EVENT_QUIT](#) = 0x0040, [CACAPI_EVENT_ANY](#) = 0xffff }
- User event type enumeration.*

pt3em

- 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_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.

- enum `CACA_CONIO_COLORS` { `CACA_CONIO_BLINK` = 128, `CACA_CONIO_BLACK` = 0, `CACA_CONIO_BLUE` = 1, `CACA_CONIO_GREEN` = 2, `CACA_CONIO_CYAN` = 3, `CACA_CONIO_RED` = 4, `CACA_CONIO_MAGENTA` = 5, `CACA_CONIO_BROWN` = 6, `CACA_CONIO_LIGHTGRAY` = 7, `CACA_CONIO_DARKGRAY` = 8, `CACA_CONIO_LIGHTBLUE` = 9, `CACA_CONIO_LIGHTGREEN` = 10, `CACA_CONIO_LIGHTCYAN` = 11, `CACA_CONIO_LIGHTRED` = 12, `CACA_CONIO_LIGHTMAGENTA` = 13, `CACA_CONIO_YELLOW` = 14, `CACA_CONIO_WHITE` = 15 }

DOS colours.

- enum `CACA_CONIO_CURSOR` { `CACA_CONIO_NOCURSOR` = 0, `CACA_CONIO_SOLIDCURSOR` = 1, `CACA_CONIO_NORMALCURSOR` = 2 }

DOS cursor modes.

- enum `CACA_CONIO_MODE` { `CACA_CONIO_LASTMODE` = -1, `CACA_CONIO_BW40` = 0, `CACA_CONIO_C40` = 1, `CACA_CONIO_BW80` = 2, `CACA_CONIO_C80` = 3, `CACA_CONIO_MONO` = 7, `CACA_CONIO_C4350` = 64 }

DOS video modes.

Functions

- `caca_canvas_t` * `caca_create_canvas` (int, int)

Initialise a libcaca canvas.

- int `caca_manage_canvas` (`caca_canvas_t` *, int(*) (void *), void *)

Manage a canvas.

pt3em

- int [caca_unmanage_canvas](#) ([caca_canvas_t](#) *, int(*) (void *), void *)
unmanage a canvas.
- int [caca_set_canvas_size](#) ([caca_canvas_t](#) *, int, int)
Resize a canvas.
- int [caca_get_canvas_width](#) ([caca_canvas_t](#) const *)
Get the canvas width.
- int [caca_get_canvas_height](#) ([caca_canvas_t](#) const *)
Get the canvas height.
- uint32_t const * [caca_get_canvas_chars](#) ([caca_canvas_t](#) const *)
Get the canvas character array.
- uint32_t const * [caca_get_canvas_attrs](#) ([caca_canvas_t](#) const *)
Get the canvas attribute array.
- int [caca_free_canvas](#) ([caca_canvas_t](#) *)
Free a libcaca canvas.
- int **caca_rand** (int, int)
- char const * [caca_get_version](#) (void)
Return the libcaca version.
- int [caca_gotoxy](#) ([caca_canvas_t](#) *, int, int)
Set cursor position.
- int [caca_wherex](#) ([caca_canvas_t](#) const *)
Get X cursor position.
- int [caca_wherey](#) ([caca_canvas_t](#) const *)
Get Y cursor position.
- int [caca_put_char](#) ([caca_canvas_t](#) *, int, int, uint32_t)
Print an ASCII or Unicode character.
- uint32_t [caca_get_char](#) ([caca_canvas_t](#) const *, int, int)
Get the Unicode character at the given coordinates.
- int [caca_put_str](#) ([caca_canvas_t](#) *, int, int, char const *)
Print a string.
- int [caca_printf](#) ([caca_canvas_t](#) *, int, int, char const *,...)
Print a formatted string.
- int [caca_vprintf](#) ([caca_canvas_t](#) *, int, int, char const *, va_list)
Print a formatted string (va_list version).
- int [caca_clear_canvas](#) ([caca_canvas_t](#) *)
Clear the canvas.
- int [caca_set_canvas_handle](#) ([caca_canvas_t](#) *, int, int)
Set cursor handle.
- int [caca_get_canvas_handle_x](#) ([caca_canvas_t](#) const *)
Get X handle position.
- int [caca_get_canvas_handle_y](#) ([caca_canvas_t](#) const *)
Get Y handle position.

pt3em

- int `caca_blit` (`caca_canvas_t *`, int, int, `caca_canvas_t` const *, `caca_canvas_t` const *)
Blit a canvas onto another one.
- int `caca_set_canvas_boundaries` (`caca_canvas_t *`, int, int, int, int)
Set a canvas' new boundaries.
- int `caca_disable_dirty_rect` (`caca_canvas_t *`)
Disable dirty rectangles.
- int `caca_enable_dirty_rect` (`caca_canvas_t *`)
Enable dirty rectangles.
- int `caca_get_dirty_rect_count` (`caca_canvas_t *`)
Get the number of dirty rectangles in the canvas.
- int `caca_get_dirty_rect` (`caca_canvas_t *`, int, int *, int *, int *, int *)
Get a canvas's dirty rectangle.
- int `caca_add_dirty_rect` (`caca_canvas_t *`, int, int, int, int)
Add an area to the canvas's dirty rectangle list.
- int `caca_remove_dirty_rect` (`caca_canvas_t *`, int, int, int, int)
Remove an area from the dirty rectangle list.
- int `caca_clear_dirty_rect_list` (`caca_canvas_t *`)
Clear a canvas's dirty rectangle list.
- int `caca_invert` (`caca_canvas_t *`)
Invert a canvas' colours.
- int `caca_flip` (`caca_canvas_t *`)
Flip a canvas horizontally.
- int `caca_flop` (`caca_canvas_t *`)
Flip a canvas vertically.
- int `caca_rotate_180` (`caca_canvas_t *`)
Rotate a canvas.
- int `caca_rotate_left` (`caca_canvas_t *`)
Rotate a canvas, 90 degrees counterclockwise.
- int `caca_rotate_right` (`caca_canvas_t *`)
Rotate a canvas, 90 degrees counterclockwise.
- int `caca_stretch_left` (`caca_canvas_t *`)
Rotate and stretch a canvas, 90 degrees counterclockwise.
- int `caca_stretch_right` (`caca_canvas_t *`)
Rotate and stretch a canvas, 90 degrees clockwise.
- uint32_t `caca_get_attr` (`caca_canvas_t` const *, int, int)
Get the text attribute at the given coordinates.
- int `caca_set_attr` (`caca_canvas_t *`, uint32_t)
Set the default character attribute.
- int `caca_unset_attr` (`caca_canvas_t *`, uint32_t)
Unset flags in the default character attribute.
- int `caca_toggle_attr` (`caca_canvas_t *`, uint32_t)

pt3em

Toggle flags in the default character attribute.

- int [caca_put_attr](#) ([caca_canvas_t](#) *, int, int, uint32_t)

Set the character attribute at the given coordinates.

- int [caca_set_color_ansi](#) ([caca_canvas_t](#) *, uint8_t, uint8_t)

Set the default colour pair for text (ANSI version).

- int [caca_set_color_argb](#) ([caca_canvas_t](#) *, uint16_t, uint16_t)

Set the default colour pair for text (truecolor version).

- uint8_t [caca_attr_to_ansi](#) (uint32_t)

Get DOS ANSI information from attribute.

- uint8_t [caca_attr_to_ansi_fg](#) (uint32_t)

Get ANSI foreground information from attribute.

- uint8_t [caca_attr_to_ansi_bg](#) (uint32_t)

Get ANSI background information from attribute.

- uint16_t [caca_attr_to_rgb12_fg](#) (uint32_t)

Get 12-bit RGB foreground information from attribute.

- uint16_t [caca_attr_to_rgb12_bg](#) (uint32_t)

Get 12-bit RGB background information from attribute.

- void [caca_attr_to_argb64](#) (uint32_t, uint8_t[8])

Get 64-bit ARGB information from attribute.

- uint32_t [caca_utf8_to_utf32](#) (char const *, size_t *)

Convert a UTF-8 character to UTF-32.

- size_t [caca_utf32_to_utf8](#) (char *, uint32_t)

Convert a UTF-32 character to UTF-8.

- uint8_t [caca_utf32_to_cp437](#) (uint32_t)

Convert a UTF-32 character to CP437.

- uint32_t [caca_cp437_to_utf32](#) (uint8_t)

Convert a CP437 character to UTF-32.

- char [caca_utf32_to_ascii](#) (uint32_t)

Convert a UTF-32 character to ASCII.

- int [caca_utf32_is_fullwidth](#) (uint32_t)

Tell whether a UTF-32 character is fullwidth.

- int [caca_draw_line](#) ([caca_canvas_t](#) *, int, int, int, int, uint32_t)

Draw a line on the canvas using the given character.

- int [caca_draw_polyline](#) ([caca_canvas_t](#) *, int const x[], int const y[], int, uint32_t)

Draw a polyline.

- int [caca_draw_thin_line](#) ([caca_canvas_t](#) *, int, int, int, int)

Draw a thin line on the canvas, using ASCII art.

- int [caca_draw_thin_polyline](#) ([caca_canvas_t](#) *, int const x[], int const y[], int)

Draw an ASCII art thin polyline.

- int [caca_draw_circle](#) ([caca_canvas_t](#) *, int, int, int, uint32_t)

Draw a circle on the canvas using the given character.

pt3em

- int [caca_draw_ellipse](#) ([caca_canvas_t](#) *, int, int, int, int, uint32_t)
Draw an ellipse on the canvas using the given character.
- int [caca_draw_thin_ellipse](#) ([caca_canvas_t](#) *, int, int, int, int)
Draw a thin ellipse on the canvas.
- int [caca_fill_ellipse](#) ([caca_canvas_t](#) *, int, int, int, int, uint32_t)
Fill an ellipse on the canvas using the given character.
- int [caca_draw_box](#) ([caca_canvas_t](#) *, int, int, int, int, uint32_t)
Draw a box on the canvas using the given character.
- int [caca_draw_thin_box](#) ([caca_canvas_t](#) *, int, int, int, int)
Draw a thin box on the canvas.
- int [caca_draw_cp437_box](#) ([caca_canvas_t](#) *, int, int, int, int)
Draw a box on the canvas using CP437 characters.
- int [caca_fill_box](#) ([caca_canvas_t](#) *, int, int, int, int, uint32_t)
Fill a box on the canvas using the given character.
- 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.
- int [caca_draw_thin_triangle](#) ([caca_canvas_t](#) *, int, int, int, int, int, int)
Draw a thin triangle on the canvas.
- 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.
- int [caca_fill_triangle_textured](#) ([caca_canvas_t](#) *cv, int coords[6], [caca_canvas_t](#) *tex, float uv[6])
Fill a triangle on the canvas using an arbitrary-sized texture.
- int [caca_get_frame_count](#) ([caca_canvas_t](#) const *)
Get the number of frames in a canvas.
- int [caca_set_frame](#) ([caca_canvas_t](#) *, int)
Activate a given canvas frame.
- char const * [caca_get_frame_name](#) ([caca_canvas_t](#) const *)
Get the current frame's name.
- int [caca_set_frame_name](#) ([caca_canvas_t](#) *, char const *)
Set the current frame's name.
- int [caca_create_frame](#) ([caca_canvas_t](#) *, int)
Add a frame to a canvas.
- int [caca_free_frame](#) ([caca_canvas_t](#) *, int)
Remove a frame from a canvas.
- [caca_dither_t](#) * [caca_create_dither](#) (int, int, int, int, uint32_t, uint32_t, uint32_t, uint32_t)
Create an internal dither object.
- 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.
- int [caca_set_dither_brightness](#) ([caca_dither_t](#) *, float)

pt3em

Set the brightness of a dither object.

- float [caca_get_dither_brightness](#) ([caca_dither_t](#) const *)

Get the brightness of a dither object.

- int [caca_set_dither_gamma](#) ([caca_dither_t](#) *, float)

Set the gamma of a dither object.

- float [caca_get_dither_gamma](#) ([caca_dither_t](#) const *)

Get the gamma of a dither object.

- int [caca_set_dither_contrast](#) ([caca_dither_t](#) *, float)

Set the contrast of a dither object.

- float [caca_get_dither_contrast](#) ([caca_dither_t](#) const *)

Get the contrast of a dither object.

- int [caca_set_dither_antialias](#) ([caca_dither_t](#) *, char const *)

Set dither antialiasing.

- char const *const * [caca_get_dither_antialias_list](#) ([caca_dither_t](#) const *)

Get available antialiasing methods.

- char const * [caca_get_dither_antialias](#) ([caca_dither_t](#) const *)

Get current antialiasing method.

- int [caca_set_dither_color](#) ([caca_dither_t](#) *, char const *)

Choose colours used for dithering.

- char const *const * [caca_get_dither_color_list](#) ([caca_dither_t](#) const *)

Get available colour modes.

- char const * [caca_get_dither_color](#) ([caca_dither_t](#) const *)

Get current colour mode.

- int [caca_set_dither_charset](#) ([caca_dither_t](#) *, char const *)

Choose characters used for dithering.

- char const *const * [caca_get_dither_charset_list](#) ([caca_dither_t](#) const *)

Get available dither character sets.

- char const * [caca_get_dither_charset](#) ([caca_dither_t](#) const *)

Get current character set.

- int [caca_set_dither_algorithm](#) ([caca_dither_t](#) *, char const *)

Set dithering algorithm.

- char const *const * [caca_get_dither_algorithm_list](#) ([caca_dither_t](#) const *)

Get dithering algorithms.

- char const * [caca_get_dither_algorithm](#) ([caca_dither_t](#) const *)

Get current dithering algorithm.

- int [caca_dither_bitmap](#) ([caca_canvas_t](#) *, int, int, int, int, [caca_dither_t](#) const *, void const *)

Dither a bitmap on the canvas.

- int [caca_free_dither](#) ([caca_dither_t](#) *)

Free the memory associated with a dither.

- [caca_charfont_t](#) * [caca_load_charfont](#) (void const *, size_t)

- int [caca_free_charfont](#) ([caca_charfont_t](#) *)

pt3em

- `caca_font_t * caca_load_font` (void const *, size_t)
Load a font from memory for future use.
- `char const *const * caca_get_font_list` (void)
Get available builtin fonts.
- `int caca_get_font_width` (caca_font_t const *)
Get a font's standard glyph width.
- `int caca_get_font_height` (caca_font_t const *)
Get a font's standard glyph height.
- `uint32_t const * caca_get_font_blocks` (caca_font_t const *)
Get a font's list of supported glyphs.
- `int caca_render_canvas` (caca_canvas_t const *, caca_font_t const *, void *, int, int, int)
Render the canvas onto an image buffer.
- `int caca_free_font` (caca_font_t *)
Free a font structure.
- `int caca_canvas_set_figfont` (caca_canvas_t *, char const *)
load a figfont and attach it to a canvas
- `int caca_set_figfont_smush` (caca_canvas_t *, char const *)
set the smushing mode of the figfont rendering
- `int caca_set_figfont_width` (caca_canvas_t *, int)
set the width of the figfont rendering
- `int caca_put_figchar` (caca_canvas_t *, uint32_t)
paste a character using the current figfont
- `int caca_flush_figlet` (caca_canvas_t *)
flush the figlet context
- `caca_file_t * caca_file_open` (char const *, const char *)
Open a file for reading or writing.
- `int caca_file_close` (caca_file_t *)
Close a file handle.
- `uint64_t caca_file_tell` (caca_file_t *)
Return the position in a file handle.
- `size_t caca_file_read` (caca_file_t *, void *, size_t)
Read data from a file handle.
- `size_t caca_file_write` (caca_file_t *, const void *, size_t)
Write data to a file handle.
- `char * caca_file_gets` (caca_file_t *, char *, int)
Read a line from a file handle.
- `int caca_file_eof` (caca_file_t *)
Tell whether a file handle reached end of file.
- `ssize_t caca_import_canvas_from_memory` (caca_canvas_t *, void const *, size_t, char const *)
Import a memory buffer into a canvas.

pt3em

- ssize_t [caca_import_canvas_from_file](#) (caca_canvas_t *, char const *, char const *)
Import a file into a canvas.
- ssize_t [caca_import_area_from_memory](#) (caca_canvas_t *, int, int, void const *, size_t, char const *)
Import a memory buffer into a canvas area.
- ssize_t [caca_import_area_from_file](#) (caca_canvas_t *, int, int, char const *, char const *)
Import a file into a canvas area.
- char const *const * [caca_get_import_list](#) (void)
Get available import formats.
- void * [caca_export_canvas_to_memory](#) (caca_canvas_t const *, char const *, size_t *)
Export a canvas into a foreign format.
- void * [caca_export_area_to_memory](#) (caca_canvas_t const *, int, int, int, int, char const *, size_t *)
Export a canvas portion into a foreign format.
- char const *const * [caca_get_export_list](#) (void)
Get available export formats.
- [caca_display_t](#) * [caca_create_display](#) (caca_canvas_t *)
Attach a caca graphical context to a caca canvas.
- [caca_display_t](#) * [caca_create_display_with_driver](#) (caca_canvas_t *, char const *)
Attach a specific caca graphical context to a caca canvas.
- char const *const * [caca_get_display_driver_list](#) (void)
Get available display drivers.
- char const * [caca_get_display_driver](#) (caca_display_t *)
Return a caca graphical context's current output driver.
- int [caca_set_display_driver](#) (caca_display_t *, char const *)
Set the output driver.
- int [caca_free_display](#) (caca_display_t *)
Detach a caca graphical context from a caca backend context.
- [caca_canvas_t](#) * [caca_get_canvas](#) (caca_display_t *)
Get the canvas attached to a caca graphical context.
- int [caca_refresh_display](#) (caca_display_t *)
Flush pending changes and redraw the screen.
- int [caca_set_display_time](#) (caca_display_t *, int)
Set the refresh delay.
- int [caca_get_display_time](#) (caca_display_t const *)
Get the display's average rendering time.
- int [caca_get_display_width](#) (caca_display_t const *)
Get the display width.
- int [caca_get_display_height](#) (caca_display_t const *)

pt3em

Get the display height.

- int [caca_set_display_title](#) ([caca_display_t](#) *, char const *)

Set the display title.

- int [caca_set_mouse](#) ([caca_display_t](#) *, int)

Show or hide the mouse pointer.

- int [caca_set_cursor](#) ([caca_display_t](#) *, int)

Show or hide the cursor.

- int [caca_get_event](#) ([caca_display_t](#) *, int, [caca_event_t](#) *, int)

Get the next mouse or keyboard input event.

- int [caca_get_mouse_x](#) ([caca_display_t](#) const *)

Return the X mouse coordinate.

- int [caca_get_mouse_y](#) ([caca_display_t](#) const *)

Return the Y mouse coordinate.

- enum [caca_event_type](#) [caca_get_event_type](#) ([caca_event_t](#) const *)

Return an event's type.

- int [caca_get_event_key_ch](#) ([caca_event_t](#) const *)

Return a key press or key release event's value.

- uint32_t [caca_get_event_key_utf32](#) ([caca_event_t](#) const *)

Return a key press or key release event's Unicode value.

- int [caca_get_event_key_utf8](#) ([caca_event_t](#) const *, char *)

Return a key press or key release event's UTF-8 value.

- int [caca_get_event_mouse_button](#) ([caca_event_t](#) const *)

Return a mouse press or mouse release event's button.

- int [caca_get_event_mouse_x](#) ([caca_event_t](#) const *)

Return a mouse motion event's X coordinate.

- int [caca_get_event_mouse_y](#) ([caca_event_t](#) const *)

Return a mouse motion event's Y coordinate.

- int [caca_get_event_resize_width](#) ([caca_event_t](#) const *)

Return a resize event's display width value.

- int [caca_get_event_resize_height](#) ([caca_event_t](#) const *)

Return a resize event's display height value.

- int [caca_getopt](#) (int, char *const [], char const *, struct [caca_option](#) const *, int *)

- char * [caca_conio_cgets](#) (char *str)

DOS conio.h cgets() equivalent.

- void [caca_conio_cleol](#) (void)

DOS conio.h cleol() equivalent.

- void [caca_conio_clrscr](#) (void)

DOS conio.h clrscr() equivalent.

- int [caca_conio_cprintf](#) (const char *format,...)

DOS conio.h cprintf() equivalent.

- int [caca_conio_cputs](#) (const char *str)

pt3em

- DOS conio.h cputs() equivalent.*
- int [caca_conio_cscanf](#) (char *format,...)
DOS stdio.h cscanf() equivalent.
- void [caca_conio_delay](#) (unsigned int)
DOS dos.h delay() equivalent.
- void [caca_conio_delline](#) (void)
DOS conio.h delline() equivalent.
- int [caca_conio_getch](#) (void)
DOS conio.h getch() equivalent.
- int [caca_conio_getche](#) (void)
DOS conio.h getche() equivalent.
- char * [caca_conio_getpass](#) (const char *prompt)
DOS conio.h getpass() equivalent.
- int [caca_conio_gettext](#) (int left, int top, int right, int bottom, void *destin)
DOS conio.h gettext() equivalent.
- void [caca_conio_gettextinfo](#) (struct [caca_conio_text_info](#) *r)
DOS conio.h gettextinfo() equivalent.
- void [caca_conio_gotoxy](#) (int x, int y)
DOS conio.h gotoxy() equivalent.
- void [caca_conio_highvideo](#) (void)
DOS conio.h highvideo() equivalent.
- void [caca_conio_incline](#) (void)
DOS conio.h incline() equivalent.
- int [caca_conio_kbhit](#) (void)
DOS conio.h kbhit() equivalent.
- void [caca_conio_lowvideo](#) (void)
DOS conio.h lowvideo() equivalent.
- int [caca_conio_movetext](#) (int left, int top, int right, int bottom, int destleft, int dest-top)
DOS conio.h movetext() equivalent.
- void [caca_conio_normvideo](#) (void)
DOS conio.h normvideo() equivalent.
- void [caca_conio_nosound](#) (void)
DOS dos.h nosound() equivalent.
- int [caca_conio_printf](#) (const char *format,...)
DOS stdio.h printf() equivalent.
- int [caca_conio_putch](#) (int ch)
DOS conio.h putch() equivalent.
- int [caca_conio_puttext](#) (int left, int top, int right, int bottom, void *destin)
DOS conio.h puttext() equivalent.
- void [caca_conio__setcursortype](#) (int cur_t)
DOS conio.h _setcursortype() equivalent.

pt3em

- void [caca_conio_sleep](#) (unsigned int)
DOS dos.h sleep() equivalent.
- void [caca_conio_sound](#) (unsigned int)
DOS dos.h sound() equivalent.
- void [caca_conio_textattr](#) (int newattr)
DOS conio.h textattr() equivalent.
- void [caca_conio_textbackground](#) (int newcolor)
DOS conio.h textbackground() equivalent.
- void [caca_conio_textcolor](#) (int newcolor)
DOS conio.h textcolor() equivalent.
- void [caca_conio_textmode](#) (int newmode)
DOS conio.h textmode() equivalent.
- int [caca_conio_ungetch](#) (int ch)
DOS conio.h ungetch() equivalent.
- int [caca_conio_wherex](#) (void)
DOS conio.h wherex() equivalent.
- int [caca_conio_wwherey](#) (void)
DOS conio.h wherey() equivalent.
- void [caca_conio_window](#) (int left, int top, int right, int bottom)
DOS conio.h window() equivalent.

Variables

- int **caca_optind**
- char * **caca_optarg**
- int [caca_conio_directvideo](#)
DOS direct video control.
- int [caca_conio__wscroll](#)
DOS scrolling control.

14.1.1 Detailed Description

pt3emAuthor

Sam Hocevar <sam@hocevar.net> 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

pt3em

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

14.1.3.3 typedef struct caca_charfont caca_charfont_t

character font structure

14.1.3.4 typedef struct caca_font caca_font_t

bitmap font structure

14.1.3.5 typedef struct caca_file caca_file_t

file handle structure

14.1.3.6 typedef struct caca_display caca_display_t

libcaca display context

14.1.3.7 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.

pt3em

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.

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.

pt3em

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.
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.

14.1.4.3 enum **CACA_CONIO_COLORS**

This enum lists the colour values for the DOS conio.h compatibility layer.

14.1.4.4 enum **CACA_CONIO_CURSOR**

This enum lists the cursor mode values for the DOS conio.h compatibility layer.

14.1.4.5 enum **CACA_CONIO_MODE**

This enum lists the video mode values for the DOS conio.h compatibility layer.

pt3em